Prediction-Based Multi-Agent Reinforcement Learning for Inherently Non-Stationary Environments

Andrei Marinescu

A Dissertation submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

2016

Declaration

Andrei Marinescu

Dated: April 4, 2016

Acknowledgements

There are a great deal of people to whom I owe thanks in one way or another, therefore I apologise in advance if your name is not on the non-exhaustive list below, there's only so many names I can remember at one time.

To my supervisor, Siobhán Clarke, for all the assiduous work, attention, and great care given throughout the supervision process, it was a privilege. Few people would have allocated so much time to correcting my work, especially when it comes to the English language.

To Ivana Dusparic, for being a great co-supervisor, colleague, neighbour and friend (not necessarily in that order). Thanks for always making time to help out with my work, and always being there when needed (your desk being two meters away from my own might have helped in that matter as well, there was no escape from my pestering).

To Colin Harris, who provided great co-supervision in the early years of my research.

To Vinny Cahill, thanks for getting me on board of the LAMP project.

I probably wouldn't have been here if it weren't for all the support from my brother, Dan, in particular in the first year of my doctorate. Dan, you have had an incredible contribution to this thesis! A big thank you also goes to my parents, Daniela and Cornel, who have always supported me throughout all these years: this wouldn't have been possible without your help.

To Joanna Pötz, for being the closest person to me (both literally and figuratively) through the last years of my doctorate: thanks for bearing with me through all the frustrations of this process, and for providing all that extra puff in moments of need. Without you, words like lady-bug, chef, bit, and strudel would have never made their way into this thesis.

To Adam Taylor, for being a research brother-in-arms during these years, and without whom the understanding of reinforcement learning would have been a much less accelerated process. It has been a pleasure sharing the same office with you over all these years, from the quiet early years of research to the elevated later ones.

To Niall O'Hara, for always being there for a piece of advice, and the rigorous attention to nutrition through early lunch scheduling and fluid early dinners. To Mithileash Mohan, for his special sense of humour, and for showing me that the time required to obtain a doctorate degree can extend to irrational lengths. To Dirk Hasselbalch and Franck Franck, for being probably the best two Danes in the world, it has been a pleasure sharing the same office and beer with you. To Vivek Nallur, for his timely office notifications, and for forgiving my whiskey debt. To Nicolás Cardozo, for spicing up research with his Colombian charm. To Michael Clear, for all the insightful talks, and the interesting views on other areas of research. To Fatemeh Golpayegani, for keeping me going with tasty Iranian treats during office hours.

To Neil O'Connor, for all those early morning swimming sessions and hot sauna talks, and many great advices in between. To Jenny Munnelly, for being the definition of Irish craic. To Raymond Cunningham, for all the football enthusiasm and his unmatched "hands-in-pockets" skills on the pitch, despite his groin drawbacks. To Jan Čurn, for the joyful and creative drawings that every now and then lightened up the gloomy moments of research. To Andronikos Nedos, for showing great archaeological skill when his help was required for the assistance of this thesis, and overall great advices on how to approach research and academia. To Tim Walsh, without whom these years would have been much less pinteresting. To Hafiz Hussein, thanks to whom the gastronomical experience acquired throughout the time-span of this thesis will be unmatched. To Shiu Lun Tsang, for always being the responsible person of the group, and for providing additional thesis material in times of need.

To Carlo Galiotto, for a friendship that extends over country borders and transcends time, the latter mostly thanks to his distorted view of it (except for the moments when he is on the football pitch). A big thank you goes to all the amazing guys from the CTVR gang - you are too many to name here, and I risk missing some out. Nevertheless, it has been a great pleasure getting to know you all unnamed soldiers.

And finally, a great thank you to all my friends, without whom I would have finished this thesis one year earlier, but without whom these years would have been much duller.

Andrei Marinescu

University of Dublin, Trinity College

October, 2015

Abstract

Multi-agent algorithms are increasingly used for autonomous decentralised control in large scale systems (e.g., computer networks, vehicular traffic or smart grids). Such systems need to be resilient to unexpected disruptions, which means that agents must autonomously adapt to changes in their operating environments. Many multi-agent approaches enable agents to learn suitable actions for each different situation encountered in the environment. Often, the systems they control operate in environments which are continuously evolving, and where agents' actions are non-deterministic, so called inherently non-stationary environments. Agents' knowledge in such environments becomes outdated, which results in them taking suboptimal actions while adapting.

This thesis focuses on a specific learning technique known as multi-agent reinforcement learning (MARL), which enables multiple agents to learn by trial-and-error how to address situations encountered in the environment, in a decentralised manner. In MARL, agents undergo an initial stage of exploration where they learn to address potential situations that can occur in the environment. However, when acting in inherently non-stationary environments, as the environment changes, agents face additional situations for which they have not been trained. In such cases, further exploration needs to be performed to accommodate changes. Existing approaches only detect such changes after they occur in the environment, and then trigger re-learning. However, during re-learning, agents underperform while exploring new solutions, and their actions negatively impact the environment. The hypothesis of this thesis is that the performance of MARL in inherently non-stationary environments can be improved by estimating future environment behaviour, to prepare agents for upcoming changes.

This thesis proposes an approach called P-MARL, whose contribution is the integration of prediction and pattern-change detection in MARL, minimising the effect of non-stationarity in the environment. In P-MARL, the environment's behaviour is modelled as a time-series, and future estimates of this behaviour are provided using prediction techniques. Prediction is implemented in two steps. In the first step, an initial estimate of the environment's future behaviour is provided through prediction. In the second step, its accuracy is continuously monitored through pattern-change detection and matching techniques. Pattern-change detection captures any significant deviation of this prediction from actual behaviour and pattern matching analyses the active environment to find a more appropriate prediction. Agents use this prediction to learn, offline, optimal actions through trial-and-error, by training in a simulation based on the predicted environment's behaviour. Afterwards, agents use the acquired knowledge to improve their performance when operating in the actual environment.

The performance of P-MARL is evaluated in a number of scenarios from a real-world inspired smart grid environment, where different parameters are predicted. This environment is characterised by non-stationary power demand patterns caused by household users and intermittent renewable energy sources. In the first scenario, the objective of P-MARL is to evenly distribute the power demand of a group of electric vehicles (EVs) over periods of low demand, while ensuring that each EV charges sufficiently for the next day's trip. In the second scenario, the objective of P-MARL is to efficiently use available renewable energy while charging a group of EVs. The scenarios are based on accurate predictions of power consumption, and availability of renewable energy sources, respectively. P-MARL is evaluated over varying levels of prediction accuracy, and with two different types of agent interaction in the MARL process: one where agents' actions are taken simultaneously, and one where they are taken in sequential order. Results show that in each situation, traditional MARL is outperformed in terms of aggregate Pareto efficiency and number of agents fulfilling their charging objectives. Higher levels of interaction allow for better performance of P-MARL, as shown by the fact that sequential P-MARL achieves the best results.

Publications related to this Ph.D.

Andrei Marinescu, Ivana Dusparic, Vinny Cahill, and Siobhán Clarke. P-MARL: Prediction-based multi-agent reinforcement learning in inherently non-stationary environments. In *Transactions on Autonomous and Adaptive Systems*, (under review), ACM, 2015.

Ivana Dusparic, Adam Taylor, **Andrei Marinescu**, Vinny Cahill, and Siobhán Clarke. Maximizing Renewable Energy Use with Decentralized Residential Demand Response. In *International Smart Cities Conference (ISC2)* (pp. 1-6), IEEE, 2015

Andrei Marinescu, Ivana Dusparic, Adam Taylor, Vinny Cahill, and Siobhán Clarke. P-MARL: Prediction-based multi-agent reinforcement learning for non-stationary environments (*Short Paper*). In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (pp. 1897-1898), IFAAMAS, 2015.

Adam Taylor, Ivana Dusparic, Colin Harris, **Andrei Marinescu**, Edgar Galvan-Lopez, Fatemeh Golpayegani, Siobhan Clarke, and Vinny Cahill. Self-organising algorithms for residential demand response. In *Conference Technologies for Sustainability (SusTech)* (pp. 55-60)), IEEE, 2014.

Andrei Marinescu, Ivana Dusparic, Colin Harris, Vinny Cahill, and Siobhán Clarke. A dynamic forecasting method for small scale residential electrical demand. In *International Joint Conference on Neural Networks (IJCNN)* (pp. 3767-3774), IEEE, 2014.

Andrei Marinescu, Ivana Dusparic, Colin Harris, Siobhán Clarke and Vinny Cahill. A hybrid approach to very small scale electrical demand forecasting. In *Conference on Innovative Smart Grid Technologies (ISGT)* (pp. 1-5), IEEE, 2014.

Colin Harris, Ivana Dusparic, **Andrei Marinescu**, Edgar Galván-López, Vinny Cahill, and Siobhán Clarke. Set point control for charging of electric vehicles on the distribution network. In *Conference on Innovative Smart Grid Technologies (ISGT)* (pp. 1-5), IEEE, 2014.

Ivana Dusparic, Colin Harris, **Andrei Marinescu**, Vinny Cahill, and Siobhán Clarke. Multi-agent residential demand response based on load forecasting. In *Conference on Technologies for Sustainability (SusTech)* (pp. 90-96), IEEE, 2013.

Andrei Marinescu, Colin Harris, Ivana Dusparic, Siobhán Clarke, and Vinny Cahill. Residential electrical demand forecasting in very small scale: An evaluation of forecasting methods.

In *International Workshop on Software Engineering Challenges for the Smart Grid* (pp. 25-32), IEEE, 2013.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

This thesis presents P-MARL, a multi-agent reinforcement learning algorithm that couples prediction techniques with reinforcement learning to improve learning performance in inherently non-stationary environments. Agents use estimates of future environment behaviour to train offline and acquire important knowledge on addressing upcoming changes in the environment. As such, P-MARL enables agents to adapt to changes in non-stationary environments without negatively affecting their real-time performance. This chapter first introduces the application domain used in this thesis, smart grids, an inherently non-stationary environment which was the initial starting point for this work. Afterwards, the chapter presents the motivation for this work by introducing the issues encountered in general by multi-agent reinforcement learning (MARL) in inherently non-stationary environments. Following this, a set of requirements for a MARL system to address these issues is defined, and the objectives for a solution that fulfils this set of requirements are drawn. Furthermore, the chapter introduces the contribution of this thesis, P-MARL, while motivating the design choices behind it. Finally, the chapter concludes with a roadmap of the remainder of this thesis.

1.1 Smart Grids

The smart grid is an emerging paradigm in the area of power systems (Tsoukalas and Gao, 2008), which brings a shift from traditional power systems by enabling new capabilities such as two-way communication; resilience to anomalies; self-monitoring and self-healing abilities; and distributed control and automation (Farhangi, 2010). In addition, many countries envisage a future where renewable electricity is the predominant energy source, as attempts are made to break away from diminishing fossil fuels and reduce CO_2 emissions (Müller et al., 2011; Turner, 1999). Ireland's smart grid roadmap has targets of 40% of electricity from renewables by 2020 and 80% by 2050 (Sustainable Energy Authority of Ireland, 2015). Renewable supply will come from wind, wave, hydropower and biomass and there will be significant increase in demand mainly due to the electrification of vehicles and heating appliances. Achieving these targets will require new engineering solutions and new ways of operating the grid. Demand side management will be key in this new smart grid, enabling energy efficiency, control of demand to match intermittent renewable supply and control of demand to avoid congestion and overloading. The significant increase in demand will require the distribution network to be significantly upgraded unless an approach to coordinating the demand is engineered.

To meet these new challenges, smart grids are required to have the following key properties:

- **robustness**: smart grids need to be resilient to perturbations, and avoid single point of failure situations. Since the smart grid needs to be self-healing, the elements of the grid have to be able to adapt to changes autonomously.

- **distributed control**: smart grids are characterised by the distributed nature of power networks, which have a large geographical spread. Subcomponents of the smart grid should be able to operate independently in case of blackouts or hazards.

- **agent interaction**: the communication infrastructure of the smart grid facilitates interactions between the grid's subcomponents (e.g., smart-meters which exchange information with electric utility companies have already been deployed). This enables efficient restoration of power network in case of faults.

Recent developments in technology are bringing down the price for powerful computing units to very accessible levels, to the point where adding them to enable intelligent control for devices can cost as low as 5 dollars (Upton, 2015). Such advances make inexpensive agent-based control feasible for smart grid devices.

Smart grid environments have already been investigated as potential real-world application fields for multi-agent systems (MAS) (Kantamneni et al., 2015; McArthur et al., 2007a,b). Moreover, because energy demand is influenced by humans, who exhibit a stochastic demand pattern at individual/household level, and because there is intermittent availability of renewable energy, the smart grid is inherently non-stationary[1].

While the initial motivation of this work was to address non-stationarity in the smart grid domain, the literature review conducted for the thesis revealed that non-stationarity is actually a broader problem in multi-agent systems. Therefore the motivation of this work extends to the general case of non-stationary environments in multi-agent systems, which is presented in the next section.

1.2 Motivation

Advances in multi-agent algorithms have enabled large-scale systems to perform complex tasks without requiring human assistance (Huebscher and McCann, 2008). Such systems need to self-configure, self-optimize, self-heal and self-protect (Kephart and Chess,

[1]Power demand in smart-grids can be affected by irregular patterns in household consumption, as this constantly changes over time. Furthermore, it can be impacted by external events such as blackouts or particular climatic phenomena.

2003). Multi-agent systems (MAS) comprise multiple autonomous entities, known as agents, that interact with an environment and/or each other (Ferber, 1999). MAS can be used to solve problems in a distributed manner, when centralised control becomes infeasible. Often, autonomy is achieved by these systems continuously learning from their interaction with the environment, rather than relying on predefined behaviour (Stone and Veloso, 2000). For this purpose, many multi-agent approaches comprise agents that are presented with target goals and sets of actions available at design time (Jennings et al., 1998). This enables agents to learn through exploration by trial-and-error which actions best meet their goals. Learning is accomplished by trying various actions in the environment and analysing their effect (Russell and Norvig, 2003). After sufficient exploration is performed, an agent can decide appropriate actions for each type of situation encountered in the environment (Alonso et al., 2001). However, learning is challenging when multi-agent systems operate in complex environments, as non-stationary behaviour affects the results of their actions (Busoniu et al., 2008). Non-stationary behaviour can occur for two reasons:

- **agent-contributed non-stationarity:** when several agents explore, simultaneously, actions within the same environment, their actions' effect is non-deterministic, since the environment reacts differently to each unique combination of actions (Shoham and Leyton-Brown, 2008).
- **environment-induced non-stationarity:** when the environment continuously evolves by itself, the result of an agent's action is also affected by the independent evolution of the environment (Klügl et al., 2005). This type of environment is further referred to as an *inherently non-stationary environment*.

Large scale systems in current operating environments demand increased levels of autonomy to maintain required quality-of-service levels (Northrop et al., 2006). Many such environments are inherently non-stationary and comprise a large number of interacting autonomous entities, therefore non-stationarity is both agent-contributed and environment-induced (Weyns et al., 2005). Learning plays an important part, as new knowledge is continuously incorporated to improve performance. However, learning is difficult when many autonomous agents attempt to learn simultaneously. When exploring different actions, agents not only affect the environment, but each other's learning process as well (Shoham and Leyton-Brown, 2008). In multi-agent setups, agents' actions' long-term benefit requires actions to be repeatedly tried out, thus resulting in longer periods of exploration (Thrun, 1992). Furthermore, when acting in inherently non-stationary environments, agents face previously unencountered situations. Once agents detect a change in environment dynamics, they go through a readaptation stage where they learn strategies that address the changes (Thrun and Möller, 1992). While learning, such a system performs sub-optimally, which is undesirable in many real-world applications.

1.3 Multi-Agent Reinforcement Learning

Multi-agent reinforcement learning (MARL) is a widespread learning-based technique for decentralised control in large scale systems, which enables multiple agents to learn how to perform well in shared environments through trial-and-error (Tan, 1993). MARL has been applied in domains such as computer networks, vehicular traffic, and power networks (Riedmiller et al., 2001; Tillotson et al., 2004; Wiering, 2000). Although there are theoretical guarantees of optimality for single agent learning in stationary environments, or in very particular multi-agent cases of deterministic environments with static repeated games (Busoniu et al., 2008), these are no longer valid when MARL is applied in more complex real-world environments. As the environment continuously evolves, learned information becomes outdated given new environment dynamics. When the environment's dynamics changes, it reacts differently to agents' actions.

One of the ways of mitigating agent-contributed non-stationarity in MARL is by aggregating the learning process of multiple agents within a joint action space (Claus and Boutilier, 1998). However, when considering joint action spaces where multiple agents interact, the complexity of MARL grows exponentially with the number of agents involved, since each agent contributes to the joint space (Busoniu et al., 2008). Learning in MARL is more difficult when large numbers of agents are interacting within the same environment, because of the time needed to explore the entire joint action space (Panait and Luke, 2005). When MARL is not based on joint spaces, each new agent adds non-stationarity to MARL, since all agents are learning simultaneously and affect each other's exploration. This problem has been approached by modelling other agents' behaviour in order to predict their actions (Bowling and Veloso, 2000). However, the complexity of learning in such systems increases significantly when there is a large number of different agents within the same environment.

1.3.1 MARL in Inherently Non-stationary Environments

Besides agent-contributed non-stationarity, multi-agent systems can face an additional type of non-stationarity. When applying MARL to real-world problems, non-stationarity does not only emerge from multiple agents learning within the same environment. The independent behaviour of the environment itself also has to be factored in (Weyns et al., 2005). If the environment is inherently non-stationary, its state can change at each time-step, independently of the agents' actions in the environment (e.g., the stock market, where prices of shares also depend on external factors such as overseas trading, currency fluctuations or political climate). Furthermore, if the environment is also continuously evolving, it can also present new dynamics (e.g., vehicle driving). This impacts on the performance of MARL, as agents face previously unencountered situations, for which they have not been prepared in the exploration stages (Basso and Engel, 2009).

1.3.2 Requirements

MARL is a suitable technique for decentralised control in large scale systems (Busoniu et al., 2008). However, it presents a series of limitations when the environment in which it operates is inherently non-stationary and continuously evolving, as described in the survey by (Busoniu et al., 2008). In environments such as smart grids, agents should be able to react in a timely manner to address changes in the power network. To achieve this, exploration in the actual smart grid environment should be reduced to a minimum, as it can also negatively impact the stability of the grid.

Inspired by this application domain and further analysis of the MARL field conducted for this thesis, the following requirements for MARL in continuously evolving non-stationary environments are derived:

- **R1: Minimize Online Learning** Agents should be able to take efficient actions in any current environment state, without negatively impacting the environment, by minimizing the online learning time.

- **R2a: Detect Sudden Changes** Agents should detect when sudden changes are occurring in the environment.

- **R2b: Estimate Change Type** Agents should identify the type of change occurring in the environment, and should be able to estimate the environment's future dynamics.

- **R3: Prepare for Changes** Agents should pre-train for possible future environment dynamics, even when these have not been encountered previously.

1.4 Thesis Aims and Objectives

These requirements motivate **the research question** addressed by this thesis, which is: what extensions are needed for a MARL algorithm to improve learning in inherently non-stationary environments, and minimize the negative impact on the environment when adapting to changes?

For non-stationary environments whose behaviour can be modelled as a time-series, this thesis investigates a hypothesis that time-series analysis techniques can benefit MARL approaches. By predicting future environment behaviour, agents can learn optimal policies offline, based on a simulation of the environment. This way they are already prepared to handle the upcoming changes in the environment. This thesis aims to address the requirements introduced in Section 1.3.2 by providing techniques that:

- supply agents with additional environment information ahead of changes through advanced prediction techniques;
- enable accurate offline simulation of the environment before real-time operation in order to allow MARL agents to interact and self-organize *a priori*;
- improve agents' performance during changes with updated knowledge achieved through offline training sessions.

1.5 P-MARL

The thesis hypothesis is addressed by P-MARL, an active prediction-based learning technique that improves the performance of MARL in inherently non-stationary environments, comprising two key components: a primary prediction component, based on a fusion of artificial neural networks (ANNs) (McCulloch and Pitts, 1943) and auto-regressive integrated moving average models (ARIMA) (Box et al., 1970), and a secondary pat-

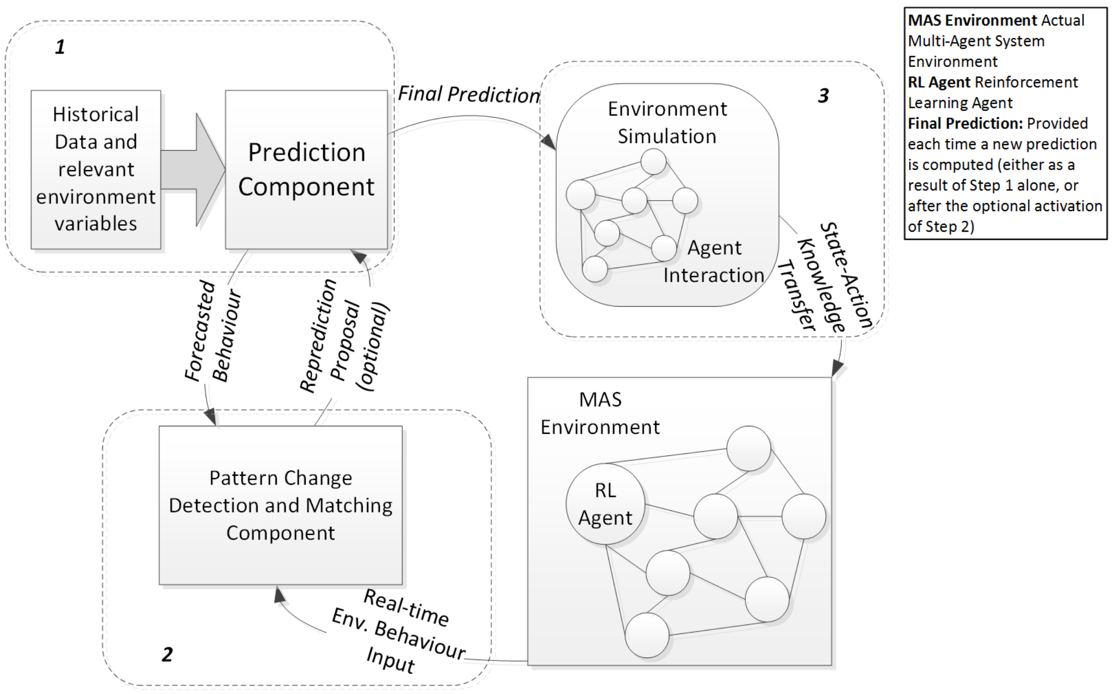

1
Historical Data and relevant environment variables
Prediction Component
Final Prediction
3
Environment Simulation
Agent Interaction
State-Action Knowledge Transfer
MAS Environment Actual Multi-Agent System Environment
RL Agent Reinforcement Learning Agent
Final Prediction: Provided each time a new prediction is computed (either as a result of Step 1 alone, or after the optional activation of Step 2)
Forecasted Behaviour
Reprediction Proposal (optional)
MAS Environment
RL Agent
Pattern Change Detection and Matching Component
Real-time Env. Behaviour Input
2

Fig. 1.1: MARL Algorithm Architecture

tern change detection and matching component based on self-organizing maps (SOM) (Kohonen, 1990). The first component provides predicted estimates of environment behaviour at set intervals, while the second component actively monitors the accuracy of the provided solution. Anomalous events can cause the MARL system to underperform, as it encounters unexpected dynamics where its actions have undesired effects. If the prediction accuracy considerably degrades along the prediction horizon (i.e, the forecasted sequence of time-steps), the second component detects this and matches the type of anomaly causing the decrease in accuracy. When an anomaly is matched, the second component triggers a reprediction procedure based on the type of anomaly detected. This process is visualised in Figure 1.1.

Once details about future changes in environment dynamics are supplied, the MARL system benefits from a separate training period before interacting with the actual environment. Agents explore offline how to perform well without negatively impacting the environment, while learning and interacting within a simulation based on an environment estimate. This way, agents prepare suitable actions to address the potential states to be encountered in the future environment's behaviour.

When MARL achieves good performance[2] the system is ready to act on the actual environment. The hypothesis is that prediction of future environment behaviour can provide agents with a sufficiently good *a priori* training model for offline learning to improve their performance in online mode (Dusparic et al., 2013). Predictive-MARL (P-MARL) addresses this. P-MARL is a MARL technique augmented with environment prediction and pattern change detection abilities for inherently non-stationary environments.

1.6 Evaluation

This work focuses on improving MARL in complex large scale real-world inspired scenarios. In this thesis, a demand side management problem for a realistic evaluation of P-MARL is proposed. Specifically, the two chosen scenarios comprise a community of households and a business park, which contain a set of electric vehicles (EVs) that need to be sufficiently charged for the next day's trip, or need to efficiently use the available renewable energy, respectively. The charging process needs to be accomplished without generating peaks in demand over the aggregated baseload. The scenarios are based on real-world data, where actual power demand is employed from a smart-meter trial in Ireland (Comission for Energy Regulation, 2011). To simulate the smart grid scenario, GridLAB-D[3] was employed, an open-source power system simulation environment. The GridLAB-D code was modified to integrate power demand prediction and agent-based

[2]There are several ways to determine performance in MARL. Convergence is one criteria used for deciding when a system is performing well. Other options include making use of a separate performance measuring procedure that can validate the system once it reaches a certain threshold, or stop the training after a certain amount of episodes.

[3]U.S. Department of Energy at Pacific Northwest National Laboratory

control for EVs. The performance of P-MARL is empirically evaluated in GridLAB-D under the assumption that direct communication between agents is not possible, and under three different prediction accuracy levels: simple prediction, which is provided by the initial prediction component; SOM reprediction, which is provided when anomalous events are detected and matched; and perfect prediction, which assumes that the prediction perfectly matches the demand occurring. Note that the latter case is used for comparison purposes only. The simulation scenarios enable agents to interact only indirectly, via the environment. Two main types of such indirect interaction have been identified, depending on the degree of restrictions within the environment: a type where agents act simultaneously, without knowledge of each other's action, and one where their actions are sequential, with the previous agents' effect on the environment being observable. In these scenarios, P-MARL is evaluated against a traditional MARL approach, which employs historical information instead of prediction.

1.7 Contribution

The main contribution of this work is the integration of prediction abilities in the MARL process. This prepares agents for upcoming changes in the environment and addresses adaptation shortcomings in non-stationary environments. Specifically:

1. Current MARL approaches only learn to adapt online, when they encounter different environment dynamics. Learning online has negative consequences on the environment, as throughout the exploration stages agents take both good and bad actions in the environment to discover which actions are best suited for each situation. In P-MARL, future environment behaviour is provided as a forecast for a predefined horizon, enabling agent offline learning, where a solution to address the changes in the environment is prepared offline, without affecting the actual

environment.

2. Current MARL approaches detect when an agent performs badly only when there are changes to the rewards it receives for taking an action, and thus infer that a change has occurred in the environment. Because changes in rewards need to be consistent over several episodes for an agent to detect a change, it takes time for the agent to detect a change in the environment. In P-MARL, sudden changes in the environment are detected using environment monitoring techniques, and agents are informed when previously expected dynamics are outdated. This way, agents are made aware of changes in the environment faster than by inspecting changes in the rewards received.

3. Current MARL approaches keep models only of previously encountered situations. When new dynamics are encountered in the environment, new models need to be learned online, and agents perform suboptimally while doing so. In P-MARL, the type of anomalous change occurring in the environment is further analysed (even though this type of dynamics has not been encountered previously), and a predicted estimate of its impact is provided to agents. Agents can use this estimated dynamics to prepare before acting online.

4. In current MARL approaches change detection and adaptation occurs online, which negatively impacts the environment due to the potentially bad decisions taken by agents while adapting. In P-MARL, agents' adjustment to sudden anomalous changes in the environment implies a preliminary offline session, to minimize the negative effect on the environment when adapting to changes. A special short-term horizon[4] with the expected environment behaviour is provided to the agents to rapidly find suitable actions.

[4]This horizon is shorter and within the initial prediction horizon.

5. Current MARL approaches address agent-contributed non-stationarity online. P-MARL reduces agent-contributed non-stationarity by enabling an offline simulation of the environment, where agents act upon the simulated environment over repeated episodes, and learn how to achieve efficient strategies while interacting in multi-agent setups.

1.8 Assumptions

A series of assumptions are made when designing and evaluating P-MARL. Considering an operating environment which is continuously evolving, one of the assumptions is that relevant information is available to assist prediction. This implies that the environment's non-stationary behaviour is not completely random, and that there are external influencing factors besides agents' actions, rendering the future environment's behaviour as being (partly) predictable through additional information sources. Furthermore, it is assumed that the environment's behaviour can be modelled as a time-series.

Agents interact indirectly, only through the environment, without any agent-to-agent communication abilities. Most of the multi-agent situations occurring in real-world situations only present a form of indirect interaction between agents (Keil and Goldin, 2006). Examples include insects interacting through pheromone trails, vehicles driving on the highway which have to adapt (among others) to other vehicles, manufacturing processes where robots adapt to changes performed on the product by other robots previously, or stock markets where sellers and buyers interact indirectly to negotiate prices.

All agents participating in the environment are built on the same type of underlying process, reinforcement learning, and are assumed to be cooperative with regard to the global objective. None of the agents engages in malicious behaviour, and all are assumed to be failure-free. Additionally, agents are not affected by any significant time delays when acting on the environment, therefore the state of the environment does not change between their observation of it and the action they take.

1.9 Thesis Roadmap

The remainder of this thesis is organised as follows:

- Chapter 2 introduces reinforcement learning, reviews existing research on MARL in non-stationary environments, presents current time-series prediction techniques employed in environment modelling, and showcases potential applications of multi-agent systems in smart grid environments.
- Chapter 3 presents P-MARL, the proposed approach for MARL in non-stationary environments which employs predicting behaviour and detecting changes in the environment as input elements into a MARL algorithm.
- Chapter 4 describes the particular implementation of P-MARL in a real-world inspired smart grid environment.
- Chapter 5 displays the results obtained by P-MARL after being evaluated in a set of smart grid scenarios.
- Chapter 6 presents concluding remarks and avenues for future work.

Chapter 2

State of the Art

This chapter introduces the concept of decentralised control, with particular focus on multi-agent systems. Background concepts are presented as follows: Section 2.1 provides a general introduction to decentralised control and multi-agent systems; Section 2.2 details reinforcement learning, the underlying technique used in multi-agent reinforcement learning (MARL); and Section 2.5 presents time-series prediction techniques, to provide the necessary background for P-MARL, the prediction-based MARL approach proposed in this thesis for decentralised control in inherently non-stationary environments.

The state of the art in MARL with focus on non-stationary environments is presented in Section 2.3. Section 2.4 further reviews other techniques employed for optimization in non-stationary environments. Section 2.6 presents model predictive control (MPC), another method that employs prediction to enhance control, where its drawbacks are also discussed. Finally, in Section 2.7, the chapter introduces the main application domain used in this thesis, smart grids, and the various forecasting mechanisms and multi-agent systems developed for it.

2.1 Decentralised Control and Multi-Agent Systems

Decentralised control is a form of control where the subcomponents of a system operate on local information to accomplish local and/or global goals, without requiring coordination from a central unit (Khosrow-Pour, 2008). This type of control is desirable when centralised solutions become infeasible due to scalability, robustness concerns, and geographical separation (Sandell et al., 1978). The functionality of the subcomponents depends on each one's interaction with the operating environment. To operate efficiently in dynamic environments, some of these subcomponents require a certain level of autonomy. These type of autonomous entities are known as *agents*. An agent is capable of perceiving the environment and uses this information to effect actions on it so as to achieve the best (expected) outcome (Russell and Norvig, 2003). A simplified architecture of an agent is illustrated in Fig. 2.1.

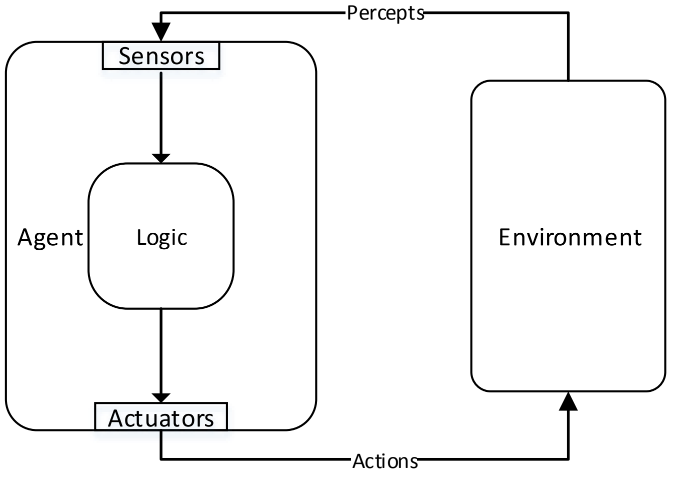


Fig. 2.1: Agent Interacting with the Environment

Agents can operate in many different types of environments. The main categories are summarised below, in mutually excluding pairs, based on the definitions provided by (Russell and Norvig, 2003)[1]:

- **static vs. dynamic**
  - *static environments*: the environment in which the agent operates reacts only based on the agent's action.
  - *dynamic environments*: the environment can change without input from the agent, to potentially unknown states.
- **fully vs. partially observable**
  - *fully observable environments*: the full state of the environment is available to the agent at each point in time.
  - *partially observable environments*: parts of the state of the environment are unobservable to the agent.
- **deterministic vs. stochastic**
  - *deterministic environments*: an action taken in the current state fully determines the next state of the environment.
  - *stochastic environments*: an action taken in the current state can lead to different states.
- **stationary vs. non-stationary**
  - *stationary environment*: a stationary environment does not evolve over time and has a predefined set of states.

[1]An additional pair of environment definitions is included (*stationary* vs. *non-stationary*), which extends the definition of dynamic environments, and is relevant for this thesis.

  - *non-stationary environment*: a non-stationary environment evolves over time and can lead an agent to previously unencountered states.

When multiple agents operate within the same environment, they form a *multi-agent system* (MAS). Even though an environment can be static and deterministic from a single agent perspective, when multiple agents act on it, the environment becomes dynamic and stochastic[2]. There are two main types of MAS, cooperative and competitive[3]. Agents in competitive MAS have conflicting objectives, or are competing for the same resources (e.g., market operators) (Ferber, 1999). Agents in cooperative MAS have a common goal, which is, essentially, decentralised control (Ferber, 1999). This thesis focuses exclusively on cooperative MAS.

2.1.1 MAS Architectures

There are different types of interaction possible between agents in multi-agent systems. When agents exchange messages directly, this is known as *direct interaction.* However, agents can also interact through the changes operated on the shared environment by their actions, where these changes can be observed by other agents (Keil and Goldin, 2006). This type of interaction is known as *indirect interaction.* Most of the interactions in real-world situations are of the indirect type (Keil and Goldin, 2006). Furthermore, agents can be organised in different types of architectures, where they interact at different levels, as follows:

- *centralised architectures:* have a main controlling agent that directs the tasks to other agents. Ultimately, this type of architecture can be thought of as a single-

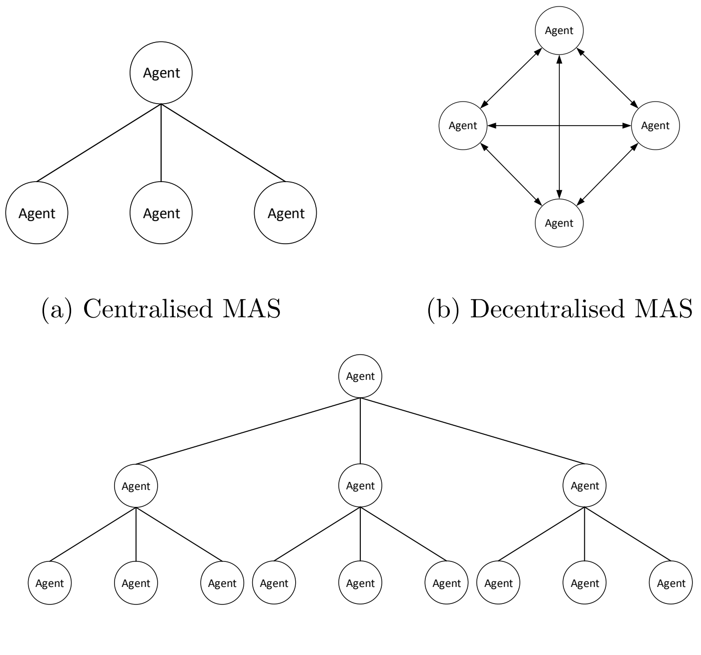
[2]Except for the particular case when agents act on separate, unrelated parts of the environment (e.g., independent workers sharing the same office building, several slot machines in a casino).

[3]The remaining types are actually variations of competitive and cooperative agents, or even combinations between the two (e.g., two football teams playing against each other, where players in each team cooperate with each other, while competing against the players from the other team). These other types are: simple collaboration (no requirement of coordination), coordinated collaboration, pure individual competition, pure collective competition (implies formation of coalition).

agent system with an agent at a high level, and different types of control available at a low level (Ferber, 1999). This type of architecture is illustrated in Fig. 2.2a.

- *decentralised architectures:* there is no central unit that influences other agents. Communication is done only on an agent-to-agent basis. This type of architecture is illustrated in Fig. 2.2b.

- *hierarchical architectures:* a combination between the previous two architectures. Agents are organised in a decentralised fashion at one level, but need to fit the directives imposed by an agent at a higher level. Agents at the higher level can communicate with same-level agents, or report to higher level agents. This type of architecture is illustrated in Fig. 2.2c.



(a) Centralised MAS (b) Decentralised MAS

(c) Hierarchical MAS

Fig. 2.2: MAS Architectures

MAS can be used to solve complex problems in a decentralised fashion, without requiring any agent to know the general problem being solved. As such, MAS present several *self-managing* properties: self-configuration, self-optimization, self-protection, self-healing and self-organization (Kephart and Chess, 2003). When an agent fails, its functionality can be replaced by other agents. MAS control systems are suitable for applications such as power networks, urban traffic networks, computer networks, flexible manufacturing networks, robotics and economics (Ferber, 1999; Sandell et al., 1978). Examples of MAS already exist in nature, e.g., ant colonies, bee swarms, fish schools or immune systems. While many of these examples have been used as an inspiration for designing MAS, one of the most commonly-used technique as the underlying process for agents is *reinforcement learning*[4].

2.2 Reinforcement Learning

Reinforcement Learning (RL) is an *unsupervised* machine learning approach that enables an agent to learn how to perform well in an unknown environment through a process of trial and error (Sutton and Barto, 1998). A reward is provided to the agent for each action taken in a given state. This process is pictured in Fig. 2.3, where X is the state transition function of the environment, s is the state transition function of the agent, a is the action taken by the agent and r is the reward received. Even though some immediate actions may lead to higher rewards than others, an agent's goal is to maximise its long term reward. This delayed reward plays a important role in an agent's attempt to reach optimal behaviour and avoid local optima. An agent needs to explore all states and actions combinations in order to learn the highest rewarding sequence of actions that can be taken from a particular state. This sequence is known as the agent's policy. To formally find an optimal policy, an agent needs to try each state-action

[4]Note however that also reinforcement learning can be considered as deriving from nature, since it is a technique with roots in behaviourist psychology.

combination an infinite number of times (Sutton and Barto, 1998). However, RL can achieve good performance once sufficient exploration through different states and actions is performed. In such situations, the key is to find a good balance between exploration and exploitation.

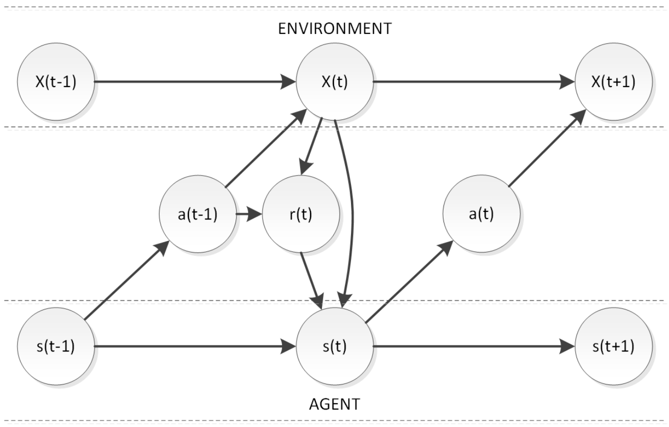


Fig. 2.3: Reinforcement Learning: Agent and Environment View

A RL process is formally defined as a Markov Decision Process (MDP), comprising the following basic elements:

- a finite set of state $S = s(1), s(2), ..., s(n)$
- a finite set of actions $A = a(1), a(2), ..., a(m)$
- a reward function $R : S \times A \rightarrow R$, where $R(s(t), a(t), s(t+1))$ is the reward received as a result of transitioning to state $st + 1$ after taking action $a(t)$ in state $s(t)$.

This is the basic form of RL, which is referred to as model-free RL. Additionally, an MDP can also have a set of rules that define the probability of transitioning from state $s(t)$ to state $s(t + 1)$ after taking action $a(t)$. In Markov models, transitions to state $s(t + 1)$ are independent of any states or actions previous to $s(t)$ and $a(t)$, respectively. In order to include information about transitions in the learning process, some RL techniques attempt to additionally model the environment. These are know as model-based RL approaches (Kaelbling et al., 1996). Model-based RL requires learning a model of state transitions to define the environment and derive appropriate actions for each state. However, model-based RL encounters difficulties when the environment in which it operates is dynamic and continuously changing. Model-free RL can adapt faster to changes in such situations. One of the most popular model-free RL algorithms is Q-Learning (Watkins and Dayan, 1992), which is presented in the following subsection.

2.2.1 Q-Learning

Q-Learning is formally defined as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big[r_{t+1} + \gamma \underset{a}{max}\, Q(s_{t+1}, a) - Q(s_t, a_t) \Big] \tag{2.1}$$

where $Q(s_t, a_t)$ is the Q-value that is updated when taking action a_t in state s_t (and represents the value of taking action a_t in state s_t), α is the learning rate, r_{t+1} is the immediate reward received at the next time-step for taking action a_t in state s_t, γ is the discount factor, and $max_a\, Q(s_{t+1}, a)$ is the maximum Q-value that can be achieved from state s_{t+1} after taking an action a. The two parameters α and γ influence how much the algorithm learns from new experience and how much weight a delayed reward holds, respectively. These parameters can only take values from the $[0, 1]$ interval, to ensure that the algorithm converges to an optimal policy. The higher the value of a parameter, the faster the algorithm is willing to learn, and the lesser the impact of previous actions is considered to be. Q-Learning is very efficient in terms of memory usage, since it recursively updates its value without the need to separately store each previous instance of the same state-action pair. However, a tradeoff between exploration and exploitation needs to be made, since formally finding an optimal policy implies that the reinforcement learning tries each state-action an infinte number of times. To address this, some techniques involve ϵ-greedy action-selection, where the agent tries a random action a set number of times, as opposed to choosing the currently available best action. Unfortunately, the randomly selected action has an equal chance to be the second-best action or the worst action. To enable a more reasonable selection of random actions, some methods have adopted *softmax* action-selection rules. One such example is the Boltzmann rule, where actions are chosen based on a parameter known as the *temperature* (Duff and Bradtke Michael, 1995). When temperatures are high, the probability of selecting any action is (nearly) equal. When the temperature has low values, actions providing high rewards yield higher chances of being selected. As an agent explores, its Boltzmann temperature decreases, therefore when the temperature falls down to 0 the agent is most likely to follow the most rewarding policy available (i.e., it moves to the exploitation stage).

If external changes occur in an environment, it will react differently to an agent's actions, therefore an agent's learned information can become outdated. In such cases, the environment is non-stationary. When an environment in which an agent is acting is non-stationary, the agent needs to re-learn in order to adapt to new changes.

2.3 MARL in Non-Stationary Environments

When several reinforcement learning agents interact within the same environment, the single agent reinforcement learning problem is extended to MARL. However, when multiple agents interact with each other the result of their actions becomes non-stationary. Non-stationarity in MARL can arise for two reasons:

1. several agents are simultaneously exploring or exploiting within the same environment, therefore the effect of their actions is non-deterministic since the environment reacts differently for each combination of actions

2. the environment continuously evolves by itself, therefore the result of an agent's action is also affected by the independent evolution of the environment

These two issues are further referred to as *agent-contributed non-stationarity* and *environment-induced non-stationarity*, respectively, and are presented in the following sections.

2.3.1 Agent-Contributed Non-Stationarity

In its single agent form, Q-Learning has been proven to reach an optimal solution in stationary environments (Watkins and Dayan, 1992). However, when several Q-Learning agents interact within the same environment, optimality is compromised because of the non-stationarity involved. Fig. 2.4 illustrates an expansion of the process in Fig. 2.3. Note that now multiple agents act upon the same environment. As such, state $X(t)$ is influenced by a set of actions $a_{1...n}(t)$, where n is the total number of agents acting within the same environment.

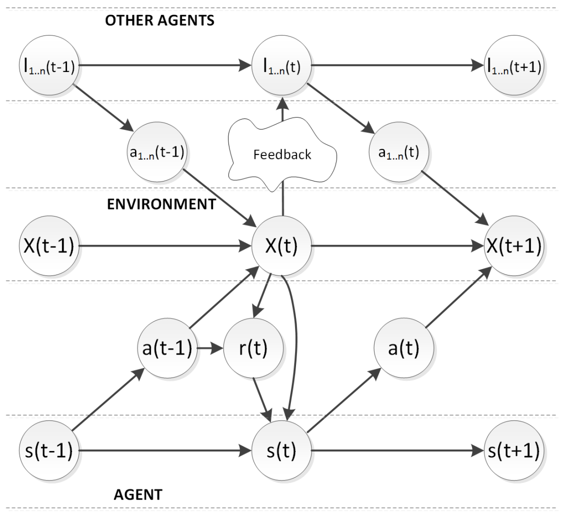


Fig. 2.4: Agent-Contributed Non-Stationarity

There has been significant research focusing on this problem from a game theoretic perspective, where an agent is acting in the presence of other non-stationary agents (Nowé et al., 2012; Shoham and Leyton-Brown, 2008). The likelihood of MARL converging to optimal equilibria in complex games has been explored (Claus and Boutilier, 1998), where it was shown that convergence guarantees of such systems are not practical. Several MARL algorithms were shown to converge to equilibria under particular conditions. One example is Nash Q-Learning (Hu and Wellman, 2003), a MARL algorithm that was proved to converge in a general-sum stochastic game where highly restrictive assumptions are made during learning. A more extended approach for general-sum games was proposed in the Friend-or-Foe algorithm (Littman, 2001). NSCP-learner (Weinberg and Rosenschein, 2004) follows the guidelines of the "AI Agenda" (Shoham et al., 2003), which suggest a focus on best-response policies instead of equilibrium, and is a MARL approach that can lead to best-response solutions for general-sum stochastic games. An extended version of Nash Q-Learning that considers Stackelberg equilibrium addresses a wider range of problems (Laumônier and Chaib-draa, 2005). Some of the more recent approaches are based on learning and adapting to an opponent's strategy, such as FAL-SG (Elidrisi et al., 2014). Also, a recently proposed framework models non-stationary opponent strategies through continuously updating decision trees (Hernandez-Leal et al., 2013).

Many of these and other multi-agent algorithms proposed are validated in simple environments, where only two agents are considered, and only the other non-stationary agent's behaviour is modelled or predicted. This is not the case in most real-world applications.

More complex large scale tasks such as load balancing have also been tackled in a distributed manner through MARL algorithms. Both non-collaborative methods (Galstyan et al., 2004; Schaerf et al., 1995) and collaborative methods (Wu et al., 2011; Zhang et al., 2009) have been shown to yield effective solutions, although there are no more guarantees of optimality. While some recent work focuses on extending prediction to model several agents' behaviour (Bazzan, 2014), very few environment prediction techniques have been suggested, and those with prediction do so only where changes can be characterized by simple binary time-series (Cetnarowicz et al., 1996) (where an environment switches only between two states but at non-stationary time intervals).

2.3.2 Environment-Induced Non-Stationarity

Non-stationarity does not only occur due to agents' actions; an environment can have this property inherently. This property is denoted as a separate non-stationary component ϵ. Fig. 2.5 illustrates how this affects the typical reinforcement learning process, previously presented in Section 2.2, where now environment state X(t) is influenced by both action $a(t-1)$ and the non-stationary component $\epsilon(t)$.

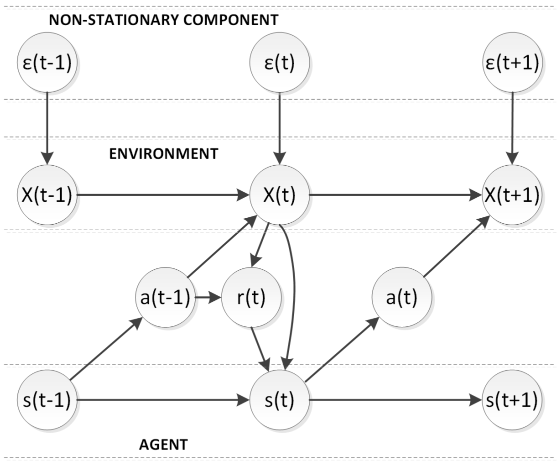


Fig. 2.5: Environment Induced Non-stationarity

2.3.2.1 Cyclical Non-Stationary Environments

Previous research approaches non-stationary environments by creating partial models of the environment (Choi et al., 2001; Doya et al., 2002). The environment behaviour is categorized into multiple partial modules, and suitable actions for each individual module are learned separately. The RL algorithms developed based on this switch between learned modules (or combinations of these) when the performance degrades. However, these solutions only address environments with repeatable dynamics. In a continuously evolving environment, new dynamics can be encountered, where previously defined models are no longer reliable.

2.3.2.2 Continously Evolving Non-Stationary Environments

Follow up research suggests Reinforcement Learning with Context Detection (RL-CD) (Da Silva et al., 2006), a context detection based approach that builds on multiple partial models. Each different environment dynamic is defined as a separate context. In RL-CD, multiple models are continuously evaluated when facing a specific environment dynamic, and the model performing best is chosen to further dictate actions. If no model performs satisfactorily, RL-CD starts learning a new model for the particular environment dynamics. The new model is triggered only when an agent consistently performs sub-optimally. In essence, RL-CD creates new models on demand. This approach was further extended through the addition of a neural architecture (Basso and Engel, 2009). The extended version detects context changes sooner than RL-CD, and thus adapts faster to changes in environment dynamics. Other techniques have been proposed to detect context changes in the environment more efficiently (Elwell and Polikar, 2009; Hadoux et al., 2014). Furthermore, another approach that combines multiple views of the environment and knowledge transfer has been proposed to address new dynamics (Trung, 2013). This is achieved by employing CMDP, an extension of MDP with factored states, where the environment is modelled with actions' effects. The idea of combining several previously learned abstract contexts to deal with changing environment dynamics is also explored by (Rosman, 2014).

Such problems have also been approached from the environment perspective in traffic (Salkham and Cahill, 2010), where the SOILSE method is proposed. Fluctuations in environment behaviour are continuously monitored using CUSUM, a moving average filter. Once a change is detected, SOILSE's learning parameters are changed in order to allow agents to adapt to the new conditions.

2.3.3 Analysis

Table 2.1: Reinforcement Learning Techniques Analysis

		Stationary Environments	**Non-Stationary Environments**	
			Agent-Contributed	Environment-Induced
Single Agent	Modelling	(Watkins and Dayan, 1992)	*N/A*	Partial (only model known dynamics): (Doya et al., 2002) (Choi et al., 2001) (Da Silva et al., 2006)
	Change Detection	*N/A*	*N/A*	(Da Silva et al., 2006) (Basso and Engel, 2009) (Hadoux et al., 2014)
Multi-Agent	Modelling	(Galstyan et al., 2004) (Wu et al., 2011) (Zhang et al., 2009)	Partial (only model one agent): (Hu and Wellman, 2003) (Littman, 2001) (Weinberg and Rosenschein, 2004) (Laumônier and Chaib-draa, 2005)	✗
	Change Detection	*N/A*	Partial (only model one agent): (Elidrisi et al., 2014) (Hernandez-Leal et al., 2013) (Bazzan, 2014)	✗

A summary of the approaches employed in MARL to mitigate non-stationarity is presented in Table 2.1. Many of the techniques employed model non-stationarity caused by other agents. However, modelling the behaviour of other agents is not feasible in complex non-stationary environments, where many agents act upon the environment and the effect of a single agent is not transparent. In situations where the environment itself is complex and non-stationary, agents' learning cannot converge to stationary policies, therefore gaining knowledge about the environment can be key to successful implementation of MARL (Klügl et al., 2005). The techniques in which the environment is modelled explore while learning a new model of an environment, so their overall performance is affected during this time. This is an undesirable situation in real-world applications where performance decrease comes at a cost, as is the case, for example, in industrial processes or vehicular traffic. From this the first requirement for MARL in non-stationary environments is derived: **R1: Minimize Online Learning**. Training offline can improve performance (Tesauro et al., 2006), but is directly affected by the quality of information provided. In current approaches, agents adjust to changes based on decreases in rewards obtained, but if changes are detected faster, directly from the environment, the system can adjusts more rapidly. This leads to requirement **R2a: Detect Suddent Changes**. Furthermore, once a change is detected the MARL could also estimate what kind of change will occur, which represents requirement **R2b: Estimate Change Type**. This information about the environment can be employed to prepare agents ahead of time, for them to take suitable actions once changes in the environment have taken place. This represents requirement **R3: Prepare for Changes**.

From this analysis, the list of requirements to improve MARL performance in inherently non-stationary environments is derived.

2.4 Optimization in Non-stationary Environments

The previous sections presented only RL approaches employed in non-stationary environments. However, other techniques have been used for optimization in inherently non-stationary environments. The techniques are presented below, depending on the algorithm at the basis of the technique.

Evolutionary Algorithms Evolutionary algorithms (EAs) were extended to non-stationary problems by the addition of diversity maintenance mechanisms and redundant genetic materials (Trojanowski and Michalewicz, 1999), which improved the evolutionary search process. Extensions to EAs were used with different purposes when dealing with dynamic environments: change detection (Cobb, 1990), injection of diversity when changes occur (Nguyen, 2011), memory usage (Goldberg and Smith, 1987), or even predicting when a change is occurring, based on the cyclical patterns found in the environment (Hatzakis and Wallace, 2006).

Particle Swarm Optimization Another technique inspired from nature, comprising a particle swarm optimizer (PSO) hybridised with a dynamic macromutation operator (Esquivel and Coello, 2004), was proposed to mitigate dynamic changes with different degrees of complexity and changing fitness landscapes. Additionally, SDPSO was suggested, where PSO was extended through individual updates of particles to enable local environment changes discovery (Cui et al., 2009). A similar technique to the one presented in the EA algorithms section, which involved the addition of diversification in to respond to dynamic changes, was further proposed (Hu and Eberhart, 2002).

Ant Colony Optimization Ant colony optimization (ACO) was employed in the DHCIAC approach (Dréo and Siarry, 2006), where a hybridisation of an interacting ant colony is created to solve continuous dynamic optimization problems. Memory-based approaches were further used to improve ACO in dynamic environments (Mavrovouniotis and Yang, 2011). Another ACO hybrid was developed to provide instantaneous rerouting when encounering dynamic changes (Chitty and Hernandez, 2004)

Artificial Immune System DynamicCS (Kim and Bentley, 2002), an artificial immune system (AIS) inspired approach, which builds on dynamic clonal selection, was proposed to tackle environments where behaviour changes occur at certain periods, and which are only partially observable. Additionally, a multi-population AIS solution was proposed to find new optima in changing environments (De França and Von Zuben, 2009).

Others Alife, an approach where autonomous agents are modelled as cells that self-organize and reproduce (Annunziato et al., 2001), was proposed for online adaptive optimisation in complex processes. This approach enables directly learning from measurements and follows the system's evolution. Bayesian optimization algorithms were employed to improve performance in dynamic environments (Kobliha et al., 2006). ODHC mitigates dynamic changes in environments (Zeng et al., 2007), and is a hill climbing-based algorithm that explores new peaks where convergence of the search for a better solution is hastened. Evolutionary game theory was employed to gain insight into the environment dynamics in MARL systems (Bloembergen et al., 2015).

Table 2.2: Optimization in Non-Stationary Environments

Technique	**Adaptation Type**
EA	diversity injection (Trojanowski and Michalewicz, 1999) memory usage (Cobb, 1990)
PS	macromutation operator (Esquivel and Coello, 2004) local change discovery (Cui et al., 2009)
ACO	enhanced communication (Dréo and Siarry, 2006) memory usage (Mavrovouniotis and Yang, 2011)
AIS	dynamic clonal selection (Kim and Bentley, 2002)
Other	self-organization and reproduction (Annunziato et al., 2001) Bayesian optimization (Kobliha et al., 2006)

2.4.1 Analysis

A summary of these techniques is presented in Table 2.2. Generally, the techniques focus on detecting changes and adapting to the environment's new behaviour. Many of them rely on memory to accommodate cyclical environment changes, but need to extend this memory when non-cyclical changes occur. As such, when previously unencountered changes occur, these require a period of adjustment until suitable solutions are found, during which they take suboptimal exploratory actions. In this thesis, non-cyclical changes are modelled ahead of time through time-series analysis techniques.

2.5 Time-Series Prediction

In order to better explain this thesis's approach to non-stationary environments, some general concepts about time-series need to be first introduced. This section briefly introduces prediction from the perspective of time-series, while in a following section, its importance in power networks, the main application domain of this thesis, is highlighted.

This thesis assumes a non-stationary environment whose behaviour can be modelled as a time-series. Such environments are commonly found in financial markets, meteorology, process monitoring, control engineering, signal processing and power networks. A time-series is a discrete sequence of points set at fixed time intervals whose values represent successive measurement of a specific variable. As such, the property X of an environment behaviour over an interval of length t can be defined as $(X_1, X_2, ..., X_i, ..., X_t)$, where X_i is the measurement of the property X taken at time i.

Many time-series analysis techniques assume that there is a relationship between X_t and $X_1, ..., X_{t-1}$, unless the series is completely random (i.e., representing white noise). Mathematical regression techniques can use previous samples of the time-series as parameters in order to forecast future samples (Makridakis et al., 1998). This process, known as auto-regression, is shown in Eq. 2.2:

$$X_t = F(X_1, ..., X_{t-1}) \tag{2.2}$$

However, in reality many time-series are related to other external variables, which can influence the evolution of the time-series. For example, the daytime length is influenced by the time of the year and the latitude where it is observed, water usage tends to increase when there are high temperatures, and currency values are impacted by the economical status of their country of origin. As such, these additional factors also need be considered when predicting future values. Eq. 2.3 extends Eq. 2.2 to accommodate these *covariates*:

$$X_t = F(< X >, < Y >, ..., < Z >) \tag{2.3}$$

where $< X >$ is a vector of previous values $X_1, ..., X_{t-1}$, and $< Y >, ..., < Z >$ are a set of vectors of other independent variables that can influence the evolution of the time-series. The mathematical technique that involves prediction by adding covariates into the equation is known as multi-regression.

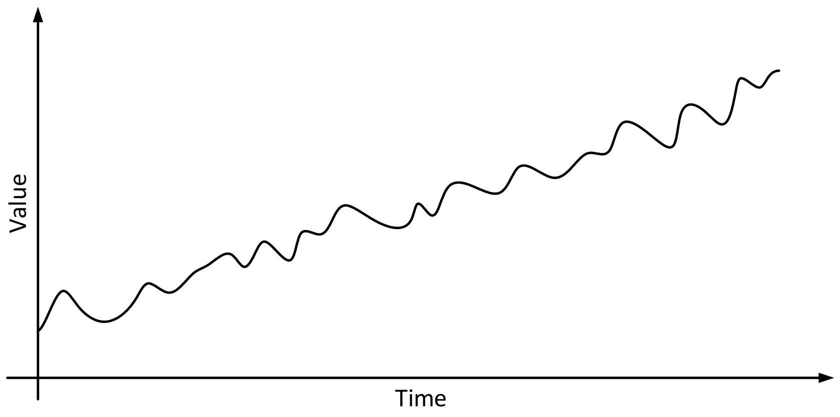


Fig. 2.6: A time-series with trend

Time-series can be further characterised by two main properties, the presence of *trend* and *seasonality.* Time-series that have the same mean and variance (over a set amount of consecutive samples) over long periods of time are considered to be stationary [5]. On the other hand, time-series that show a growing or decreasing mean over time present a trend (e.g., the evolution of the human population of Earth), and are considered to be non-stationary. An example of a time-series that exhibits trend is illustrated in Fig. 2.6.

Additionally, some time-series can exhibit cyclical behaviour, with patterns that tend

[5]However, this definition is valid for the more general set known as *weakly* stationary time-series. A more strict set, known as *strongly* stationary time-series, implies that all the statistical properties of the time-series do not change over time, therefore the joint statistical distribution of any two sequences from the time-series is the same.

to repeat (e.g., a sinusoide). This property is known as seasonality. An example of a time-series that presents seasonality is presented in Fig. 2.7. To adjust for trend and seasonality, prediction techniques use a mechanism where recent observations hold more weight than older ones.

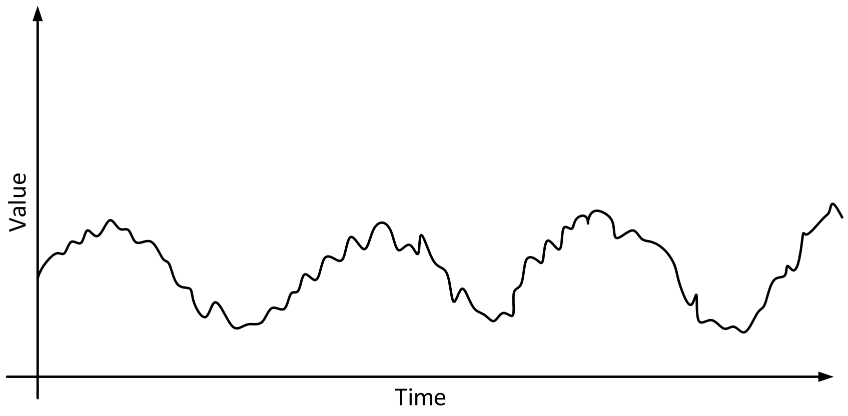


Fig. 2.7: A time-series with seasonality

Widely used approaches for predicting future values of time-series (in the above example the values $X_{t+1}, X_{t+2}, ...$) include statistical methods based on regression (multiple regression, auto-regression, generalized additive models), artificial neural networks (Zhang et al., 1998), support vector machines (Vapnik, 2013), Holt-Winters exponential smoothing methods (Winters, 1960), and fuzzy logic techniques (Brown and Harris, 1994). This section further details some of the most used methods.

2.5.1 Multiple Regression

Multiple regression techniques are an extension of simple regression. In simple regression, to predict a variable Y, a predictor variable X can be linearly related to it, as shown in

Eq. 2.4:

$$Y_t = \alpha + \beta X_t + \epsilon \tag{2.4}$$

where Y_t and X_t are the values of variables Y and X at time t, respectively, α and β are fixed (unknown) parameters, and ϵ is the forecasting error.

Eq. 2.4 can be extended to include a vector of predictor values $< X >$, which results in Eq. 2.5:

$$Y_t = \alpha + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_i X_t + \epsilon \tag{2.5}$$

This equation can be rewritten in compact form as:

$$Y_t = \alpha + < \beta, X > + \epsilon \tag{2.6}$$

where $< \beta, X >$ is the dot product of vectors $< \beta >$ and $< X >$. Eq. 2.6 can be further extended to include other predictor variables that influence the value of Y_t. The resulting equation is the generalised form of multi-regression:

$$Y_t = \alpha + < \beta', X' > + < \beta'', X'' > + ... + \epsilon \tag{2.7}$$

where X',X'', ... are the predictor variables or covariates, and β',β'',... their associated sets of parameters.

2.5.2 Auto-Regression

Statistical auto-regressive methods are established methods for time-series prediction in weakly stationary or non-stationary time-series, with the auto-regressive moving average (ARMA) model being first introduced (Whittle, 1951), and later popularised together with auto-regressive integrated moving average (ARIMA) as Box-Jenkins approaches

(Box et al., 1970). The models analyse random processes and linearly relate the output of the prediction system based on previous values of the time-series. The series is decomposed using a formula that relates individual coefficients with the former chosen p values. The auto-regressive part is actually a form of multi-regression that considers a predefined number of previous values of variable Y as predictors. The formula for auto-regression (AR) is shown in Eq. 2.8:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \epsilon_t \tag{2.8}$$

where c is a constant term, ϕ_j is the jth auto-regressive term, p is the order of the model, and ϵ_t the error term at time t.

ARMA and ARIMA additionally include a moving average part, where another set of coefficients is considered for the moving average (MA) model component. The computations for a MA model are presented in Eq. 2.9:

$$Y_t = c + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q} + \epsilon_t \tag{2.9}$$

where c is a constant term, θ_j is the jth moving average term, ϵ_{t-j} is the error term at time $t - j$, and q the order of the MA model.

The ARMA model resulting from combining Eq. 2.8 and Eq. 2.9 is:

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t \tag{2.10}$$

While ARMA can be used with weak-stationary systems, ARIMA applies differentiation on a non-stationary time series, thus removing the non-stationary component and treating the result as stationary. To address trends, ARIMA further extends the ARMA formula in Eq. 2.10. Note that Eq. 2.10 can be rewritten with *backshift* notation as:

$$(1 - \sum_{i=1}^{p} \phi_i L^i) X_t = (1 + \sum_{i=1}^{q} \theta_i L^i) \epsilon_t \tag{2.11}$$

where L is the lag operator, which operates on an element of the time-series to produce the previous one. Once differencing is applied, this equation becomes:

$$(1 - \sum_{i=1}^{p} \phi_i L^i)(1 - L)^d X_t = (1 + \sum_{i=1}^{q} \theta_i L^i) \epsilon_t \tag{2.12}$$

where d is the degree of differencing. Variations of the ARIMA model can be used to address seasonality and trends in time-series.

2.5.3 Holt-Winters Exponential Smoothing

The models based on Holt-Winters exponential smoothing address, in particular, time-series with seasonality and trends. These methods build on the MA concept, and employ a weighting scheme where older observations have decreased weight. There are three equations at the basis of the method: the level equation 2.13a, which forecasts a future value as a function of the previously observed value and the error between the previous forecasted value and the actual observed value, the trend equation 2.13b and the seasonal equation 2.13c.

$$L_t = \alpha(Y_t - S_{t-s}) + (1 - \alpha)(L_{t-1} + b_{t-1}) \tag{2.13a}$$

$$b_t = \beta(L_t - L_{t-1}) + (1 - \beta) b_{t-1} \tag{2.13b}$$

$$S_t = \gamma(Y_t - L_t) + (1 - \gamma) S_{t-s} \tag{2.13c}$$

These three equations are further used to generate the forecast in Eq. 2.14:

$$F_{t+m} = L_t + b_t m + S_{t-s+m} \tag{2.14}$$

where s is the season's length, L_t the level of the series, b_t the trend, S_t the seasonal component, and F_{t+m} the forecast for the mth value after time t. The *smoothing* characteristic of this method stems from factor L_t, which represents a *smoothed* (averaged) value of the series without the seasonality component. The factor γ accounts for the randomness in the Y time-series.

2.5.4 Artificial Neural Networks

An artificial neural network (ANN) is a supervised learning technique which is effective in prediction models where other statistical methods fail, due to the advantages brought by its non-linear approach. ANNs are inspired from nature (specifically biological neural connections), and comprise a layer of input neurons and a layer of output neurons. In between these two layers, a neural network can have one or more intermediate (or *hidden*) layers of neurons, which provide links between the input neurons and the output neurons. The structure is illustrated in Fig. 2.8. Note that this particular example represents a fully connected network, but in other cases some links between adjacent neurons can be missing.

Through an adaptive weighting mechanism at each neuron, neural networks can be trained to match an unknown function's operation. The initial values from the input neurons are combined at the hidden layer through a non-linear process known as the *activation function.* Considering a set of input neurons $Y_1, ..., Y_n$, the first formula used for combining the input neurons is shown in Eq. 2.15:

$$Z = b + \sum_{i=1}^{n} w_i Y_i \tag{2.15}$$

This is in turn used as input into the non-linear function in Eq. 2.16:

$$S(Z) = \frac{1}{1 + e^{-aZ}} \tag{2.16}$$

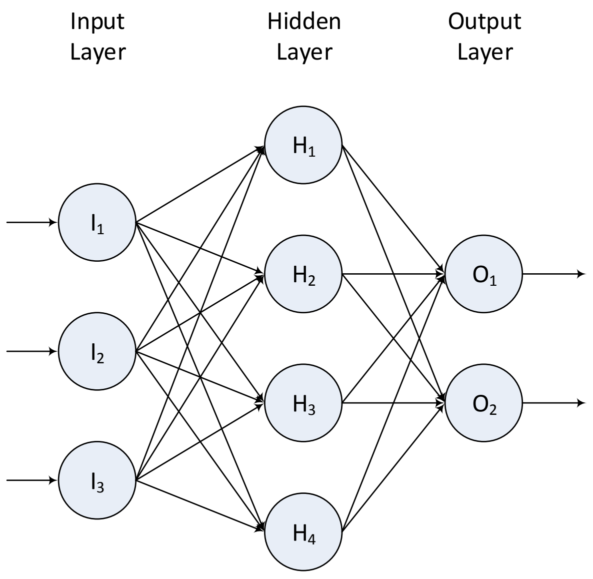


Fig. 2.8: Artificial Neural Networks Structure

where a,b are a set of parameters, and $w_1, ..., w_n$ are the weights used for each neuron. Note that some weights can be of value 0 if the network is not fully connected. The values resulting from $S(Z)$ in the hidden layer are then linearly combined to provide an output neuron. In general, the input layer comprises both previous values of the time-series and related variables/predictors. ANNs with several hidden layers are at the basis of *deep learning* (Deng and Yu, 2014), which has become popular in recent years due to the increasing computing power of microprocessors and dedicated devices.

The training process is achieved using a provided set of inputs and outputs, known as a training set. One of the most popular techniques employed for this is backpropagation (Werbos, 1994). Backpropagation calculates the gradient of a loss function while taking into account all the weights in the network. The gradient is further used in the optimization function, where the weights are updated, in an attempt to minimize the loss function. To avoid overfitting, the training process also employs a validation set, where the trained network is tested.

Once training is completed, the neural network provides similar outputs to the unknown function when faced with equivalent sets of inputs. ANNs have a "black box" nature. Even though they can provide accurate estimations of functions, the underlying process is not fully transparent as there is no explicit model of it.

2.5.5 Complementary Techniques

While directly predicting future behaviour can be very efficient, most successful techniques combine several methods to improve overall results. Among them, one of the most noticeable and effective additions is the classification of historical behaviour into different sets, generally accomplished using self-organising maps (Kohonen, 1990), k-means clustering (Hartigan and Wong, 1979), or support vector machines (Cortes and Vapnik, 1995). These techniques rely heavily on past behaviour, but key environmental variables can also play an important role. Monitoring the key variables can help estimate future changes in the environment. The following sections detail some of the most important supporting techniques.

2.5.5.1 Time-series Decomposition

Time-series can be decomposed into subsets, which, through addition, reconstruct the aggregated series. Techniques from signal processing such as the Fourier transform (Cooley and Tukey, 1965) can be employed, where the time-series is decomposed into sub-signals of different frequencies. This technique is at the basis of Wavelet Neural Networks (WNNs), which first decomposes the signal into a set of sub-signals, then performs prediction on each of the sub-signals through neural networks, and afterwards adds the resulting predictions to construct a main forecast. Another widely-used decomposing technique is Seasonal-Trend decomposition (STL) (Cleveland et al., 1990), where through an iterative process based on the *loess smoother* (Cleveland, 1981) a time-series is decomposed into three sub-signals: the trend component, the seasonal component, and the remainder component.

2.5.5.2 Pattern-Change Detection

Particular changes in time-series can affect the quality of prediction. When unexpected changes occur, these can be detected through special pattern-change detection techniques. Changes might imply deviations in statistical values such as mean, variance, correlation, or spectral density. These changes can be detected through techniques such as:

- **Bayesian inference** (Dempster, 1968), where the probability of the change hypothesis is continuously updated based on a set of underlying variables.
- **Cumulative Sum Control Chart** (CUSUM) (Page, 1961), where the values of the time-series are compared against a mean value. If the latest observed values (sliding window mechanism) deviate too much from the mean (i.e., the sum of differences from the mean goes over a certain threshold), a change is detected.
- **Cluster Analysis**, where a set of latest observations are compared against clustered groups of previous patterns (arranged based on similar characteristics). If the latest set is matched into a different cluster than previously, a change is detected.

2.5.5.3 Classification Techniques

Classification techniques are used to group sets of samples into similar classes. These can be used to improve prediction, as anomalous behaviours in time-series can be matched with previous occurrences. Popular classification methods include:

- **Self-organizing Maps** (SOM), a type of ANN based on *unsupervised learning* that categorises the information into a form of multi-dimensional map (typically bi-dimensional). Initially, input samples are randomly distributed across the map, and after several iterations similar samples are grouped together, with closely related groups neighbouring each other. After training is accomplished, SOMs can be used to classify new samples into corresponding groups.

- **K-Means clustering**, which partitions samples into k clusters where each observation falls into the cluster with the nearest mean.

- **Support Vector Machines** (SVMs), which are supervised learning techniques that can categorize data into one of two classes. The classes are divided based on a straight separation line where the biggest gap between samples is found. To increase the class number, SVMs can be extended to generate subclasses of a parent class.

2.5.6 Measuring Forecasting Accuracy

Forecasting errors are measured depending on the difference between forecasted values and actual observed values, and are often used in the training and evaluation process of a forecasting technique. Consider a series of values $X_1, X_2, ..., X_n$, and its estimates $\hat{X}_1, \hat{X}_2, ..., \hat{X}_n$, where X_i is the observed value and $\hat{X}_i$ the forecast. The simplest method to compute the average error is by using the mean error (ME) formula, as show in Eq. 2.17a. However, if there are evenly spread positive and negative errors from the forecast, these cancel each other out and result in a low ME. As this is an unrealistic estimation of error, other methods have been devised to consider errors in absolute values, such as mean absolute error (MAE), presented in Eq. 2.17b, mean squared error (MSE), shown in Eq. 2.17c, or root mean squared error (RMSE), expressed in Eq. 2.17d.

$$ME = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i) \tag{2.17a}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|X_i - \hat{X}_i| \tag{2.17b}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2 \tag{2.17c}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{X}_i)^2} \tag{2.17d}$$

However, the methods presented so far are not suitable when comparing forecasting accuracy between time-series of different scales. One solution is to normalize the RMSE measurement between the boundary values of the time-series. The formula for normalized root mean squared error (NRMSE) is presented in Eq. 2.18a. A drawback of this method is that, in case X_{max} and X_{min} are outliers, the computed error becomes misleadingly low. Another solution is to compute the forecast error as a percentage value, by dividing the resulting error to the observed value. This is the basis of the mean absolute percentile error (MAPE), which is expressed in Eq. 2.18b.

$$NRMSE = \frac{RMSE}{X_{max} - X_{min}} \tag{2.18a}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{X_i - \hat{X}_i}{X_i}\right| \tag{2.18b}$$

However, the MAPE measurement can be misleading when forecasting very low values (in absolute terms) of a series, as the percentile error can be very high, even though the forecast is close in relative terms.

2.5.7 Analysis

In this section several time-series related techniques were presented. Future time-series values can be predicted based on past behaviour, and based on their relation with correlated external variables. Some of the times-series techniques presented linearly relate available information (e.g., regression), while others make use of non-linear modelling techniques to provide forecasts (e.g., ANNs). Furthermore, these can also model the seasonality and trend present in a time-series. However, each of these methods is suited for particular types of time-series. The accuracy of a forecast can be measured through several methods, although it has been shown that each of them has drawbacks, depending on the type of time-series forecast evaluated.

Besides the ability to forecast future behaviour, some techniques can be used to detect changes in the time-series. These are useful when the time-series is non-stationary, and where the type of changes occurring can be classified.

2.6 Model Predictive Control

An alternative method to learning that combines prediction with control theory is Model Predictive Control (MPC). MPC is an advanced technique for process control, which employs models of an environment's future behaviour to optimize actions over a prediction horizon (Camacho and Alba, 2013). It is used in dynamic systems, where an agent controls the future behaviour of a system on the basis of predictions of the effect of its actions on it. MPC employs a mathematical model of the system, and involves an objective function that needs to be satisfied with respect to a set of constraints. At each time-step, based on the information currently available, MPC solves an optimization problem to determine the set of actions that need to be taken for the control agent to accomplish its goal. Only the first action is taken before the agent moves to the next time-step, where the whole process is repeated.

Model predictive control has been further extended to decentralised setups, where several control agents attempt to solve an MPC problem in a distributed manner (Camponogara et al., 2002). This technique is known as distributed MPC (DMPC), and focuses on MPC problems that are decomposed into a set of subproblems, where each subproblem is assigned to a different control agent. DMPC has applications outside of classical industrial control process problems, to large-scale systems such as traffic or power networks (Negenborn and Maestre, 2014). However, one issue is that a DMPC problem often needs to be divided into subproblems at a central unit, and assigned to different agents. If an agent is disconnected from the system, the DMPC problem's distribution needs to be reconfigured, which remains an important problem in current MPC research (Christofides et al., 2013). Furthermore, MPC problems are often solved through computationally intensive mathematical approaches such as mixed integer linear programming, which are NP-hard. When involving long prediction horizons, the MPC problems become intractable since the complexity of computing a solution increases exponentially with the number of time-steps. Some of these issues could be addressed through combining MPC with learning based techniques such as RL (Ernst et al., 2009).

2.6.1 Analysis

MPC employs a prediction of the environment's future behaviour to prepare suitable actions for control. However, there is a significant complexity penalty when the prediction horizon is long. DMPC can be used to solve MPC problems in a distributed manner, but rely on a central unit to plan the subtasks of each element, which makes it prone to failure if the central unit is disconnected. RL can be used to find suitable actions faster than MPC (Ernst et al., 2009), and can further benefit from techniques employed in MPC. Prediction can be used to enhance the learning process in MARL.

2.7 Smart Grids

The smart grid has recently emerged as a new paradigm for the electrical grid. The currently deployed electrical grid relies on a system that is mostly unidirectional in nature, based on centralised large power plants that generate the bulk of our daily demand. Fossil fuel and nuclear power plants are large systems that generate electricity in the order of gigawatts or hundreds of megawatts. The electricity produced is then used by consumers, who are located hundreds or even thousands of kilometres away from the source. The current architecture of electrical grids defines a highly centralised network, one which relies heavily on a few large power generators and the connected transmission lines. The emergence of renewable sources is a key component in the attempt to match demand requirements with supply, reduce CO_2 emissions, and decrease our dependency on fossil fuels.

In general, electrical energy demand is increasing due to the electrification of appliances and the growing number of electrical vehicles, the latter which could account for 20%-50% of the total vehicle market share by 2030 (Kay et al., 2013). Unfortunate events throughout history have proven that, due to accidents or natural disasters, large power plants can be completely disabled. This leads to power losses of several gigawatts that, comparably, can account for the total electricity generation of a small country (Andersson et al., 2005). Such events lead to severe situations that can incur lengthy blackouts and affect life-critical systems. On the other hand, in a grid that employs many generators based on renewable sources, the failure of one element can be quickly covered by other sources, drastically reducing the possibility of failure (Farhangi, 2010). Besides the drawbacks of a centralised system, traditional grids encounter difficulties in integrating renewables without affecting the nominal voltage and frequency levels of the grid (Bevrani et al., 2010). Other issues of traditional grids that need to be addressed are: transmission losses and wasted heat, which account for one third of the total energy produced by the large power plants (Farhangi, 2010); CO_2 emissions resulting from fossil fuels; and the development of self-healing abilities.

To address these challenges, the re-engineering of traditional power systems as smart grids has brought a new set requirements:

- **robustness**: smart grids need to be resilient to perturbations, and avoid single points of failure. Since the smart-grid needs to be self-healing, when some subsystems fail, other elements of the grid have to be able to replace their functionality.
- **distributed control**: smart grids are characterised by the distributed nature of power networks, which have a large geographical spread. Subcomponents of the smart grid should be able to operate independently in case of blackouts or hazards.
- **agent interaction**: the communication infrastructure of the smart grid should facilitate interactions between the grid's subcomponents (e.g., smart-meters which exchange information with electric utility companies have already been deployed).
- **managing inherently non-stationary environments**: smart grids are an inherently non-stationary environment, which continuously change and evolve. Non-stationarity can occur both at supply and demand ends. At the supply side, non-stationarity occurs due to the intermittent nature of renewable sources such as wind and solar, or when power plants malfunction. On the demand side, non-stationarity emerges from the variable power usage of consumers. Both demand and supply sides of the smart grid should cater for non-stationarity.

Demand side management will be key in this new smart grid, enabling energy efficiency, control of demand to match intermittent renewable supply and control of demand to avoid congestion and overloading (Ramchurn et al., 2012). A significant increase in demand will require the distribution network to be significantly upgraded, unless an approach to coordinating the demand is engineered.

2.7.1 Microgrids

A potential solution to address these requirements is the organization of the grid in a decentralised fashion, where it relies mainly on distributed electricity resources such as generators and storage facilities belonging to microgrids or virtual power plants (VPPs) (Backhaus et al., 2015; Robu et al., 2012). A microgrid consists of a set of power generation and storage units such as wind turbines, solar panels, combined heat and power (CHP) systems, fuel cells, batteries and fly wheels. Microgrids are geographically close to consumers, leading to lower transmission costs and enabling the use of additionally generated heat from CHP stations (Hatziargyriou et al., 2007). When blackouts occur in the electrical grid, microgrids can disconnect from the main power network and function in islanding mode, thus continuing to provide electricity to its own community of users and allowing the main grid to perform recovery operations without interfering with the customers. After the reconfiguration of the network, microgrids can be reconnected individually at different time steps to maintain the stability of the network.

2.7.2 Multi-Agent Systems in Smart Grids

Multi-agent Systems (MAS) provide decentralised solutions for problems where centralised control is not feasible due to scalability and robustness concerns. Autonomous intelligent agents are in control of separate subcomponents of a system, and can interact and collaborate to achieve a global goal. Local information is gathered by each agent, thereby eliminating the need for a central controller or a shared knowledge centre. Some of the targeted applications include electricity market operations (Bhuvaneswari et al., 2010; Kok, 2010; Weidlich and Veit, 2008), flow management (Lauri et al., 2013), grid protection (Pang et al., 2010), power restoration (Kumar et al., 2008; Lambert-Torres et al., 2009; Nagata et al., 2003), demand management (Nunna and Doolla, 2013), or small scale power grid control (Chatzivasiliadis et al., 2008; Dimeas and Hatziargyriou, 2007; Dou and Liu, 2013). Additionally, MAS were employed for supplying ancillary services in power networks Vandael et al. (2013b), in order to maintain grid stability while providing a scalable solution. MAS have also been used to support the general control and operation of microgrids (Eddy et al., 2015; Ghazvini et al., 2014; Zhao et al., 2015). A microgrid operation solution integrated consideration of grid market price, resources availability, local demand and environmental and efficiency objectives (Colson and Nehrir, 2013). (Colson et al., 2014a) further extended this to employ storage systems, while later (Colson et al., 2014b) also incorporated multiple objectives in the control solution. Furthermore, energy trading strategies under Nash equilibrium are modelled through a genetic algorithm to maximize user utility (Faqiry et al., 2014).

MAS techniques are also increasingly used in energy Demand Side Management (DSM) solutions to reduce peak power demand and thus minimize CO_2 emissions (Dusparic et al., 2013; Logenthiran et al., 2011), and better integrate renewable energy sources (Gomes et al., 2014; Mao et al., 2014). There are also important financial implications to DSM, as customers (and generators) can reduce their overall costs. Furthermore, decentralised control for DSM also has the potential to better manage scalability, robustness, reliability and privacy concerns (Amin and Wollenberg, 2005). While centralised solutions are optimal in such cases, they are also NP-hard, and are thus infeasible for real-time applications. To address this, some solutions involve combining centralised and decentralised control to minimize computational load in dynamic environments while maintaining similar levels of performance as the centralised solution Vandael et al. (2013a).

In DSM, particular attention has been given to electric vehicle (EV) charging problems, which have high power consumption and charging time flexibility. Examples include agent-based control algorithms based on grid prices and emergent coordination of agents (Ramchurn et al., 2011), where a 17% reduction of peak demand for domestic consumers was obtained; imbalance cost reductions of 44% through a multi-agent solution for coordinated plug-in hybrid EVs (Vandael et al., 2011); investigation into the use of electric vehicle fleets as virtual power plants for energy trading in wholesale markets (Kahlen et al., 2014), where it was concluded that agent-coordinated control can increase profits and reduce CO_2 emissions for EV fleet owners; and research into individual EVs, where owners benefit from smart charging strategies based on learning agents in order to achieve electricity bill savings and peak demand reductions (Valogianni et al., 2014).

However, many of the presented techniques were designed for stationary demands, where the demand is assumed to be known in advance. This is not the case in reality, as demand evolves in a non-stationary way on a day-to-day basis. Furthermore, unexpected changes can occur that significantly affect the demand when compared to initial expectations. One such example is a cold weather front that occurred in December 2010 in Ireland, which caused increased power demand compared to usual daily consumption patterns of December days (Comission for Energy Regulation, 2011).

2.7.3 Power Demand Forecasting in Smart Grids

In power networks, demand forecasting is used to estimate future energy consumption, and as such is a critical element for national grid operators and DSM. Forecasting techniques are employed to schedule generators (mainly large power plants) on a day-ahead basis so as to match supply and demand.

At a smaller scale, such as microgrids, demand forecasting has also been employed for efficient use of renewable sources, matching available demand with supply, and for minimizing end users' energy costs where dynamic energy pricing mechanisms are in use

(Hernandez et al., 2014). However, in smaller scales there is a noticeable increase in stochasticity with regard to demand patterns, as individual users have higher impact on the aggregated demand. This leads to decreased levels of precision in demand estimation (Marinescu et al., 2013; Tidemann et al., 2013).

Several different forecasting systems have been developed for power networks (Alfares and Nazeeruddin, 2002). Although high precision, as low as 1.97% mean absolute percentage error (MAPE), has been achieved on large scale (for example national and municipal level (Amjady, 2007; Beccali et al., 2004; Motamedi et al., 2012; Taylor and Mcsharry, 2008)), microgrid, VPP and transformer level forecasting has only recently emerged as a research interest (Amjady and Keynia, 2010; Fatimie et al., 2010; Hernandez et al., 2014; Llanos et al., 2012; Lloret and Valencia, 2013). The results are not very encouraging, with errors ranging from and 5.15% MAPE at university campus level (Fatimie et al., 2010), where power demand peaks at 8 MW during the day, and 7.92% MAPE at university building level (Borges et al., 2011) - up to 13.8% MAPE at village level, where power demand peaks at 15 kW (Llanos et al., 2012). Short term load forecasting (STLF) has further been done at microgrid level, with forecasting errors of 3.69% MAPE (Wai et al., 2011)[6], 6.7% MAPE (Shimoda et al., 2012), 7.92% MAPE (Chaouachi et al., 2013) and 15.12% MAPE (Chan et al., 2011).

2.7.4 Analysis

Forecasting has been employed in electricity grids both at large scale (e.g.,national, municipal level) and small scale (e.g., microgrid, campus, village). Forecasting in small scale is very important for efficiently matching demand with supply in microgrids and virtual power plants, but small scale forecasting is more difficult, as the amount of noise in the demand patterns is higher, thus resulting in greater forecasting errors.

[6]This work only forecasts over a period of 4 months, which is considerably shorter than other cases which consider yearly data.

Residential communities can take advantage of forecasted demand to achieve demand response (Dusparic et al., 2013).

To achieve efficient demand response, accurate forecasts need to be provided and demand needs to be continuously monitored to detect when real-time demand significantly deflects from the predicted demand. The information about demand is a key element in deferring the additional load that occurs in the power system. The control of the additional loads can be done in a decentralised manner, to avoid single-point of failure situations. This way each additional deferrable load has autonomous control. The systems in charge of managing the deferrable loads' need to be interconnected, to achieve global consensus offline before applying their solutions in real-time. This ensures that the global effect of their actions does not negatively impact the environment. As such, P-MARL is a suitable solution for enabling decentralised control in this domain.

In this thesis, P-MARL is applied to a smart grid problem where its task is to evenly distribute the demand of a group of electric vehicles within a residential community of households, and as such makes use of advanced forecasting techniques to model the demand of the community and detect changes in consumption patterns. This provides the basis of learning in the MARL process in the smart grid scenario, which follows the steps presented in Section 1.7, i.e., forecast the energy demand, continuously evaluate the forecast against the real-time demand of the environment to detect changes, propose change type matches if changes are detected, and employ the forecast and potential change type information to prepare agents offline for suitable actions to be taken in the environment.

2.8 Summary and Analysis

This chapter introduced multi-agent systems while highlighting their application in decentralised control. Reinforcement learning (RL), one of the most used techniques for autonomous agent learning implemented in multi-agent systems (MAS), is also presented. Particular focus is given to Q-Learning, a model-free reinforcement learning technique suitable for dynamic environments. Afterwards, the type of MAS comprising multiple RL agents interacting within the same environment, known as multi-agent RL (MARL), is described. The issues encountered by MARL in non-stationary environments are identified, and the two types of non-stationarity are presented: agent-contributed non-stationarity, and environment-induced non-stationarity. A review on current literature, presented in Section 2.3.3, analyses the methods employed for mitigating non-stationary behaviour in MARL, where it is concluded that most of the methods address only agent-contributed non-stationarity.

Further analysis of the techniques designed for MARL in inherently non-stationary environments reveals that changes in the environment are only detected, but not immediately addressed. When these changes lead to already encountered situations, agents employ previously acquired knowledge to adjust. Otherwise, agents trigger re-learning and adapt online, which negatively impacts the environment. Since the type of non-stationarity induced by the environment refers to continuously evolving environments, these environments can lead to previously unencountered situations. A summary of this analysis is presented in Table 2.3. Additionally, other techniques that address environment-induced non-stationarity are presented and discussed. However, related approaches require adaptation and exploration when new changes occur, which involve taking suboptimal actions.

Because the hypothesis being examined in this thesis relates the applicability of

Table 2.3: Adaptation in Non-Stationary Environments

	Type of Environment			**Adaptation Type**	
	Stationary	**Non-Stationary**		**Previously Encountered Dynamics**	**New Dynamics**
		Agent-Contributed	**Environment-Induced**		
Behaviour Change Detection	*N/A*	Detect change in other agents actions	Detected through change in rewards	Memory based approaches (keep previous models of environments)	Learns new dynamics online
Behaviour Prediction	*N/A*	Predict agents behaviour (limited number of agents and dynamics modelled)	MPC approach	Mixed integer linear programming (NP-hard)	✗

predictive analysis to environment-induced non-stationarity, the chapter presented some of the widely used time-series prediction techniques, and also introduced complementary methodology employed in time-series analysis. Most of the techniques presented rely on time-series analysis at most for detecting when changes occur. However, prediction is used in model predictive control. Distributed model-predictive control (DMPC), a control theory method that combines decentralised control and prediction, is further presented. This relies on a central unit to divide subtasks, and is an NP-hard problem whose complexity increases exponentially with the prediction horizon. The chapter concludes with an introduction of smart grids, the main application field of this thesis, and the multi-agent systems and forecasting techniques employed in this specific domain.

Chapter 3

Design

The state of the art in multi-agent reinforcement learning (MARL) in non-stationary environments was presented in the previous chapter. The analysis revealed that the currently proposed solutions focus on mitigating *agent-contributed non-stationarity*, while *environment-induced non-stationarity* is only addressed based on previously observed dynamics, when the new dynamics encountered matches a previously observed dynamics.

While in general, MARL research tackling non-stationarity focuses on modelling agent behaviour (the agent-contributed non-stationarity case), attention to the environment plays an equally important part. Model-based RL approaches use experience to model the environment's reaction to an agent's actions, and thus agents are able to implement strategies based on the developed model to maximize their performance. However, when the environment is inherently non-stationary, the accumulated experience becomes misleading. Agents need to acquire new experience in order to learn a new model, and while doing so they underperform. Model-free RL techniques can adapt faster to the changes in the environment, since they do not need to accumulate experience for modelling the environment, but will underperfom during the learning stage.

Agents need to re-adapt to new situations, and at the same time attempt to minimize losses. Solutions such as those based on RL-CD and SOILSE, which were presented in Section 2.3, are no longer feasible, as these detect changes in the environment and then trigger learning, and will underperfom when re-adapting.

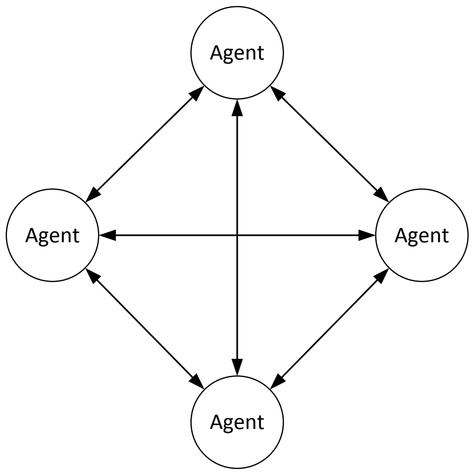
This chapter presents the main contribution of this thesis, P-MARL, an approach that addresses environment-induced non-stationarity in MARL through advanced prediction techniques. The chapter introduces the aims for a MARL algorithm to mitigate environment-induced non-stationarity, and the design approach for a MARL that enables decentralised control in this type of environment. The P-MARL solution is then presented, and its main comprising elements are further described.

3.1 Requirements for MARL in Non-stationary Environments

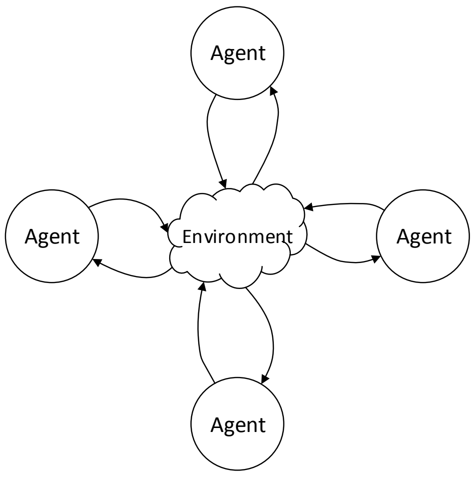
The techniques discussed in Chapter 2 only partially address the problems of non-stationarity. Environment non-stationarity is only addressed through partial models of previously encountered dynamics. The environment models are incrementally extended as new situations in the environment are encountered, but require additional learning phases, which happen online, at a cost. When evaluated against the list of requirements presented in Section 1.3.2, current approaches fully address *R2a: Detect Sudden Changes*, while only partially fulfilling *R1: Minimize Online Learning* and *R2b: Estimate Change Type*. Furthermore, *R3: Prepare for Changes* is not addressed at all.

A MARL system that fulfils all these requirements should have the following functionality:

- resolve agent-contributed non-stationarity, without affecting the environment;



(a) Direct Interaction in MAS



(b) Indirect Interaction in MAS

Fig. 3.1: MAS Interactions

- address environment-induced non-stationarity without negatively impacting the environment.

The first functionality is only partially fulfilled in literature, under restrictive assumptions, and in small multi-agent systems (MAS) (e.g., (Elidrisi et al., 2014; Weinberg and Rosenschein, 2004)). In MAS involving direct interaction, non-stationarity is mitigated by agents directly agreeing between each other which actions to take next, as can be seen in Fig. 3.1a. However, in situations where only indirect interaction exists, agents can cooperate only while interacting through the environment, which has negative consequences on the environment while an agreement is reached. This type of situation is pictured in Fig. 3.1b.

The second functionality is only fulfilled when the type of non-stationary changes occurring lead to previously encountered dynamics. To meet the requirements, the following design approach is proposed:

1. **D1: Resolve agent-based non-stationarity offline** Agent-based non-stationarity should be resolved through a simulation of the environment, offline, to avoid negatively affecting the environment.

2. **D2: Model future environment behaviour** The system should be able to generate a model of the environment for the simulation, even when the environment presents non-stationary characteristics which have not been encountered before.

3. **D3: Monitor environment behaviour and detect changes** The environment's behaviour estimate should be continuously compared against actual behaviour, to reduce the amount of inaccurate information provided to agents, and consequently the chance of agents taking suboptimal actions. As such, the system should be able to detect deviations in the environment from expected behaviour.

4. **D4: Estimate change type** The system should be able to estimate how the environment would evolve after a change, to ensure that agents address upcoming changes with suitable actions.

3.2 P-MARL Design

P-MARL employs three key components to address the design specifications presented in the previous section. These components are:

1. Prediction

2. Pattern change detection and matching

3. Offline MARL training based on prediction

Prediction is employed to address *D2: Model future environment behaviour*, **Pattern change detection and matching** is used to address *D3: Monitor environment* *behaviour and detect changes* and *D4: Estimate change type*, while **offline MARL training based on prediction** addresses *D1: Resolve agent-based non-stationarity offline*.

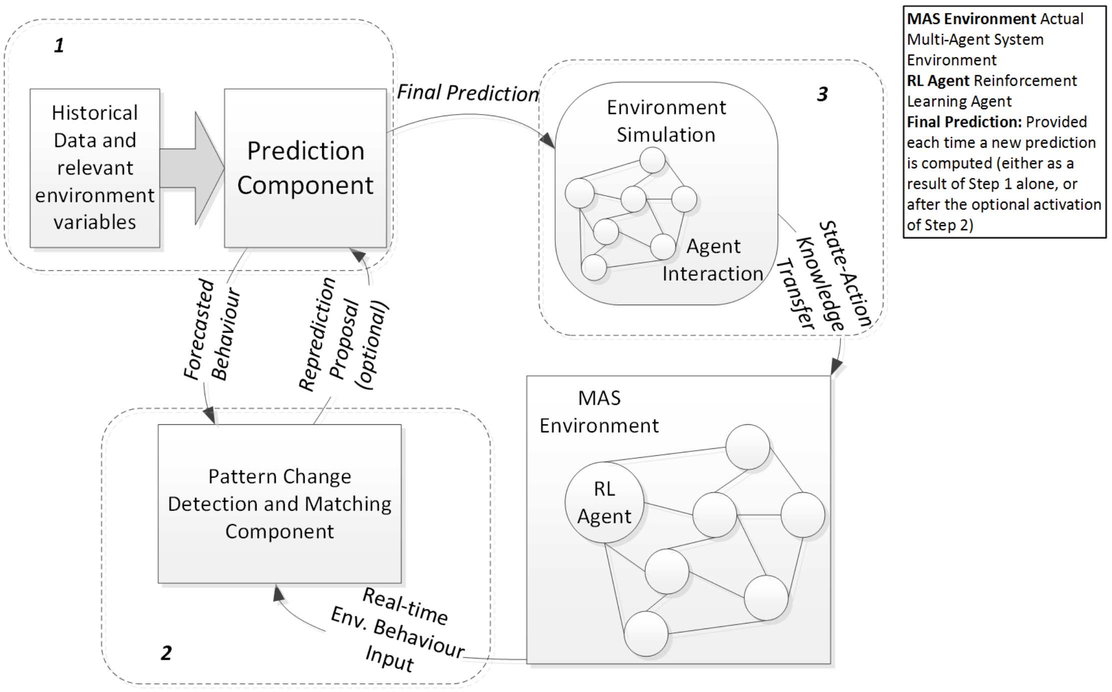

1
Historical Data and relevant environment variables
Prediction Component
Final Prediction
3
Environment Simulation
Agent Interaction
State-Action Knowledge Transfer
MAS Environment Actual Multi-Agent System Environment
RL Agent Reinforcement Learning Agent
Final Prediction: Provided each time a new prediction is computed (either as a result of Step 1 alone, or after the optional activation of Step 2)
Forecasted Behaviour
Reprediction Proposal (optional)
MAS Environment
RL Agent
Pattern Change Detection and Matching Component
Real-time Env. Behaviour Input
2

Fig. 3.2: MARL Algorithm Architecture

The relationship between these components is illustrated in Fig. 3.2. In the first component, the future environment's behaviour is predicted based on historical data and other influencing environment variables. Further details about this process are presented in Section 3.2.1. In the second component, this prediction is continuously evaluated against the real-time behaviour of the environment through a pattern-change detection mechanism. If a change from expected behaviour occurs, the type of change is matched and re-prediction is triggered using this updated information. The alternative, when a match is not found, is discussed later in this section. The second component is presented in Section 3.2.2. In the third component, the MARL system trains offline in a simulation of the environment, which is based on the prediction. This way agents acquire knowledge on how to address the future environment's behaviour offline (i.e., update their state-action table offline based on the prediction), and then apply this knowledge in real-time (i.e., the state-action knowledge acquired offline is applied on the actual environment). The processes occurring in this component are presented in Section 3.2.3.

Prediction is integrated in the MARL process to provide a forecast of the environment's future behaviour. In the first step, environment estimates are provided periodically, at set intervals. Each estimate is given for a predefined horizon (e.g., one minute ahead, one hour ahead, one day ahead etc.). This process is illustrated in Fig. 3.3. The estimate is provided as a discrete sequence of points, $(\hat{X}_t, \hat{X}_{t+1}, ..., \hat{X}_{t+h})$, where h is the prediction horizon and X_i is the state of the environment sampled at time i (e.g., for a day ahead horizon, with 24 sequence point estimates, there is one estimate for each hour of the day). This sequence is then used in the MARL process as a training base, to simulate the environment's future behaviour.

In parallel, once an estimate has been provided by the primary prediction component, its accuracy is continuously evaluated against the real-time behaviour of the environment. If the prediction accuracy falls under a certain threshold before the end of the

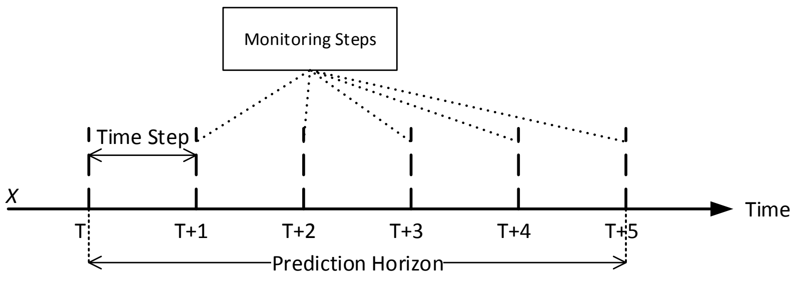


Fig. 3.3: Prediction Horizon With Discrete Time-Steps

predefined horizon, a new estimate needs to be computed for the remaining interval of the prediction horizon based on the latest observations. Once an updated estimate is provided, it is sent to the MARL component as input. If the estimate is sufficiently accurate during the entire horizon, the MARL component can continue performing online, without the need for re-training before the prediction for the next horizon is provided. In this case, current knowledge is considered to be sufficient.

The second prediction step, which triggers the reprediction of the environment behaviour when pattern changes are detected, is needed when encountering anomalous events that significantly influence the environment. These are situations for which agents were not prepared, where their actions can lead to suboptimal behaviour. The two steps are further described in the next sections, and are illustrated at a high level in Algorithm 1.

3.2.1 Prediction Component

The fundamental part of P-MARL is the prediction component. Providing a prediction of the environment's future behaviour is the **first contribution** of this thesis. The prediction component can function independently of the pattern-change detection and matching component. The prediction model considers recent historic values and key environment variables that are related to the environment's behaviour, in order to provide an estimate of future behaviour. To obtain good estimates, forecasting techniques can be combined to increase the accuracy of a prediction model (Clemen, 1989). In particular, there are two main categories of forecasting techniques which approach prediction differently. Linear models forecast only based on historical data about the time-series, while non-linear models employ additional information from correlated variables.

A prediction model should take advantage of both linear and non-linear prediction techniques' abilities. The most effective non-linear forecasting technique based on the

Algorithm 1: Prediction System Algorithm

```
1  currentState ← gatherEnvVars();
2  histBehaviour ← updateHistDatabase(currentState);
3  TSparams ← evaluateSeasonalityTrend(histBehaviour);
4  envPrediction ← predict(N time-steps,TSparams, histBehaviour);
5  simulateMARL(envPrediction);
6  while current time-step < N do
7      changeDetect ← evaluatePrediction(envPrediction, real-time behaviour);
8      if changeDetect > threshold then
9          currentState ← gatherEnvVars();
10         histBehaviour ← updateHistDatabase(currentState);
11         changeType ← matchChangeType(currentState, histBehaviour,
             envPrediction);
12         newEnvPrediction ← computeNewPrediction(changeType, histBehaviour,
             currentState);
13         simulateMARL(newEnvPrediction);
14     end
15 end
```

literature surveyed in Section 2.5 was artificial neural networks (ANNs), as they adjust to noisy time-series and make use of external variables in the forecasting process (Makridakis et al., 1998). However, when dealing exclusively with predicting future time-series behaviour based on past behaviour, auto-regressive integrated moving average (ARIMA) methods have been found very effective (Makridakis et al., 1998). As a result, the particular model designed in this thesis is a hybrid solution that combines both ANN and ARIMA methods. This model takes advantage of both techniques' strengths for time-

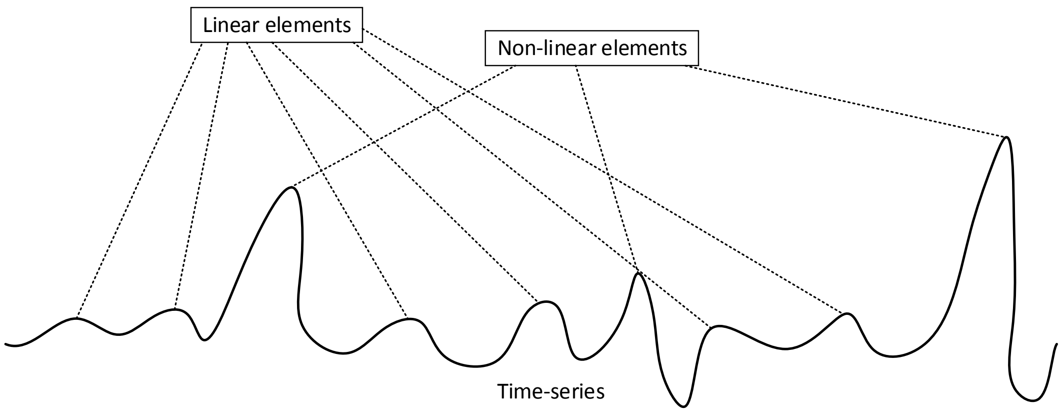


Fig. 3.4: A Time-series with Linear and Non-linear Elements

series prediction (Marinescu et al., 2014b), as one is more effective at specific subcomponents of time-series than the other, due to the linearity of the relations between some variables. This property is visualised in the example in Fig. 3.4. As a result, a combined solution is more accurate overall than individual methods when predicting the full time-series. For example, given an estimate with a horizon of 5 time-points ($\hat{X}_{t+1}, ..., \hat{X}_{t+5}$), ANNs could be better suited to predict elements $\hat{X}_{t+1}$ and $\hat{X}_{t+3}$, while ARIMA methods could be better suited to predict elements $\hat{X}_{t+2}$, $\hat{X}_{t+4}$ and $\hat{X}_{t+5}$. Best performance of sub-techniques needs to be investigated on the specific training dataset, as the hybrid solution is challenging to implement when aiming for very accurate estimates.

In the prediction algorithm, several techniques suitable for time-series prediction[1] are first evaluated, and then best performing methods over different subsections of the time-series are fed into a combined model. This evaluation process is presented in Algorithm 2. First, the available historical data is divided into two sets, a training set and a validation set (line 1). The different prediction techniques considered for the combined

[1]There needs to be a high level evaluation of the time-series with regard to trend, seasonality and correlation with other variables before deciding which methods are suitable for predicting the particular time-series.

Algorithm 2: predict(N time-steps, TSparams, histBehaviour)

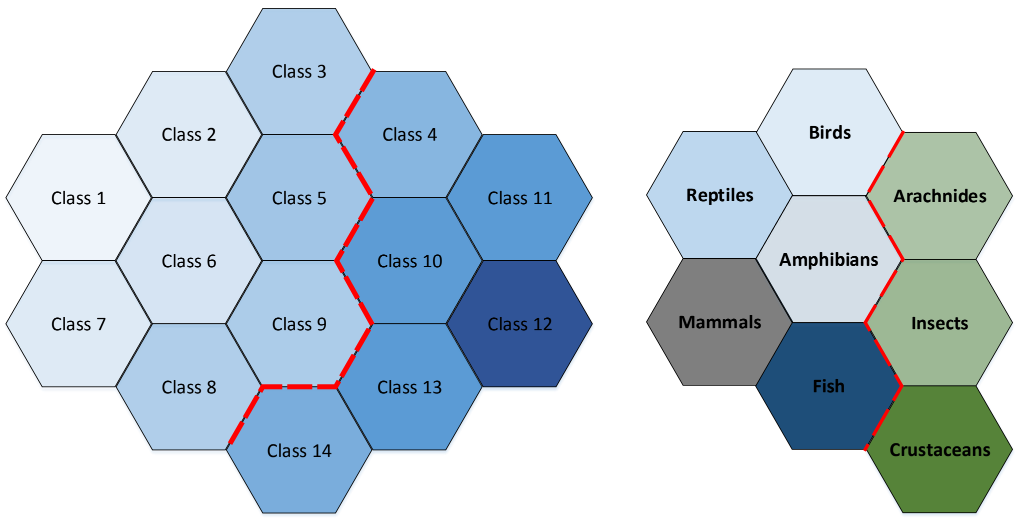
```
1  [trainingData, validationData] = divide(histBehaviour);
2  foreach predictionTechnique i in predictionSet do
3      prediction(i) ← implementPrediction(predictionTechnique, TSparams);
4  end
5  foreach predictionTechnique i in predictionSet do
6      predictedTS(i) ← computePrediction(trainingData, predictionTechnique(i));
7      accuracy(i) ← evaluatePrediction(predictedTS(i), validationDATA);
8  end
9  foreach time-step j in predictionHorizon do
10     bestMethod(time-step) ←
        selectBestPerformingMethod(accuracy,time-step(j));
11 end
12 foreach new prediction do
13     foreach predictionTechnique i in predictionSet do
14         predictedTS(i) ← computePrediction(trainingData,
            predictionTechnique(i));
15     end
16     foreach time-step j in predictionHorizon do
17         aggregatedPrediction(j) ← predictedTS(bestMethod(j));
18     end
19 end
20 finalPrediction ← aggregatedPrediction;
```

method are implemented and trained on the training set, and their accuracy is evaluated on the validation set. The method performing best for each particular time-step over most of the predicted sequences (i.e., prediction horizons) is selected to provide future predictions over the next prediction horizons for that particular time-step (line 10). The combined prediction for future horizons is afterwards computed (line 12). As time passes, time-series information in the training and validation datasets is updated, and the process presented in Algorithm 2 is repeated to update the hybrid solution.

While ARIMA provides additional metrics about forecast points such as prediction intervals, this is not valid for ANNs. ANNs only provide forecast results in batches, without individual information about single forecast points. In a model combining both techniques, only some of the forecast time-series points would benefit of these additional metrics. As a result, aggregated forecast metrics for a hybrid solution cannot be provided anymore.

3.2.2 Pattern Change Detection and Matching Component

This component is used to detect and react to situations where the prediction model fails to provide accurate estimates of the future behaviour of the environment. Detecting changes from expected values in the environment's future behaviour represents the **second contribution** of this thesis. Several techniques for pattern-change detection and matching were presented in Section 2.5.5. Some techniques used for pattern-change detection and the techniques used for pattern matching share a common ground. Specifically, cluster analysis relies on classification techniques to detect changes in behaviour. Self-organising maps (SOMs) can be used to classify time-series segments through on an unsupervised learning process based on ANNs (Kohonen, 1990). This process maps a high-dimensional space (i.e, the time-series segment used as input) into a low-dimensional space (i.e., the classes/clusters). The time-series segments are time sequences of the same number of time-steps, and are fragments of the overall behaviour of the environment, divided based on some periodic criteria (e.g., seasonality or prediction horizon). Examples of SOM usage include daily temperature classification, weekly sales, or the price of mutual funds and stocks (Fu, 2011).



(a) Self-organising Map Classes Distribution (b) Self-organising Map Example

Fig. 3.5: Self-organising Maps

SOMs are trained through unsupervised learning, where they go through a process of self-organisation (thus the name). Throughout the training stage, time-series segments sampled from past environment behaviour get clustered based on similarity, and cluster characteristics (i.e., weight vectors) are being defined for each class. After the training stage, when a new time-series segment is recorded from the environment, this can be classified based on its properties into one of the already configured clusters[2]. As such, SOMs can also be used for pattern-change detection, by detecting in which class a new time-series segment is classified when compared to the previous one. If the new segment is classified into a different class than the previous segment, a pattern-change has occurred.

[2]These properties (i.e., weights) are computed by the self-organising map, and the class with the closest weight to the sample is chosen as the matching class.

Recall from Section 2.5.5 that a SOM has a number of classes available for classification, which are decided at the implementation stage. These depend on the approximate number of different types of samples[3]. Once the training process is finished, each new sample fed into the SOM is assigned into one of the classes. A self-organising map additionally shows the closeness between different neighbouring classes. This is illustrated in Fig. 3.5a, where similar classes are pictured with close nuances. In the particular example from Fig. 3.5a, it can be noticed that the classes can be separated into subgroups, for example one in the upper left side and another one in the lower right side of the SOM. An example of a SOM is pictured in Fig. 3.5b, where animals are classified based on their similarity, and where it can be seen that there are particular groups within the SOM.

Since there can be many different classes within a SOM, a design choice was to group these into two main categories: normal classes, encompassing most of the samples, and anomalous classes, comprising particular samples that are outside of the ordinary. The classes are arranged in the SOM based on the closeness between them. Normal classes share more properties between each other than anomalous classes, and are placed geographically close. Normal classes encompass most of the samples from the database, while anomalous classes comparatively contain only a small amount. This information helps delimit the normal group from the anomalous group. Once the map training process is finished, real-time environment behaviour can be fed into the SOM, and will be classified in one of the two main categories.

The predicted horizon $(\hat{X}_t, \hat{X}_{t+1}, ..., \hat{X}_{t+h})$ provided by the primary prediction component is used to detect anomalies. The initial assumption is that, for this horizon, the environment behaviour will not be anomalous, therefore the predicted horizon ends up classified into one of the normal classes after being evaluated by the SOM. As the environ-

[3]The number of classes is defined by the user.

ment evolves from time t to a future time $t+i$, where $i < h$, the recently acquired actual information about the environment is overlapped over the predicted horizon. Therefore, the sequence $(\hat{X}_t, \hat{X}_{t+1}, ..., \hat{X}_{t+h})$ becomes $(X_t, X_{t+1}, ..., X_{t+i}, \hat{X}_{t+i+1}, ..., \hat{X}_{t+h})$. At each time-step $t+i$, the newly obtained sequence (comprising both real and predicted values) is inserted into the SOM. If the sequence ends up classified into an anomalous class before the actual time-line passes the prediction horizon, an anomaly is detected. This detection triggers further adjustments to the prediction mechanism, as employing only historic data can lead to inaccurate predictions in anomalous cases. The mechanism for this procedure is based on preliminary studies performed for this thesis (Marinescu et al., 2014a).

A change is detected when a time sequence is classified into an anomalous class instead of a normal class. This is then compared with similar sequences from the corresponding anomalous class. Once a close match is found (based on the smallest distance between the two vectors), it is used to replace the remaining time-steps in the predicted sequence starting from the current time $t+i$. These remaining time-steps are adjusted to take into account past environment variables encountered when the last such anomaly occurred, and also the value of these variables in the current/future environment state. Predicting the type of change from expected behaviour in the environment represents the **third contribution** of this thesis.

To summarise, when an anomaly is detected, the closest previously encountered such sample is used from the fitting class of anomalies in order to increase reprediction accuracy. This triggers a new estimate which is particular to anomalous cases.

3.2.3 The Multi-Agent System

P-MARL uses techniques from reinforcement learning and time-series forecasting. To further explain the concepts behind P-MARL, the RL notation introduced in Section

2.2.1 is expanded on to include non-stationary concepts. Recall the Q-Learning equation presented in Section 2.2.1:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\Big] \tag{3.1}$$

This equation is based on the tuple $(s_t, a_t, r_{t+1}, s_{t+1}, a)$. In Q-Learning, once an agent moves towards the exploitation stage, it attempts to maximise its overall reward by choosing the most long-term rewarding action from state s_t. Once sufficient exploration is performed, action a_t is considered to be the optimal action from this state. In the case of stationary environments, s_{t+1} is a function of the action taken at time t:

$$s_{t+1} = F(a_t) \tag{3.2}$$

In a non-stationary environment, where a change has just occurred, an agent takes action a_t in state s_t, expecting to maximise its long term reward by moving into state s_{t+1}. Since the environment is non-stationary, it influences the result of action a_t, and the new state reached is instead s'_{t+1}. P-MARL considers that s'_{t+1} is a function of both the agent's action and the environment's independent behaviour:

$$s'_{t+1} = F(a_t, X_{t+1}) \tag{3.3}$$

Thus, value X_{t+1} affects the outcome s'_{t+1} together with action a_t. In stationary environments, X_{t+1} can be learned and integrated in the rewarding scheme, so Eq. 3.3 can be reduced to Eq. 3.2. However, when the environment changes, the previously integrated value X_{t+1} can be very different from the actual state of the environment. In this case, action a_t can lead towards a suboptimal state with a different reward than expected, so Q-values need to be readjusted. In such situations, further exploration is needed for the Q-Values to be re-learned, until an optimal set of actions is found again.

The environment's evolution needs to be taken into account separately, to improve the agent's performance. This can be further explained through Eq. 3.4:

$$X_{t+1} = \bar{X}_{t+1} + \varepsilon_{t+1} \tag{3.4}$$

where $\bar{X}_{t+1}$ is the already integrated value in Q-Learning, and ε_{t+1} is the difference between the integrated value of the environment and the one actually occurring due to the environment's independent evolution. From Eq. 3.3 and Eq. 3.4, it can be deduced that:

$$s'_{t+1} = F(a_t, \bar{X}_{t+1} + \varepsilon_{t+1}) \tag{3.5}$$

However, since ε_{t+1} is a constant, this can be written as:

$$s'_{t+1} = F(a_t, \bar{X}_{t+1})) + \varepsilon_{t+1} \tag{3.6}$$

If $|\varepsilon_{t+1}| << |\bar{X}_{t+1}|$, the Q-values obtained by employing $\bar{X}_{t+1}$ should not affect the choice of actions as the state reached $s'_{t+1} = s_{t+1}$, but this assumption does not hold for non-stationary environments, where ε_{t+1} can be large. This fact is key to the proposed approach, which intends to provide an estimate $\hat{X}_{t+1}$ to closely match X_{t+1}, the future environment's behaviour. This estimate can be used to train an RL agent offline, within a simulation of the environment based on the estimate.

This work proposes to employ advanced forecasting techniques for non-stationary environments, where a close estimate $\hat{X}_{t+1}$ can be predicted, so that $\hat{X}_{t+1} \approx X_{t+1}$.

The particular application environment addressed in this thesis is inherently non-stationary. This means that the underlying generating function of the environment changes over time. The problem can be simplified if the outcome of the generating function of the environment is modelled based on recently observed behaviour. At every time step t, there is historic data available:

$$X = (X_1, X_2, ..., X_t)$$

where X_i is the state of the environment X at time i. X_{t+1} is predicted based on past observations and other variables related to the environment. This is a difficult task, since a non-stationary environment does not present only fully repeatable patterns. Moreover, the uncertainty present in the environment needs to be accounted for separately, as anomalous events in the environment can lead to unexpected changes. The accuracy of predictions is critical in the training process of agents, as inaccurate predictions can make the agents take sub-optimal actions.

Once an estimate of the environment is provided, this is used to create a simulation of the environment. Agents train offline within this estimate-based simulation. The offline training session is performed when agents do not need to change their actions on the environment, as during this time their previously taken action is still valid in online mode. This process is illustrated in Fig. 3.6. Adjusting to environment changes offline, without affecting the actual environment, represents the **fourth contribution** of this thesis. Throughout the training process, the simulated behaviour of the environment is repeated over several episodes, until the agents reach an agreement for their combined sets of actions and are considered ready to act online[4]. The agreement is reached once every agent repeats the same actions in the current episode as it had in the previous episode, thus meaning that they are content with their actions. This consensus is reached between agents offline, thus reducing agent-contributed non-stationarity in the actual environment, and represents the **fifth contribution** of this thesis. This process is described in line 6 and line 13 of Algorithm 3.

The assumption of this approach is that the environment's states occurring in real-

[4]However, if no consensus can be reached between agents, the simulation stops after a set number of episodes.

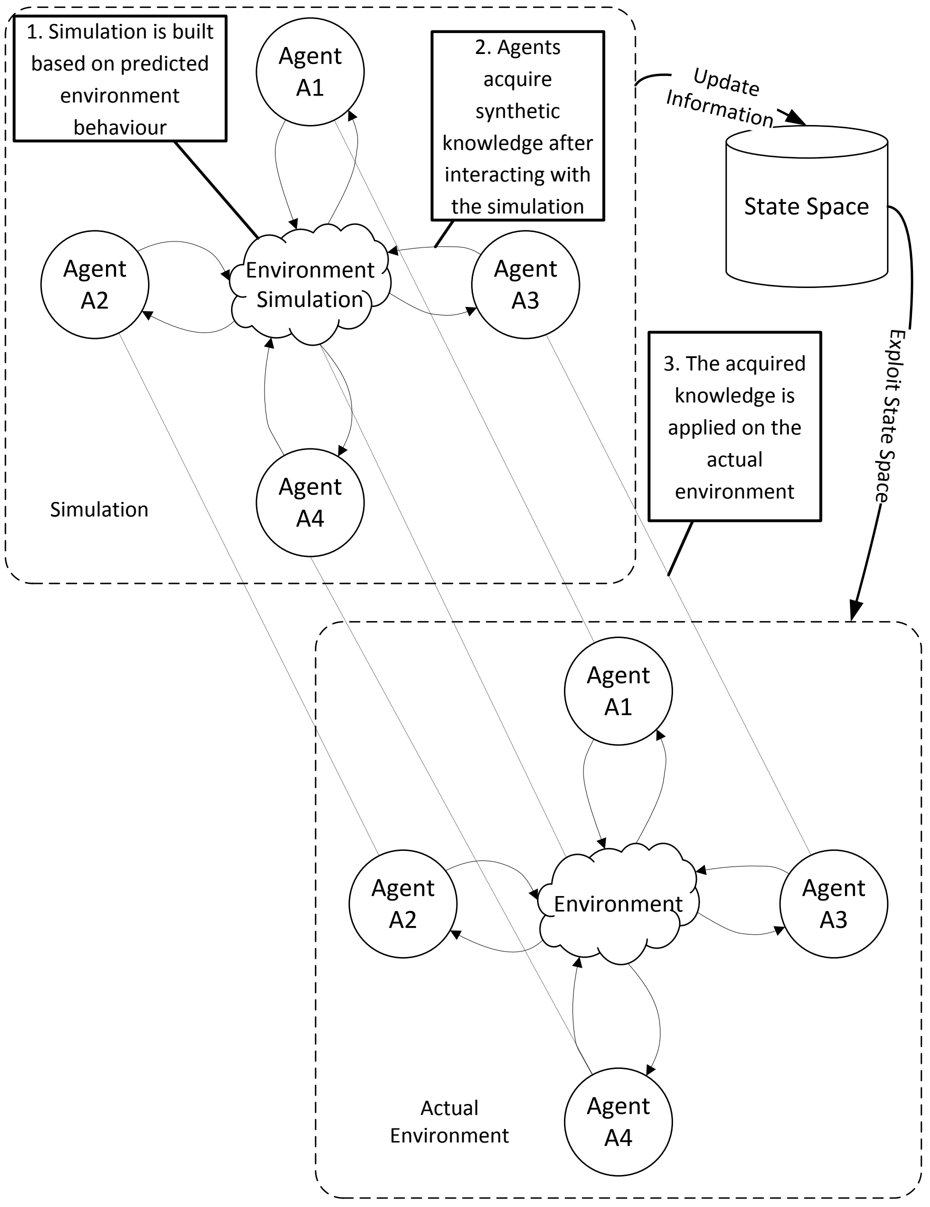

1. Simulation is built based on predicted environment behaviour
2. Agents acquire synthetic knowledge after interacting with the simulation
3. The acquired knowledge is applied on the actual environment
Agent A1
Agent A2
Agent A3
Agent A4
Environment Simulation
Simulation
Update Information
State Space
Exploit State Space
Agent A1
Agent A2
Agent A3
Agent A4
Environment
Actual Environment

Fig. 3.6: MARL Adaptation Concept for Non-stationary Environments

Algorithm 3: P-MARL Algorithm

```
1  foreach new prediction horizon do
2      envVars ← updateEnvData();
3      histData ← updateHistoricBehaviour();
4      prediction ← makeInitialPrediction(envVars,histData);
5      evaluatePrediction(prediction,currentEnvBehaviour);
6      marlKnowledge = learnBestBehaviour(finalPrediction);
7      while timeStep < predictionHorizonEnd do
8          if no significant change detected then
9              finalPrediction ← prediction
10         else
11             anomalyType ← matchChangeType();
12             finalPrediction ← repredict(anomalyType,envVars,histData);
13             marlKnowledge = reLearnBestBehaviour(finalPrediction);
14         end
15         exploit(marlKnowledge, actualEnv);
16     end
17 end
```

time will be mapped by the RL agents in their state-space in the same way as the states of the simulated environment[5] were mapped offline. This enables agents to follow a similar policy (i.e, take the same set of actions) as in the last episode of the simulation. This assumptions holds if $\hat{X}_i$ is sufficiently close to X_i for the agent to reach s_i after taking action a_{i-1} (based on its pre-computed policy), instead of reaching a different state s_j (where $s_j \neq s_i$).

[5]This simulated environment is based on the predicted behaviour.

The relationship between the components was illustrated in Fig. 3.2. An initial prediction, which is based on historic data and current environment evolution, is supplied by the prediction component to the pattern change detection component, which decides on the final estimate to be used for MARL training. More details of this process are presented in Algorithm 3. A final estimate of the environment's future expected behaviour is provided by the prediction system. The MAS agents evaluate their performance based on the expected future behaviour and configure their state-space values. A process of exploration-exploitation is performed by the agents based on the provided environment estimate. This helps them achieve a best-response function to the expected future behaviour of the environment. Once agents' solutions converge, they are ready to operate in online mode. Even though the actual environment they will face will be somewhat different from the estimate, the hypothesis is that the previously obtained knowledge will help them perform well in similar conditions to the ones provided by the estimate. In the case of Q-Learning, this translates to obtaining Q-values that allow agents to maintain the same policy in between the estimate and the actual environment.

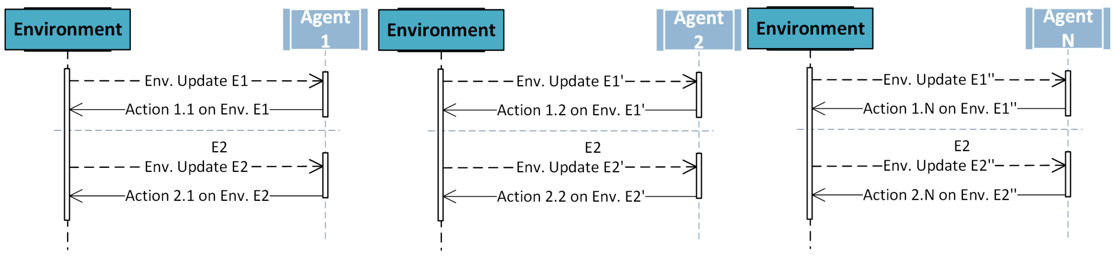
Once each agent receives a prediction of the environment behaviour, several types of learning can occur during the exploration stage of agents, depending on the communication restrictions imposed on the system. P-MARL implements three different kinds of learning, between which the algorithm can switch as a function of the restrictions imposed:

- single agents acting separately on the environment;
- multiple agents acting simultaneously on the environment;
- multiple agents acting sequentially on the environment.

These are further described in detail in the following sections.

3.2.3.1 Single Agents Acting Separately on the Environment

Fig. 3.7: Single Agents Acting Separately

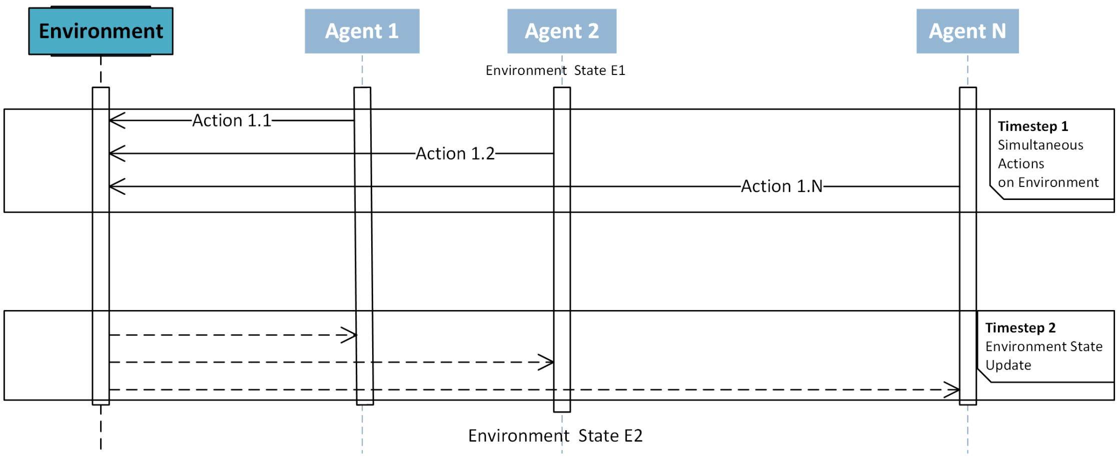


In this case, during exploration, each agent reaches a policy solely based on the effects occurring on the environment due to its own actions. Once an agent has taken an action, it receives the updated state of the environment, which results exclusively based on its own action and the environment's independent evolution. An agent cannot see any of the effects of other agents on the environment. While the environment is updated only locally, the final state of the environment will aggregate all actions taken, but this aggregated state is not accessible to agents (e.g., several slot-machine players attempting to win money from a casino). This is the case with most severe communication restrictions. In fact, it can be summarised as a single agent problem applied to multiple agents. This type of interaction is shown in Fig. 3.7.

3.2.3.2 Multiple Agents Acting Simultaneously on the Environment

In this case, during exploration, an agent's action on the environment is taken into account together with the cumulative effect of all agents' actions. The agent can see this cumulative effect only after it has chosen an action, therefore agents choose actions simultaneously. Once an agent has taken an action, it can only see the aggregated effect of all agents' actions on the environment when it receives the update. This case assumes

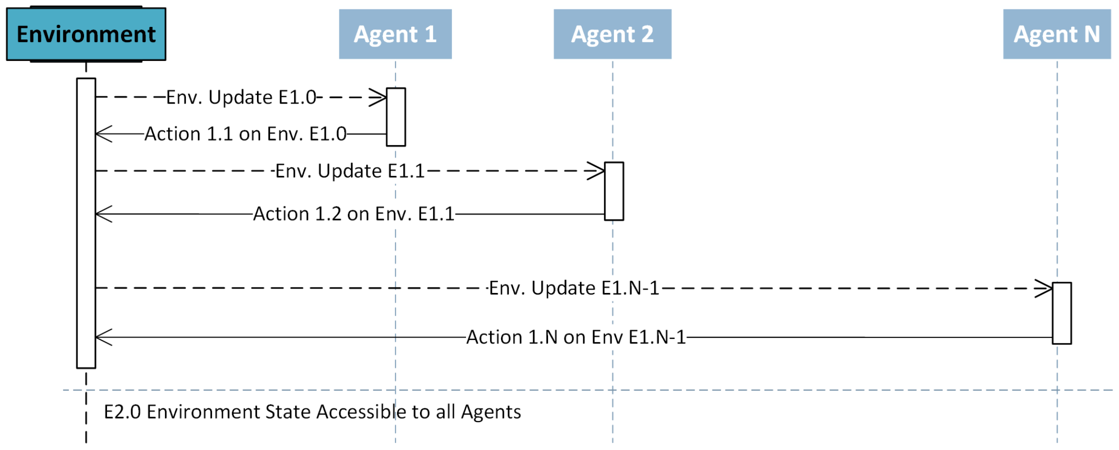
Fig. 3.8: Multiple Agents Acting Simultaneously



that there is only post-communication towards the agents, coming from an environment update once all agents have taken action (e.g., the stock market, where brokers take actions simultaneously). This type of interaction is shown in Fig. 3.8.

3.2.3.3 Multiple Agents Acting Sequentially on the Environment

Fig. 3.9: Multi-Agent Acting Sequentially



In this case, during exploration, agents take decisions at each time-step in a sequential manner. An agent will see the cumulative result of the previous agents' actions on the environment, and then decide which action to take. Once it has taken a decision, the updated cumulative effect on the environment is passed on to the following agent. The order of this sequence is random at each of the environment's behaviour time-step[6], therefore no agent is favoured during the length of a learning episode. This case assumes that agents take actions in sequence, and each agent whose turn it is to take an action can see the cumulative effect of previous agents before making its decision (e.g., an auction, where only the current bid matters and not previous ones). This type of interaction is shown in Fig. 3.9.

3.2.3.4 Discussion

The implementation for these three cases is highly dependent on the level of interaction between agents and environment. It is worth noting that the case presented in Section 3.2.3.3 is a form of indirect communication between agents, as an agent's decision is communicated through the environment, which takes the role of broadcaster. This form of interaction requires an underlying protocol to maintain order between agents' actions and thus avoid overlapping decisions.

3.3 Summary and Analysis

This chapter first presented the design requirements for a MARL algorithm that operates in inherently non-stationary environments. To meet the design requirements, it then presented P-MARL, which integrates prediction and pattern-change detection and matching techniques in the learning process of a MARL algorithm.

[6]However, within each of the environment's time-steps, agents take sub-steps, and their actions are aggregated at the end.

Time-series analysis techniques were employed to model the environment's future behaviour, and also to monitor changes that occur from expected behaviour. Once a prediction of the environment's behaviour is provided, this is used to simulate the environment during the exploration stage of agents, offline. Three possible MARL exploration cases, in which exploration is based on the prediction provided, were identified. These depend on the level of interaction between agents and the environment, specifically: single agents acting separately on the environment; multiple agents acting simultaneously on the environment; and multiple agents acting sequentially on the environment. When agents discover suitable actions to achieve their goals, given the expected future environment's behaviour, they start acting online, on the actual environment. This way the agents' negative impact on the environment while adapting to new situations is minimized.

The three main components could have employed other alternative techniques. In the case of the prediction component, while there were many forecasting techniques to choose from, it was considered that one non-linear (ANNs) and one linear (ARIMA) forecasting technique suffice. The preliminary analysis for this thesis revealed these two methods to achieve best results in their specific categories: peak estimation and overall accuracy, respectively. In the case of the pattern-change detection and matching component, change detection could have been performed through a sliding window mechanism that detects changes based on the aggregated deviations of the actual environment's behaviour from the expected behaviour. However, if these changes were only slowly occurring, the sliding window mechanism could fail to detect changes over a long horizon. In the case of the multi-agent system component, model-based RL techniques were also considered for the agents. The problem with these however was that they adapt slower to dynamic environment, as they also have to adjusts their internal models of the environment while learning to address changes.

This approach is based on the ability to forecast future environment behaviour.

However, one limitation is that P-MARL only addresses environments that are not fully stochastic, where the environment's behaviour lends itself to prediction. If the non-stationarity induced by the environment is completely random (i.e., without any relation to past behaviour) and has no correlation with external variables (e.g., drawing a card from a deck with replacement), this approach is not applicable anymore.

The following chapter presents in detail how decentralised control is implemented through P-MARL in an inherently non-stationary environment that can be modelled through time-series techniques.

Chapter 4

Implementation

This chapter presents the implementation of P-MARL. Specifically, the three main components, *prediction*, *pattern-change detection* and the *multi-agent system* are implemented with respect to the scenarios employed in this thesis.

P-MARL is written in C++, and is based on an implementation of DWL (Dusparic and Cahill, 2012) written by Adam Taylor[1]. For the smart grid scenarios, the P-MARL architecture is integrated into GridLAB-D (Chassin et al., 2008). GridLAB-D is an open source power network simulator, also written in C++. P-MARL is integrated into a modified version of the electric vehicle class of GridLAB-D, to include agent-based control and prediction functionalities. The scenarios in which P-MARL is applied involve the optimization of energy demand management. The environment is characterised by electrical energy demand at small scale level, mainly residential neighbourhoods and business parks.

The class diagram of P-MARL is pictured in Figure 4.1. A P-MARL agent `p-marl_agent` learns an optimal policy `optimal_policy` based on the information provided by the prediction and pattern-change detection classes, `predict_env` and `change_de-`

[1]https://github.com/AdamLukeTaylor/AMAAS-master/tree/master/DWL

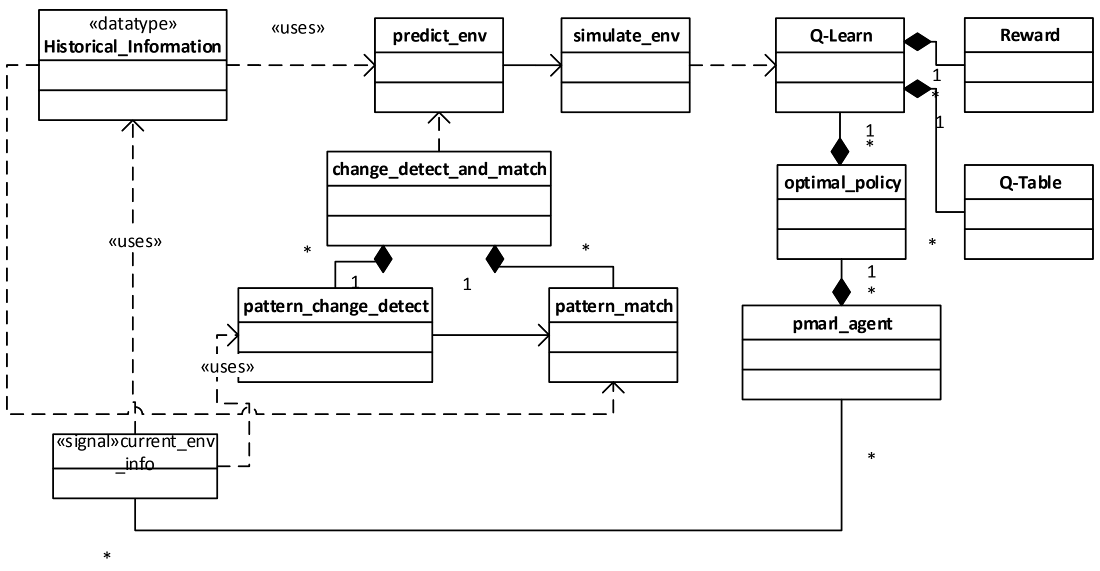


Fig. 4.1: Class Diagram of P-MARL and the RL process

`tect_and_match`, respectively. An initial prediction of the future environment's behaviour is provided based on historical information, from the database `Historical_Information` by the prediction component's class, `predict_env`. This prediction is continuously evaluated by the pattern-change detection and matching component's class, `change_detect_and_match`. The latter class has access to historical information from the database, and current environment status from the `current_env_info` signal, and can detect changes and eventually estimate the type of changes occurring. If changes are detected, a new prediction of the future environment's behaviour needs to be provided by the prediction class `predict_env` while taking into account the change type provided by the `change_detect_and_match` class.

The prediction is used in the environment simulation class, `simulate_env`, where the future environment's behaviour is simulated to enable offline agent learning. An agent explores the effect of its actions in the simulated environment through trial-and-error through the `Q-Learn` class, and the suitable actions to address the current environment's behaviour are learned and provided to the `optimal_policy` class based on the `Q-Table` data. Once the P-MARL agent `pmarl_agent` has obtained the optimal policy based on the simulation of the environment, it exploits this knowledge on the actual environment. The time sequence of this process is summarised in the sequence diagram from Figure 4.2. This represents a high level view of P-MARL.

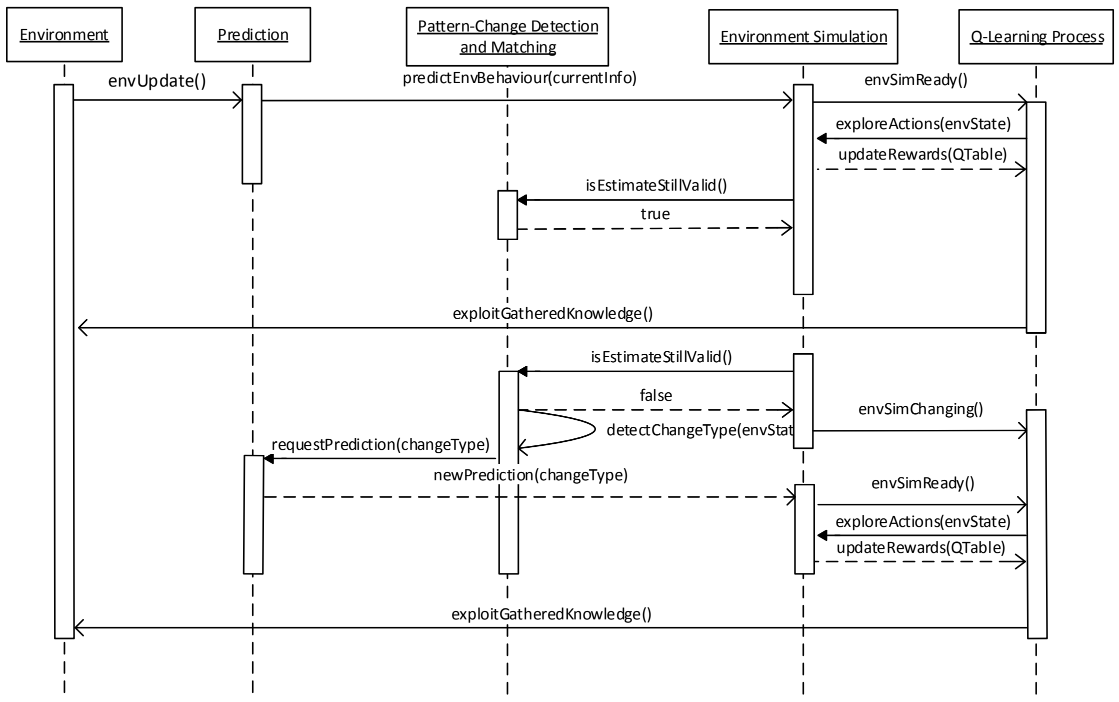


Fig. 4.2: Sequence Diagram: P-MARL High Level View

4.1 Application in Smart Grids

P-MARL is an approach designed for inherently non-stationary environments. Smart grids are an application domain which have non-stationary characteristics. This research was performed within a team who have been looking at the application of MAS to smart grids for some time, and control optimization benefits if the control solution meets the requirements presented in Section 1.3.2. As presented in Section 2.7.2, multi-agent systems (MAS) have many applications in the control of smart grids. However, while many MAS solutions detect pattern-changes and can adjust to known patterns, these do not address upcoming changes in the environment's behaviour when this is inherently non-stationary, when the environment is continuously evolving.

One of the main requirements for smart grids is *robustness*, as it provides services (i.e., energy) to critical systems such as hospitals or air traffic control centres. To be resilient in the face of perturbations (e.g., black outs, power generator failures, or particular weather phenomena), smart grids need to be able to self-heal and have elements that can replace each other. Agent-based control is a form of *distributed control*, which fits this requirement of the smart grids, as this way subcomponents of the smart grid can function autonomously. *Agent interaction* is facilitated by the underlying communication protocol of P-MARL, where agents self-organise and reach together a consensus for the next set of actions. Finally, agents benefit from prediction and pattern-change detection and matching abilities, which enable them to *address inherently non-stationary environments.* Therefore the smart grid is suitable as the evaluation domain for P-MARL.

The smart grid's main purpose is to provide energy to customers. The amount of energy produced and delivered in these power networks is determined by customer demand. For consumers, the aggregated energy demand from everyday household appliances is known as the *baseload.* This baseload demand fluctuates during the day, switching between periods of low demand and periods of high demand. In times of high demand, more generators need to be scheduled, which increases the cost of energy. Demand side management techniques attempt to avoid peak energy usage by scheduling deferrable loads to times when the baseload has the lowest energy usage. As opposed to fixed loads, which encompass most household appliances, deferrable loads can be controlled through agent-based systems, and thus can be scheduled in a decentralised manner to achieve their purposes while avoiding peak-time energy usage.

Among the devices drawing power from households, electric vehicles are by far the most power hungry (Druckman and Jackson, 2008; Information, 2014). These represent a very good case for agent-based demand management control. Therefore, in this work, all other appliances are regarded as being part of the baseload, and the focus is on EVs as deferrable loads. EVs are further modelled as agents. Two scenarios are devised to evaluate P-MARL. These scenarios involve:

1. Non-stationary power demands

2. Non-stationary solar energy supply

In the first scenario, the main purpose of an EV agent is to obtain a sufficient charge for its next day's strip. Considering a neighbourhood of residential users, many of whom own EVs, controlling the charging process for EVs becomes a multi-agent problem, as the demand should also be evenly spread between all EVs, to match the available energy provided by the smart grid. This is the basis of the main scenario chosen to implement and evaluate P-MARL. In this scenario, the baseload that defines the environment's behaviour has non-stationary characteristics that need to be accommodated for.

In the second scenario, the purpose of EV agents is to efficiently use the solar energy provided by a set of solar panels in a business park. The demand of a group of small-medium enterprises (SMEs) from a CER trial (Comission for Energy Regulation, 2011) is used to represent the stationary demand of the business park. In this secondary scenario the business park uses a set of solar panels, and the energy generated by these has non-stationary characteristics that need to be accommodated for. The solar energy produced defines the environment's behaviour.

In short, for both scenarios, flexible energy consumers such as electric vehicles should:

- train offline based on predicted demand (in between charging decisions, which occur every 15 minutes when a dynamic price change scheme is in place). This way, at every time-step when dynamic pricing updates occur and demand changes, an EV should take optimal charging decisions based on its previously acquired knowledge (*R1: Minimize online learning*);
- notice when (and what kind of) changes in demand are occurring and adjust their actions accordingly (*R2a: Detect Sudden Changes*);
- use forecasting techniques to estimate future energy demand after changes occur (*R2b: Estimate Change Type*);
- prepare suitable anomaly mitigation charging strategies (*R3: Prepare for Changes*).

4.2 Residential Neighborhood Demand Response Scenario

In the residential neighbourhood scenario (i.e., the first scenario), the personal objective for each EV agent is to achieve a desired state of charge (SOC) in order to fulfil the next day's trip. Additionally, this charging process might be constrained by periods of high demand, when electricity is expensive, and when charging is to be avoided. Such periods can change in real-time, i.e., when all EVs charge simultaneously during periods of moderate energy usage, resulting in very high energy usage. The latter is not a desirable state of the environment, and should be avoided. As a result, the policy followed by the EV agents is to evenly spread their demand during the charging period, without generating peaks in demand.

To simulate such a situation, a real-world inspired problem is proposed: a residential neighbourhood where a community of households are supplied by a single transformer.

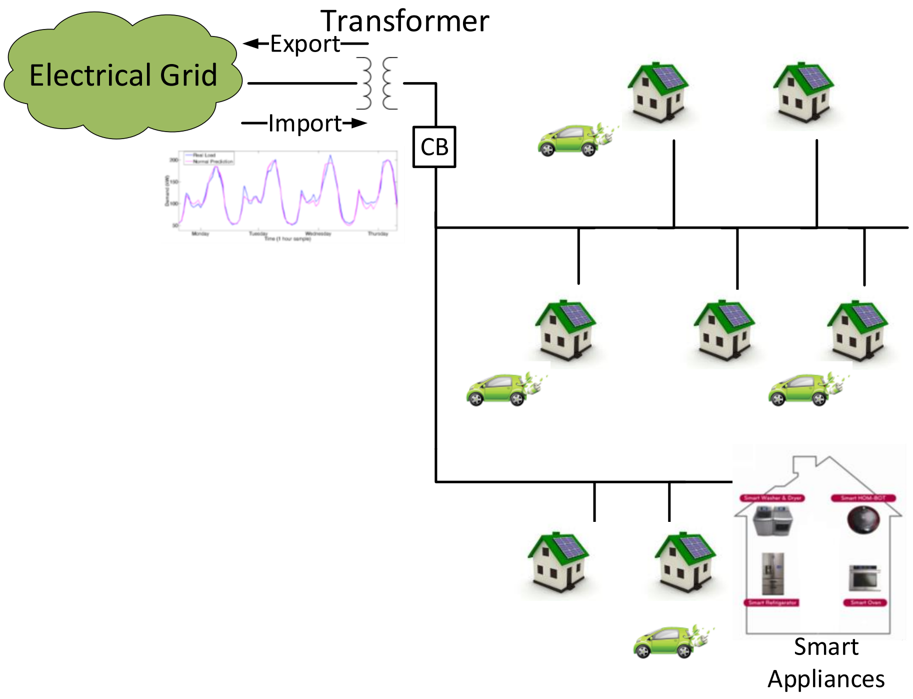


Fig. 4.3: Residential Neighbourhood Scenario

EV agents interact through the transformer, which informs them of the effect of their actions at every time-step. A time-step takes place every 15 minutes, when dynamic price changes occur. Once connected to the grid, agents attempt to achieve DSM at the residential community's level, based on the information obtained from the transformer. The general structure of such a community is presented in Fig. 4.3. Energy prices are considered to be proportional to the demand, therefore increasing with the demand. DSM techniques attempt to smooth the demand curve in order to reduce overall costs.

To meet these requirements, P-MARL's three step solution is implemented as described in the following sections. Energy demand is forecasted by the **primary prediction component**, whose implementation which is further described in Section 4.2.1. Any changes and anomalies from predicted demand are detected and matched by the

pattern-change detection and matching component, which is described in Section 4.2.2. Finally, all the knowledge about future demand and potential anomalies occurring in the grid are integrated in the offline training process of the **multi-agent system**, which is in charge of optimizing the demand of the group of EVs. The agents are severely penalised if they charge during peak time, to avoid their demand reaching the transformer limits. This training process is described in Section 4.2.3. Additionally, Section 4.4 proposes an evaluation benchmark for P-MARL, an optimal centralised solution.

The solutions are developed for a microgrid scenario, where real-world data was employed: a smart-meter trial which recorded half-hourly energy demand from individual households. The trial was conducted by CER[2] in Ireland for 17 months, between July 2009 and January 2011. Since weekdays follow a significantly different pattern than weekends by having a higher aggregated demand, and comprise a larger proportion of days in a week thus providing more samples, these were exclusively selected for the microgrid scenario.

4.2.1 Energy Demand Forecasting Component Implementation

In the residential neighbourhood scenario (i.e., first scenario), all electric vehicles arrive home in the evening and depart in the morning. Each EV agent would like to know the periods of low demand occurring while it is home, in order to be as cost effective as possible. In a non-stationary environment, though, such *a priori* knowledge is not available. Periods of low demand can change from one day to another. Furthermore, if all EV agents chose the same period for charging, based on the assumption that it has low demand, this results in high demand.

An initial analysis of the environment's evolution needs to be performed before the

[2]Commission for Energy Regulation

primary prediction component is configured. This analysis looks for any seasonality and trend patterns, and influencing environment variables. The defining characteristic of the environment is examined in relation to other environment variables. If the defining environment characteristic is found to be dependent on other variables (e.g., rain dependent on clouds), these variables should also be employed in the prediction component. Furthermore, the time-series representation of the environment evolution could present seasonal patterns and trend lines; these type of properties need to be addressed when customising the prediction model.

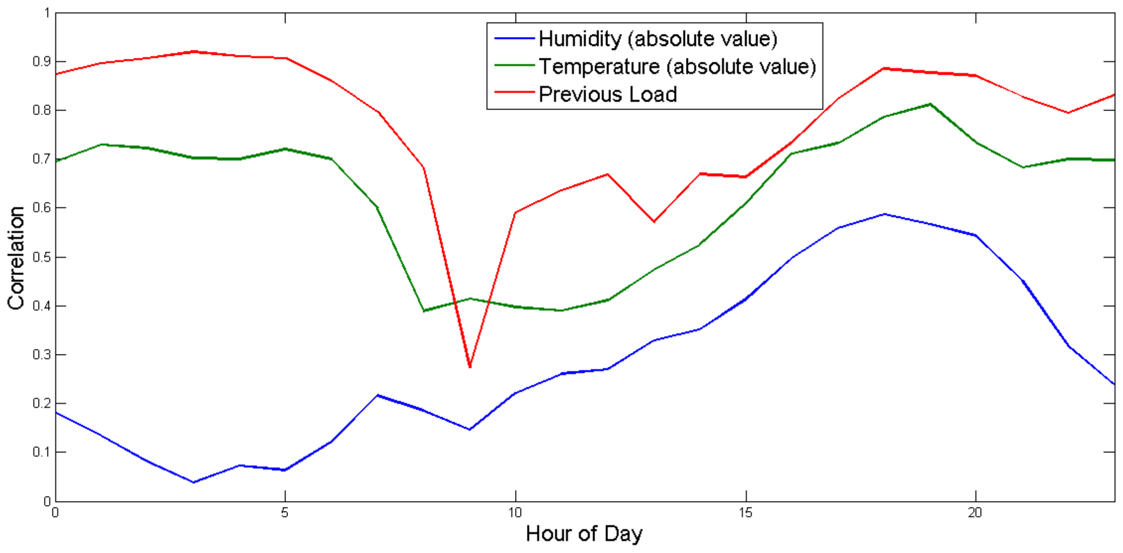


Fig. 4.4: Variables correlation

A correlation test was conducted to evaluate the relationship between the current load, the load of the previous day, the temperature of the current day, and the humidity of the current day. As illustrated in Fig. 4.4, there is a high correlation between the current load and the previous load and forecasted temperatures. Humidity is also a significant influence, which is more obvious in the second half of the day, when more people tend to be home.

Techniques such as short-term load forecasting (STLF) deal with power demand estimates, which can give good estimates of expected demand. For this problem, the most appropriate such STLF sub-technique is the one focusing on day-ahead demand forecasting, providing an estimate every 24 hours. This is because energy demand is seasonal at day level. The best methods for such forecasts rely not only on historic values of previous power demands, but also on other data such as weather variables (temperature, humidity), day of the week and public holidays (Gross and Galiana, 1987). While weekdays and weekends differ significantly in terms of demand patterns, even each weekday poses individual characteristics. Anomalous days (from the demand's perspective) also occur, mostly because of public holidays, but also because of other unexpected reasons (e.g., particular weather phenomena). These anomalous days need to be accounted for separately.

To improve load forecasting accuracy, besides employing historical load data[3], additional information about temperature and humidity was selected for the forecasting models in this thesis[4]. The performance of five forecasting methods was initially evaluated. These have been successfully used for forecasting on large scale. Particular focus was placed on Artificial Neural Networks (ANN), since they have proven very reliable in non-linear and non-stationary system predictions (Hippert et al., 2001). A set of three closely-related statistics based linear prediction methods, auto-regressive (AR), auto-regressive moving average (ARMA), and auto-regressive integrated moving average (ARIMA) were selected for comparison purposes. Furthermore, another method employed for electric load forecasting based on time-series decomposition and ANNs, wavelet neural networks (WNN), was evaluated in these scenarios.

Each method provides a prediction based on previously recorded load. ANNs and WNNs include weather information, as it was shown that weather has a considerable

[3]The historical load information was employed from the CER trial.

[4]Weather information was obtained from www.ogimet.com, a website that indexes publicly available weather data.

influence upon the energy consumption (Hernndez et al., 2012). The statistical methods rely on time-series regression in order to generate the prediction, and thus only use historical demand records.

4.2.1.1 Artificial Neural Networks

There are several approaches to day ahead power demand estimation through neural networks (Alfares and Nazeeruddin, 2002; Hippert et al., 2001). Some of the most common are networks with 24 outputs (Khotanzad et al., 1997), one for each hour of the day based on an input of previous days, and networks with 1 output (Park et al., 1991), providing the prediction for the next hour based on previous hours. The two approaches were both considered. After running a preliminary set of experiments, results showed that the 24 output version was more efficient, so the day-ahead solution was employed in this thesis. This also provides a longer prediction horizon compared to the 1 hour-ahead solution. The input of the neural network is based on the same day of the week as the one to be predicted (e.g., for predicting the next Tuesday, the recorded load from the previous Tuesday is used). The ANN also employs historical weather information and the forecasted weather for the day to be predicted. This involves dry bulb temperature and humidity. Furthermore, the input contains information about the day of the week, to cater for the differences between the five weekdays. A three-layer multilayer perceptron was designed, which involved one hidden layer.

The detailed arrangement for the neural network is presented in Fig. 4.5. There are 55 input neurons, which are divided as follows:

- 24 neurons used for previous load input, one for each hour of the day;
- 24 neurons are used for weather forecast input along the day, 12 for temperature and 12 for humidity;

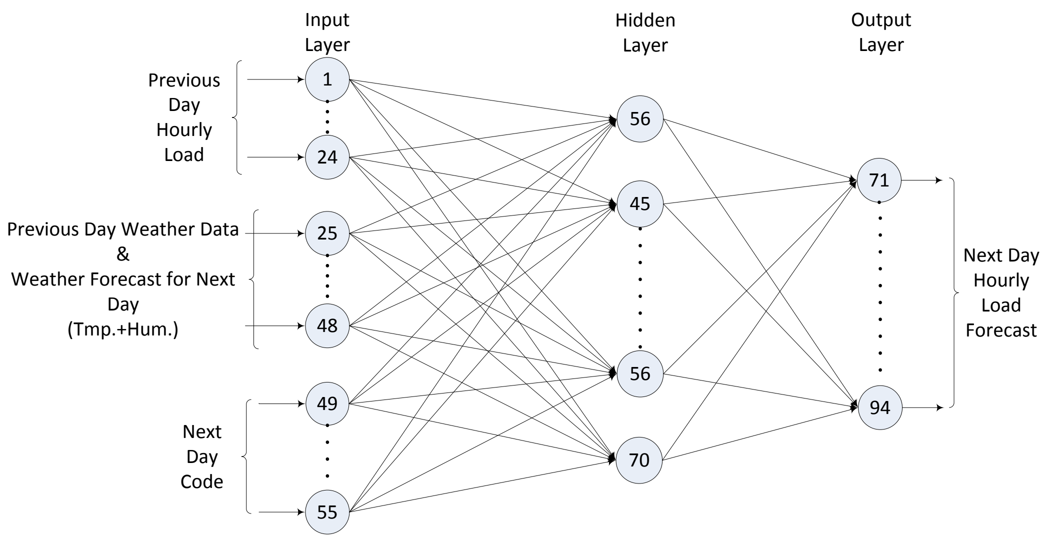

Input Layer
Hidden Layer
Output Layer
Previous Day Hourly Load
Previous Day Weather Data & Weather Forecast for Next Day (Tmp.+Hum.)
Next Day Code
1
24
25
48
49
55
56
45
56
70
71
94
Next Day Hourly Load Forecast

Fig. 4.5: Neural Network Structure

- 7 neurons are used for the day code input, for each day of the week.

The output layer comprises a total number of 24 neurons, which represent the short term load forecast. Each individual neuron provides a demand estimate belonging to an hour of the day, in consecutive order, starting from midnight. The hidden layer comprises 15 neurons, a number which was chosen based on empirical analysis (values between 9-30 were initially trialled).

Several learning algorithms were investigated for the neural network training process. Resilient backpropagation (RPROP) (Riedmiller et al., 2001) performed best when compared to QUICKPROP (Fahlman, 1988), SARPROP (Treadgold and Gedeon, 1998) and cascade training, so it was further exploited for the implementation of the prediction algorithm. Both the input and the hidden layer have an extra bias neuron for weight adjustments. The ANN is fully connected, with a total of 1224 $((55+1)*15+(15+1)*24)$ links between neurons.

Three sets of data were used: one for training, one for validation, and one for testing purposes. The training set is used to configure the weights of the neurons based on the available input, in order to reach the desired output values. The validation set is used to avoid overtraining, so that the neural network does not overfit the network to match only the selected samples of the training set. The testing set is used to evaluate the performance of the neural network after the training process is finished, where the predicted results are compared against the actual values of the forecasted day. The training set contains 210 weekdays (70% of total days), the validation set 60 weekdays (20%), and the testing set 30 (10%) weekdays. Since there is a relatively small number of samples, the vast majority of samples were selected for the training and validation of the neural network. The input layer and hidden layer weight activation functions were selected based on the results obtained on the validation set, to avoid overfitting.

4.2.1.2 Wavelet Neural Networks

Due to the very small scale of the load demand predicted, denoising techniques can be employed to remove noise from the load shape. This process should improve the training process, as the input set for the neural network has less variation, thus being better able to exploit the similarity between samples. Filtering is part of the wavelet neural network approach (WNN). In this case, the time-series is decomposed into 5 different sub-signals, based on their frequency, and the signal component with the highest frequency is processed in such a way as to retain only the larger peaks for the denoised curve.

Fig. 4.6 presents the decomposition process over 30 days. On the lower left side four unfiltered signals (from a total five) are shown, while on the lower right side are their filtered equivalents. Only the signals in the lowest two parts are filtered in order to retain the higher peaks, which are more significant than the other low level variations.

The same procedure as in the ANN case is used for the forecasting process, except for the input values which take in the smoothed version of the load. However, the desired

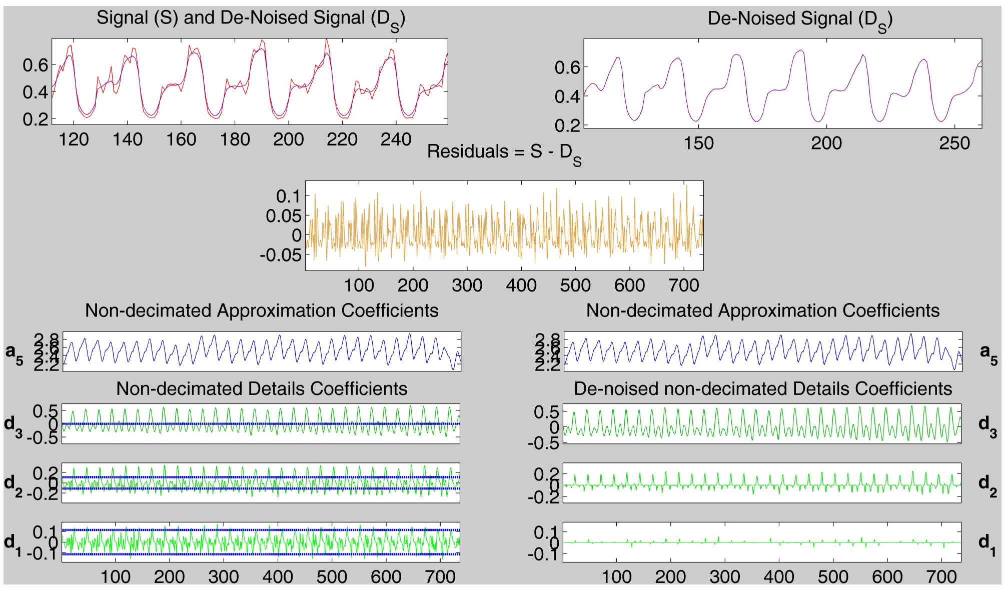


Fig. 4.6: Electricity Demand Decomposition

output of the neural network is still the same as in the ANN case (as opposed to a filtered version of the forecasted day's load).

The initial tests resulted in lower accuracy when rebuilding the signal from the two predicted parts, the denoised signal and the remaining residuals. This is due to the fact that the residual signal closely resembles white noise. Therefore the residuals are not further considered for prediction.

4.2.1.3 Auto-regressive Methods

Auto-regressive methods are able to analyse random processes and linearly relate the output of the prediction system based on previous values of the time-series. The series is decomposed through a formula that relates individual coefficients with the former n values. As opposed to auto-regression (AR), auto-regressive moving average (ARMA) and auto-regressive integrated moving average (ARIMA) additionally have a moving average part, where another set of coefficients is considered for the moving average model component. While AR and ARMA deal with weak stationary systems, ARIMA applies differencing on a non-stationary time series, thus removing the non-stationary component and treating the result as a stationary series. The AR model developed in this thesis estimates the following day's demand based on the previous 6 weeks of recorded demand ($n = 720$). The ARMA and ARIMA models both estimate the next day based only on the past week's data ($n = 120$).

4.2.1.4 Residential Neighbourhood Scenario Details

The prediction methods presented in the previous sections were evaluated on two different scenarios, that test different scales. The scenarios use information from the trial presented in Section 4.1. The trial recorded half-hourly smart-meter data from residential and commercial users. In this thesis the focus is exclusively on a control set of residential users, whose daily demand were not affected by electricity price changes over the day. The selected households do not rely on any type of demand response.

The first scenario comprises 90 houses, and the second one 230 houses. The latter one roughly encompasses the number of houses provided for by a 630 kVA transformer. The estimate was calculated based on several criteria. In the first scenario, the energy demand from 90 houses was considered, where the aggregated demand peaks at 140 kW. This scenario is based on the work by (Meschiari et al., 2013). Furthemore, capacity losses were taken into account, as presented in Eq. 4.1:

$$\begin{aligned} S &= P + jQ \\ S^2 &= P^2 + Q^2 \\ |P| &= |S||\cos\phi| \end{aligned} \tag{4.1}$$

where S is the apparent power, P the active power, Q the reactive power, j the imaginary imaginary unit, and $\cos\phi$ the power factor. The power factor is the decisive element in the conversion of apparent power towards active power. While the true (active) power circulation charges the network, the reactive power is introducing another randomly varying line congesting element, different from the active power variation. Both components of the apparent power vary independently. The reactive power is caused by inductive elements (coils) in home appliances. In this thesis, as can be seen from the considered data for the scenario, only the true power circulation is taken into account.

Transformers are designed with an overdimensioning factor of 0.5 from the maximum consumption. Considering a power factor of 0.85(inductive), the coverage of a 630 KVA should be roughly 340 kW or 230 houses[5], with respect to the proportions in the first scenario and the considered reactive power circulation.

In addition to the recorded power demand, hourly recorded weather data was employed from OGIMET (OGIMET, 2015), for the same time span as the smart meter trial. This includes hourly information about temperature and humidity reported in Dublin. The capital city was selected as a reference point for having the largest population in Ireland, assuming most of the surveyed users are from Dublin, since further information about smart meter users was anonymous [6].

Recall that this evaluation focuses only on weekdays, since their demand is higher and more unpredictable than the demand over the weekends. The load demand was normalized to fit in between 0 and 1 for easier processing in the case of neural networks, according to the formula in Eq. 4.2. The same procedure was applied to temperature and humidity values.

$$\frac{1}{1 + e^{\frac{-(x-\bar{x})}{stdev}}} \tag{4.2}$$

For the neural networks methods implementation, the Fast Artificial Neural Network

[5]These computations are based on the maximum value of the aggregated demand for 90 houses, which peaks at 140 kW.

[6]In addition to the fact that other Irish weather station had inconsistent information, with some of their recordings missing.

(FANN) library was employed (Nissen, 2003). FANN is a highly configurable open-source tool, written in C/C++, which simplified the configuration process of the neural network, and also enabled the integration of it in P-MARL.

The AR, ARMA and ARIMA methods were implemented through MATLAB's System Identification toolbox. Additionally, the load denoising employed in WNNs was accomplished through the Wavelet toolbox.

4.2.1.5 The Hybrid

The resulting forecasting method is a hybrid solution which exploits the best features of the previously evaluated forecasting techniques: artificial neural networks, wavelet neural networks and auto-regression. The hybrid solution uses as input previously recorded power demands, past day's temperature and humidity information, temperature and humidity forecasts for the day to be predicted, and information about the day of the week. The output is the next day's power demand estimate, provided as a sequence of 24 data points, one for each hour of the day.

Based on the previous results, a system was devised to combine three methods for a day-ahead forecast, each method selected at the best performing intervals. The algorithm runs tests over four previous weeks of real and forecasted data, and selects the best method on average for each of the 24 hours of the day. The four weeks period is selected to avoid the potential involvement of outliers among the tested days (i.e., anomalous days).

4.2.2 Energy Demand Pattern Change Detection and Matching Component Implementation

In situations involving unquantifiable uncertainties, quality of service guarantees cannot be provided. Despite the previous model generating accurate predictions, there are particular times when forecasts fail to closely match actual behaviour. These failures happen when anomalous events occur that affect the demand. In smart grids, anomalous situations are caused by anomalous events such as unexpected climate phenomena. The pattern-change detection component makes the system more robust in the face of such events.

Once the prediction model provides a forecasting estimate for the next day, as the forecasted day progresses, the estimate is compared against the actual demand along the first hours of the morning (the evening part is the period of highest demand, and needs to be accommodated for separately). The custom self-organising map (SOM) used for classification in this case comprises four classes: two classes for normal days (one for cold/winter days and one for warm/summer days), and two anomalous classes (one for public holidays and one for other particular days).

If significant deviations from the actual demand occur, the day is classified into one of the anomalous classes. In this case, the pattern change detection mechanism triggers reprediction, since the demand estimate is regarded as flawed. This process is pictured in Fig. 4.7, where the short-term load forecast (STLF) is the final output. A match is chosen based on similarly previously encountered patterns, which are found in a database of historic recordings. After the self-organising map classifies the type of anomaly detected, it provides the closest previously encountered match from the fitting class as a suggestion for re-prediction.

This section presents an adapting load forecasting mechanism based on pattern-change detection and matching. This mechanism is an additional contribution of this thesis, motivated by state of the art research involving load forecasting and SOMs where anomalous demands are only predicted based on calendar events. This technique detects anomalous power usage behaviours on the fly and triggers an appropriate re-prediction mechanism, as pictured in Fig. 4.7. Short term load forecasting is used to make ahead

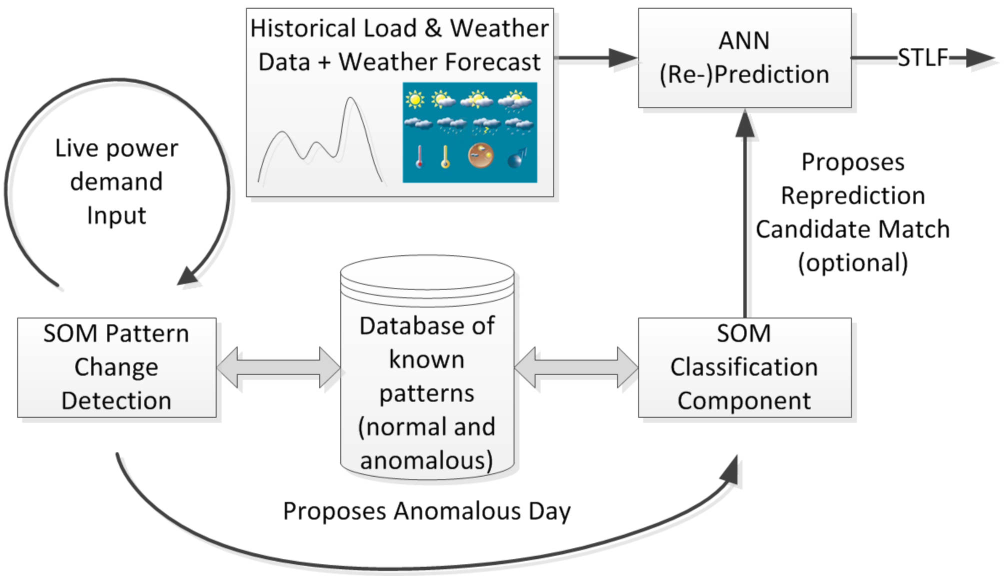

Historical Load & Weather Data + Weather Forecast
ANN (Re-)Prediction
STLF
Live power demand Input
Proposes Reprediction Candidate Match (optional)
SOM Pattern Change Detection
Database of known patterns (normal and anomalous)
SOM Classification Component
Proposes Anomalous Day

Fig. 4.7: Pattern Change Detection Mechanism

estimates in between 2 weeks and 24 hours. Some anomalies cannot be anticipated within a day ahead. These can occur as the day progresses, and actions taken at the point of anomaly detection can be critical for the optimal operation of the microgrid. Unlike the calendar-based approaches of the state-of-the-art forecasting methods, the proposed technique addresses unanticipated anomalous days. The pattern change detection component continuously monitors power demand during a day to detect if it becomes anomalous, the presumption being that it was not previously marked as anomalous. As a seemingly normal day progresses and anomalous power demands occur, the pattern change detection (PCD) mechanism detects changes from the expected behaviour. Once the type of change is detected, the PCD proposes the re-prediction of the demand based on a similarly previously encountered pattern found through SOM classification. This process is presented in Section 4.2.2.1.

4.2.2.1 SOM Classification

An anomalous day (i.e., a 24 hour time sequence) is detected with 100% accuracy only once it has ended, as anomalies can also occur at the end of the day. When detected, these anomalous days can only be of use at a further date, which is the case in state of the art forecasting approaches. The proposed approach employs self-organising maps for classification and pattern change detection of anomalous days before the day reaches its end, and more importantly before the critical evening peak. As such, information is provided in a timely manner and assists anomaly mitigation strategies.

SOMs group similar samples into clusters (also known as classes). Initial empirical analysis involving a large number of classes led to disparate clusters of samples due to the relatively low number of samples available. As a result, public holidays and anomalous days were scattered across the map. Through experimentation, bringing the number of classes down to 4 led to all Irish bank holidays being clustered into a single class, and non-calendar based anomalous days into another class. The remaining two classes comprise only normal days. Therefore, in this thesis a self-organising map with 4 classes (1a, 1b, 2a, 2b) was further employed.

The input layer of the SOM contains 48 half-hourly measures of power demand, i.e., one sample of power demand recorded every half an hour over the course of a 24 hour period. This input layer is illustrated in Fig. 4.8a. An input sample with 48 values represents one day of demand. Once the dispersion of the samples (one sample per each day) is established, these are grouped into the four different classes based on their similarity. This process is visualised in Fig. 4.8b. However, the four classes share some similarities between themselves. Fig. 4.8c pictures the four different classes with blue hexagons. Classes with similar properties are linked by lighter coloured (elongated) hexagons. There is number 368 days available for the training of the SOM. From these, the total number of samples contained by each class is shown in Fig. 4.8d.

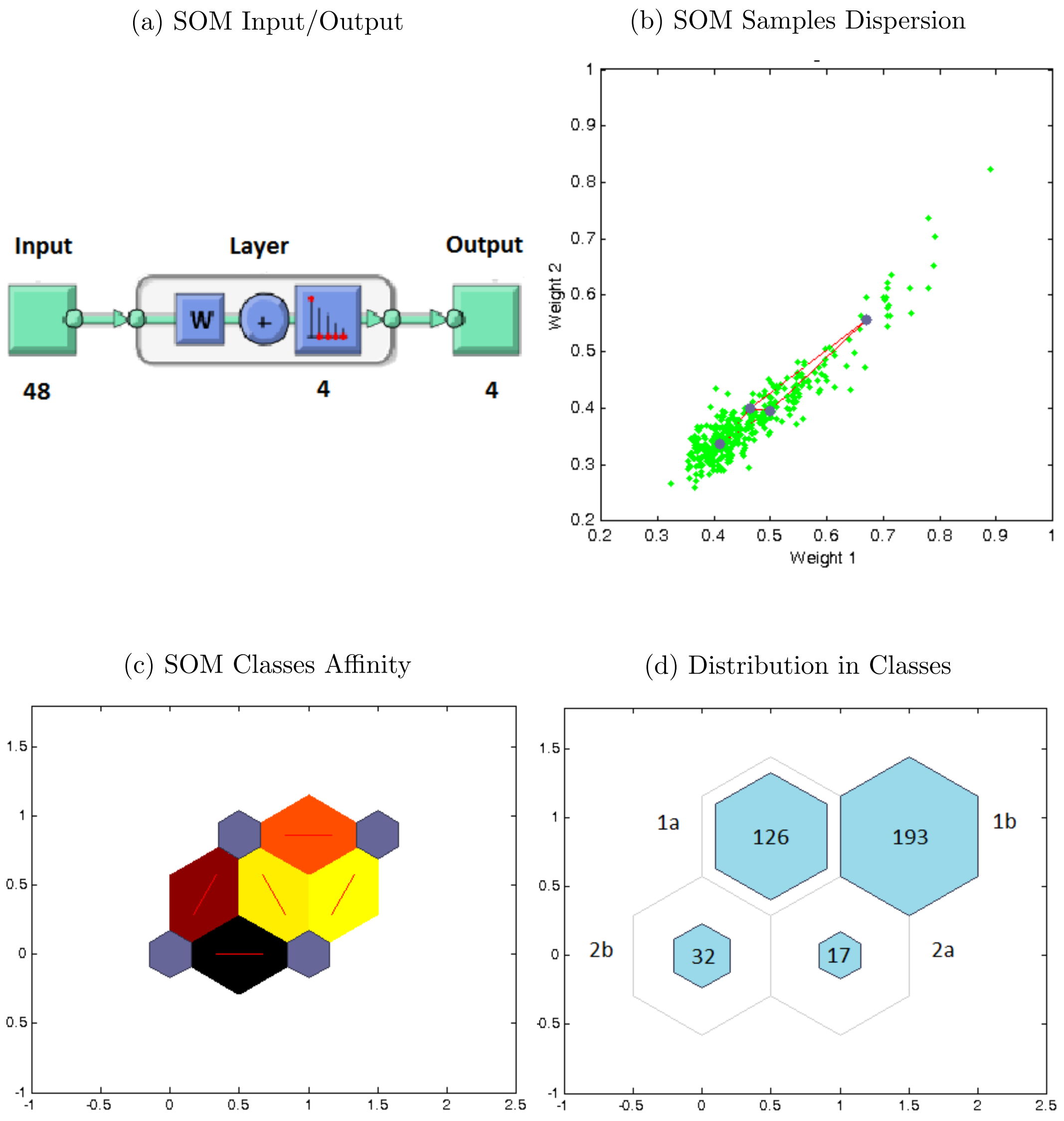


Fig. 4.8: Self-Organising Map

Close inspection based on the historical load input reveals that the SOM has further clustered samples into:

- **1a)** normal days with higher power demands (cold season);
- **1b)** normal days with lower power demands (warm season);

- **2a)** anomalous days occurring during bank holidays;
- **2b)** anomalous days outside calendar based events.

The SOM classifies all the 13 occurring bank holidays in the available interval (between 01-08-2009 and 31-12-2010) into a single class, together with a few more days around them, particularly in the Christmas/New Years period. The most detached class is the one containing only anomalous days, outside of the holidays range (that cannot be explained by public holidays or proximity to these holidays), which sum up to 32 days. While some anomalous days occur in one of the years (Nov-Dec 2010), these do not occur in the other one (Nov-Dec 2009). Part of the anomalous days occur around public holidays. These have a particular demand shape, as some residential consumers take additional days off while some do not, resulting in unique demand patterns. The remaining days are divided between the other two classes depending on seasonality, as there are particular differences between summer and winter days. The lower the daily temperature, the higher the demand, as many household heating units rely on electricity in Ireland. During summertime, as temperature does not sufficiently rise to create thermal discomfort among household users, HVAC systems are not employed and as a result lower power demand patterns occur.

4.2.2.1.1 Discussion The resulting SOM encompasses four classes, representing two main groups: normal and anomalous. The two normal classes, *1a* and *1b* are divided based on season (warm/cold) and contain similar patterns, with the main differences occurring in the amplitude of the shapes. However, there are only two classes for the anomalous samples: one that comprises bank holidays (*2a*), and one comprising other anomalies (*2b*). While *2a* presents a clear similarity between samples, *2b* contains samples with much more variation between each other. Due to the small number of anomalous samples, more classes could not be defined. However, with more samples available, similarities between anomalous samples in class *2b* could be found and this class could be further divided in subclasses. This would give the option of accurately identifying the type of anomaly, and in case of new anomalous types occurring, the option of expanding the number of overall anomalous classes to accommodate these.

4.2.2.2 SOM Pattern Change Detection

A pattern change detection system was implemented to deal with anomalous days on the fly, by providing additional information about the state of the day (normal or anomalous). State of the art forecasting approaches that deal with anomalous day prediction employ just a classification component that post-processes the day when this has already passed. The assumption is that the pattern of the passed day can be used at a further date, possibly same time next year (e.g., for Christmas day).

A SOM detects anomalous days with 100% accuracy in post-processing mode, when all the days have ended. Based on the previous section, experiments were conducted to detect when a trade-off between accuracy and reaction time would be sufficient for an anomaly detection algorithm to react sooner, with satisfactory accuracy. When the anomaly detection rate is approximately 50%, it is more likely that a day is anomalous than not, and further actions are needed to address this .

For this purpose, the average demand shape over 24 hours was initially computed. Samples were taken every half hour, represented in Fig. 4.9 by the blue curve. This average demand is based on all the available historical data from the smart-meter trial. This shape is much smoother compared to real days values due to the averaging process. For comparison purposes, an example of a normal day is pictured in Fig. 4.9 with a red line.

The average shape fits in the upper two classes (1a and 1b) of the SOM, the ones belonging to normal days. Each value in the 48 element vector representing the average

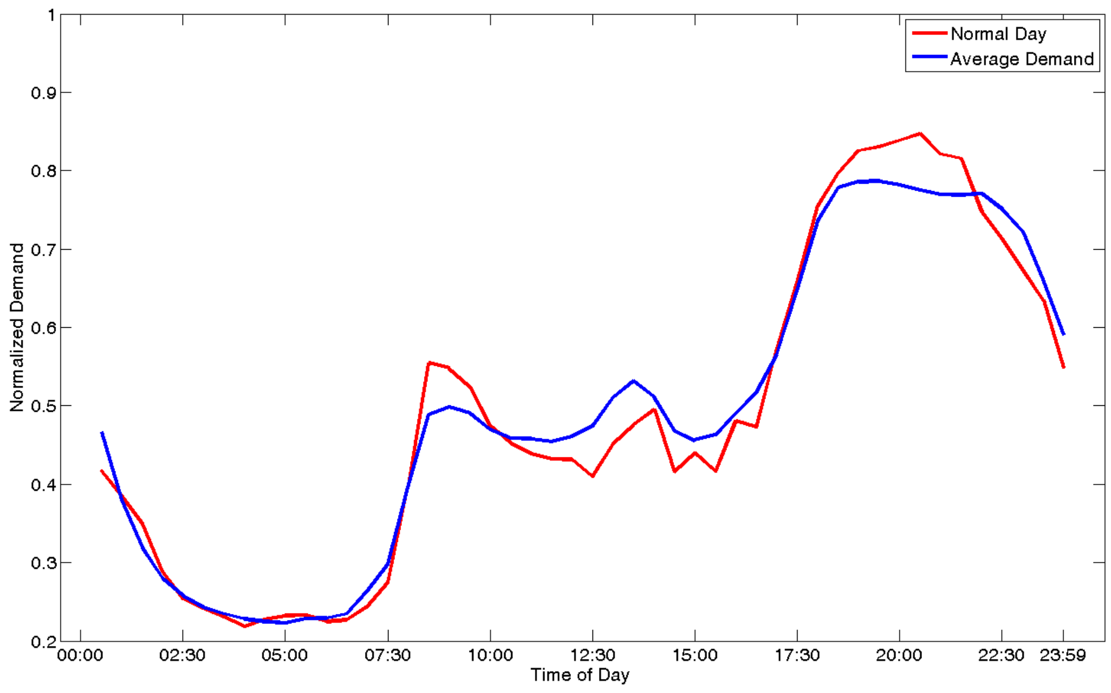


Fig. 4.9: Average Shape

shape is replaced with values obtained from the demand of the day in progress, starting from midnight. For example, if the hour is 06:10, 12 values from the actual day from midnight up to and including 06:00 replaced the first 12 values in the average shape, and are followed by 36 samples of the average day representing values between 6:30 up to and including 23:30.

The re-prediction component is based on an ANN which adjusts its historic load input part (24 neurons, one for each hour of the day) to a combination between the demand observed so far of the anomalous day [7] and the remaining demand from the closest match provided by the SOM component. This way it is able to provide more accurate results for the evening demand interval.

[7]This interval is 00:00-14:30, which is motivated later in Section 5.2.2.

4.2.2.2.1 Discussion The pattern change detection component will detect changes in the evolution of the environment after the morning has passed. In a residential scenario, demand in the morning is significantly less than in the late afternoon and evening period (as can be seen in Fig. 4.9), therefore preparation for this specific period is done at the expense of the morning period. However, in other applications early detection of changes can be important. For these cases, the pattern change detection component could be adjusted to use a sliding window mechanism instead of SOMs. This mechanism compares the environment's behaviour with the forecast for the last n steps observed, and notices if there are any significant differences between the two. If significant differences occur, a change is detected and a new forecast needs to be computed.

4.2.2.3 ANN Prediction and Re-prediction

In the prediction component from Section 4.2.1 several techniques are first to be evaluated and afterwards combined in an adaptive hybrid method to increase forecasting accuracy. This approach is more computationally intensive than its subcomponents, and requires several consecutive (normal) days to accurately forecast a following (normal) day. For the purpose of this section, where anomalous days are dealt with, only one of these techniques was selected for implementation: a simplified version of the neural network component presented in Section 4.2.1.1. ANNs do not require additional learning once trained, thus forecasting is instantaneous when input data is provided. This is a useful feature in time-critical applications. The component involves a multilayer perceptron artificial neural network (ANN) trained through resilient backpropagation. The implementation is based on the Fast Neural Network Toolbox (Nissen, 2003).

The ANN presented in Section 4.2.1.1 is more effective to predict normal days, therefore it tends to overfit for these. To avoid overfitting and allow for better forecasting of

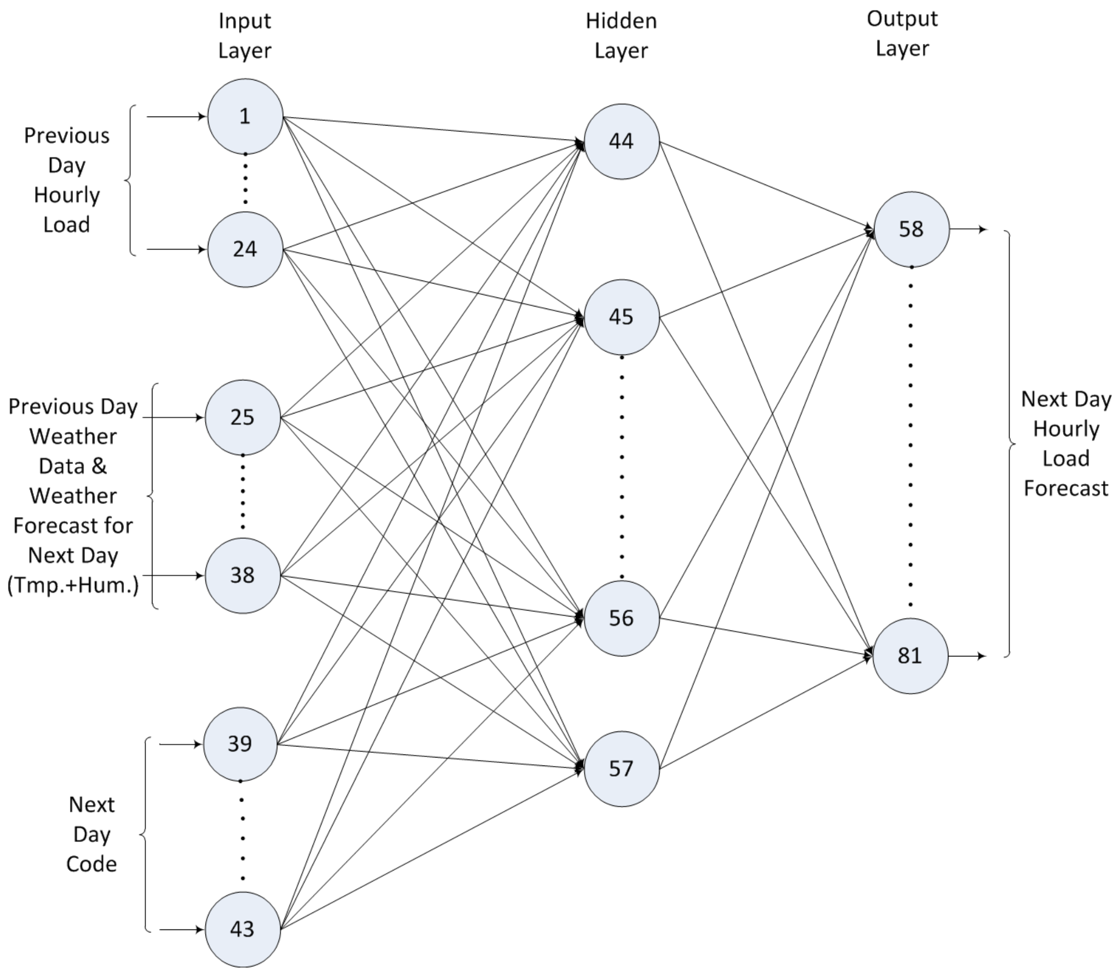

Input Layer
Hidden Layer
Output Layer
Previous Day Hourly Load
Previous Day Weather Data & Weather Forecast for Next Day (Tmp.+Hum.)
Next Day Code
Next Day Hourly Load Forecast
1
24
25
38
39
43
44
45
56
57
58
81

Fig. 4.10: Re-Prediction Neural Network Structure

anomalous days, the total number of neurons in the ANN was reduced. While the ANN in Section 4.2.1.1 uses 55 neurons for input, these were reduced to 43. The simplified ANN is illustrated in Fig. 4.10. The input neurons are further represented as follows:

- 24 neurons are used for previous load input (one for each hour).
- 14 neurons (down from 24) are used for weather forecast input along the day. Several consecutive neuron values from the previous model were further averaged and used in a single neuron (as these have less impact than the historical load),

thus resulting in 8 neurons for temperature and 6 for humidity; more inputs were chosen for temperature as it is more relevant than humidity according based on the correlation tests.

- 5 neurons (down from 7) are used for the day code input, since the scenarios focus exclusively on weekdays.

The neurons in the input layer employed for load, temperature, and humidity use extrapolated values from samples taken during a whole day, depending on their correlation level with the load. The neurons' values corresponding to the former load represent average demand for an hour of the day (24 hours in total). The neurons corresponding to temperature contain 3 consecutive hourly values averaged for each neuron. The neurons corresponding to humidity contain 4 consecutive hourly values averaged per neuron, since they are less relevant than the load and temperature neurons.

Another change from the initial ANN described in the previous section occurs in the training process. Due to the small number of anomalous days in the sample set, the number of samples was artificially increased three-fold, by adding small random variations to the demand of each real day recorded. The validation and testing set were not affected by this measure, and comprise only actual recorded demands.

The output layer contains 24 neurons, which represent the short term load forecast. Each of the output neurons provides a demand estimate corresponding to the equivalent hour of the day.

Each prediction is based on the weather forecast for the day it attempts to predict, together with the historical recorded demand occurring over the same day of the previous week.

For the re-prediction mechanism component, as opposed to the ANN presented in Section 4.2.1.1, the input part with regard to the historical load changes to accommodate the shape provided by the pattern change detection and matching mechanism, instead of using the previous day's load. Since the network does not require any more training at this stage, re-prediction is instantaneous once the network is provided with the appropriate anomaly matching input from the SOM. In the case of re-prediction, the first 14 neurons are substituted with the values obtained from the day in progress up to hour 14 (the hour when a match can be found, after sufficient actual demand hours are entered in the SOM), while the last 10 are based on the closest fit found by the pattern matching mechanism (the fit found by the SOM).

4.2.3 Multi-Agent System Implementation

P-MARL is developed by integrating the previous prediction and pattern change components with a MARL system. The reinforcement learning process is implemented using Q-Learning. The Q-Learning process can be represented as a *single-objective* problem for each agent: to make sure that a desired charge is reached, rewarding the agent when deciding to charge. This results in a greedy behaviour, with the agent charging at every time-step until the electric vehicle is fully charged. This has undesirable effects, as the EV can charge during periods of high demand. As such, the charging process needs to be constrained within periods of low demand to comply with DSM targets. In order to achieve this, a demand forecast is provided to each agent, and based on it an agent can decide which load levels are most desirable for charging.

These observations, and the scenario, result in the following list of constraints:

- an agent can decide whether to charge or not at 15 minutes intervals; thus an agent is charging in time-slots of 15 minutes each;
- an agent can charge at home only between 18:00 and 9:00 the following morning, therefore it has a maximum of 15 hours available for charging;

- an agent needs to charge sufficiently(i.e., have sufficient state of charge (SOC) of the battery) for its next day's trip;
- an agent has to avoid peak-time charging when possible.

These are the agent's pre-requisites. To meet these constraints, an *additional objective* was added for the agent, which rewards the EV agent when it decides to charge in periods of low demand, and penalises it when it decides to charge in periods of high demand. In essence, this represents a cost minimisation policy, as under dynamic pricing schemes the price increases with the demand.

Algorithm 4 illustrates the implementation of P-MARL in the smart grid scenario with respect to these conditions. The stopping criteria for exploration in line 22 can be defined to be either when a certain number of training episodes have occurred, when a certain level of overall performance has been achieved, or when there are no changes in actions from one episode to another[8].

The MARL system can be further modelled to employ information about the future state of the environment. This is provided by the prediction components. The forecasted day-ahead load is analysed and afterwards discretized into ten different levels (or states)[9], depending on the amount of power used. The lowest states will present the highest reward to the agent, therefore leading the agent to charge during intervals of low demand.

Agent learning is integrated in the code, with different types of learning options depending on the level of interaction between agents. Specifically, there are three types of learning that can occur, depending on the information provided by the transformer:

1. Only present baseload information is provided to an EV agent. As a result, its reward is based exclusively on the effect of its own action on the baseload. The

[8]This means that a Nash equilibrium was reached between agents.

[9]Only ten states were used, since the state-action table grows exponentially with the number of states involved, and requires more learning. However, more computationally expensive solutions can be devised, for example with 100 levels of quanta, to achieve better performance.

Algorithm 4: P-MARL Algorithm Implementation: Smart Grid Scenario

```
1  forecastedLoad ← obtainFinalPrediction(predictionSystem);
2  foreach EV do
3      chargeNeeded ← computeChargingRequirementsEV(SOC,tripDistance);
4      slottedLoad ← sortLoadLevelsBySlot(forecastedLoad);
5      desirableSlots ← computeDesirableSlots(slottedLoad,chargeNeeded);
6  end
7  repeat
8      simEnv ← startEnvironmentSimulation(forecastedLoad);
9      foreach slot in simEnv do
10         for each EV do in parallel
11             decision ← exploreQ-LearningCharging(EV,desirableSlots,slot);
12             resultingAction ← takeActionOnEnv(decision);
               // EV decision is whether to charge or not in current slot
13             updateChargeStatus(resultingAction);
14         end
15         updateLoadStatus(slot,decisions);
16         for each EV do in parallel
17             rewardEnv ← getRewardEnv(decision, envLoadStatus);
18             rewardCharge ← getRewardEVAgent(decision,chargeStatus);
19             Q-Tables ← updateQ-
                 Value(rewardEnv,envLoadStatus,rewardCharge,chargeStatus);
20         end
21     end
22 until stoppingCriteriaReached;
```

agent has a localized view of the environment. This represents the implementation for the case presented in Section 3.2.3.1, *single agents acting separately on the environment.* The transformer aggregates all the localized environment views into a global version, but this state is not sent to agents.

2. Present baseload information is provided to an EV agent, and after all agents take actions, the aggregated load status is further provided. The reward is based on an agent's action and the resulting aggregated load. This represents the implementation for the case presented in Section 3.2.3.2, *multiple agents acting simulatenously on the environment.*

3. An agent receives the state of the environment's load after other EV agents have already taken action. This way an agent takes an action, and receives a reward based only on the action taken and the state of the environment reached afterwards (i.e., as a result of its action exclusively). Once it has taken an action, the state of the environment is updated and another agent takes action afterwards, in sequential order. This is the implementation for the case presented in Section 3.2.3.3, *multiple agents acting sequentially on the environment.*

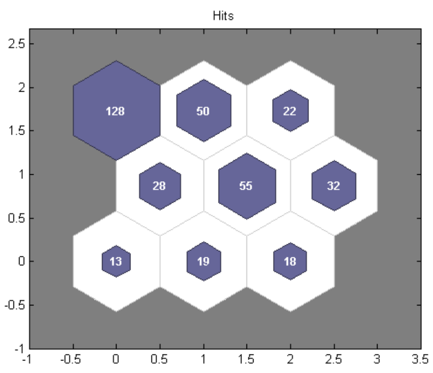
4.3 Solar Energy Usage Optimization Scenario

A secondary scenario is proposed to further test requirement *R1: Minimize Online Learning* for P-MARL. The scenario comprises a group of SMEs, which on aggregate emulate the behaviour of a business park. This group of SMEs benefits of solar energy generation systems (i.e., solar panels). In this scenario, the baseload is assumed to be stationary, but the amount of solar energy available is non-stationary.

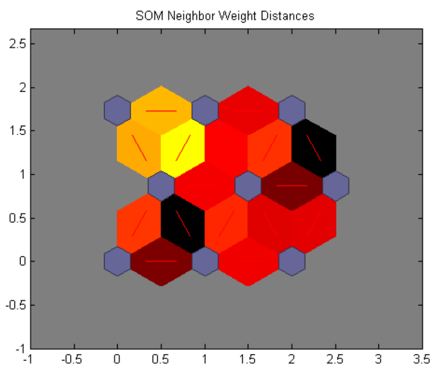
The scenario's details are as follows:

- the power demand from 200 SMEs is employed from the CER trial. These are selected from the control group. Their aggregated demand represents the demand of the business park.

- the business park is considered to be in the nearness of Dublin airport, therefore weather data from the airport is further employed. This was obtained from the Irish meteorological service website[10].
- two variables from the weather dataset are employed: amount of sunshine per hour, which is considered to be directly related to solar energy output (solar irradiance per hour), and amount of clouds per hour, which is considered to assist in forecasting the former.
- each SME is provided with renewable energy by solar panels with a surface of 10 square meters, and an efficiency of 20%.



(a) SOM Samples Dispersion



(b) SOM Classes Affinity

Fig. 4.11: Solar Energy Scenario Daily Samples

The 365 available samples (one for each day of 2010) were used to train a SOM with 9 classes. The classes illustrated in Fig. 4.11a are numbered as follows, from left to

[10]www.met.ie

right: 1,2,3 in the lower level, 4,5,6 in the middle layer, and 7,8,9 in the upper layer. The patterns for the samples in the 9 classes are as follows:

- class 1 shows high energy production in the morning, but low for the rest of the day.
- class 2 shows high energy production overall, with brief plunges in production during the day.
- class 3 shows high energy production throughout the day, with a dip in production during the evening period.
- class 4 shows low energy production throughout the day, while showing some spikes in production for short periods mostly during midday.
- class 5 shows increased energy production midday, but decreased in the morning and evening period.
- class 6 shows high energy production overall, without any particular dips or spikes.
- class 7 shows average to low production days, with some of the days spiking in the morning and some of the days spiking in the evening.
- class 8 shows average to low production days, with production spiking midday.
- class 9 shows high energy production overall, with low energy production in the morning.

Only the 32 samples from class 6 from Fig. 4.11a (the rightmost class from the middle line) show a clearly repeating pattern. These days had mostly sunny hours, and thus presented a bell-shaped pattern for the energy production throughout a day, a typical output for sunny days. The peak of the bell occurs at midday, when there is the highest amount of energy produced. For the samples fitted in the other classes, there was no unique pattern. For example, a day with low energy production in the morning (when a deviation from the ideal case of class 6, which has high energy production in the morning, is detected) can either have:

- high production for the rest of the day (class 9).
- high production during midday (class 5)
- low production midday (class 4)
- average production during midday (class 7)

As such, pattern matching in this case can lead to inconclusive results, therefore is not applicable. As a result, only requirement **R1: Minimize Online Learning** can be fulfilled by P-MARL in this scenario.

4.3.1 Solar Energy Forecasting Implementation

The solar energy produced depends on the following factors:

- location (latitude and longitude)
- date and time of day
- amount of sunshine received (clouds affect the amount of energy produced)

The first and second factor determine the total amount of solar radiation (irradiance) received in a particular location on Earth, at a particular time and date. The total amount of irradiance can be computed based on these, but represents an ideal value, which is influenced by weather conditions. The last factor, amount of sunshine received, is the factor that inserts non-stationarity into the equation. This factor depends on the amount of clouds in the sky. However, the latter follows a non-stationary pattern.

Hourly weather information for the year 2010 was obtained for Dublin airport[11]. Among other variables such as temperature and humidity, the data contained two highly relevant variables: sunshine per hour (values between 0-1, low to high), and amount of clouds per hour (values between 0 and 8, low to high). The amount of sunshine per hour can multiplied with the ideal solar irradiance to compute the actual amount of solar energy produced.

The tests performed over the dataset showed a correlation of approximately 0.53 between the amount of sunshine per hour and amount of clouds per hour. The amount of sunshine per hour was considered to be the end result used in the computation of the irradiance, while the amount of clouds the day-ahead forecasting measure for the amount of sunshine. However, this type of forecast gives approximately 20% forecasting error. To improve on this forecast, a neural network with the following structure was created.

- 1 input, for the ideal solar irradiance for the hour to be forecasted
- 1 input, for the amount of clouds during the hour to be forecasted
- 1 output, for the sunshine for the hour to be forecasted

This network was trained over one year, with the hours selected for each day to be in between sunrise and sunset times.

4.3.2 Multi-Agent System Implementation

The multi-agent system implementation for the solar energy scenario follows the same steps as the one presented in Section 4.2.3. However, instead of using only the baseload (non-stationary in the residential scenario) as input for the environment simulation, the input for the MAS is the baseload (stationary in this SME scenario) from which the

[11]www.met.ie

energy produced by the solar panels (non-stationary) is subtracted. The hours when solar energy charging is available are between 05:00 and 22:00, which represent the earliest sunrise and latest sunset hours in Ireland during the year 2010, respectively.

4.4 Demand Response Benchmark: Optimal Centralised Solution

A Pareto optimal performance was defined for this problem to evaluate against P-MARL. While a centralised solution is not suitable in such cases, it is guaranteed to be optimal with respect to a defined set of constraints. For this, a system of dynamic pricing is assumed, where the solution leads to EVs using power at the lowest demand times (considering energy cost to be directly proportional with the system power load). The resulting constrained optimization function is presented in Eq. 4.3:

$$\min F(x) = \min \sum_{j=1}^{m} \left[\sum_{i=1}^{n} \left(x_{ij} + C_j \right) \right] x_{ij} \tag{4.3}$$

where $F(x)$ is the cost function, n the total number of electric vehicles, m the total hours available for charging (assuming the same availability schedules for EVs), x_{ij} the charging decision of vehicle i at time j (0 for not charging and 1 for charging), and C_j the initial cost of energy at time j (based on baseload).

The function's optimal solution is a Pareto front of a large number of possible optimal solutions, since this is an underdetermined system of equations where the unknowns x_{ij} can only take binary values. While solutions are achievable, the purpose of the benchmark is not to obtain individual solutions for each agent, but to define the aggregated optimal charging solution. This is a valley-filling problem, such as the one presented by (Gan et al., 2011).

The problem can be solved by taking each EV in turn, computing the minimum amount of charging slots required, and then allocating these charging slots in the periods of low demand. Each EV incrementally updates the overall demand until all EV charging slots are allocated. The valley-filling algorithm is presented in Algorithm 5.

Algorithm 5: Valley-Filling Algorithm

```
1  foreach $EV$ do
2      energyRequired ← energyRequiredByVehicle ;
3      energyUsed ← 0 ;
4      while $energyUsed < energyRequired$ do
5          desiredSlotIndex ← slotIndexOfLowestEnergyCost ;
6          while $numOfUsedEnergySlots(desiredSlotIndex) \geq$
            $maxEnergySlotsPerTimeSlot$ do
7              desiredSlotIndex ← slotIndexOfNextLowestEnergyCost;
8          end
9          SlotsOfEnergyUsed(desiredSlotIndex)++;
10         energyUsed++;
11     end
12 end
```

The best performance that can be achieved by the MARL technique should have the same aggregated effect as the valley-filling algorithm. In order to evaluate the optimality of the MAS solution, a formula derived from the mean absolute percentile error (MAPE) was used. This is shown in Eq. 4.4.

$$M = \frac{1}{m}\sum_{j=1}^{m}\left(1 - \frac{|X_j - \hat{X}_j|}{TotalNoOfEVs}\right) \tag{4.4}$$

where X_j is the total number of EVs charging at time-slot j, and $\hat{X}_j$ the optimal amount of EVs that should be charging at time-slot j.

The resulting aggregated behaviour outputted by this algorithm defines the optimal line from Section 5. If a solution fully matches this line, it is considered to have 100% Pareto efficiency. Any deviations from the optimal line are penalised.

4.5 Summary

This chapter presented the implementation of P-MARL for two problems involving of decentralised control: one where P-MARL is tasked with the charging of a group of electric vehicles located in a residential microgrid, and one where it is tasked with optimizing the usage of available solar energy.

In the first scenario, energy demand is forecasted on a day-ahead basis by the prediction component. The actual energy demand is continuously evaluated against the forecast. Any changes from expected demand are detected and matched by the pattern-change detection and matching component. The information about expected future demand is provided to the multi-agent system, which uses it to create a simulation of the environment where EV agents learn optimal charging strategies. This knowledge is exploited later, on the actual environment. The performance of P-MARL in this scenario will be evaluated against a Pareto front defined by an optimal centralised solution.

In the second scenario, solar energy production is forecasted on a day-ahead basis by the prediction component. In this scenario, there is no implementation of a pattern-matching component, as solar energy fluctuations do not respect any specific patterns. As such, only the energy production prediction is provided to the multi-agent system, where this is used to generate a simulation of the environment where EV agents learn how to optimize usage of the available solar energy. Similarly to the previous scenario, the performance of P-MARL in this case will also be evaluated against a Pareto front defined by the optimal centralised solution.

Chapter 5

Evaluation

This chapter presents the evaluation of P-MARL in a set of smart grid scenarios. The performance of the three components of P-MARL, *prediction*, *pattern-change detection and matching* and *the multi-agent system* is evaluated against a Pareto front, determined by an optimal centralised solution. The optimal solution represents an ideal case, where the environment's behaviour is perfectly predicted, and agents make optimal charging decisions with respect to the EVs charging objectives and the demand response target. P-MARL is evaluated against the MARL requirements presented in Section 1.3.2.

Prediction of the environment's future energy demand is used in an offline simulation of the environment, where EV agents train together to achieve the first target, *R1: Minimize Online Learning.* Only after they have trained offline, EV agents apply this knowledge on the actual environment.

The predicted demand is monitored against the actual demand, to meet *R2a: Detect Sudden Changes.* If any change is detected, the change type is analysed to acquire insight about the environment's future behaviour from this point on, so as to fulfil *R2b: Estimate Change Type.*

Once a particular change demand is detected and matched, the change type is provided to agents for them to re-train offline, in order to prepare for upcoming changes in the demand. This is done to fulfil *R3: Prepare for Changes*.

5.1 Experiments

P-MARL is applied in a set of scenarios under several types of prediction: initial prediction, prediction after pattern-change detection and matching, and perfect prediction, as an additional benchmark. Initial prediction is used to evaluate *R1*, while prediction after pattern-change detection and matching is used to evaluate *R2a*, *R2b* and *R3*.

Three types of agent interactions are evaluated in the experiments, to determine P-MARL's performance under different levels of agent-contributed non-stationarity. These interactions are based on those described in Section 3.2.3.

1. *Single-agents acting separately*: EV agents see only the effect on their own demand and charging actions on the environment, without access to the aggregated demand which results from other agents' actions. This case does not capture agent-contributed non-stationarity.

2. *Multiple agents acting simultaneously*: EV agents see only the cumulative effect of the group's charging decisions on the environment (i.e., only the aggregated demand). This case illustrates agent-contributed non-stationarity when actions are simultaneous.

3. *Multiple agents acting sequentially on the environment*: Each EV agent can see the effect of its own individual action on the aggregated demand. This case illustrates agent-contributed non-stationarity when actions are sequential.

Two scenarios are evaluated, where:

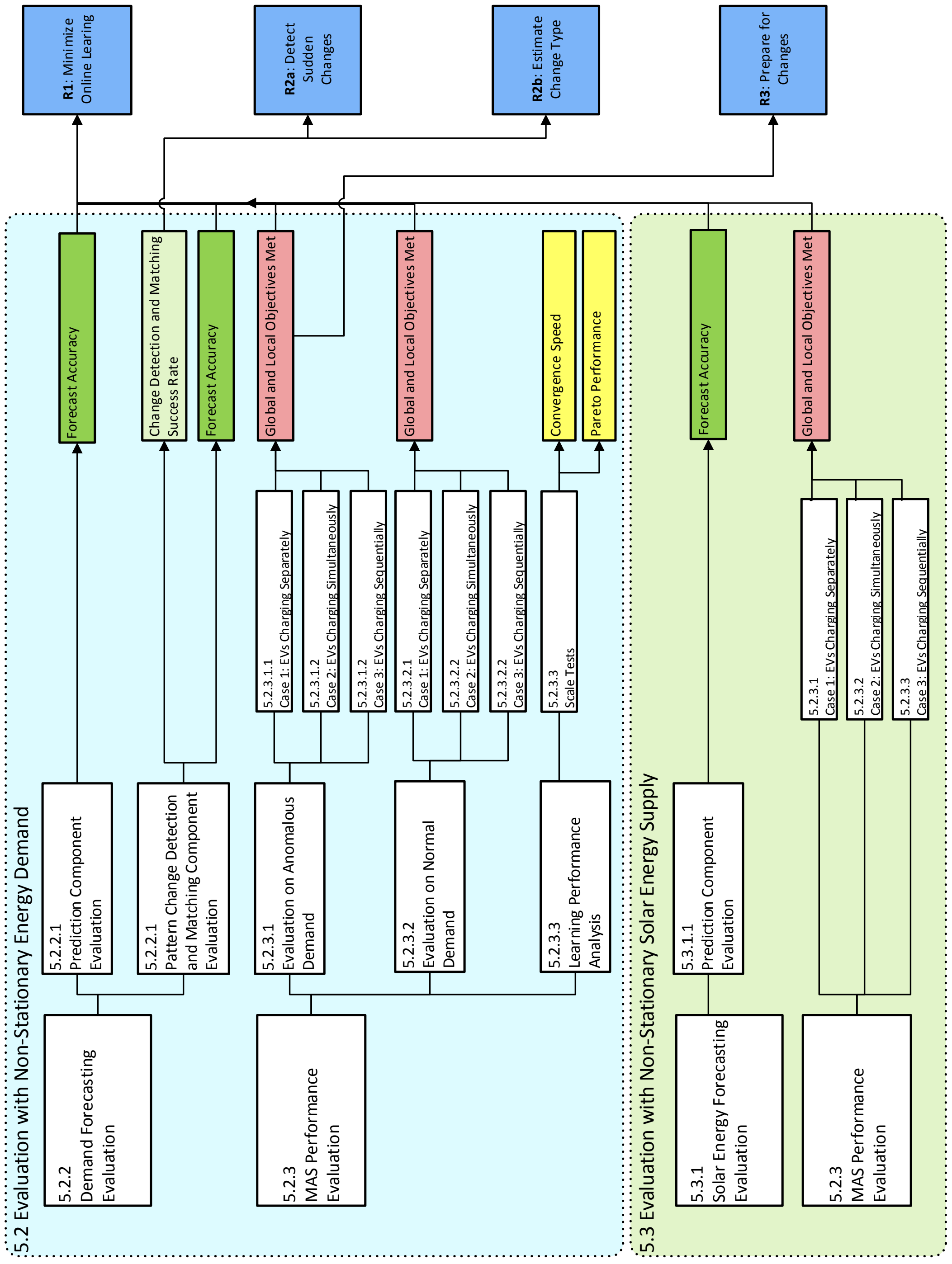

R1: Minimize Online Learing
R2a: Detect Sudden Changes
R2b: Estimate Change Type
R3: Prepare for Changes
5.2 Evaluation with Non-Stationary Energy Demand
5.2.2 Demand Forecasting Evaluation
5.2.3 MAS Performance Evaluation
5.2.2.1 Prediction Component Evaluation
5.2.2.1 Pattern Change Detection and Matching Component Evaluation
5.2.3.1 Evaluation on Anomalous Demand
5.2.3.2 Evaluation on Normal Demand
5.2.3.3 Learning Performance Analysis
5.2.3.1.1 Case 1: EVs Charging Separately
5.2.3.1.2 Case 2: EVs Charging Simultaneously
5.2.3.1.2 Case 3: EVs Charging Sequentially
5.2.3.2.1 Case 1: EVs Charging Separately
5.2.3.2.2 Case 2: EVs Charging Simultaneously
5.2.3.2.2 Case 3: EVs Charging Sequentially
5.2.3.3 Scale Tests
Forecast Accuracy
Change Detection and Matching Success Rate
Forecast Accuracy
Global and Local Objectives Met
Global and Local Objectives Met
Convergence Speed
Pareto Performance
5.3 Evaluation with Non-Stationary Solar Energy Supply
5.3.1 Solar Energy Forecasting Evaluation
5.2.3 MAS Performance Evaluation
5.3.1.1 Prediction Component Evaluation
5.2.3.1 Case 1: EVs Charging Separately
5.2.3.2 Case 2: EVs Charging Simultaneously
5.2.3.3 Case 3: EVs Charging Sequentially
Forecast Accuracy
Global and Local Objectives Met

Fig. 5.1: Experiment Map

1. a group of electric vehicles need to be charged at home in a residential neighbourhood while avoiding peak demand, and under non-stationary energy demand.

2. a group of electric vehicles need to be charged while maximizing the use of solar energy in an industrial park and avoid peak demand, under non-stationary availability of solar energy .

The first scenario is used to investigate P-MARL under requirements R1, R2a, R2b and R3, while the second scenario is used to conduct further investigation on requirement R1.

The full set of experiments where the requirements are addressed is illustrated in the diagram in Fig. 5.1. The elements in red boxes represent the sections where P-MARL is evaluated against the list of requirements, while the other elements in the column represent the sections where the performance of the sub-components of P-MARL are separately evaluated. The relation between the separate elements and the list of requirements is shown in the last column.

5.2 Evaluation with Non-stationary Energy Demand

For the evaluation of P-MARL, the power demands from a community of 230 households were employed. An EV penetration rate of 40% is assumed (Nemry and Brons, 2010), resulting in a total of 90 EVs. A daily trip is 50 km (EPA, 2008), while the EV specifications are based on the Nissan Leaf characteristics (Information, 2014). The vehicles can choose 15 minutes charging slots anytime between 18:00-09:00. This smart grid scenario was implemented in GridLAB-D, a power distribution system simulator (U.S. Department of Energy at Pacific Northwest National Laboratory, 2015), where a standard charging rate of 1.4kW was used.

For the experiments, two days were selected from the dataset: a normal day and an anomalous day, from the perspective of demand. The anomalous day is selected to identify P-MARL performance differences between employing only the primary prediction component (which targets **R1**), and a combined system that also involves pattern-change detection and matching abilities (which additionally targets **R2a**, **R2b** and **R3**), while the normal day is used to evaluate P-MARL performance when no deviations from predicted behaviour occur (the typical case when only **R1** needs to be met). The full set of experiments are listed below:

1. Anomalous Day (a sudden change occurs). P-MARL evaluated under:
    (a) no agent-contributed non-stationarity, first in a case when the sudden change is not matched, and second in a case when the sudden change is matched (Section 5.2.3.1.1).
    (b) agent-contributed non-stationarity with simultaneous actions, first in a case when the sudden change is not matched, and second in a case when the sudden change is matched (Section 5.2.3.1.2).
    (c) agent-contributed non-stationarity with sequential actions, first in a case when the sudden change is not matched, and second in a case when the sudden change is matched (Section 5.2.3.1.3).
2. Normal Day (no sudden changes occur). P-MARL evaluated under:
    (a) no agent-contributed non-stationarity (Section 5.2.3.2.1).
    (b) agent-contributed non-stationarity with simultaneous actions (Section 5.2.3.2.2).
    (c) agent-contributed non-stationarity with sequential actions (Section 5.2.3.2.3).

All 90 EV agents begin the training sessions with a state of charge (SOC) of 0% (i.e., empty battery). Their purpose is to sufficiently charge for the next day's trip; each vehicle has to achieve a SOC that allows for 50 kms of travelling. At the end of each training day, the EVs' SOC is again reset to 0. Agents are trained offline over a period of 100 days, in a simulation of the environment. The simulation is based on a prediction of the environment's future behaviour. After the training session, agents are tested on the actual environment. Each set of experiments is performed 10 times and averaged.

5.2.1 Forecasting Metrics

Other work presents results in the non-normalised RMSE (e.g., (Espinoza et al., 2005; Llanos et al., 2012; Tidemann et al., 2013)), but NRMSE is a better way to compare forecasting accuracy, as the power demand scale between tested cases does not have to be the same when comparing accuracy rates (Wijaya et al., 2014). Therefore, even particularly high power demands are comparable with other evaluated demands, when normalised. Additionally, NRMSE poses a more realistic estimation of errors when compared to MAPE[1] at very small scale, the latter being very strict when it comes to deflections of the forecast from the true load for low demand periods (i.e., low morning peaks) while not as strict at periods of high demand (evening peak). This is evidenced later in this section by the high discrepancy in MAPE (7.59% to 12.04%) when compared to the equivalent values of NRMSE (7.31% to 7.37%, respectively) when comparing whole days from Table 5.3 versus the evening periods in Table 5.4. As a result, NRMSE was chosen as a more evenly distributed way of evaluating forecasting accuracy in very small scale and therefore is employed as the main measure of forecasting accuracy in the evaluation section.

[1]This is another widely used demand forecasting method, as shown in the survey by (Hernandez et al., 2014).

5.2.2 Demand Forecasting Evaluation

This section presents the demand forecasting results achieved in the small scale power network represented by a residential neighbourhood. This demand contains only the baseload, and does not involve the demand of the 90 EVs. The forecast is required by the EV agents since it is used as a basis for learning in the offline environment simulation. The forecasting accuracy of the prediction component is first evaluated in Section 5.2.2.1, while the performance of the complementary pattern-change detection and matching component is evaluated in Section 5.2.2.2.

5.2.2.1 Prediction Component Evaluation

The hybrid approach to forecasting proposed in this thesis is tested on two scenarios of different scales, under non-stationary energy demand patterns. The first one covers 90 houses, while the second one covers 230 houses. These represent the simulation of a rural and urban case, respectively, encompassing the demand of a single transformer in different population density conditions. The model chosen for the urban case was a 630 kVA transformer that supplies roughly 230 houses (Marinescu et al., 2013).

In the dataset used for training and testing, both the individual forecasting methods and the resulted hybrid rely on recorded information from a smart-meter field trial held by the Commission of Energy Regulation (CER) in Ireland[2]. Even though in the recordings there are data from users with different tariff plans and pricing systems, there is also a set of control users where power demand was not affected by electricity price changes over the day. A subset of households was chosen from this control set, since the included households did not benefit from any demand side management programmes and therefore demand restrictions did not apply. Additionally, in order to fulfil the requirements of weather information in the neural networks, hourly recordings from

[2]More details about this are available in Section 4.1.

OGIMET were employed comprising temperature and humidity information (OGIMET, 2015). Details about this data are provided in Section 4.2.1.4.

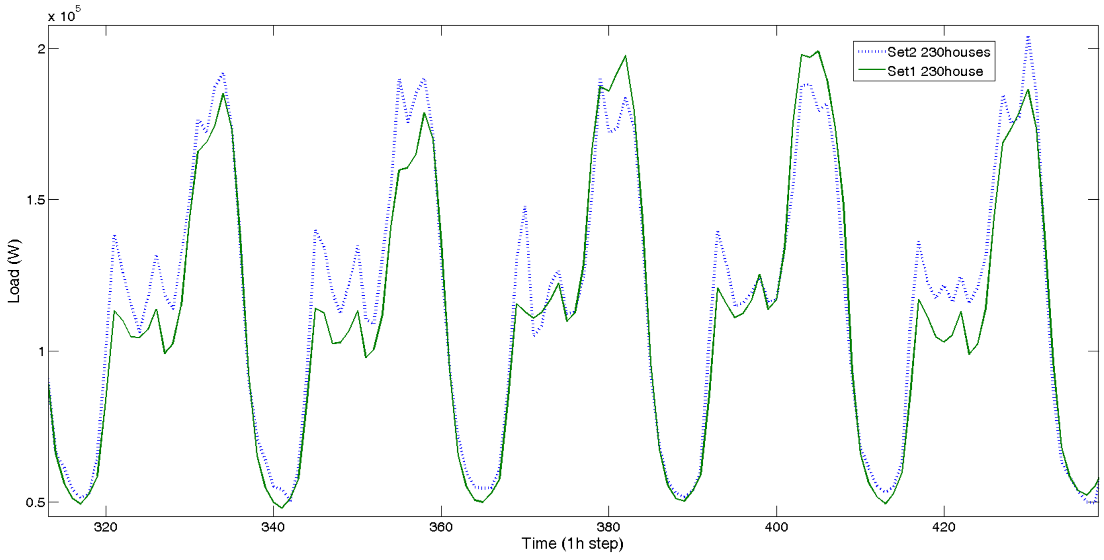


Fig. 5.2: Set Differences

The half-hourly smart-meter information was down-sampled through averaging to 24 recording per day, and further normalised. The normalizing procedure is a requirement for neural networks input, and was also applied in the case of weather information.

Training and forecasting is performed on the same group of houses, since each cluster of houses has its own particular pattern. As a result, training on one set and forecasting on another one leads to significant losses in accuracy. For comparison purposes, power demand shapes for two 230 houses clusters are presented in Fig. 5.2. While the second set (dashed line) presents a higher demand than the second set (solid line) during the morning peaks, the two shapes have similar amplitudes for the evening peaks, although with different variations at the peak points.

The concept behind the algorithm used for the hybrid is introduced in Algorithm 2, described in Section 4.2.1.5. The preliminary study for this thesis, leading to this particular hybrid solution, was carried out in (Marinescu et al., 2014b).

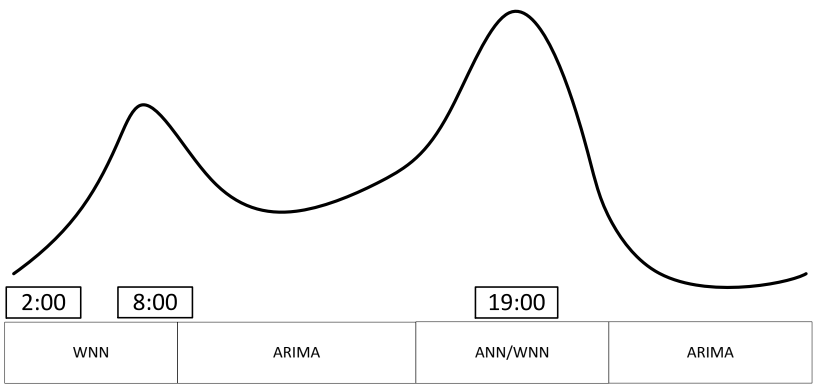
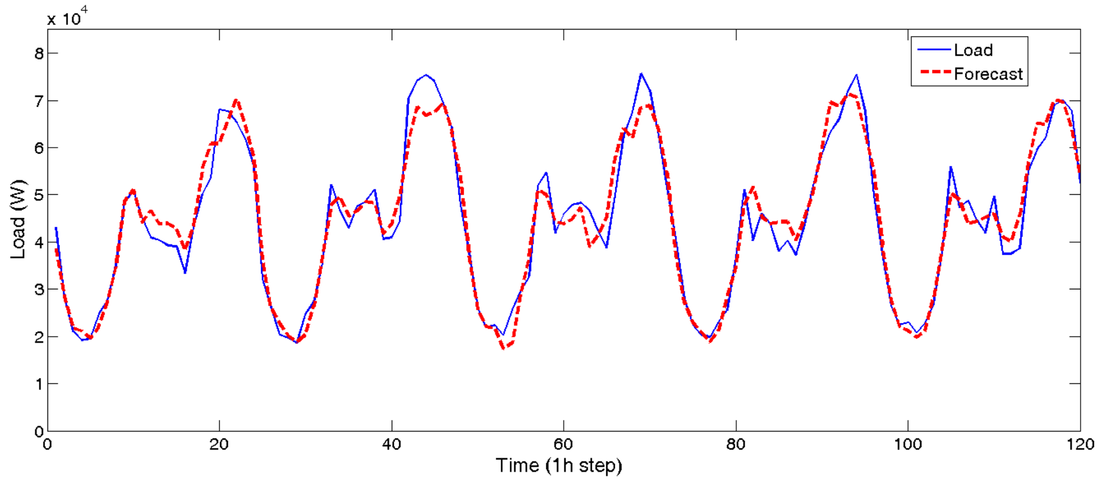


Fig. 5.3: Methods Combination over 24 hours

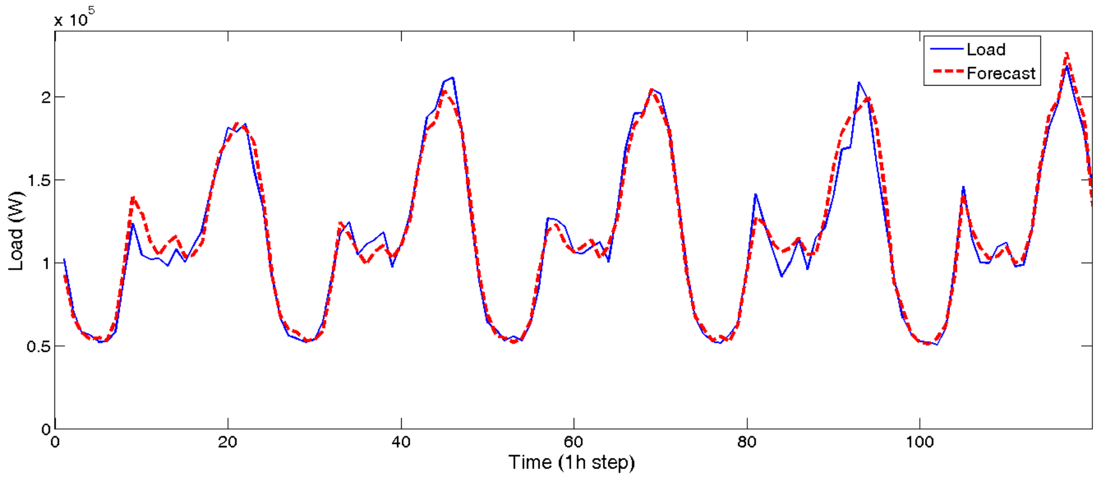
The hybrid solution is pictured in Fig. 5.3. The hybrid layout is the following: night time (02:00) until and including the morning peak (07:00-08:00) is represented by the WNN; the time between the morning peak and the evening peak (08:00-17:00) is modelled by ARIMA; the evening peak time (17:00-21:00), with a longer time-span than the morning peak and a higher demand, is modelled by ANNs and WNNs, while the final part of the day (until 02:00) is again modelled by ARIMA. These results are also presented in a preliminary study for this thesis (Marinescu et al., 2014b). The shape represented in Fig. 5.3 is a rough approximation of the daily demand.

The hybrid has devised in both scales a dominant ARIMA component, with a complementary ANN/WNN component in the evening peak. WNN has shown very good results in the 90 houses scenario, where noise is eliminated and therefore accuracy is higher when compared to the more traditional ANN. The decision process for the morning peak estimation switches between ARIMA, ANN and WNN. For the smaller scale scenario, both ANN and WNN have a comparatively higher contribution to the morning peak estimation.

Fig. 5.4 presents the forecast for both scenarios over one week, using the hybrid method. A better estimation is achieved in the 230 houses scenario, where the overall demand behaviour of the users is smoother because the higher number of consumers makes it easier to predict. The more chaotic behaviour in smaller scale can be attributed to the considerable impact of individual users.

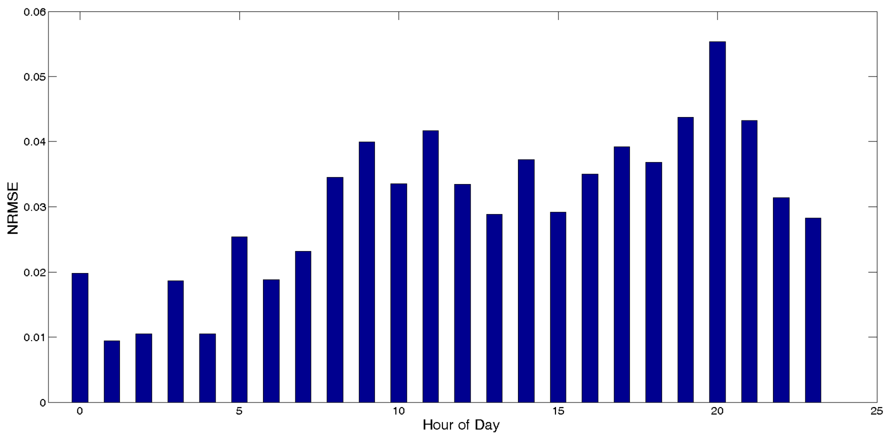


(a) 90 houses

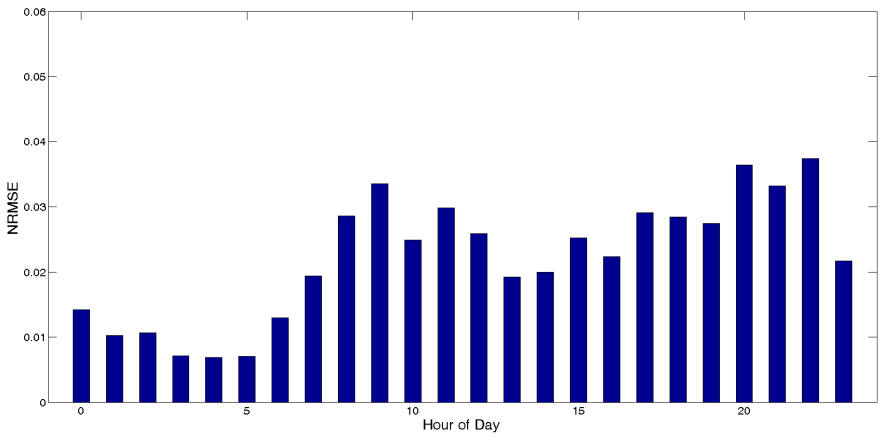


(b) 230 houses

Fig. 5.4: Hybrid Forecasting over One Week



(a) 90 houses



(b) 230 houses

Fig. 5.5: NRMSE over Two Weeks

5.2.2.1.1 Forecasting Results The results illustrated in Fig. 5.5 show the forecasting accuracy over each hour of the day for both scenarios. The highest NRMSE errors are observed around the two critical points, morning and evening peaks, although the forecasting differences between hours are smaller on the larger scale scenario when compared to the 90 houses scenario. Noticeably, the best prediction is found around night time. For comparison purposes between each separate method, twenty consecutive weekdays were chosen during August/September 2010. As shown in Table 5.1, the hybrid outperforms all individual methods, with improvements ranging from ≈13% in the ARIMA case, and up to ≈20% in the WNN case. The result achieved on average over one day was 2.39% for the 230 houses scenario, with results being between 1.66%-3.25% NRMSE with 95% confidence. Although the differences in overall accuracy between the hybrid and ARIMA method might not seem significant, these are due to the increased accuracy over the two short intervals of the critical peaks, which are of particular interest as the highest demand occurs during this time.

Table 5.1: Scenario comparison

Method	**NRMSE (%)**	
	90 houses	**230 houses**
ANN	3.93	2.89
WNN	3.89	2.98
ARIMA	3.62	2.74
Hybrid	3.23	2.39

5.2.2.1.2 Analysis The accuracy achieved by the hybrid goes as low as 1.6% NRMSE for particular days, while on average reaching 3.23% NRMSE in the 90 houses scenario and 2.39% NRMSE in the 230 houses scenario. The results obtained suggest that the hybrid method is a potential forecasting approach for very small scale systems such as microgrids or VPPs.

5.2.2.2 Pattern Change Detection and Matching Component Evaluation

The methods forecasting tests were divided into two separate categories, where the accuracy of prediction was tested during normal days (the vast majority of the days from the sampled data) and during the special case of anomalous days. The previous component deals exclusively with normal days, and as such tests were performed only over such intervals.

Table 5.2: Prediction Accuracy over 20 Consecutive Normal Days (Aug/Sept 2010)

Method	**RMSE (kW)**	**NRMSE (%)**	**MAPE (%)**
ANN	7.81	2.89	4.83
Hybrid	6.94	2.39	4.55

When the ANN prediction algorithm was evaluated over normal days in the 230 houses scenario, a forecasting accuracy that was close to the one of the best performing hybrid approach (within 0.5% NRMSE) was achieved. Over a testing period of 20 consecutive weekdays (months of August and September), without any anomalous days involved, an accuracy of 2.89% NRMSE (4.83% in Mean Absolute Percentage Error (MAPE)) was obtained. This is shown in Table 5.2. A sample of ANN predictions over 4 consecutive days are pictured in Fig. 5.6. However, the purpose of this component is to employ the ANN approach only for anomalous days, when triggering time-critical re-prediction. This is done to avoid the more intensive computations for the hybrid approach when given short notice, such as the few seconds/minutes available in the middle of the day when unexpected changes occur.

Possible anomalies in the dataset start to be detected from 8:00 onwards by the pattern-change detection component. Fig. 5.7 shows the false positive anomalies reported by the SOM, pictured with a green line. The blue line marks the number of actual anomalies (true positives) detected, as determined by the SOM at the end of the day. The total number of days are represented on the Y axis on the left. The red line shows the total accuracy rate, which is the ratio between the possible anomalies detected which turn out to be real anomalies over the total number of proposed anomalies,

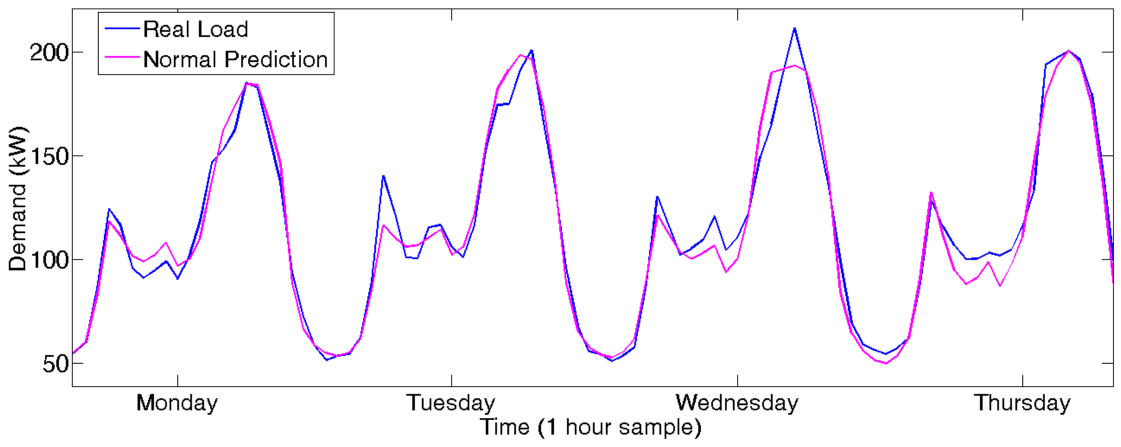


Fig. 5.6: Prediction over 4 consecutive days (Aug 2010)

including false positives. Its values are represented on the Y axis of Fig. 5.7, in percentages, on the right side. In the early morning period (midnight-8AM) a lot of anomalous days (about 10 out of 50) are detected quite accurately (40% accuracy), due to their particularly different demand patterns over the beginning of the day. However, there is also a significant number of false positives. As the time progresses towards midday, many more false positives appear, due to the seasonality factor, and the overall accuracy decreases because of the inclusion of these false detections.

The total number of false positives detected starts to significantly decrease at 11:00, and at 12:30 when the blue line intersects the green line, the total number of false positives is the same as the number of true positives. At this point there is 50% accuracy in anomaly detection, which was the initial target. The SOM needed only 4.5 hours to reach this level of accuracy, starting from 7:30. This is because during night time the demand tends to be less significant, thus the small variations in this interval do not cause any anomalies. While this SOM method sacrifices accuracy for the morning peak, it prepares the forecasting mechanism for the most critical part of the day, the evening peak, where the highest demand occurs.

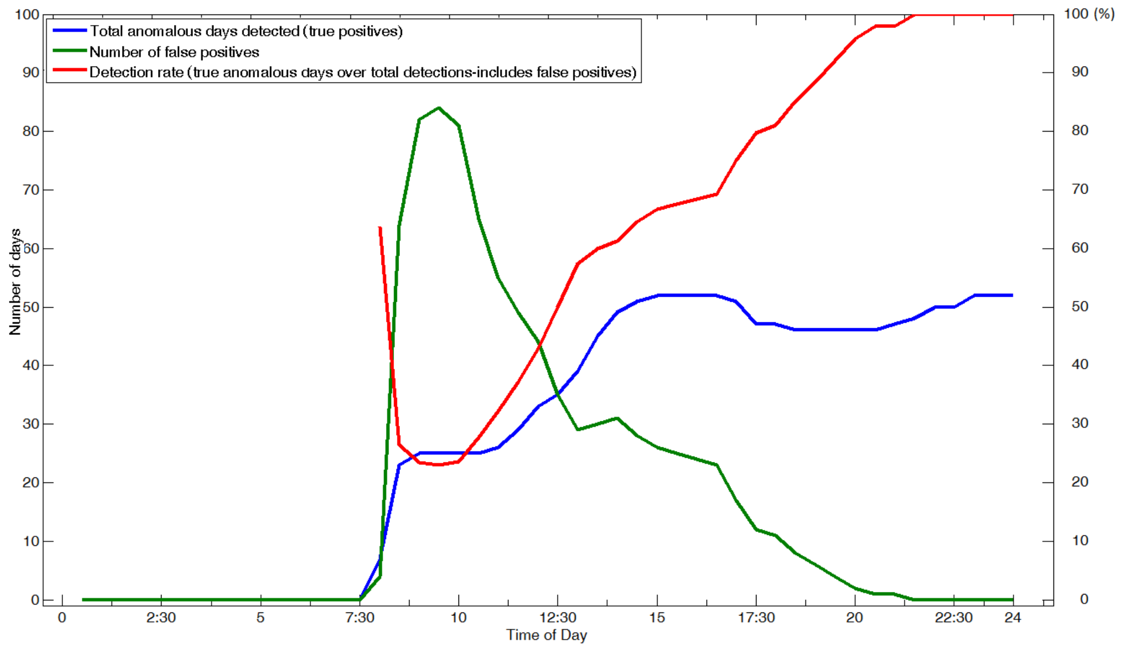


Fig. 5.7: Detection rates

Under specific conditions, the re-prediction stage described in Section 4.2.2.3 can be accomplished just by employing a SOM anomaly detection algorithm that overlaps values on top of the average shape from midnight up to 12:30, in order to trigger the re-prediction in case it detects anomalies in demand. Even though at this stage an anomaly is detected with 50% accuracy, the SOM requires another 2 hours of demand to properly match the type of anomaly, after comparing it with previously encountered anomalous patterns. Another observation is that, in these samples, the afternoon/evening period (14:30-23:59) accounts for more than half of the total energy consumption, more precisely 55%. Given this, at 14:30 there is a 65% detection accuracy rate, with 98% of the true positives detected, along with a few more false positives.

At 14:30 the components look for a similarly encountered demand in the database of anomalous days. Once the closest match is found, the demand obtained so far and the rest of the demand belonging to the closest match are fed into the re-prediction system.

The pattern change detection mechanism requires only 5 hours from the beginning of the day (i.e., 07:30) in order to be able to detect anomalous days with a 50% accuracy rate, a value regarded as important enough to consider the re-evaluation of prediction for the day. By detecting an anomalous day in the morning, the microgrid can be prepared in advance for the change from expected demand. This is particularly useful to address the critical evening peak, which is by far the period of highest power demand during the day.

5.2.2.2.1 Forecasting Results In the case of anomalous days, three kinds of prediction are evaluated: the initial ANN presented in Section 4.2.2.3; an ANN that generates a reprediction once a change is detected (without change-matching, just based on the information up to the change), which is further referred to as ANN+REP; and the ANN that generates a re-prediction once a change is not only detected but also matched, which is further referred to as ANN+SOM+REP. The detailed results for these three methods are presented in Table 5.3, where the prediction accuracies during normal days versus anomalous days are shown. The particular days forecasted here occur during the same time of the year, specifically the last two months of the year 2010.

Table 5.3: Prediction Error Normal vs. Anomalous Days (Nov/Dec 2010)

Method	**Normal Days**		**Anomalous Days**	
	NRMSE (%)	MAPE (%)	NRMSE (%)	MAPE (%)
ANN	3.03	5.03	7.37	12.04
ANN+REP	2.83	4.50	5.41	6.49
ANN+SOM+REP	2.81	4.36	3.63	4.71

Fig. 5.8 visualises the forecasting accuracy obtained for each day, between the 10th of November and 29th of December 2010. Note the two periods of anomalies, one in the

beginning of December, which is possibly due to a very cold North Atlantic front which caused increased energy usage. The second period of anomalies is due to the Christmas holidays and New Year's eve. During the first anomalous period, the first five days produced high errors in forecasting of up to 11% NRMSE in the standalone prediction. Standalone re-prediction (ANN+REP) reduces this accuracy error after the anomaly detection, while the SOM enhanced re-prediction (ANN+SOM+REP) minimizes it to similar levels as the ones of normal days. After five weekdays, the normal prediction algorithm (ANN) adjusts itself by using the input of an anomalous day for the prediction of the 6th day (the actual demand from one week ago). This can be noticed by the reduced forecasting errors of the standalone prediction ANN on Monday (the 6th of December) and Tuesday (the 7th), in Fig. 5.8.

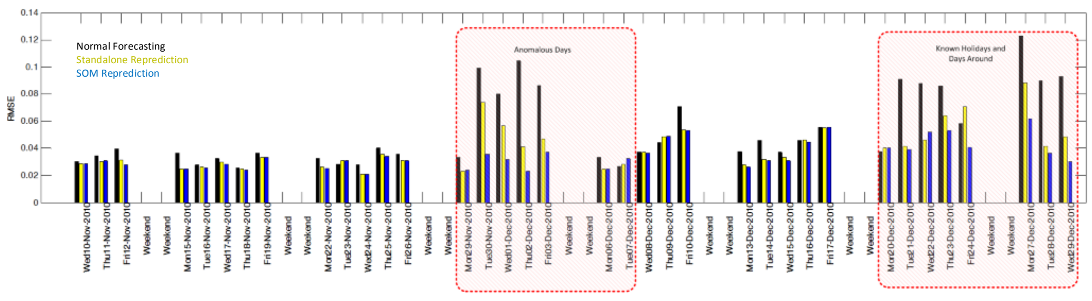


Fig. 5.8: Anomalous Month

The anomalous days' evaluation was performed over one week (5 consecutive weekdays) occurring at the end of November and beginning of December. The results are illustrated in Table 5.3. The simple ANN based prediction algorithm shows significant decreases in accuracy over that time, with values of 7.37% NRMSE, 12.04% MAPE, and 22.18 kW RMSE. These are, however, still an improvement over the results from the closest related work from the state of the art (Tidemann et al., 2013). While this might be considered somewhat satisfactory, compared to results over normal days the error is large enough to consider the involvement of re-prediction through pattern matching techniques.

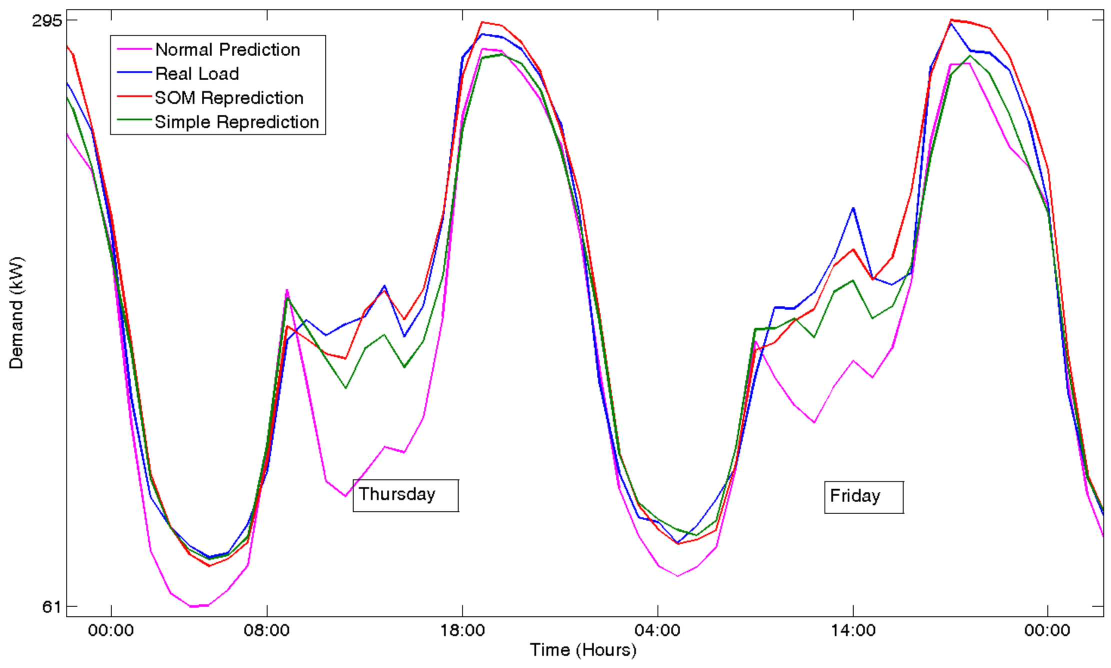


Fig. 5.9: Predictions Over Two Anomalous Days

Fig. 5.9 illustrates the prediction of two anomalous days. Here, the real demand is higher than initially expected. However, this is not the only anomalous behaviour for the two days presented. Typical features of the aggregated residential demand includes two peaks, a bigger one that occurs in the morning and a smaller one in the early afternoon, before the very high evening peak. The morning peak is higher in the normal days, but in this particular case there is a higher demand during the early afternoon peak, particularly in the second day (Friday) presented in Fig. 5.9.

The pattern change detection and matching mechanism triggers re-prediction over the five anomalous days, which brings down the RMSE to 10.89 kW (3.63% NRMSE or 4.71% MAPE), from a previous value of 22.18 RMSE (7.37% NRMSE or 12.04% MAPE). More details can be seen in Table 5.3, which also includes the results obtained by ANN+REP, the prediction which is accomplished without the involvement of pattern matching techniques.

The more relevant testing period occurrs after the 14:30 interval, which is the moment when re-prediction is triggered. The results obtained are shown in Table 5.4. During this specific interval (14:30-23:59), the pure ANN prediction algorithm provides an accuracy of 9.68% MAPE (8.28% NRMSE) over the anomalous days. ANN+REP (re-prediction with pattern change detection enabled but without the pattern matching enhancements) achieved 7.59% MAPE (7.55% NRMSE) in the same given period, while the SOM enhanced re-prediction, ANN+SOM+REP, reached 4.99% MAPE (5.00% NRMSE). The demand for two anomalous days, together with their predictions, are presented in Fig. 5.9.

Table 5.4: Prediction Error Anomalous Days: 14:30-23:59 Interval (Nov/Dec 2010)

Method	**Anomalous Days**	
	NRMSE (%)	MAPE (%)
ANN	8.02	9.68
ANN+REP	7.31	7.59
ANN+SOM+REP	4.84	4.99

While ANN+SOM+REP shows the best accuracy out of the three evaluated forecasting methods (ANN, ANN+REP, ANN+SOM+REP), the errors that occur in its forecasts also tend to be overestimates of the evening peak when compared to the other methods, which underestimate it. This is important, as it is relevant in the particular cases where the transformer's capacity limits are reached.

The power demand forecasted in this evaluation ranges between 40 kW and 340 kW (depending on the season), which is close both in demand patterns and power usage to the demand at distribution substation level presented in (Tidemann et al., 2013). As previously mentioned, it is difficult to compare results with previous work because of the differences in scale. However, for illustration purposes, note that (Tidemann et al., 2013) have at distribution substation level a demand which is between 100 and 300 kW. These values are based on the evaluation period presented in their graphs. When the RMSE is normalized based on these values, it results in approximately 10.72% NRMSE, with the best result of 21.43 kW RMSE obtained through the auto-regressive (AR) model.

For the previously evaluated months for normal days, between August and September, the demand is actually between 40 kW and 220 kW. The neural network forecasting algorithm provides a 7.81 kW RMSE over a period of 4 consecutive weeks (20 weekdays), in comparison to the best result of 21.43 kW RMSE from (Tidemann et al., 2013) (obtained with AR).

Another interesting test with regard to power demand level comparisons were forecasting evaluations performed during 2 consecutive weeks (10 weekdays) in November 2010, which is a highly variable period, close to holidays. Here the power demands range between 50 kW and 340 kW. These demand values are close to the ones in the previously mentioned work (Tidemann et al., 2013). The ANN forecasting algorithm developed in this thesis provided a RMSE of **9.11** kW. This accuracy was reached despite the slightly wider range (subject to a higher RMSE) and the high variability in the given period. The variability of demand can be explained by the fact that after this period of 10 consecutive weekdays, there are several days marked as anomalous by the SOM, with the weekdays in question being close to or on the borderline of anomalous days. For other methods of forecasting error measurements, the results were of 5.03% MAPE. This represents an insignificant decrease in accuracy when compared to the results obtained over the more settled summer period, which were presented in Table 5.2.

5.2.2.2.2 Analysis The results obtained by the dynamic forecasting method are better than state of the art approaches in small scale. Combined with the hybrid forecasting method presented in Section 5.2.2.1, prediction for energy demand for the residential neighbourhood achieves accuracy within approximately 5% MAPE. This is shown in Table 5.5, which presents the accuracy for forecasting for both normal and anomalous days. A normal summer August month was predicted from 2010. For the anomalous case, the winter month of December was predicted, which has mostly anomalous days. The table shows both the average forecasting results, and the 95% confidence interval. The last two columns show the prediction accuracy achieved by the hybrid vs. the prediction accuracy achieved through the dynamic reprediction solution presented in this section. However, the final combined prediction solution comprises only the dynamic reprediction component (achieved through pattern-change detection and matching) during anomalous days, and not the hybrid prediction.

Table 5.5: Prediction Accuracy: Normal vs. Anomalous Days

	Forecasting Accuray (MAPE)		
	Summer Month (Normal)	**Winter Month** (Anomalous)	
		Hybrid Prediction	**Dynamic Reprediction**
Average	5.01%	8.84%	5.53%
95% Conf. Inter.	3.49%-6.51%	6.40%-11.29%	3.58%-7.49%

To the author's knowledge, this is the only forecasting approach in small scale that deals with normal days as well as anomalous days without classification on a predetermined basis, thus enabling on-the-fly anomaly detection, pattern matching, and reprediction techniques in case of unanticipated anomalous days occurring. The electrical demand forecasting results achieved are very good at residential transformer level, which in this case is considered to be of up to 350 kW.

Classification techniques can be improved to classify demands based on seasonality and day of the week patterns. For such improvements to be implemented, a larger dataset is required, one which spans several years, unlike the present case which was limited to 17 months and therefore did not allow for much flexibility with regard to the SOM classes.

5.2.3 MAS Performance Evaluation

The load from the same day of the previous week and three different load predictions are provided in each set of experiments as input to P-MARL: simple prediction (no pattern matching), anomaly-matching prediction (SOM Prediction, entitled ANN+SOM+SOM in Section 5.2.2.2), and perfect prediction (i.e., the estimate is the same as the actual environment behaviour). The load of the previous week is used in a traditional MARL implementation, which is based only on previously encountered situations. Perfect prediction represents an idealised situation, where the environment induced non-stationarity is removed from the MARL problem. This latter case is used only for comparison purposes as an additional benchmark, to differentiate between the levels of performance achievable by the first two prediction types. Additional tests of statistical significance were performed through two-tailed t-tests. Results where p-values are lower than 0.05 were considered to be statistically significant.

5.2.3.1 Evaluation on Anomalous Demand

The evening and early morning periods of a particularly anomalous day, together with its predictions and the same day of the previous week, are illustrated in Fig. 5.10. The anomalous day occurs in December 2010; a comparison with other winter days reveals a higher amount of energy usage in the anomalous day. This anomaly occurs because of an

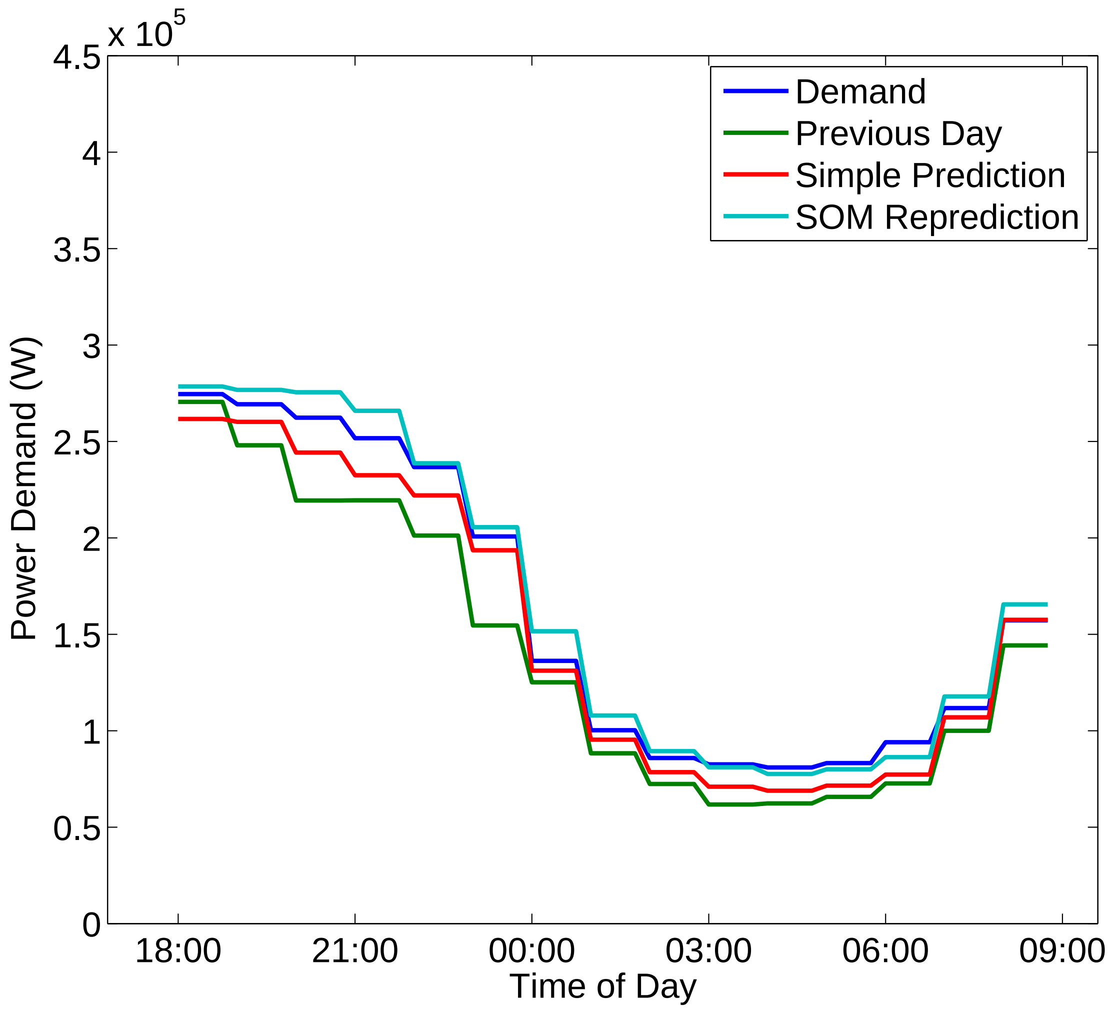


Fig. 5.10: Predictions of an Anomalous Day

unexpected cold front advancing from the North Atlantic, which caused increased power usage. The previous day and simple prediction underestimate the actual demand, while the SOM reprediction overestimates the demand. In this case, SOM reprediction takes a more conservative measure. The previous day has a forecasting error of 14.87% MAPE, simple prediction achieves a forecasting error of 7.65% MAPE, while SOM reprediction achieves 4.66% MAPE.

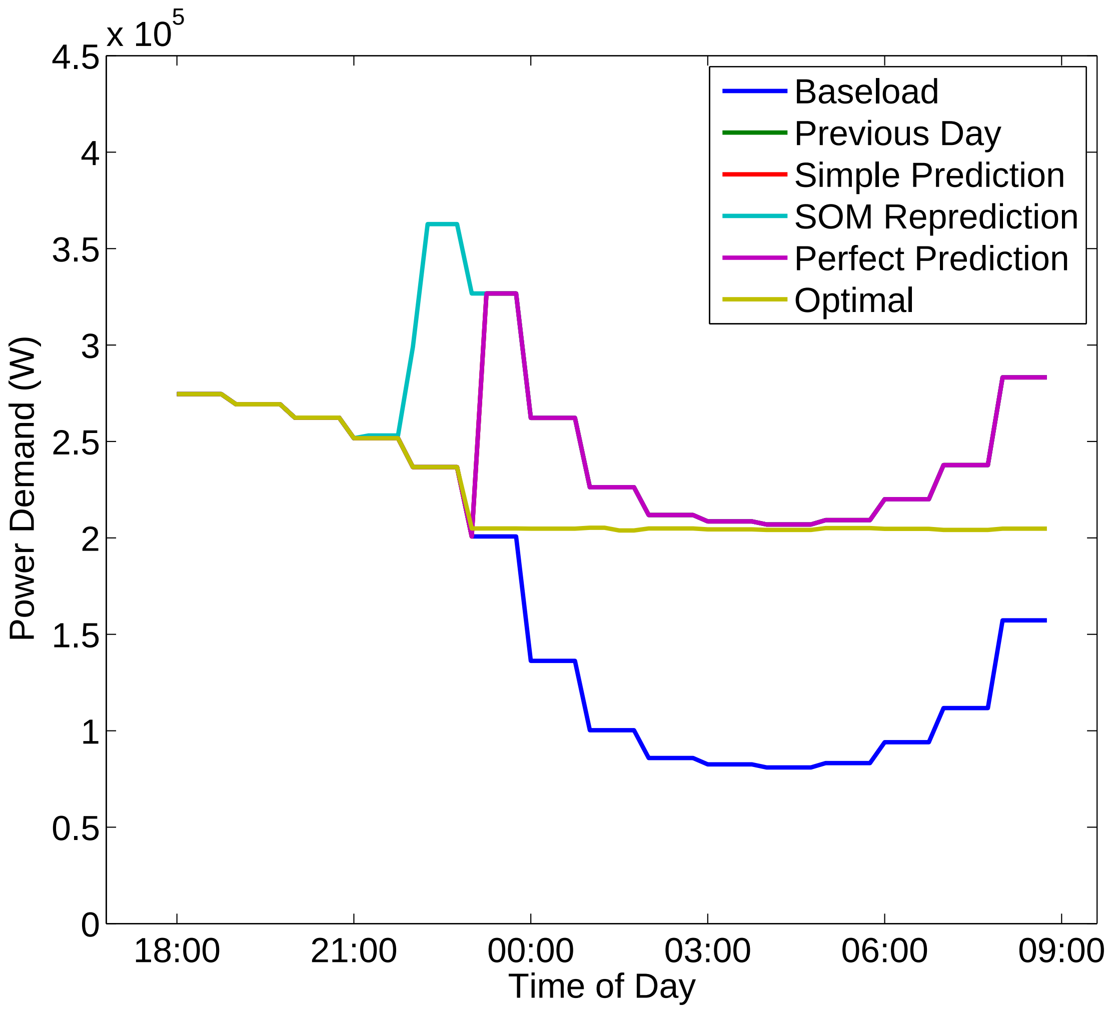


Fig. 5.11: EVs Charging Separately

5.2.3.1.1 Case 1: EVs Charging Separately An agent only has a localized view of the environment. As such, an agent can observe only the effect of its own action on the environment, and not the effect of other agents. An EV agent decides whether to charge or not based on the effect of its previous action on the environment. The results of such a constrained charging process are pictured in Fig. 5.11. The previous day's load, simple prediction and perfect prediction lead to the same demand, therefore these overlap in the figure. Their performance in terms of Pareto optimality is presented in

Table 5.6. The performance differences between simple prediction/perfect prediction and SOM reprediction are statistically significant, with a p-value of 1.21e-11 over the 10 different runs. The number of EVs achieving their individual targets is presented in Table 5.7. Since all vehicles achieve their targets there is no statistical difference between numbers of charged vehicles.

All four methods generate peaks in demand that can cause problems in the power network. SOM reprediction-based agents start charging earlier than others (22:00 as opposed to 23:00), creating the highest peak. This is because agents overestimate the demand, and therefore the real demand occurring at 22:00 is considered low enough for EVs to start charging. A positive aspect is that in this particular case, all EV agents will achieve and even surpass their target SOC, as can be observed in Table 5.7. However, this type of charging might cause transformer failures during peak demand which could potentially lead to blackouts.

5.2.3.1.2 Case 2: EVs Charging Simultaneously Agents take decisions simultaneously, and only after an agent takes an action can it observe the cumulative effect all agents' actions had over the environment. An agent decides whether to charge or not based on the current status of the aggregated load (comprising baseload and demand of EVs currently charging). The results of this type of charging interaction are pictured in Fig. 5.12.

EV agents trained both on the previous day's load and the simple prediction underestimate the demand. This leads agents to incorrectly assume that they are charging at periods of high demand, and therefore a few of them take conservative measures and stop charging at the next time-step. As soon as they stop charging, they realise that the demand becomes low enough for them to charge again. Therefore the whole group decides to charge at the next time-step, which brings them to the same situation as two

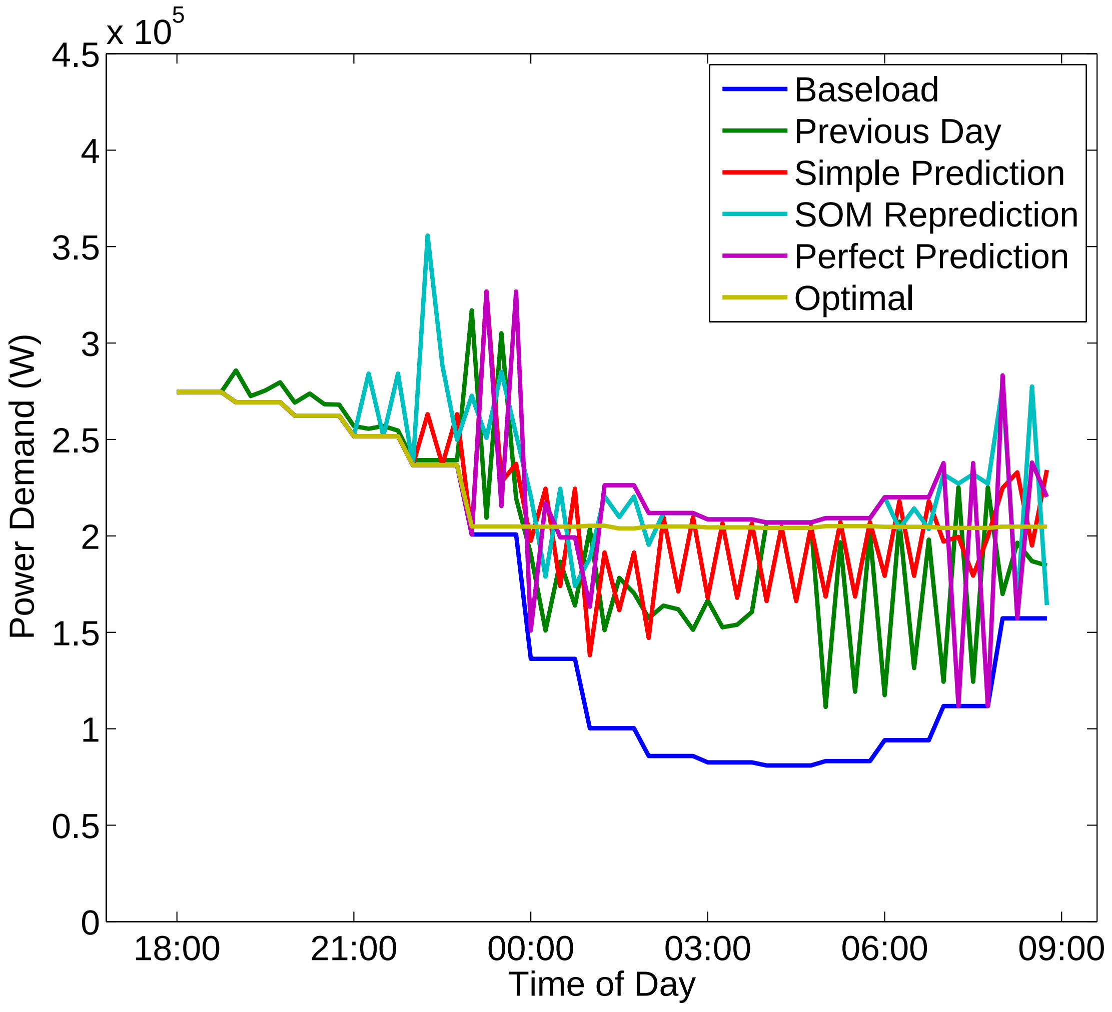


Fig. 5.12: EVs Charging Simultaneously

time-steps before. This alternating behaviour can be noticed in Fig. 5.12. Because of this, some agents do not achieve their target SOC, as shown in Table 5.7. On average, 4.3% of EVs will not have sufficient charge for the next day in the simple prediction situation, while a much larger 83.6% of EVs will fail when training based on previous load. Only 16.4% of EVs manage to charge in the traditional MARL case because the previous day's load is considerably lower than the actual load, which means agents avoid charging during a much higher number of time-steps when compared to their training episodes.

The difference in Pareto performance between perfect prediction and SOM reprediction runs is statistically significant, with a p-value of 0.0058, while simple prediction and SOM reprediction runs failed to show any significant differences, with a p-value of 0.10. In terms of percentage of vehicle charged, the difference was noticed to be statistically different between simple prediction and the SOM reprediction/perfect prediction cases, with a p-value of 0.0002 (the p-value is the same because in the latter cases all vehicles end up charged).

EV agents trained on SOM reprediction present similar behaviour to the ones in case 1. They generate a peak in demand at 22:00, but immediately realise this and most of them back off. Alternating behaviour can still be noticed in the next few hours, but the aggregated demand is smoothed from midnight onwards. This is because the night-time SOM estimate accurately matches the actual demand occurring. As a result all agents achieve their target SOC.

In the perfect prediction situation, agents will also generate peaks in demand, although their amplitude is lower than those generated by SOM reprediction trained EV agents. The bulk back-off behaviour can still be noticed before midnight and during morning hours, when demand only allows for a few agents to charge before aggregate demand surpasses the acceptable load levels. Also, in this situation all EV agents achieve their target SOC.

5.2.3.1.3 Case 3: EVs Charging Sequentially In this case, agents take decisions one after another, in turns. An EV agent can see the aggregated effect of previous agents on the power demand, before deciding whether to charge or not. Once a decision is taken, it becomes another agent's turn to decide. This following agent takes into account the now updated demand load before taking its own decision - and so forth. The resulting behaviour is illustrated in Fig. 5.13. In this case aggregate demand in all prediction

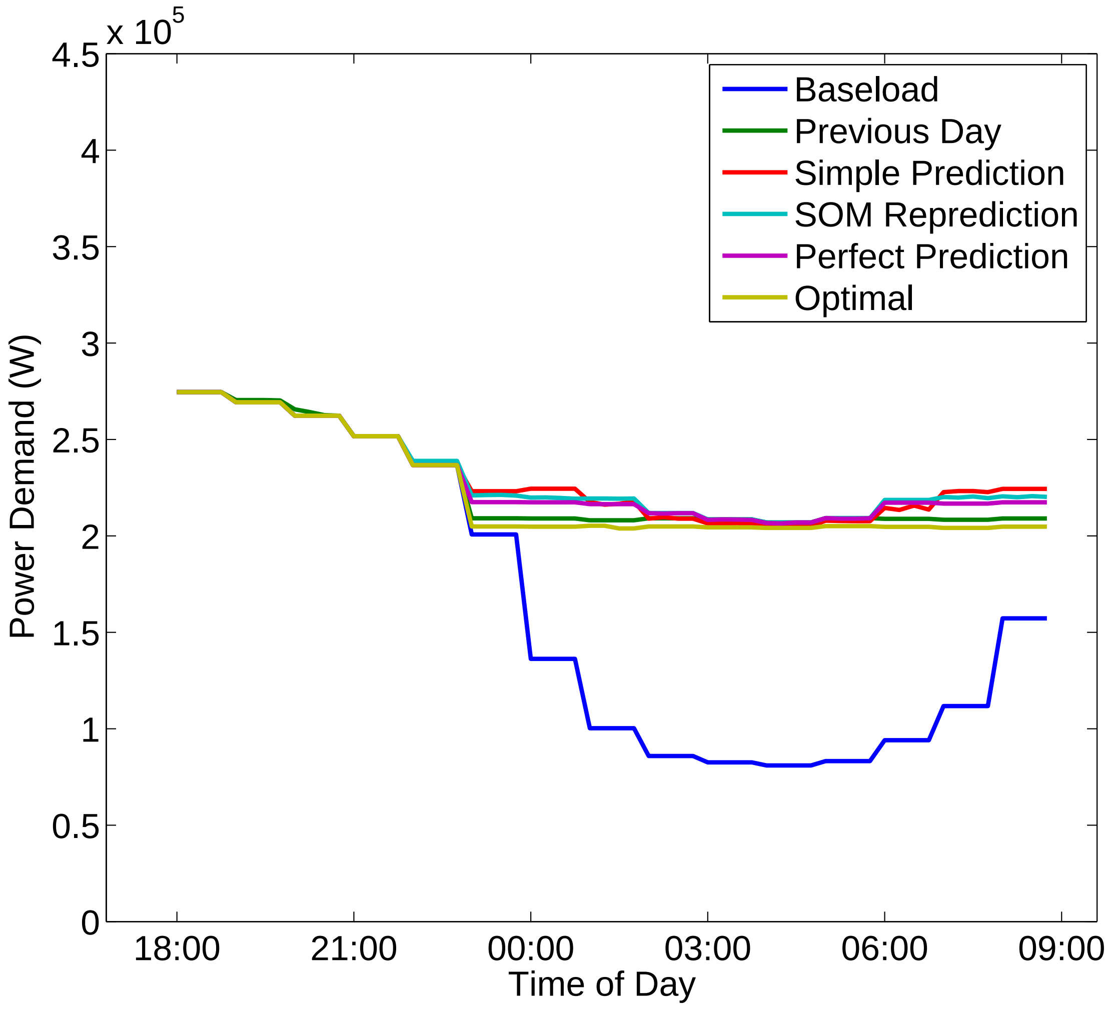


Fig. 5.13: EVs Charging Sequentially

situations is much closer to the optimal line, as can also be seen in Table 5.6.

EV agents trained on the previous day's load and simple prediction tend to charge less due to the underestimated demand, and several are left with insufficient SOC for the next day: 17.7% and 16.1% of EVs, respectively, will not achieve their target charge, as shown in Table 5.7. Even though the aggregate demand in this case is higher than the optimal aggregate, this is because other agents end up charging more than needed. When the sufficiently charged agents act before the insufficiently charged ones, this forces the latter agents to take decisions in the threshold region. Some agents avoid charging in such situations.

When EV agents are trained on the SOM reprediction, all of them achieve their target charge. The difference between these results and the ones obtained by using simple prediction are statistically significant, with a p-value of 1.25e-7. While performance in terms of Pareto optimality is not as good as in the simple prediction situation, this is because agents charge more on aggregate as the actual load is lower on average than during the training phase. The Pareto optimality differences between the two situations are very small, however still statistically significant, with a p-value of 0.0005.

EV agents trained on perfect prediction obtain best results both in terms of Pareto optimality and amount of agents reaching their targets (Table 5.6 and Table 5.7). This statement is also supported by a statistically significant difference between perfect prediction runs and SOM reprediction runs, with a p-value of 2e-9 with regard to Pareto optimality. Since agents train on the same estimate as the actual load, they are able to develop efficient charging strategies that help them reach target SOC at high Pareto optimality.

Table 5.6: Algorithms Efficiency in Different Prediction Conditions for Anomalous Demand

MARL Agent Interaction	**Pareto Performance**			
	MARL	**P-MARL**		
	Prev. Day	Simple Pred.	SOM Repred.	Perfect Pred.
1. Separate Actions	83.22%	83.22%	75.78%	83.22%
2. Simultaneous Actions	78.94%	86.95%	85.88%	86.71%
3. Sequential Actions	97.76%	94.24%	93.68%	95.20%

Table 5.7: Percentage of EVs Charged Before Departure for Anomalous Demand

MARL Agent Interaction	**Percentage of Charged EVs**			
	MARL	**P-MARL**		
	Prev. Day	Simple Pred.	SOM Repred.	Perfect Pred.
1. Separate Actions	100%	100%	100%	100%
2. Simultaneous Actions	16.4%	95.7%	100%	100%
3. Sequential Actions	82.3%	83.9%	100%	100%

5.2.3.1.4 Analysis The quality of prediction and an agent's level of interaction with the environment and other agents has a high impact on the aggregate P-MARL performance. When agents are not informed of the effect of other agents' decisions (case 1), the aggregate demand has negative consequences on the environment. The peak demand that results in this case forces the scheduling of additional power generation units, or even a blackout at neighbourhood level. This peak effect is noticeable regardless of the level of prediction accuracy.

Once agents have access to aggregate charging decisions (case 2), they notice their cumulative effect. When their aggregated behaviour leads to peaks, agents will back-off at the next time-step. The problem is that they can all resume charging two time-steps later as they register a low demand once all have backed off. This alternating behaviour is noticeable in particular in the simple prediction situation. Here, since the night-time prediction is underestimated, agents believe that their aggregate behaviour is generating new peaks. This is not the case in reality, and as a result some agents do not achieve their target charge because they expect time-slots of lower demand to follow. These results are pictured in Fig. 5.14.

Informing agents of the cumulative effect of other agents before making their own decision leads to a considerable performance improvement (case 3). Peak demand is

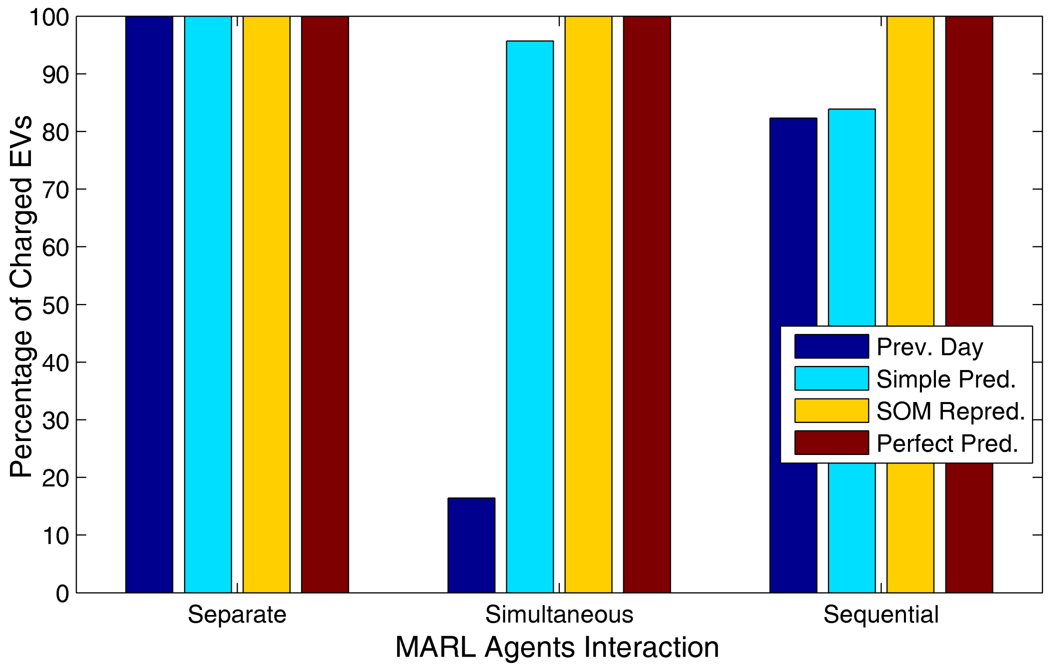


Fig. 5.14: Percentage of EVs Charged

avoided, and agents' aggregate performance closely matches the optimal line. In this case, prediction quality also affects the number of EVs achieving their target SOC. Some EVs trained on the previous day and simple prediction cases do not charge enough before their departure, despite the high Pareto efficiency achieved on aggregate.

An investigation of the 10 different runs reveals consistent results in case 1 and 3. In case 2 there is a larger amount of variability involved, in particular during midnight and morning hours. This behaviour can be visualized in the boxplot from Fig. 5.15. The time-slots with higher variability occur because of the effect agents have when acting in bulk. If all agents charge in these particular time-slots, their aggregate demand can result in peak demand, a situation in which all agents are penalised.

P-MARL provides improved performance over the traditional MARL approach with regard to the agents achieving their charging objectives, which is the primary objec-

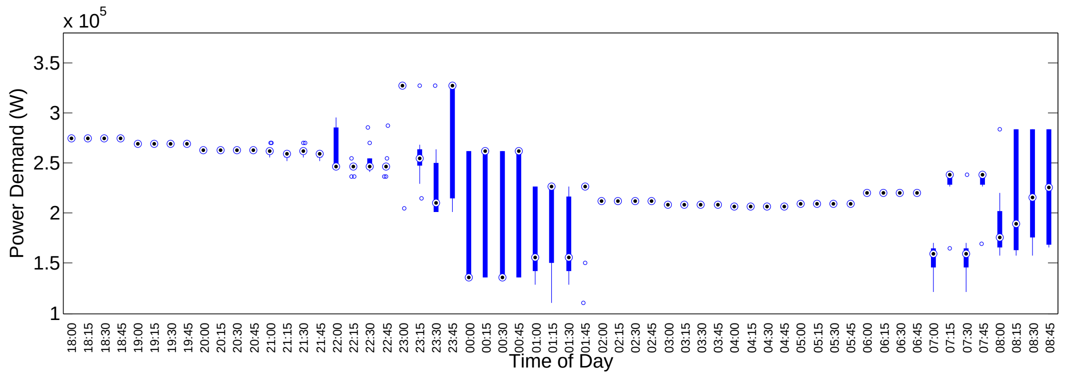


Fig. 5.15: Simultaneous Actions Statistics

tive of an EV agent. When SOM reprediction is provided and agents interaction exists (case 2 and 3), all P-MARL agents achieve their objectives, as opposed to only 16.4% of agents in simultaneous MARL and 82.3% in sequential MARL when only the initial prediction was used as a training basis. This shows that P-MARL not only detects changes (**R2a**) and matches them (**R2b**), but can also use this information to prepare agents for these upcoming changes (**R3**) through SOM reprediction. In terms of Pareto efficiency, P-MARL performance is similar to traditional MARL, except for the simultaneous actions case, where MARL performs considerably worse. This is because MARL agents consistently back-off as they wrongly assume that their charging is generating peaks in demand.

5.2.3.2 Evaluation on Normal Demand

The evening and early morning periods of a normal day, that day's predictions and the same day of the previous week, are illustrated together in Fig. 5.16. The normal day is selected from August 2010, a period of relatively settled days from the perspective of demand. As there are no significant changes from expected demand, the initial estimate provided by the prediction component through the hybrid holds throughout the whole

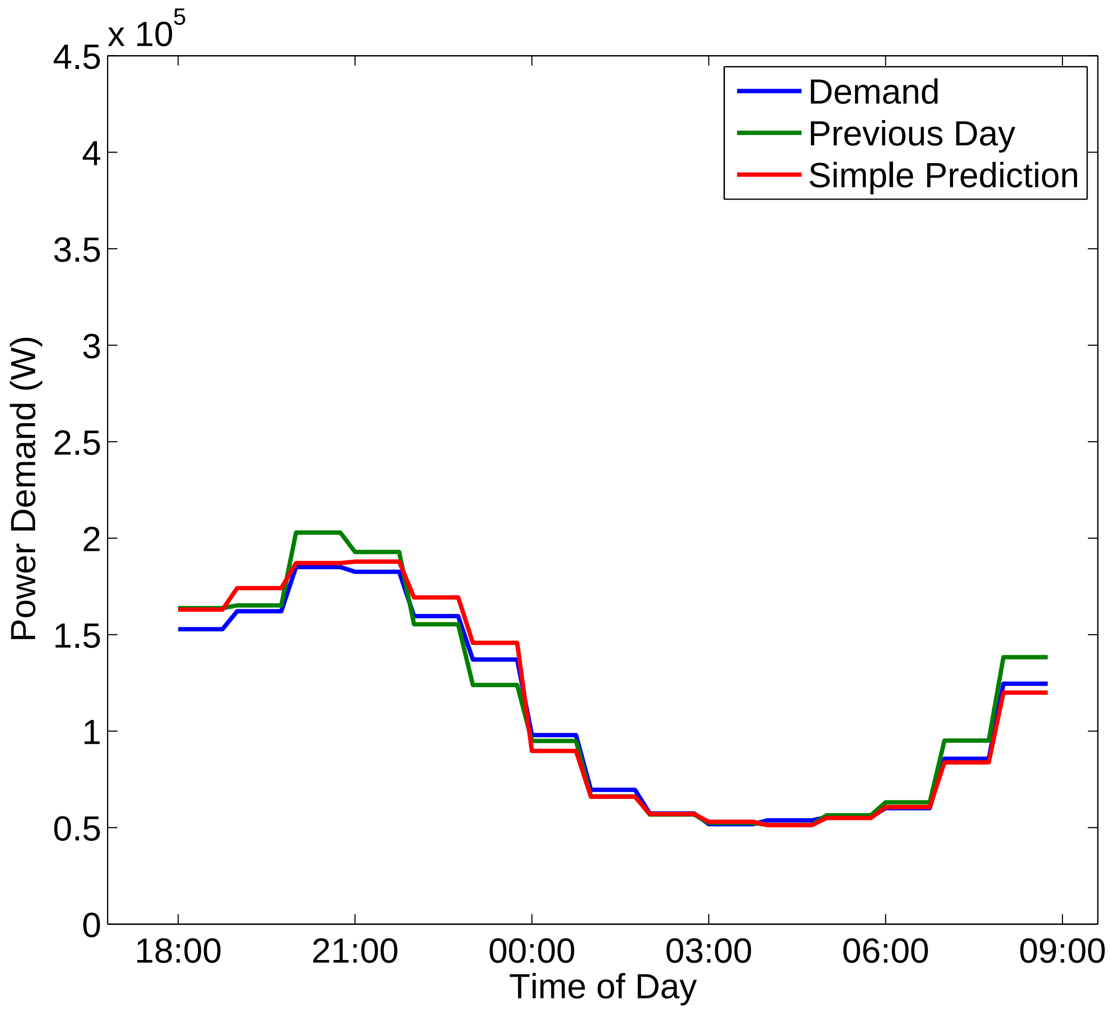


Fig. 5.16: Predictions for a Summer Day

prediction horizon, therefore there is no need for additional predictions based on pattern-change detection and matching. The prediction and the demand of the same day from the previous week tend to somewhat overestimate the demand occurring. The previous weekday has a forecasting error of 5.57% MAPE, while the hybrid prediction has a forecasting error of 4.33% MAPE.

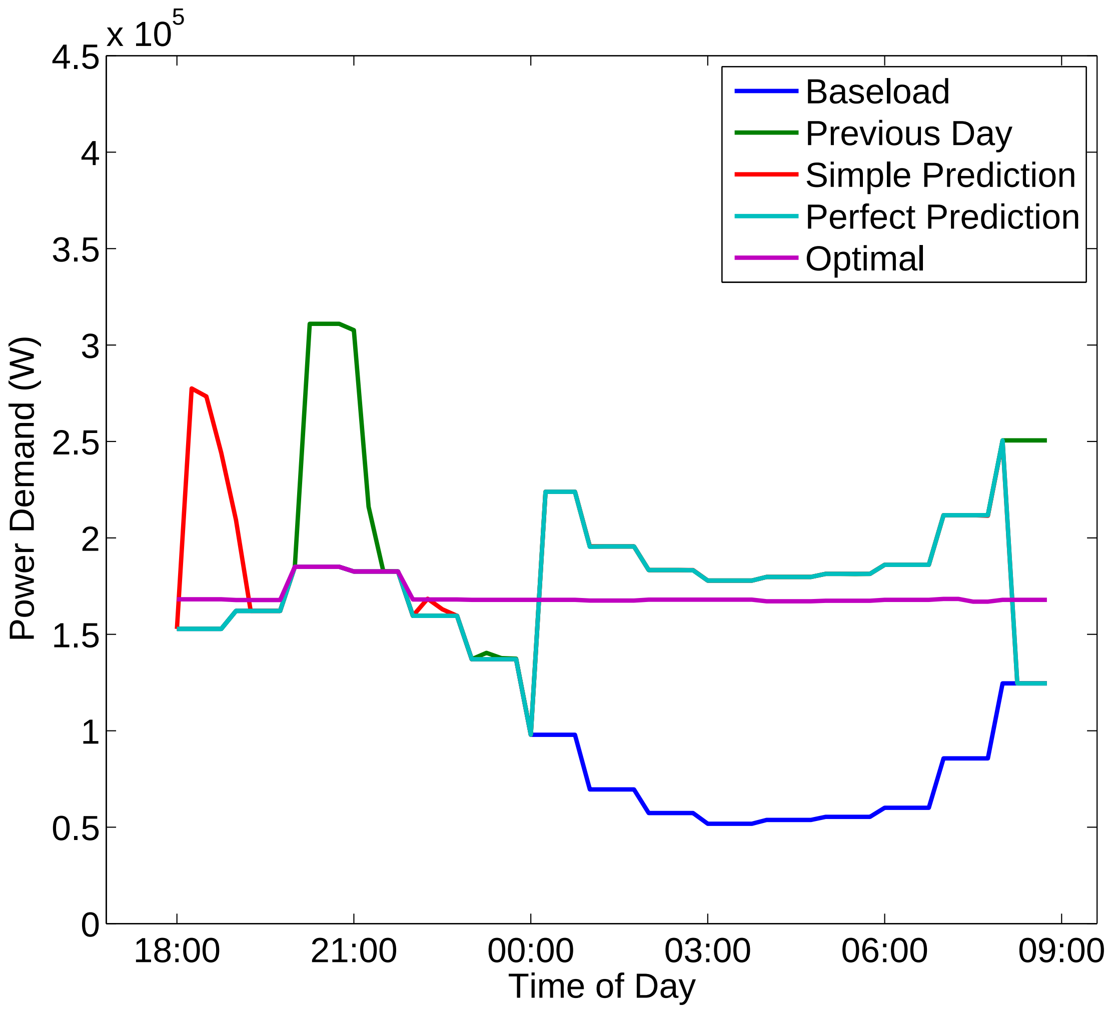


Fig. 5.17: EVs Charging Separately over a Normal Day

5.2.3.2.1 Case 1: EVs Charging Separately The result of EVs charging separately, where their learning stage and the results of their actions are not affected by other agents, is pictured in Fig. 5.17. Since the demand is overestimated by the two predictions (i.e., previous day and simple prediction), EV agents start charging earlier and thus created peaks in demand when these predictions were used as a training base. In the case when the previous day is employed as a training base, the peak is higher because the demand is overestimated more than in the hybrid prediction case. Compared to the anomalous day's case presented in Section 5.2.3.1, the optimal valley-filling based solution spreads the demand more evenly over the available charging period. Since the baseload is significantly lower than in the anomalous' day case (between 50-180 kW as opposed to 80-280 kW), with demand values between slots closer to each other, the EV agents need to spread their charging demand more. This results in a few EVs charging as soon as they arrive home. Even if they start charging at the beginning of the charging period, they back off later and completely avoid charging during the highest period of the evening's peak (20:00-22:00 in this case). In all the prediction cases, the EVs manage to achieve their target charge. This is shown in Table 5.9. All the EV agents manage to achieve their targets this time because of the overestimated demand, as opposed to the case presented in Section 5.2.3.1.1. However, this performance comes at a cost in terms of Pareto efficiency, illustrated in Table 5.8, as the values are slightly lower than those presented in Table 5.6. The Pareto performance difference between simple prediction and perfect prediction are statistically significant, with a p-value of 6.65e-14, while the performance difference between simple prediction and previous day are also statistically significant, with a p-value of 7.47e-15.

5.2.3.2.2 Case 2: EVs Charging Simultaneously The case of EV agents charging simultaneously, when the result of their actions is affected also by other agents, is illustrated in Fig. 5.18. The behaviour is similar to the anomalous day scenario presented in Section 5.2.3.1.2: EVs alternate between charging and non-charging periods, as they realise that their group behaviour has a negative impact on the environment. As the previous day overestimates the demand, the peaks created when this is used as a training base are the largest. When the simple (i.e., hybrid) prediction is used as a training base, the peaks generated by EVs charging have lower values and the troughs are deeper. In the perfect prediction case, the charging behaviour is the closest one to

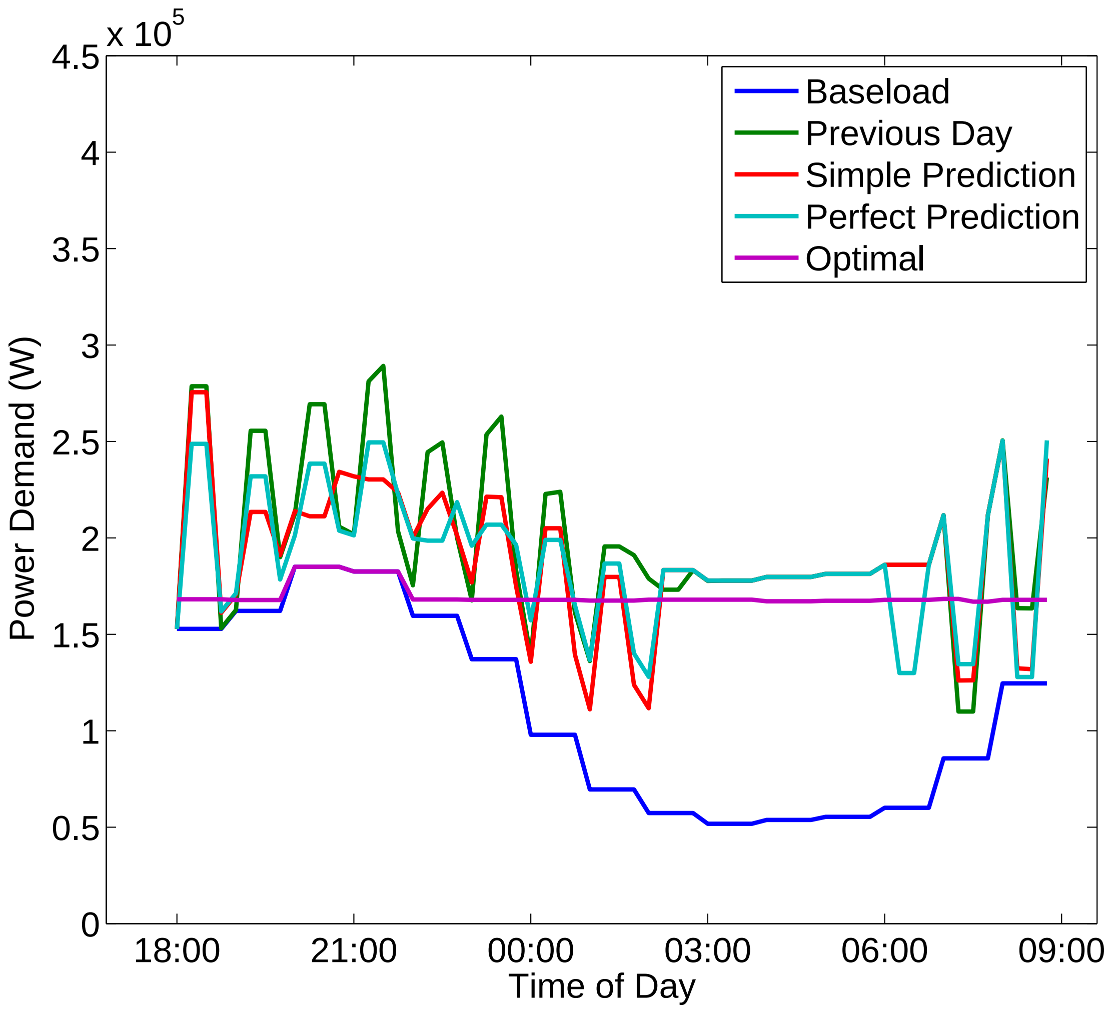


Fig. 5.18: EVs Charging Simultaneously over a Normal Day

follow the optimal behaviour, as it comprises the smallest peaks and troughs. Again, all the EVs manage to achieve their charging targets in all the three cases, as presented in Table 5.9. However, similarly to the previous section, these obtain a lower Pareto efficiency than the ones obtained in Section 5.2.3.1.2, as presented in Table 5.8. The performance difference between perfect prediction and simple prediction runs was statistically significant, with a p-value of 0.017, while simple prediction and previous day runs failed to show any significant difference, with a p-value of 0.213.

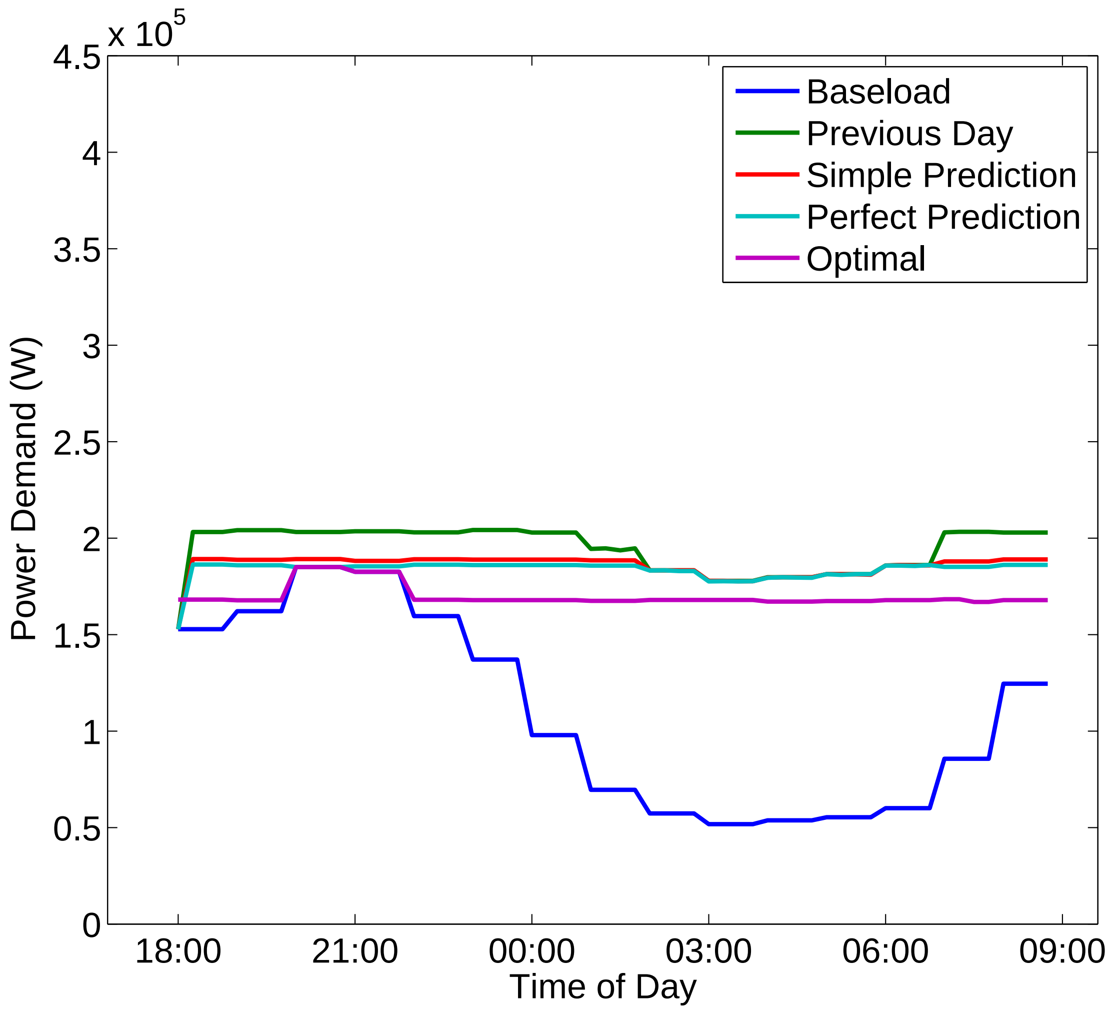


Fig. 5.19: EVs Charging Sequentially over a Normal Day

5.2.3.2.3 Case 3: EVs Charging Sequentially The case of EVs charging sequentially, when each agent acts after another one and can see the results of other agents' actions before taking an action, is pictured in Fig. 5.19. In each of the three prediction cases EV agents decide to overcharge. However, the less accurate the prediction (i.e., more overestimated), the more EVs tend to overcharge. As a result the EV agents trained on the previous day's demand generate the highest demand, even though this is not very far away from the optimal line and does not create peaks in demand. Since the EVs overcharge, all of them achieve (and surpass) their charging targets. This result is shown in Table 5.9. Because of overcharging and the wider time-spread charging options, the Pareto efficiency of the solutions in this case are lower than those presented in Section 5.2.3.1.3. The results in terms of Pareto efficiency are presented in Table 5.8. The difference in performance between simple prediction and previous day was statistically significant, with a p-value of 2.24e-23, while the difference in performance between perfect prediction and simple prediction was also statistically significant, with a p-value of 1.67e-16.

Table 5.8: Algorithms Efficiency in Different Prediction Conditions for Normal Demand

MARL Agent Interaction	**Pareto Performance**		
	MARL	**P-MARL**	
	Prev. Day	Simple Pred.	Perfect Pred.
1. Separate Actions	75.44%	80.74%	84.04%
2. Simultaneous Actions	70.45%	74.38%	76.25%
3. Sequential Actions	79.10%	85.64%	86.86%

Table 5.9: Percentage of EVs Charged Before Departure for Normal Demand

MARL Agent Interaction	**Percentage of Charged EVs**		
	MARL	**P-MARL**	
	Prev. Day	Simple Pred.	Perfect Pred.
1. Separate Actions	100%	100%	100%
2. Simultaneous Actions	100%	100%	100%
3. Sequential Actions	100%	100%	100%

5.2.3.2.4 Analysis Similarly to the anomalous day case presented in Section 5.2.3.1, the quality of prediction and level of interaction between agents and environment affects the aggregate performance of P-MARL. However, since the demand is much better matched this time as opposed to anomalous periods, EV agents are able to achieve their charging objective in all prediction cases and under all types of interaction. Their performance in terms of Pareto efficiency is illustrated in Fig. 5.20. In all three cases presented in the previous sections, better prediction levels leads to increased performance. On average, performance in the case when agents are acting simultaneously is lower than when agents acts separately. However, agents also achieve lower peaks in aggregated demand, which can be seen when comparing Fig. 5.17 and Fig. 5.18, where agents charge separately and simultaneously, respectively.

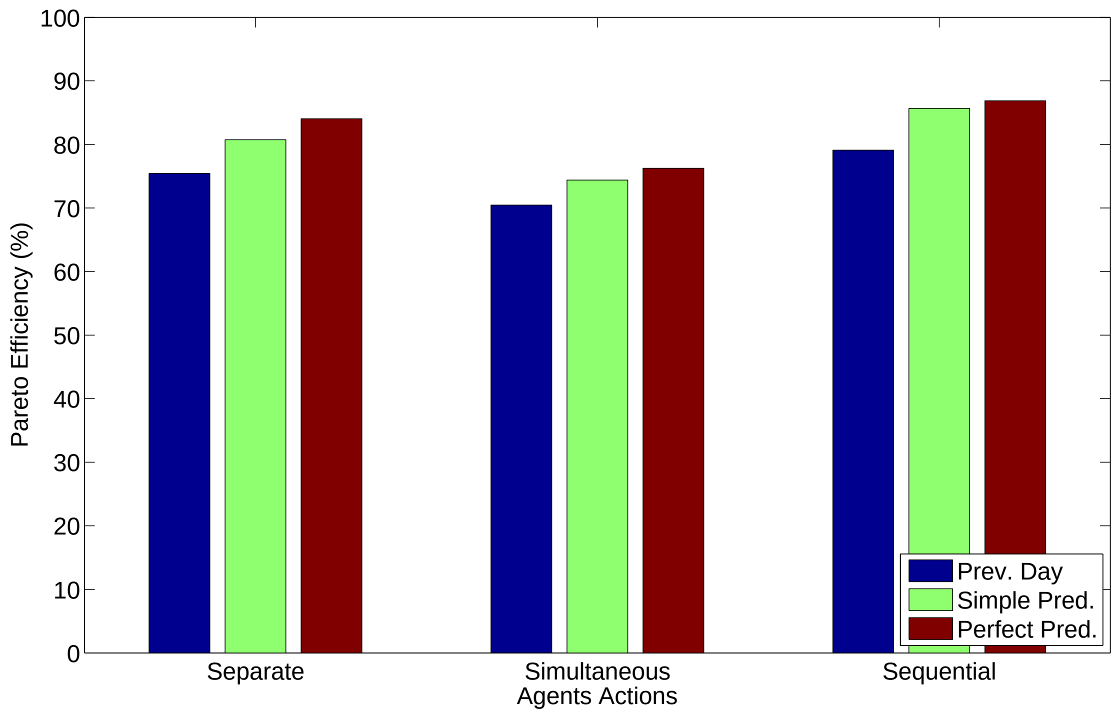


Fig. 5.20: Pareto Efficiency over Normal Day

When comparing the Pareto efficiency results obtained in the case of normal demand with the ones obtained under anomalous demand, the former have on average lower values. This is because demand values for the normal case in August over the hours

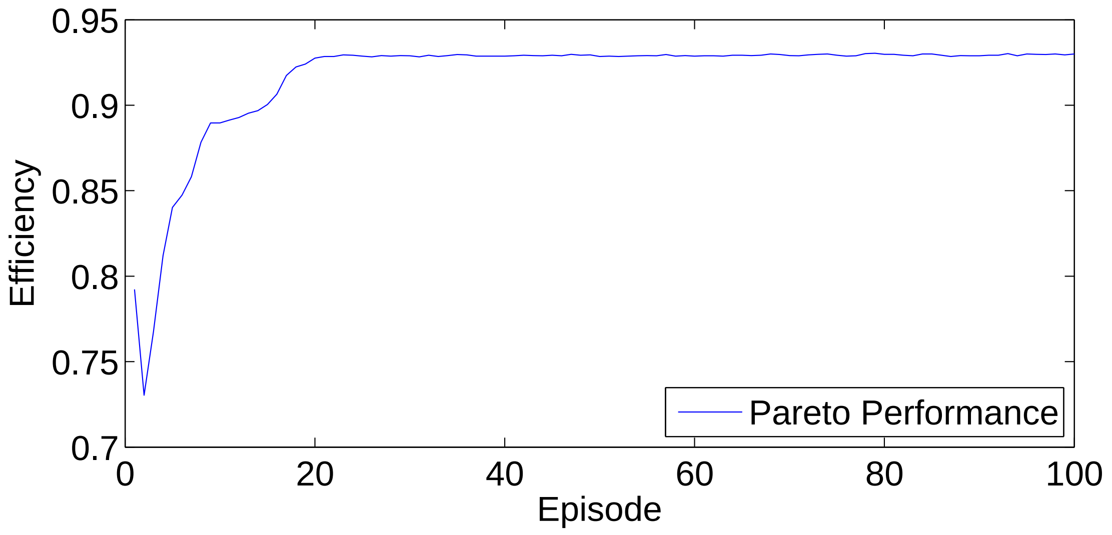


Fig. 5.21: Efficiency During Learning Episodes

available for charging, are lower and closer to each other, as opposed to those from the anomalous case in December. This is caused by less energy used for heating in summer as opposed to winter. As a result, for the normal demand case, EV agents have a wider charging period available, as their aggregated demand has a higher impact on the baseload here than in the anomalous case. With the wider charging period there is also more scope for variability in the solutions, and as a result efficiency drops when compared to the optimal solution.

5.2.3.3 Learning Performance Analysis

The results for the EV agents in this section are based on a learning period of 100 days. However, most of the learning occurs in the first 15-20 days. Fig. 5.21 illustrates the efficiency during each episode while agents are learning with sequential interaction.

After the 20 days mark, the P-MARL algorithm has converged to its final value of approximately 93%. The first 15 days of the learning stage, when most of the state-action knowledge is acquired, are pictured in Fig. 5.22. After the first 3-4 days, the

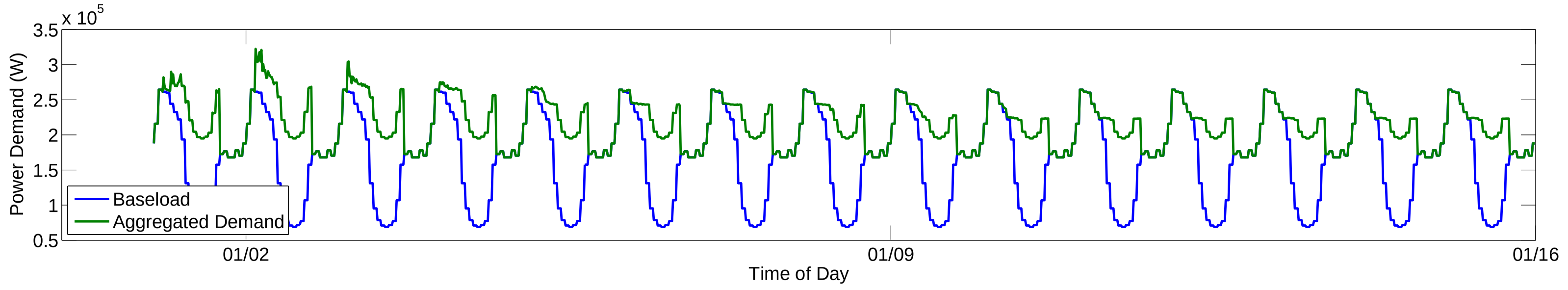


Fig. 5.22: Efficiency During First 15 Learning Episodes

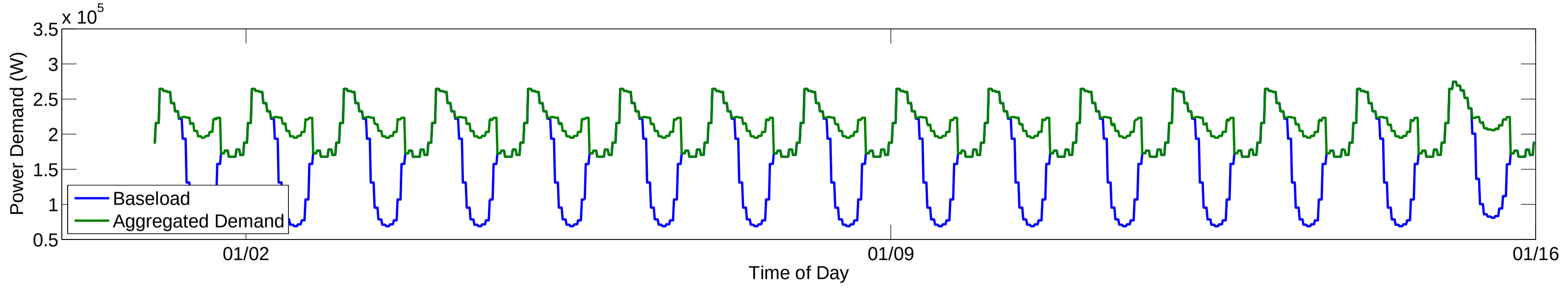


Fig. 5.23: Efficiency During Last 15 Learning Episodes

algorithm has already learned to avoid the evening peak, and continues to improve on spreading the demand over the remaining available charging hours. By the end of the interval, the demand decreases even further, as EV agents are able to achieve their charging objectives with less energy than initially used.

The last 15 days of the learning stage are shown in Fig. 5.23. At this stage, there are only very small changes between the demand of one day and the next one, which are almost unnoticeable. However, the last day represents the real test of the algorithm: the learning stage ends here, and the EV agents are presented with the actual demand occurring, i.e., agents move from the simulated environment to the actual environment. Even though the actual day has a higher demand than predicted, the agents are able to cope with this change and maintain the same behaviour as previously. As a result, their demand is evenly spread, and the Pareto efficiency does not drop in the last day/episode of the experiments, as seen in Fig. 5.21. This shows that the offline training period has achieved its target: agents do not negatively impact the environment when adjusting to changes, as they have been prepared for this in the simulated environment.

5.2.3.3.1 Scale Tests The initial tests were performed in small scale, considering 90 EVs and the demand from 230 houses. However, further tests were done to evaluate the performance of P-MARL in large scale. For this particular set of tests, only the sequential interaction was chosen, as scale is meant to have a larger impact on this type of interaction compared to the others[3], since agents act after each other. An additional change is made to Algorithm 4, after line 5: once the desirable slots are computed, the last desirable slot's value (i.e., the slot with the highest demand between all the selected slots) is saved. Initially this value is selected based on a single agent's charging requirements and the demand. The computed demand slots give the upper limit for the

[3]Scale has no impact on agents acting separately, and the bulk behaviour for agents acting simulatenously is still the same regardless of scale.

charging decisions: when demand goes over this value, the EV agent will not charge as it believes it negatively impacts the environment. When multiple EV agents charge, this limit needs to be higher to accommodate all EVs. However, an EV agent does not know how many other EV agents are involved in the environment. If it is not able to achieve its target charge for 10 learning episodes after initially respecting the upper limit, it increments this value and tries again to achieve its objective for another 10 episodes.

Three scales were employed for this particular set of tests regarding P-MARL, with increasing orders of magnitude:

- 900 EVs
- 9000 EVs
- 90000 EVs

The results of these tests are illustrated in Fig. 5.24. The results for the three scales are very similar. While there are very small differences between the 900 EVs case and the 9000 EVs case, the 9000 and 90000 cases completely overlap. The differences in the last two cases are too small to be noticed visually, as they occur only after the 5th decimal point. Every 10 episodes significant increases in efficiency can be noticed. This is because of the additional change in the algorithm, the incrementing of the upper limit if the charging objective is not achieved with the lower value. At the end of the training period (90 episodes), all EVs fulfil their charging objective. Furthermore, they achieve a Pareto efficiency of over 99.6%. Since the algorithm's efficiency converges very similarly (and even more with scale as shown by the extremely small differences between the 9000 and 90000 case), this shows that P-MARL scales well regardless of the number of agents involved.

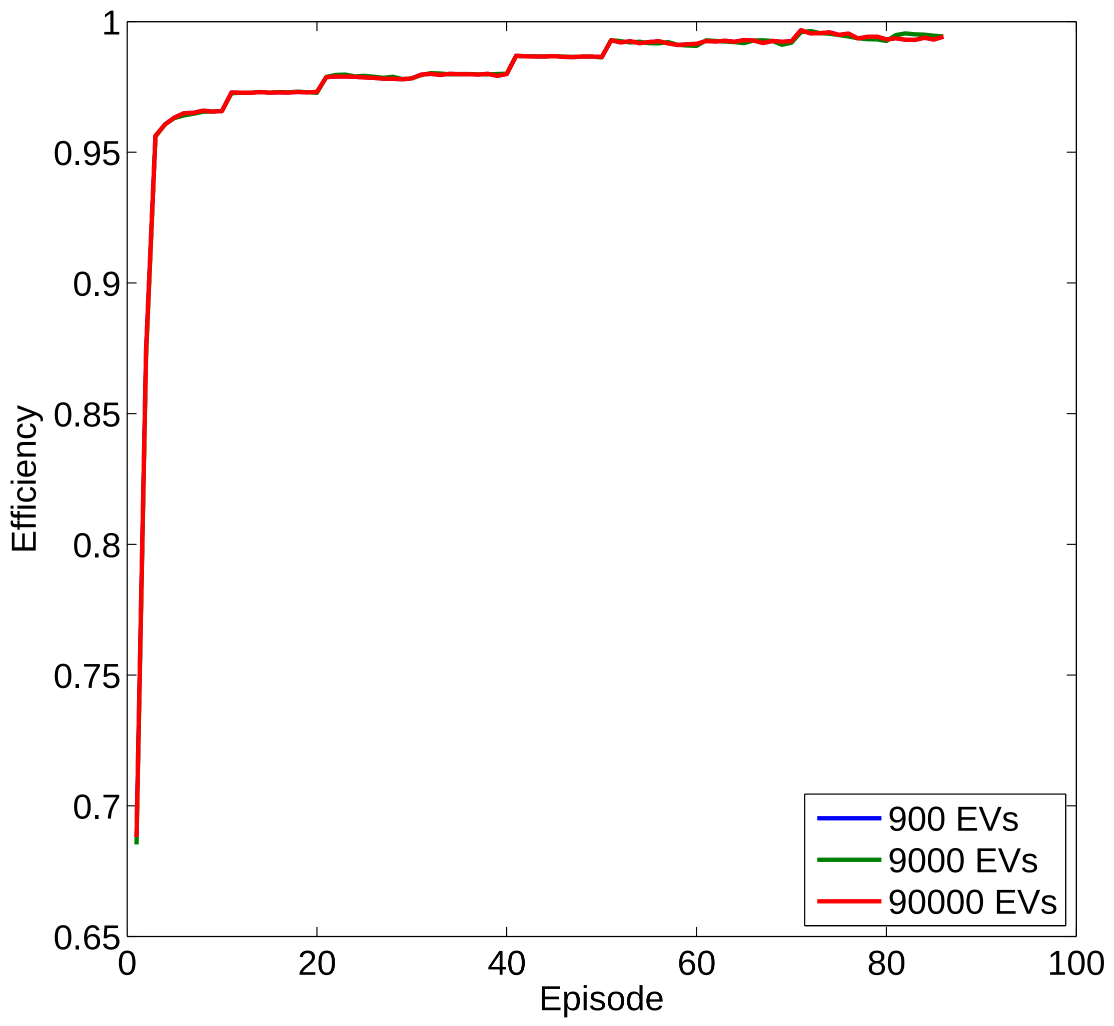


Fig. 5.24: Pareto Efficiency Over Different Scales

5.3 Evaluation with Non-Stationary Solar Energy Supply

P-MARL was also evaluated in other non-stationary conditions, when forecasting accuracy is less accurate. This particular scenario involves stationary power demands, represented by the demand created by a group of small and medium enterprises (SMEs)[4].

[4]The group comprises 200 SMEs, which were chosen from the CER trial, all being from the control group.

These SMEs operate partly on solar energy, which is generated by a set of solar panels[5]. However, the solar energy production is more volatile than energy demand (i.e., higher degree of non-stationarity), thus forecasting is more difficult than in the previous scenario. Changes can occur from expected energy supply, but since these do not depend on consumers, their sources and effects are unpredictable. This scenario is used to evaluate *R1: Minimize Online Learning* under more difficult conditions than the ones occurring in Section 5.2.

The SMEs are part of a business park. As such, they are located in the same place, and supplied by the same transformer. For the purposes of this experiment, the scenario assumes that the park has 90 parking slots where EVs can charge for free. However, these stations provide energy only depending on the solar energy availability, so it is assumed that the EVs charging there already have sufficient charge for the return trip home if there is a very cloudy day. This free charging is just meant to give them an extra charge for the next day. As a result, EV stations have only one objective when compared to the previous sections: optimize solar energy usage. To optimize energy usage, these SMEs operate in a coalition, and attempt to maximize the use of solar energy. At the same time, they have a certain amount of energy contracted from the supplier for each hour of the day which needs to be used.

Similarly to the cases presented in Section 5.2, P-MARL is again evaluated under three types of agent interactions:

1. no agent-contributed non-stationarity.

2. agent-contributed non-stationarity with simultaneous actions.

3. agent-contributed non-stationarity with sequential actions.

[5]It is assumed that each SME has a solar panel of 10 sq. meters, and that the solar panels have 20% conversion efficiency from solar energy.

5.3.1 Solar Energy Forecasting Evaluation

While cloudy days represent a significant subset, the solar energy produced during these days does not follow any specific pattern. This is because the amount of clouds per hour varies each day, and results in very few days being alike. However, there is a predominant group when it is very cloudy, when there is no real solar energy available during the day, and as a result the patterns are very similar in these cases (close to 0 energy produced). Consequently, the data gathered over one year did not allow for any pattern-change detections or pattern matching solutions in terms of solar energy production. As such, P-MARL relies only on the main prediction component in this particular scenario. Furthermore, there is no strong correlation between the previous day's amount of sunshine and the current day's amount of sunshine, as the correlation was under 0.4.

5.3.1.1 Prediction Component Evaluation

In order to optimize their energy usage, these SMEs need to have a forecast of the available solar energy during the day. This energy forecast can be provided based on the weather forecast for the next day (cloudiness for each hour of the day). However, the amount of solar energy produced each day follows a non-stationary pattern.

The ANN presented in Section 4.3.1 was able to provide a forecasting accuracy of 9.32% NRMSE over the sun hours. Detailed results are presented in Table 5.10.

Table 5.10: Solar Energy Forecasting Accuracy

	Forecasting Accuray (NRMSE)
Average	9.32%
95% Confidence Interval	0.67%-17.66%

5.3.2 MAS Performance Evaluation

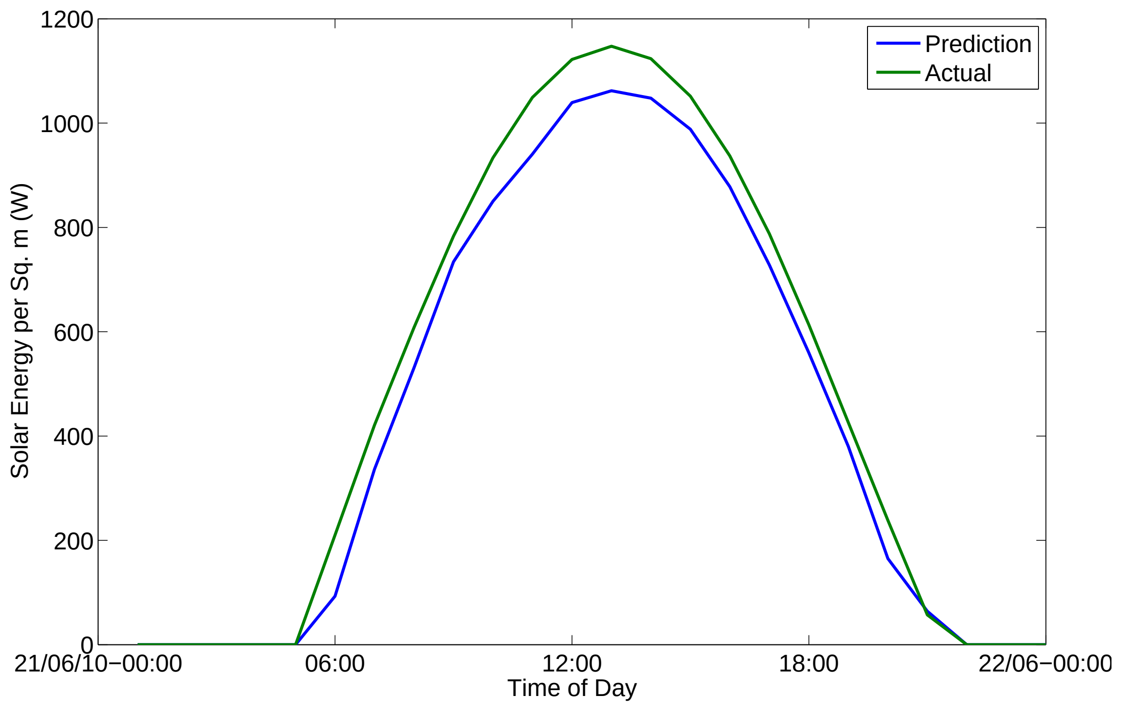


Fig. 5.25: Solar Energy Forecasting Over One Day

A day from June 2010 was chosen to evaluate P-MARL in this scenario. This day had an extended number of sun hours available, and furthermore there was also solar energy produced. The irradiance forecasting error on this particular day was 12.53% NRMSE, with 15.85% NRMSE during the sun hours. The results are illustrated in Fig. 5.25. The prediction underestimates the actual amount of energy produced, in particular at midday.

The resulting aggregated energy consumption, together with the baseload, are pictured in Fig. 5.26. In this figure the amount of energy produced is adjusted to the number of overall square meters available and the efficiency of the solar panels, and presented as an absolute value[6]. In the 07:00-08:00 interval, the amount of energy produced

[6]However, this should be represented on the other side of the X axis since it actually represents

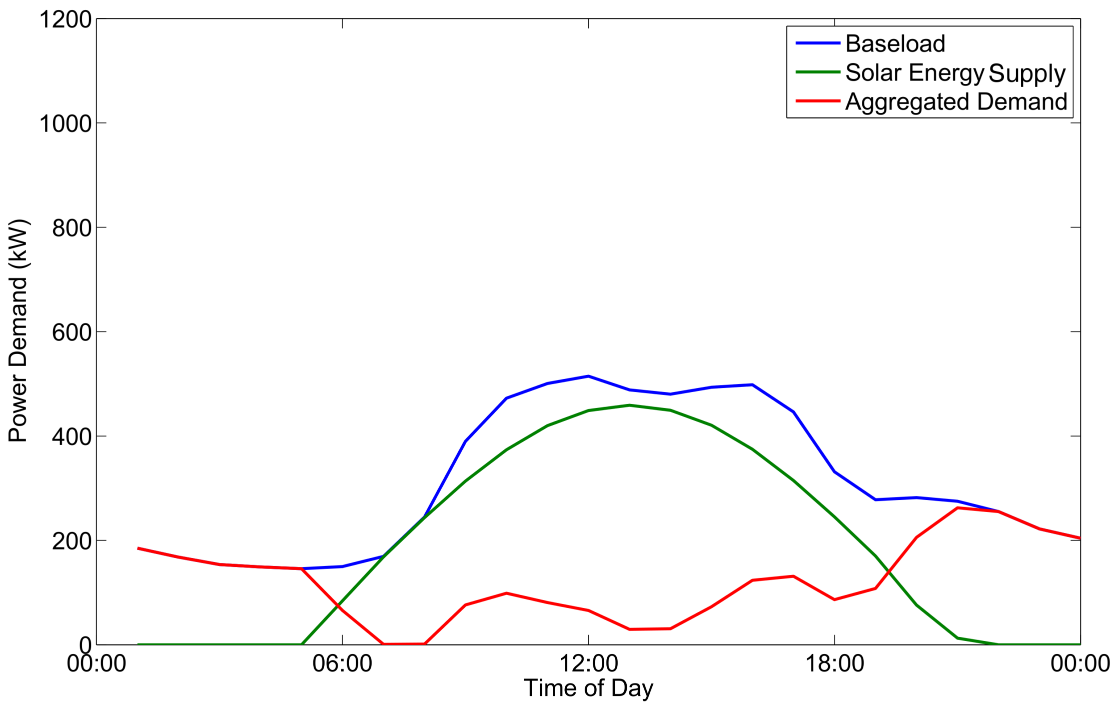


Fig. 5.26: Solar Energy Produced and Used

by the solar panels is almost equal to the amount of energy used in the business park. As such, the business park does not use any of the energy contracted and this is wasted. This interval represents a good opportunity for EV charging. Furthermore, Fig. 5.26 shows that the peaks occurring in the SMEs scenario (before and after lunchtime) are minimised due to the amount of energy produced by the solar panels.

5.3.2.1 Case 1: EVs Charging Separately

Fig. 5.27 presents the results obtained when agents act separately on the environment. When EV agents are trained on both the simple prediction and perfect prediction, these generate a somewhat higher demand than the optimal line. This is because of the cumulated effect of the agents, which are not aware of each other's actions. Since the

production, not demand, but it is presented as absolute values for illustration purposes.

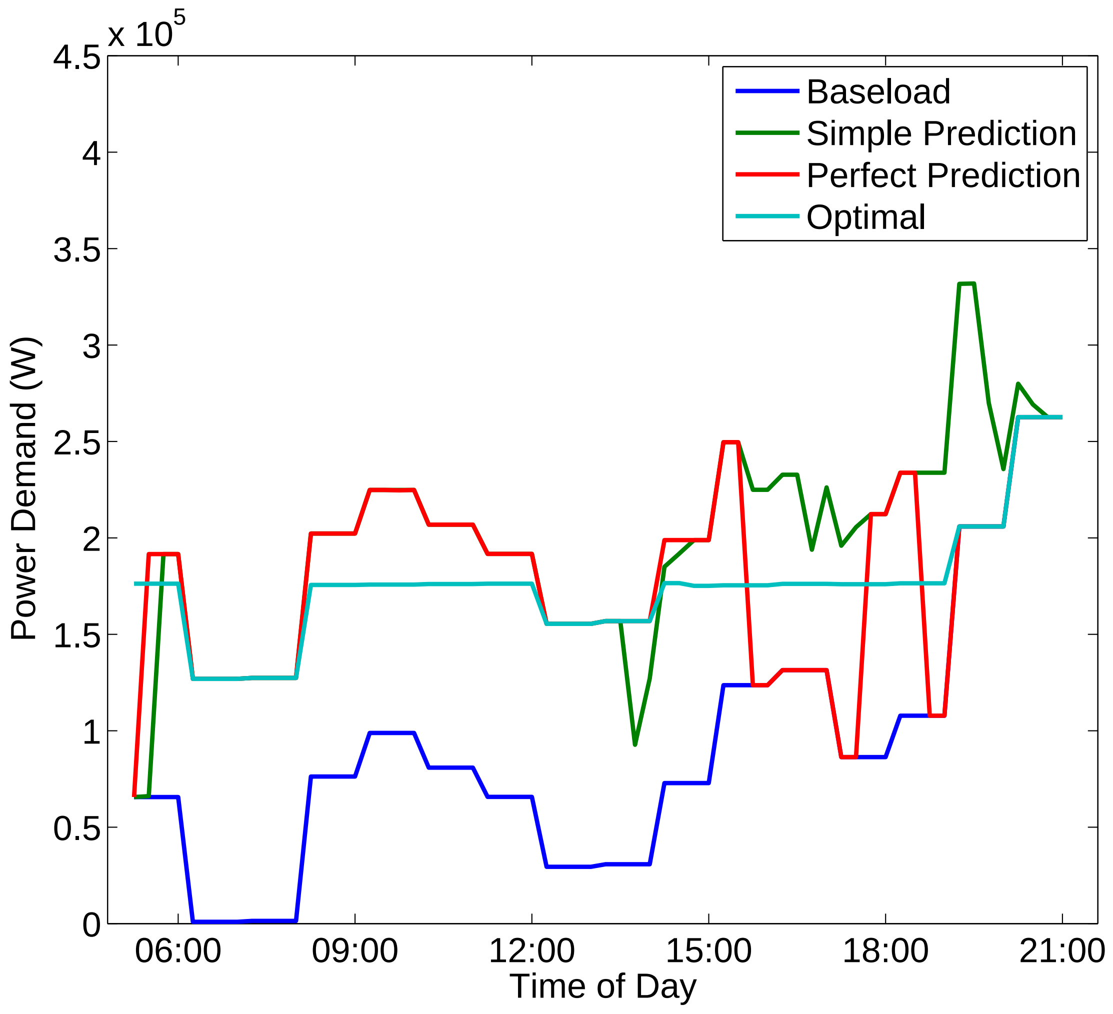


Fig. 5.27: Optimized Energy Demand: Agents Acting Separately

energy generation is underestimated in the simple prediction case, agents are faced with a lower aggregated demand than expected and decide to charge more than the optimal line compared to the agents in the perfect prediction case. This can be noticed at the end of the charging period, at about 20:00, when there is a peak in demand as agents face a comparatively higher upper limit than they trained for, and thus allow for more charging to occur. The Pareto performance of the two solutions is presented in Fig. 5.30, which shows a higher efficiency in the perfect prediction case, in line with intuition, as

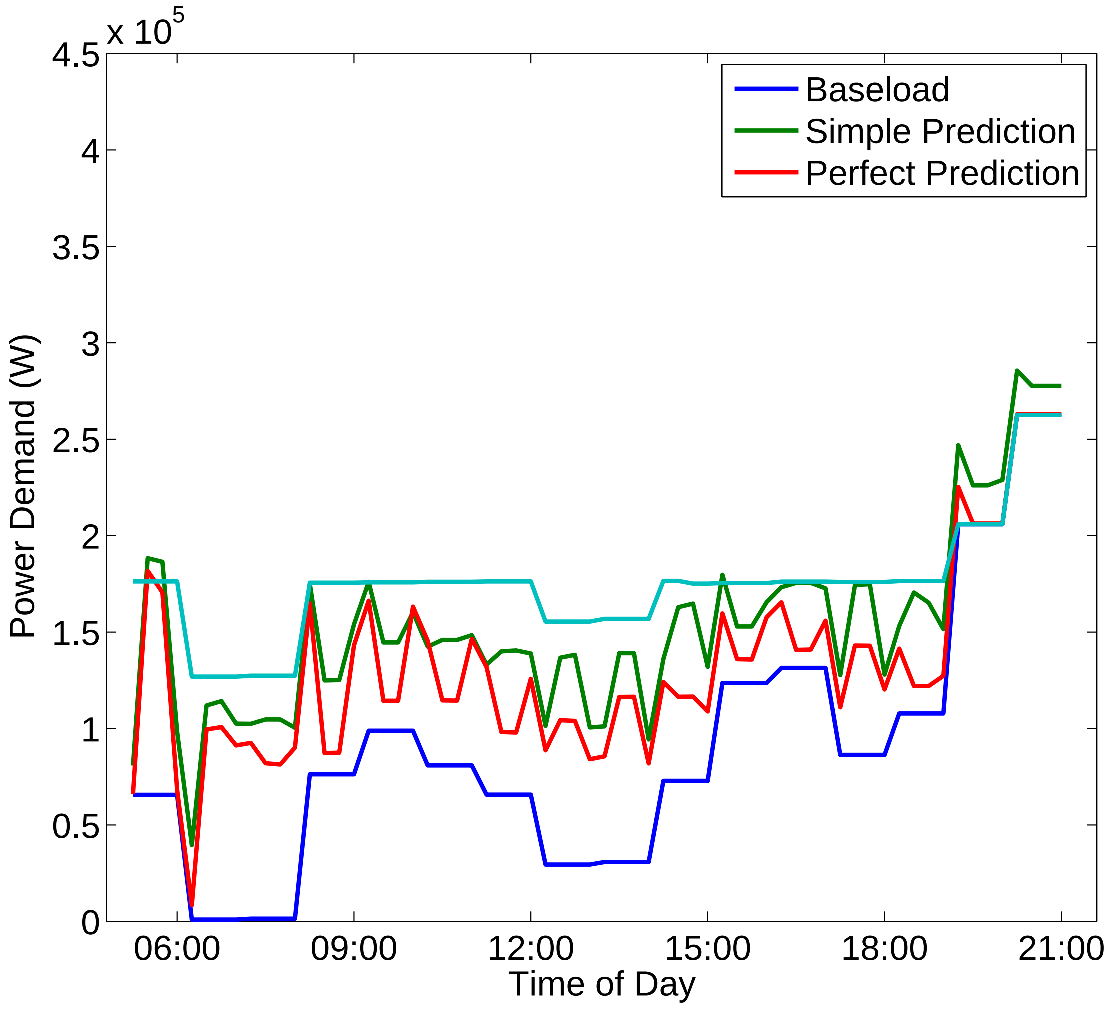


Fig. 5.28: Optimized Energy Demand: Agents Acting Simultaneously

the demand in this case is not underestimated and as a result the charging decisions are closer to the optimal line. This difference in performance is statistically significant, with a p-value of 1.77e-11.

5.3.2.2 Case 2: EVs Charging Simultaneously

When agents act simultaneously, which is the case illustrated in Fig. 5.28, they are aware of the cumulated effect of their actions and are able to keep their demand under the optimum line. However, there is still a noticeable peak at the end of the day, but this is much smaller than the one in Fig. 5.27. The variations in this case are also much smaller than in the previous one, which is more beneficial for the environment (i.e., transformer). However, as illustrated in Fig. 5.30, the agents trained on simple prediction achieve a better Pareto performance. This is because the agents trained on perfect prediction take more conservative measures, in order to avoid running over the upper limit, which coincides with the higher levels of the optimal line. Even though the performance is not as good in the perfect prediction case, agents trained here manage to avoid almost completely charging over the optimal line and do not generate any extra peaks. The difference in performance was considered to be statistically significant, with a p-value of 1.05e-5.

5.3.2.3 Case 3: EVs Charging Sequentially

Fig. 5.29 shows the aggregate charging decisions taken by agents when the actions they take are sequential. Demand is much more uniform in this case, but the EV agents trained on the simple prediction create a higher demand than the optimal solution. This is again due to the underestimated demand, as the forecast for this has an error of 15.85% during the optimization interval. The Pareto performance achieved in this case is 85%, which is in line with the forecasting error. When agents train on the perfect prediction, they more closely follow the optimal charging, achieving a Pareto performance of 92%. This difference in performance is statistically different, with a p-value of 7.54e-10. However, this charging process is again under the optimal line, as EV agents are cautious and take more conservative measures to avoid running over the upper limit.

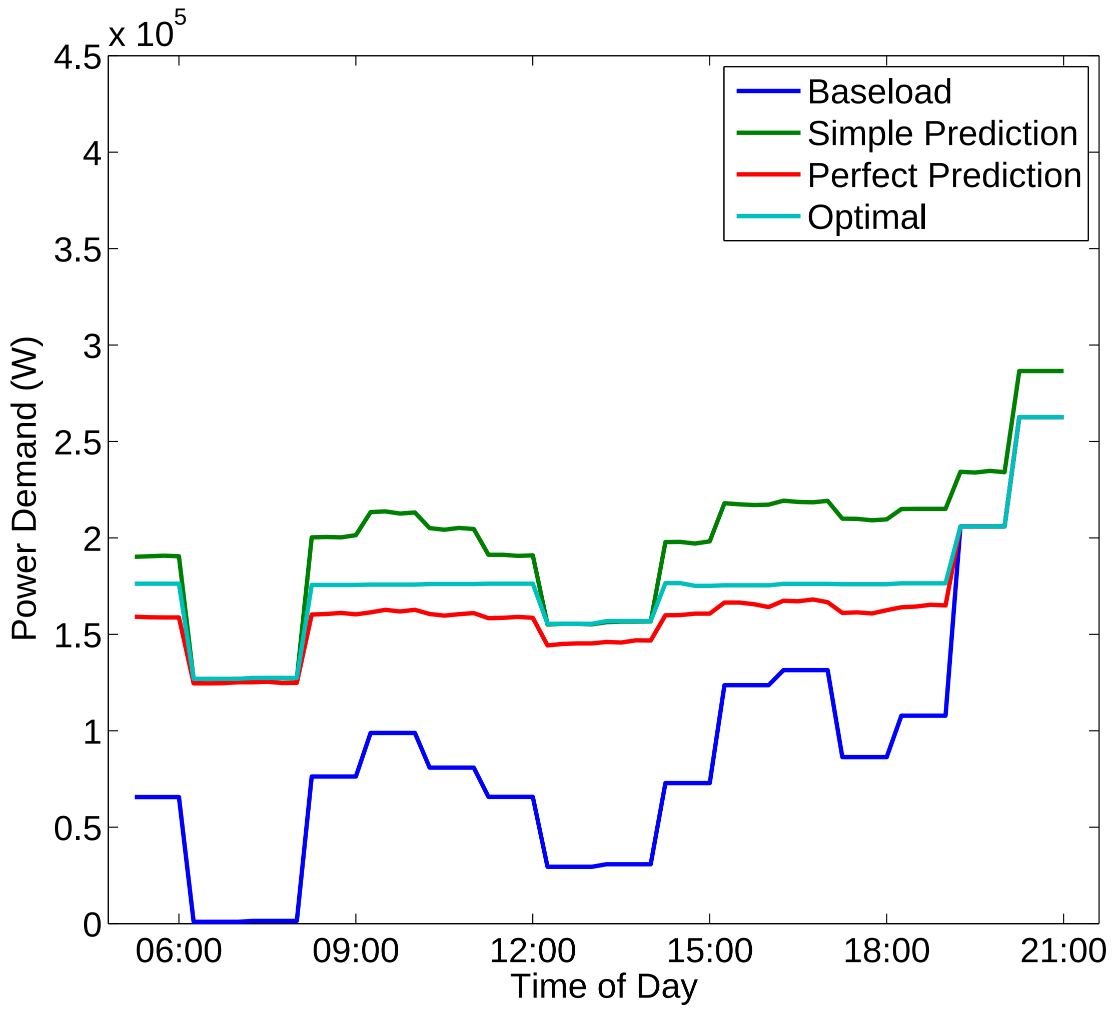


Fig. 5.29: Optimized Solar Energy Demand: Agents Acting Sequentially

5.3.3 Analysis

In this section, P-MARL performance was tested under higher forecasting errors than in the case when demand is non-stationary. Weather has a higher impact on the scenario, giving more unpredictable patterns than in the case of energy demand, which leads to increased non-stationarity in the dataset. These larger errors are shown in Table 5.10. Even though pattern-change detection and matching techniques cannot be employed in this case, P-MARL achieves good results when compared to the optimal line, and is

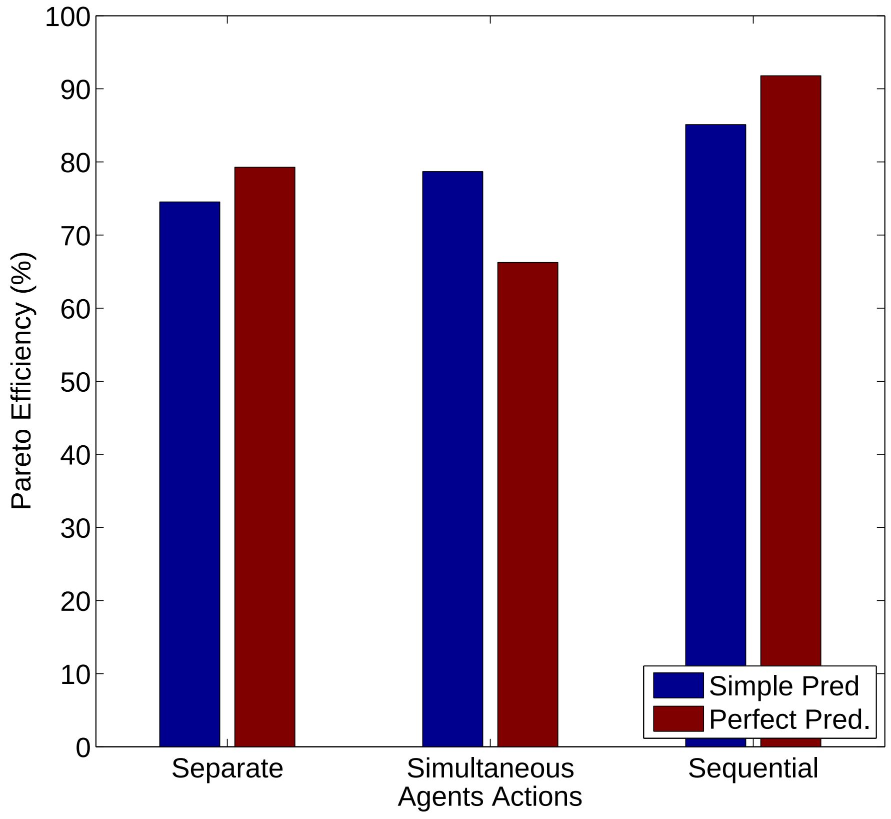


Fig. 5.30: Optimized Solar Energy Demand: Pareto Performance

able to optimize the use of available renewable energy. This is particularly noticeable in the case when agents act sequentially. As such, P-MARL is able to meet requirement *R1:Minimize Online Learning* without negatively impacting the environment even under less accurate forecasts.

5.4 Summary and Analysis

This section evaluated P-MARL in two smart grid scenarios. In the first scenario, P-MARL was evaluated against requirements *R1:Minimize Online Learning*, *R2a:Detect Sudden Changes*, *R2b: Estimate Change Type* and *R3: Prepare for Changes*, while in the second scenario P-MARL was evaluated against *R1:Minimize Online Learning* in more difficult conditions.

A residential neighbourhood scenario was employed in Section 5.2, where P-MARL was used to optimize the charging process of a group of EVs under non-stationary energy demand patterns.

A normal day from the power demand's perspective was used in Section 5.2.3.2, to evaluate P-MARL under requirement R1. It was shown that when demand prediction is used to train P-MARL offline, this provides performance improvements over traditional MARL. Furthermore, compared to the latter, P-MARL does not negatively impact the actual environment.

An anomalous day from the power demand's perspective was used in Section 5.2.3.1, to evaluate P-MARL under requirements R2a, R2b and R3. P-MARL is initially trained based on a predicted demand, but then significant deviations from the predicted demand occur in the actual environment. These changes in demand are first detected by P-MARL, and then the demand change type is matched. The information about the change type is afterwards provided to MAS agents, for them to prepare offline for the upcoming changes. P-MARL shows significant improvements in performance when this change type is used in the training process compared to the case when only the previous prediction is used. Detecting the change type and re-predicting demand makes the difference between agents achieving their objectives and not being able to fulfil their targets, as it was shown in Table 5.7.

A solar energy generation based scenario was employed in Section 5.3 to further evaluate P-MARL against requirement R1, under more difficult conditions than in the previous scenario. P-MARL was used to optimize renewable energy usage under non-stationary solar energy supply patterns. Solar energy forecasting techniques were employed for the prediction component. However, in this case, forecasting accuracy was lower than in the energy demand case, as shown when comparing the results from Table

5.10 and Table 5.3. P-MARL was shown to achieve good performance even under these conditions, which can be seen from the results illustrated in Fig. 5.30.

The performance of P-MARL also depends on the level of interaction between agents and the agent-contributed non-stationarity. All the mentioned tests were performed given: no agent-contributed non-stationarity, agent-contributed stationarity when all agents act simultaneously, and agent-contributed stationarity when agents act sequentially. In line with intuition, the environment benefits if agents are aware of other agents actions before taking an action. When agents acted sequentially, they were able to see the effects of other agents on the environment and this influenced their decisions. As a result, this type of interaction gave the best Pareto performance with regard to the environment related objective: optimizing energy usage. Finally, Section 5.2.3.3 showed that P-MARL's performance is independent of scale, as agents are able to achieve the same level of performance in the same learning time at three different agent scales when compared to the one used in the two scenarios: 900, 9000 and 90000 agents, as opposed to 90 agents, respectively. The time required to perform the simulations grows linearly with the number of agents.

These simulations were performed on just one computer, with a 3.4GHz processor and 8Gb of RAM, where all agents were emulated in a single thread. In an actual deployment case, these computations would be performed in a distributed manner, each agent performing his own computations while evaluating the next action to be taken based on the environment's status. Here there is no significant central computational component: only the result of all agent's actions is aggregated in the offline simulation of environment. This result takes the same amount of time in both the case when actions are taken separately or simultaneously regardless of the number of agents. However, there could be a small delay when involving a large number of agents in the case when agents act sequentially, because they have to wait for the effect of other agent's actions.

In the simulations performed, this was less than half a minute per aggregated decision in the case of 90000 agents. To further decrease the aggregated decision time, when involving such large numbers of agents, the algorithm could be modified so that small groups of agents take actions simultaneously, one group after another, in a combination of sequential and simultaneous decisions. This would not significantly affect the overall Pareto efficiency, while reducing the time needed to obtain an aggregated decision to sub-second level.

Chapter 6

Conclusions

You are the eventuality of an anomaly, which despite my sincerest efforts I have been unable to eliminate from what is otherwise a harmony of mathematical precision. While it remains a burden assiduously avoided, it is not unexpected, and thus not beyond a measure of control. Which has led you, inexorably, here.

The Architect, *Matrix*

This thesis presents P-MARL, a novel approach that improves multi-agent reinforcement learning (MARL) performance in inherently non-stationary environments. The approach integrates prediction and pattern-change detection and matching in the learning stage of MARL, to minimise the effect of non-stationarity. This chapter summarises the thesis while presenting its contributions to the state of the art, and concludes with a discussion on avenues for future work.

6.1 Contributions

The main aim of this thesis was to improve MARL performance when agents operate in inherently non-stationary environments. Chapter 1 motivated the problem behind this work. As an environment continuously evolves, learnt information becomes outdated given new environment dynamics. When changes occur in the environment, agents face previously unencountered situations, for which they are not prepared for in the exploration stages. While adapting to these changes, agents negatively impact the environment.

Chapter 2 presented the state of the art review of MARL in non-stationary environments. The methods used to address changes in non-stationary environments were presented and analysed. It was concluded that current solutions do not directly address non-stationary environments where new dynamics occur, only environments where changes lead to previously encountered dynamics. When the environment presents new dynamics, agents need to adapt online and thus negatively impact the environment while learning suitable actions to address these new dynamics. Time-series analysis techniques were employed in this thesis to help address these new dynamics, and backgrounds concepts about these were further provided. Furthermore, the smart grid, the main application domain investigated in this thesis, was presented.

P-MARL, the main contribution of this thesis, is presented in Chapter 3. P-MARL relies on **prediction** and **pattern-change detection and matching** components to prepare agents for upcoming changes in the environment. Agents are trained offline on a simulation of the environment, which is based on an estimate of the future environment's behaviour provided by these components. The prediction component provides an initial estimate of the environment's future behaviour for a set prediction horizon. The initial prediction is continuously evaluated by the pattern-change detection and matching component, and any significant deviations of the environment's behaviour from the expected behaviour are detected and matched. If a change is detected, a new match for the environment's behaviour is computed by the component through a dynamic reprediction method. The **MAS component** prepares agents for changes offline, in a simulation of the environment, which is based on the repredicted behaviour. P-MARL implements three types of agents interactions, depending on the level of agent-contributed non-stationarity in the environment: agents acting separately, agents acting simultaneously, and agents acting sequentially.

The implementation of P-MARL in the smart grid application domain is presented in Chapter 4, where the smart grid simulator is also introduced. P-MARL is implemented for two scenarios: one where the environment is characterised by non-stationary energy demand patterns, and another one where it is characterised by non-stationary renewable energy supply patterns. The performance of P-MARL in these scenarios is analysed in Chapter 5. P-MARL is evaluated under two types of situations: normal and anomalous. The first situation is used to evaluated P-MARL performance when the environment changes more slowly, while the second situation is used to evaluate P-MARL when significant (i.e., anomalous) changes suddenly occur. The results show that P-MARL outperforms the traditional MARL approach in all situations, with particularly noticeable differences in the case of anomalous changes. As opposed to traditional MARL, all P-MARL agents are able to achieve their objectives when they prepare offline for upcoming changes. The quality of prediction influences the performance of P-MARL, thus justifying the need for accurate environment prediction systems. Additionally, the best P-MARL performance was achieved when agents acted sequentially, as agents were able to observe the effect of other agents' actions on the environment before taking a decision. This performance was shown to be independent of scale, as tests were performed, in addition, on three different orders of magnitude[1]. Additionally, when P-MARL is implemented with sequential agent interaction, the agent-contributed non-stationarity in the actual environment is minimized, since agents interact, before, in the simulated environment over repeated episodes, where they manage to converge to an equilibrium

[1]In terms of number of agents.

before acting online.

The main contributions of this thesis are summarised below:

1. Future environment behaviour is provided as a forecast for a predefined horizon, enabling agent offline learning, where a solution to address the changes in the environment is prepared offline, without affecting the actual environment. This enables P-MARL agents to outperform traditional MARL agents when dealing with continuously evolving environments.

2. Sudden changes in the environment are detected using environment monitoring techniques, and agents are informed when previously expected dynamics become outdated. This way agents are aware that their current knowledge is becoming obsolete and need to prepare for adjustments.

3. The type of anomalous changes occurring in the environment are further analysed (even though this type of dynamics have not been encountered previously), and a predicted estimate of their impact is provided to agents. Agents can use this estimated dynamics to prepare before acting online.

4. Agents' adjustment to sudden anomalous changes in the environment implies a preliminary offline session, to minimize the negative effect on the environment when adapting to changes. A separate short-term horizon describing the expected environment behaviour is provided for agents, to find suitable actions offline.

5. P-MARL reduces agent-contributed non-stationarity by enabling an offline simulation of the environment. Agents act in the simulated environment over repeated episodes, and learn how to take suitable (combined) actions to address the environment changes while interacting in multi-agent setups.

While P-MARL is the main contribution of this thesis, the preliminary studies bring additional contributions to the state of the art with regard to energy demand forecasting. The results achieved by the hybrid prediction solution developed for small scale residential demand forecasting surpasses the state of the art results. These results were discussed in Section 5.2.2.1. Furthermore, the dynamic prediction method presented in Section 5.2.2.2 provides a novel approach to forecasting anomalous energy demand. By combining pattern-change detection and matching techniques, this prediction solution is able to significantly improve the demand forecasting accuracy of anomalous days when compared to traditional methods.

6.2 Discussion

P-MARL aimed to improve MARL performance in inherently non-stationary environments, and was shown to achieve this in the smart grid scenarios presented in this thesis. The performance improvements are particularly noticeable when agents were able to act sequentially, one after another. However, the application of P-MARL is limited to non-stationary environments whose behaviour can be characterised as time-series, and whose future behaviour is not completely random. This is a case where there are external influencing factors that render the environment's behaviour as somewhat predictable through additional information sources. If the time-series describing the environment has a fully random behaviour, without presenting any patterns, the P-MARL solution is not applicable. Furthermore, in the case of environments where sudden changes do not follow any identifiable patterns, P-MARL cannot fulfil requirement *R2b: Estimate Change Type* and as a result requirement *R3: Prepare for changes.* This was observed in the second scenario, where P-MARL was tasked with the optimization of renewable energy usage, where it could only predict future solar energy supply and detect changes from the expected demand, but could not match the type of change occurring. However, in such cases the SOM number of classes could be expanded, by including new classes for such types of changes. Therefore if an unencountered type of change starts occurring repeatedly, this would end up classified into its own class in the SOM.

In the power demand forecasting case, changes are detected later in the day. However, some significant changes can occur in the morning period that would impact on the grid's stability; this information could be useful for other applications as well, for which other more reactive types of change detection systems could be implemented (e.g., change detection through sliding window mechanisms). As for classification, other techniques could have been used for this part of solution, such as SVMs. SOMs achieved good results in the preliminary work for this thesis, therefore they were further employed, without further evaluation of other classification techniques.

An additional limitation of P-MARL is its reliance on the offline environment simulation component. If agents cannot interact offline, within a simulation of the environment, their performance is reduced to the first type of interaction presented, *single-agents acting separately.*

6.3 Future Work

In this thesis, P-MARL was evaluated in a set of smart grids scenarios. However, P-MARL could be applied in other inherently non-stationary environments such as vehicular traffic, computer networks or resource allocation problems. It would be interesting to evaluate future implementations of P-MARL in such domains, where environment time-series analysis techniques can be integrated in the learning process of MARL. These domains present a new set of challenges, where reaction time can be a critical factor, and where the offline simulation of the environment needs to have quick response time, and thus fast learning abilities (i.e., small number of training episodes).

An assumption in the evaluation of P-MARL was that all agents have the same global objectives, besides a personal objective. However, the multi-agent system's agents could have conflicting objectives, or even involve malicious agents, therefore it would be worth investigating the convergence of P-MARL in such conditions. Since agents with conflicting objectives could cause changes from the expected environment's behaviour, it would be interesting to model and predict their (group) behaviour as well, in addition to the environment's behaviour. However, modelling individual agents' behaviour would be infeasible when a large number of agents is involved.

Furthermore, all agents in P-MARL are based on reinforcement learning. Other systems could be developed based on the design of P-MARL while considering different types of underlying agent learning techniques, possibly even multiple types in one implementation of the MAS component.

The work in this thesis focused mainly on environment-induced non-stationarity, while agent-contributed non-stationarity was addressed through different types of indirect agents interactions, where agents interacted with each other only through the environment. However, the MAS component could benefit of approaches where agents exchange information directly, such as the Distributed W-Learning algorithm (Dusparic and Cahill, 2012). This is another potential extension to P-MARL which is worth investigating, where agent-contributed non-stationarity is addressed by agents directly collaborating.

6.4 Conclusion

This thesis presented P-MARL, a novel approach which extends MARL by integrating advanced time-series analysis techniques to address inherently non-stationary environments. Throughout this chapter, it was shown how these techniques enabled P-MARL to achieve improved performance when compared to traditional MARL agents in environments whose behaviour can be modelled as time-series.

Bibliography

Alfares, H. K. and Nazeeruddin, M. (2002). Electric load forecasting: Literature survey and classification of methods. *International Journal of Systems Science*, 33(1):23–34.

Alonso, E., D'inverno, M., Kudenko, D., Luck, M., and Noble, J. (2001). Learning in multi-agent systems. *The Knowledge Engineering Review*, 16(03):277–284.

Amin, M. and Wollenberg, B. (2005). Toward a smart grid: power delivery for the 21st century. *Power and Energy Magazine, IEEE*, 3(5):34–41.

Amjady, N. (2007). Short-Term Bus Load Forecasting of Power Systems by a New Hybrid Method. *IEEE Transactions on Power Systems*, 22(1):333–341.

Amjady, N. and Keynia, F. (2010). Short-Term Load Forecast of Microgrids by a New Bilevel Prediction Strategy. *IEEE Transactions on Smart Grid*, 1(3):286–294.

Andersson, G., Donalek, P., Farmer, R., Hatziargyriou, N., Kamwa, I., Kundur, P., Martins, N., Paserba, J., Pourbeik, P., Sanchez-Gasca, J., et al. (2005). Causes of the 2003 major grid blackouts in north america and europe, and recommended means to improve system dynamic performance. *Power Systems, IEEE Transactions on*, 20(4):1922–1928.

Annunziato, M., Bertini, I., Lucchetti, M., Pannicelli, A., and Pizzuti, S. (2001). Adaptivity of artificial life environment for on-line optimization of evolving dynamical systems. *Proc. EUNITE01, Tenerife, Spain.*

Backhaus, S., Swift, G. W., Chatzivasileiadis, S., Tschudi, W., Glover, S., Starke, M., Wang, J., Yue, M., and Hammerstrom, D. (2015). Dc microgrids scoping studyestimate of technical and economic benefits.

Basso, E. W. and Engel, P. M. (2009). Reinforcement learning in non-stationary continuous time and space scenarios. In *Artificial Intelligence National Meeting (Enia)*, volume 7, pages 1–8.

Bazzan, A. L. (2014). Beyond reinforcement learning and local view in multiagent systems. *KI-Künstliche Intelligenz*, 28(3):179–189.

Beccali, M., Cellura, M., Lo Brano, V., and Marvuglia, A. (2004). Forecasting daily urban electric load profiles using artificial neural networks. *Energy Conversion and Management*, 45(18-19):2879–2900.

Bevrani, H., Ghosh, A., and Ledwich, G. (2010). Renewable energy sources and frequency regulation: survey and new perspectives. *Renewable Power Generation, IET*, 4(5):438–457.

Bhuvaneswari, R., Srivastava, S. K., Edrington, C. S., Cartes, D. A., and Subramanian, S. (2010). Intelligent Agent Based Auction by Economic Generation Scheduling for Microgrid Operation. *Education*, pages 1–6.

Bloembergen, D., Tuyls, K., Hennes, D., and Kaisers, M. (2015). Evolutionary dynamics of multi-agent learning: A survey. *Journal of Artificial Intelligence Research*, pages 659–697.

Borges, C. E., Penya, Y. K., and Fernández, I. (2011). Optimal combined short-term building load forecasting. In *Innovative Smart Grid Technologies Asia (ISGT), 2011 IEEE PES*, pages 1–7. IEEE.

Bowling, M. and Veloso, M. (2000). An analysis of stochastic game theory for multiagent reinforcement learning. Technical report, DTIC Document.

Box, G., Jenkins, G., and Reinsel, G. (1970). *Time series analysis: forecasting and control*. Wiley.

Brown, M. and Harris, C. J. (1994). Neurofuzzy adaptive modelling and control.

Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172.

Camacho, E. F. and Alba, C. B. (2013). *Model predictive control*. Springer Science & Business Media.

Camponogara, E., Jia, D., Krogh, B. H., and Talukdar, S. (2002). Distributed model predictive control. *Control Systems, IEEE*, 22(1):44–52.

Cetnarowicz, K., Kisiel-Dorohinicki, M., and Nawarecki, E. (1996). The application of evolution process in multi-agent world to the prediction system. *Proceedings of the Second International Conference on Multi-agent Systems (ICMAS)*.

Chan, P. P., Chen, W.-C., Ng, W. W., and Yeung, D. S. (2011). Multiple classifier system for short term load forecast of microgrid. In *Machine Learning and Cybernetics (ICMLC), 2011 International Conference on*, volume 3, pages 1268–1273. IEEE.

Chaouachi, A., Kamel, R. M., Andoulsi, R., and Nagasaka, K. (2013). Multiobjective intelligent energy management for a microgrid. *Industrial Electronics, IEEE Transactions on*, 60(4):1688–1699.

Chassin, D., Schneider, K., and Gerkensmeyer, C. (2008). Gridlab-d: An open-source power systems modeling and simulation environment. In *2008 IEEE/PES Transmission and Distribution Conference and Exposition.*

Chatzivasiliadis, S. J., Hatziargyriou, N. D., and Dimeas, a. L. (2008). Development of an agent based intelligent control system for microgrids. *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, pages 1–6.

Chitty, D. M. and Hernandez, M. L. (2004). A hybrid ant colony optimisation technique for dynamic vehicle routing. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 48–59. Springer.

Choi, S. P., Yeung, D.-Y., and Zhang, N. L. (2001). Hidden-mode markov decision processes for nonstationary sequential decision making. In *Sequence Learning*, pages 264–287. Springer.

Christofides, P. D., Scattolini, R., de la Pena, D. M., and Liu, J. (2013). Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering*, 51:21–41.

Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*, pages 746–752.

Clemen, R. T. (1989). Combining forecasts: A review and annotated bibliography. *International journal of forecasting*, 5(4):559–583.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., and Terpenning, I. (1990). Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73.

Cleveland, W. S. (1981). Lowess: A program for smoothing scatterplots by robust locally weighted regression. *American Statistician*, pages 54–54.

Cobb, H. G. (1990). An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical report, DTIC Document.

Colson, C. M. and Nehrir, M. H. (2013). Comprehensive real-time microgrid power management and control with distributed agents. *Smart Grid, IEEE Transactions on*, 4(1):617–627.

Colson, C. M., Nehrir, M. H., Sharma, R. K., and Asghari, B. (2014a). Improving sustainability of hybrid energy systems part i: Incorporating battery round-trip efficiency and operational cost factors. *Sustainable Energy, IEEE Transactions on*, 5(1):37–45.

Colson, C. M., Nehrir, M. H., Sharma, R. K., and Asghari, B. (2014b). Improving sustainability of hybrid energy systems part ii: Managing multiple objectives with a multiagent system. *Sustainable Energy, IEEE Transactions on*, 5(1):46–54.

Comission for Energy Regulation, I. (2011). Smart meter trial data.

Cooley, J. W. and Tukey, J. W. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cui, X., Charles, J. S., and Potok, T. E. (2009). A simple distributed particle swarm optimization for dynamic and noisy environments. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2008)*, pages 89–102. Springer.

Da Silva, B. C., Basso, E. W., Bazzan, A. L., and Engel, P. M. (2006). Dealing with non-stationary environments using context detection. In *ICML*, pages 217–224. ACM.

De França, F. O. and Von Zuben, F. J. (2009). A dynamic artificial immune algorithm applied to challenging benchmarking problems. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 423–430. IEEE.

Dempster, A. P. (1968). A generalization of bayesian inference. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 205–247.

Deng, L. and Yu, D. (2014). Deep learning: methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387.

Dimeas, A. and Hatziargyriou, N. (2007). Agent based control of virtual power plants. In *Intelligent Systems Applications to Power Systems, 2007. ISAP 2007. International Conference on*, pages 1–6. IEEE.

Dou, C.-X. and Liu, B. (2013). Multi-agent based hierarchical hybrid control for smart microgrid. *Smart Grid, IEEE Transactions on*, 4(2):771–778.

Doya, K., Samejima, K., Katagiri, K.-i., and Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369.

Dréo, J. and Siarry, P. (2006). An ant colony algorithm aimed at dynamic continuous optimization. *Applied Mathematics and Computation*, 181(1):457–467.

Druckman, A. and Jackson, T. (2008). Household energy consumption in the uk: A highly geographically and socio-economically disaggregated model. *Energy Policy*, 36(8):3177 – 3192.

Duff, S. J. and Bradtke Michael, O. (1995). Reinforcement learning methods for continuous-time markov decision problems. *Advances in Neural Information Processing Systems 7*, 7:393.

Dusparic, I. and Cahill, V. (2012). Autonomic multi-policy optimization in pervasive systems: Overview and evaluation. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7(1):11.

Dusparic, I., Harris, C., Marinescu, A., Cahill, V., and Clarke, S. (2013). Multi-agent residential demand response based on load forecasting. In *Technologies for Sustainability (SusTech), 2013 1st IEEE Conference on*, pages 90–96. IEEE.

Eddy, F., Gooi, H., and Chen, S. (2015). Multi-agent system for distributed management of microgrids. *Power Systems, IEEE Transactions on*, 30(1):24–34.

Elidrisi, M., Johnson, N., Gini, M., and Crandall, J. (2014). Fast adaptive learning in repeated stochastic games by game abstraction. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 1141–1148, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Elwell, R. and Polikar, R. (2009). Incremental learning of variable rate concept drift. *Multiple Classifier Systems*, pages 142–151.

EPA (2008). Average annual emissions and fuel consumption for gasoline-fueled passenger cars and light trucks. Technical report, United States Environmental Protection Agency.

Ernst, D., Glavic, M., Capitanescu, F., and Wehenkel, L. (2009). Reinforcement learning versus model predictive control: a comparison on a power system problem. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(2):517–529.

Espinoza, M., Joye, C., Belmans, R., and Moor, B. D. (2005). Short-Term Load Forecasting , Profile Identification , and Customer Segmentation : A Methodology Based on Periodic Time Series. 20(3):1622–1630.

Esquivel, S. C. and Coello, C. A. C. (2004). Particle swarm optimization in non-stationary environments. In *Advances in Artificial Intelligence–IBERAMIA 2004*, pages 757–766. Springer.

Fahlman, S. (1988). An empirical study of learning speed in back-propagation networks. Technical report, Computer Science Department, Carnegie Mellon University.

Faqiry, M. N., Kundu, R., Mukherjee, R., Das, S., and Pahwa, A. (2014). Game theoretic model of energy trading strategies at equilibrium in microgrids. In *North American Power Symposium (NAPS), 2014*, pages 1–4. IEEE.

Farhangi, H. (2010). The path of the smart grid. *Power and Energy Magazine, IEEE*, 8(1):18–28.

Fatimie, M., Baharudin, Z., Hisham, N., Faris, M., and Yunus, S. S. (2010). Electricity forecasting for small scale power system using fuzzy logic. *2010 Conference Proceedings IPEC*, pages 1040–1045.

Ferber, J. (1999). *Multi-agent systems: an introduction to distributed artificial intelligence*, volume 1. Addison-Wesley Reading.

Fu, T.-c. (2011). A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181.

Galstyan, A., Czajkowski, K., and Lerman, K. (2004). Resource allocation in the grid using reinforcement learning. In *Proceedings of the Third International Joint Con-*

ference on Autonomous Agents and Multiagent Systems-Volume 3, pages 1314–1315. IEEE Computer Society.

Gan, L., Topcu, U., and Low, S. (2011). Optimal decentralized protocol for electric vehicle charging. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 5798–5804. IEEE.

Ghazvini, M., Abedini, R., Pinto, T., and Vale, Z. (2014). Multiagent system architecture for short-term operation of integrated microgrids.

Goldberg, D. E. and Smith, R. E. (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *ICGA*, pages 59–68.

Gomes, L., Pinto, T., Faria, P., and Vale, Z. (2014). Distributed intelligent management of microgrids using a multi-agent simulation platform. In *Intelligent Agents (IA), 2014 IEEE Symposium on*, pages 1–7. IEEE.

Gross, G. and Galiana, F. (1987). Short-term load forecasting. *Proceedings of the IEEE*, 75(12):1558–1573.

Hadoux, E., Beynier, A., and Weng, P. (2014). Sequential decision-making under non-stationary environments via sequential change-point detection. *International Workshop on Learning over Multiple Contexts*.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108.

Hatzakis, I. and Wallace, D. (2006). Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1201–1208. ACM.

Hatziargyriou, N., Asano, H., Iravani, R., and Marnay, C. (2007). Microgrids. *Power and Energy Magazine, IEEE*, 5(4):78–94.

Hernandez, L., Baladron, C., Aguiar, J., Carro, B., Sanchez-Esguevillas, A., Lloret, J., and Massana, J. (2014). A survey on electric power demand forecasting: Future trends in smart grids, microgrids and smart buildings. *Communications Surveys Tutorials, IEEE*, 16(3):1460–1495.

Hernandez-Leal, P., Munoz de Cote, E., and Sucar, L. E. (2013). Learning against non-stationary opponents. In *Proceedings of the 12th international conference on autonomous agents and multiagent systems, Saint Paul, MN.*

Hernndez, L., Baladrn, C., Aguiar, J. M., Calavia, L., Carro, B., Snchez-Esguevillas, A., Cook, D. J., Chinarro, D., and Gmez, J. (2012). A study of the relationship between weather variables and electric power demand inside a smart grid/smart world framework. *Sensors*, 12(9):11571–11591.

Hippert, H., Pedreira, C., and Souza, R. (2001). Neural networks for short-term load forecasting: a review and evaluation. *IEEE Transactions on Power Systems*, 16(1):44–55.

Hu, J. and Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4:1039–1069.

Hu, X. and Eberhart, R. C. (2002). Adaptive particle swarm optimization: detection and response to dynamic systems. In *wcci*, pages 1666–1670. IEEE.

Huebscher, M. C. and McCann, J. A. (2008). A survey of autonomic computing: Degrees, models, and applications. *ACM Comput. Surv.*, 40(3):7:1–7:28.

Information, U. E. F. E. (2014). Nissan leaf.

Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1):7–38.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, pages 237–285.

Kahlen, M., Ketter, W., and van Dalen, J. (2014). Agent-coordinated virtual power plants of electric vehicles. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1547–1548. International Foundation for Autonomous Agents and Multiagent Systems.

Kantamneni, A., Brown, L. E., Parker, G., and Weaver, W. W. (2015). Survey of multi-agent systems for microgrid control. *Engineering Applications of Artificial Intelligence*, 45:192–203.

Kay, D., Hill, N., and Newman, D. (2013). Powering ahead: The future of low-carbon cars and fuels. Technical report, RAC Foundation.

Keil, D. and Goldin, D. (2006). *Indirect Interaction in Environments for Multi-agent Systems*, volume 3830 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Kephart, J. and Chess, D. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Khosrow-Pour, M. (2008). *Encyclopedia of information science and technology*, volume 1. IGI Global.

Khotanzad, A., Afkhami-rohani, R., Lu, T.-l., Abaye, A., Davis, M., and Maratukulam, D. J. (1997). ANNSTLF - A Neural-Network-Based Electric Load Forecasting System. *IEEE Transactions on Neural Networks*, 8(4):835–846.

Kim, J. and Bentley, P. (2002). Towards an artificial immune system for network intrusion detection: an investigation of dynamic clonal selection. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1015–1020.

Klügl, F., Fehler, M., and Herrler, R. (2005). About the role of the environment in multi-agent simulations. In *Environments for multi-agent systems*, pages 127–149. Springer.

Kobliha, M., Schwarz, J., and Očenášek, J. (2006). Bayesian optimization algorithms for dynamic problems. In *Applications of Evolutionary Computing*, pages 800–804. Springer.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

Kok, K. (2010). Multi-Agent Coordination in the Electricity Grid , from Concept towards Market Introduction. (Aamas):1681–1688.

Kumar, Y., Das, B., and Sharma, J. (2008). Multiobjective, multiconstraint service restoration of electric power distribution system with priority customers. *Power Delivery, IEEE Transactions on*, 23(1):261–270.

Lambert-Torres, G., Martins, H. G., Coutinho, M. P., Salomon, C. P., and Vieira, F. C. (2009). Particle swarm optimization applied to system restoration. In *PowerTech, 2009 IEEE Bucharest*, pages 1–6. IEEE.

Laumônier, J. and Chaib-draa, B. (2005). Multiagent q-learning: Preliminary study on dominance between the nash and stackelberg equilibriums. In *Proceedings of AAAI-2005 Workshop on Multiagent Learning, Pittsburgh, USA.*

Lauri, F., Basso, G., Zhu, J., Roche, R., Hilaire, V., and Koukam, A. (2013). Managing power flows in microgrids using multi-agent reinforcement learning. *Agent Technologies in Energy Systems (ATES)*.

Littman, M. L. (2001). Friend-or-foe q-learning in general-sum games. In *ICML*, volume 1, pages 322–328.

Llanos, J., Sáez, D., and Núñez, A. (2012). Load Profile Generator and Load Forecasting for a Renewable Based Microgrid Using Self Organizing Maps and Neural Networks. *International Joint Conference on Neural Networks*, pages 10–15.

Lloret, J. and Valencia, U. P. D. (2013). FUTURE SMART GRID UTASG A Multi-Agent System Architecture for Smart Grid Management and Forecasting of Energy Demand in Virtual Power Plants. *IEEE Communications Magazine*, (January):106–113.

Logenthiran, T., Srinivasan, D., and Khambadkone, A. M. (2011). Multi-agent system for energy resource scheduling of integrated microgrids in a distributed system. *Electric Power Systems Research*, 81(1):138–148.

Makridakis, S., Wheelwright, S. C., and Hyndman, R. J. (1998). *Forecasting methods and applications*. John Wiley & Sons.

Mao, M., Jin, P., Hatziargyriou, N. D., and Chang, L. (2014). Multiagent-based hybrid energy management system for microgrids. *Sustainable Energy, IEEE Transactions on*, 5(3):938–946.

Marinescu, A., Dusparic, I., Harris, C., Cahill, V., and Clarke, S. (2014a). A dynamic forecasting method for small scale residential electrical demand. In *IJCNN*, pages 3767–3774.

Marinescu, A., Harris, C., Dusparic, I., Cahill, V., and Clarke, S. (2014b). A hybrid approach to very small scale electrical demand forecasting. In *ISGT, 2014 IEEE PES*, pages 1–5.

Marinescu, A., Harris, C., Dusparic, I., Clarke, S., and Cahill, V. (2013). Residential electrical demand forecasting in very small scale: An evaluation of forecasting methods. In *Software Engineering Challenges for the Smart Grid (SE4SG), 2013 2nd International Workshop on*, pages 25–32. IEEE.

Mavrovouniotis, M. and Yang, S. (2011). Memory-based immigrants for ant colony optimization in changing environments. In *Applications of Evolutionary Computation*, pages 324–333. Springer.

McArthur, S. D., Davidson, E. M., Catterson, V. M., Dimeas, A. L., Hatziargyriou, N. D., Ponci, F., and Funabashi, T. (2007a). Multi-agent systems for power engineering applications - part i: Concepts, approaches, and technical challenges. *Power Systems, IEEE Transactions on*, 22(4):1743–1752.

McArthur, S. D., Davidson, E. M., Catterson, V. M., Dimeas, A. L., Hatziargyriou, N. D., Ponci, F., and Funabashi, T. (2007b). Multi-agent systems for power engineering applications - part ii: technologies, standards, and tools for building multi-agent systems. *Power Systems, IEEE Transactions on*, 22(4):1753–1759.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Meschiari, L., Harris, C., and Clarke, S. (2013). Analysis of approaches to coordinated charging of electric vehicles on the distribution grid. *International Conference on Smart Grids and Green IT Systems.*

Motamedi, A., Zareipour, H., and Rosehart, W. D. (2012). Electricity Price and Demand Forecasting in Smart Grids. *IEEE Transactions on Smart Grid*, 3(2):664–674.

Müller, S., Brown, A., and Ölz, S. (2011). Renewable energy: policy considerations for deploying renewables. *International Energy Agency.*

Nagata, T., Tao, Y., Sasaki, H., and Fujita, . (2003). A multi-agent approach to distribution system restoration. *IEEE Power Engineering Society General Meeting.*

Negenborn, R. R. and Maestre, J. (2014). Distributed model predictive control: An overview and roadmap of future research opportunities. *Control Systems, IEEE*, 34(4):87–97.

Nemry, F. and Brons, M. (2010). Plug-in hybrid and battery electric vehicles. market penetration scenarios of electric drive vehicles. Technical report, Institute for Prospective and Technological Studies, Joint Research Centre.

Nguyen, T. T. (2011). *Continuous dynamic optimisation using evolutionary algorithms.* PhD thesis, University of Birmingham.

Nissen, S. (2003). Implementation of a fast artificial neural network library (fann). Technical report, Department of Computer Science University of Copenhagen.

Northrop, L., Feiler, P., Gabriel, R. P., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Schmidt, D., Sullivan, K., et al. (2006). Ultra-large-scale systems: The software challenge of the future. Technical report, DTIC Document.

Nowé, A., Vrancx, P., and De Hauwere, Y.-M. (2012). Game theory and multi-agent reinforcement learning. In *Reinforcement Learning*, pages 441–470. Springer.

Nunna, H. and Doolla, S. (2013). Multiagent-based distributed-energy-resource management for intelligent microgrids. *Industrial Electronics, IEEE Transactions on*, 60(4):1678–1687.

OGIMET (2015). Professional information about meteorological conditions in the world.

Page, E. (1961). Cumulative sum charts. *Technometrics*, 3(1):1–9.

Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434.

Pang, Q., Gao, H., and Minjiang, X. (2010). Multi-agent based fault location algorithm for smart distribution grid.

Park, D., El-Sharkawi, M., Marks, R., Atlas, L., and Damborg, M. (1991). Electric load forecasting using an artificial neural network. *IEEE Transactions on Power Systems*, 6(2):442–449.

Ramchurn, S. D., Vytelingum, P., Rogers, A., and Jennings, N. (2011). Agent-based control for decentralised demand side management in the smart grid. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 5–12. International Foundation for Autonomous Agents and Multiagent Systems.

Ramchurn, S. D., Vytelingum, P., Rogers, A., and Jennings, N. R. (2012). Putting the'smarts' into the smart grid: a grand challenge for artificial intelligence. *Communications of the ACM*, 55(4):86–97.

Riedmiller, M., Moore, A., and Schneider, J. (2001). Reinforcement learning for cooperating and communicating reactive agents in electrical power grids. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, pages 137–149. Springer.

Robu, V., Kota, R., Chalkiadakis, G., Rogers, A., and Jennings, N. R. (2012). Cooperative virtual power plant formation using scoring rules. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1165–1166. International Foundation for Autonomous Agents and Multiagent Systems.

Rosman, B. (2014). Context-based Online Policy Instantiation for Multiple Tasks and Changing Environments. *Annual Symposium of the Pattern Recognition Association of South Africa*, pages 1–9.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach.* Prentice Hall.

Salkham, A. and Cahill, V. (2010). Soilse: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 531–538.

Sandell, N., Varaiya, P., Athans, M., and Safonov, M. (1978). Survey of decentralized control methods for large scale systems. *Automatic Control, IEEE Transactions on*, 23(2):108–128.

Schaerf, A., Shoham, Y., and Tennenholtz, M. (1995). Adaptive load balancing: A study in multi-agent learning. *arXiv preprint cs/9505102.*

Shimoda, E., Numata, S., Baba, J., Nitta, T., and Masada, E. (2012). Operation planning and load prediction for microgrid using thermal demand estimation. In *Power and Energy Society General Meeting, 2012 IEEE*, pages 1–7. IEEE.

Shoham, Y. and Leyton-Brown, K. (2008). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations.* Cambridge University Press.

Shoham, Y., Powers, R., and Grenager, T. (2003). Multi-agent reinforcement learning: a critical survey. *Web manuscript.*

Stone, P. and Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383.

Sustainable Energy Authority of Ireland (2015). Smart grid roadmap.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to reinforcement learning.* MIT Press.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337.

Taylor, J. W. and Mcsharry, P. E. (2008). Short-Term Load Forecasting Methods : An Evaluation Based on European Data. *EEE Transactions on Power Systems*, pages 2213–2219.

Tesauro, G., Jong, N. K., Das, R., and Bennani, M. N. (2006). A hybrid reinforcement learning approach to autonomic resource allocation. In *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, pages 65–73.

Thrun, S. B. (1992). The role of exploration in learning control. *Handbook of intelligent control: Neural, fuzzy and adaptive approaches.*

Thrun, S. B. and Möller, K. (1992). Active exploration in dynamic environments. In *Advances in neural information processing systems*, pages 531–538.

Tidemann, A., Høverstad, B., Langseth, H., Ozturk, P., et al. (2013). Effects of scale on load prediction algorithms. In *Electricity Distribution (CIRED 2013), 22nd International Conference and Exhibition on*, pages 1–4. IET.

Tillotson, P., Wu, Q., and Hughes, P. (2004). Multi-agent learning for routing control within an internet environment. *Engineering Applications of Artificial Intelligence*, 17(2):179–185.

Treadgold, N. and Gedeon, T. (1998). Simulated annealing and weight decay in adaptive learning: the sarprop algorithm. *Neural Networks, IEEE Transactions on*, 9(4):662 –668.

Trojanowski, K. and Michalewicz, Z. (1999). Searching for optima in non-stationary environments. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages –1850 Vol. 3.

Trung, N. T. (2013). Scalable model-based reinforcement learning in complex, heterogeneous environments.

Tsoukalas, L. and Gao, R. (2008). From smart grids to an energy internet: Assumptions, architectures and requirements. In *Electric Utility Deregulation and Restructuring and Power Technologies, 2008. DRPT 2008. Third International Conference on*, pages 94–98. IEEE.

Turner, J. A. (1999). A realizable renewable energy future. *Science*, 285(5428):687–689.

Upton, E. (2015). Raspberry pi zero: The $5 computer.

U.S. Department of Energy at Pacific Northwest National Laboratory (2015). GridLAB-D.

Valogianni, K., Ketter, W., and Collins, J. (2014). Learning to schedule electric vehicle charging given individual customer preferences. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 1591–1592. International Foundation for Autonomous Agents and Multiagent Systems.

Vandael, S., Boucké, N., Holvoet, T., De Craemer, K., and Deconinck, G. (2011). Decentralized coordination of plug-in hybrid vehicles for imbalance reduction in a smart grid. In *The 10th International Conference on Autonomous Agents and Multiagent*

Systems-Volume 2, pages 803–810. International Foundation for Autonomous Agents and Multiagent Systems.

Vandael, S., Claessens, B., Hommelberg, M., Holvoet, T., and Deconinck, G. (2013a). A scalable three-step approach for demand side management of plug-in hybrid vehicles. *Smart Grid, IEEE Transactions on*, 4(2):720–728.

Vandael, S., Holvoet, T., Deconinck, G., Kamboj, S., and Kempton, W. (2013b). A comparison of two giv mechanisms for providing ancillary services at the university of delaware. In *Smart Grid Communications (SmartGridComm), 2013 IEEE International Conference on*, pages 211–216. IEEE.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer Science & Business Media.

Wai, R.-J., Chen, Y.-C., and Chang, Y.-R. (2011). Short-term load forecasting via fuzzy neural network with varied learning rates. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pages 2426–2431. IEEE.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Weidlich, A. and Veit, D. (2008). A critical survey of agent-based wholesale electricity market models. *Energy Economics*, 30(4):1728–1759.

Weinberg, M. and Rosenschein, J. S. (2004). Best-response multiagent learning in non-stationary environments. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 506–513. IEEE Computer Society.

Werbos, P. J. (1994). *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*, volume 1. John Wiley & Sons.

Weyns, D., Parunak, H. V. D., Michel, F., Holvoet, T., and Ferber, J. (2005). Environments for multiagent systems state-of-the-art and research challenges. In *Environments for multi-agent systems*, pages 1–47. Springer.

Whittle, P. (1951). *Hypothesis testing in time series analysis*, volume 4. Almqvist & Wiksells boktr.

Wiering, M. (2000). Multi-agent reinforcement learning for traffic light control. In *ICML*, pages 1151–1158.

Wijaya, T. K., Humeau, S. F. R. J., Vasirani, M., and Aberer, K. (2014). Individual, aggregate, and cluster-based aggregate forecasting of residential demand. Technical report, Technical report, EPFL, Lausanne, Switzerland.

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3):324–342.

Wu, J., Xu, X., Zhang, P., and Liu, C. (2011). A novel multi-agent reinforcement learning approach for job scheduling in grid computing. *Future Generation Computer Systems*, 27(5):430–439.

Zeng, S., Shi, H., Kang, L., and Ding, L. (2007). Orthogonal dynamic hill climbing algorithm: Odhc. In Yang, S., Ong, Y.-S., and Jin, Y., editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 79–104. Springer Berlin Heidelberg.

Zhang, C., Lesser, V. R., and Shenoy, P. J. (2009). A multi-agent learning approach to online distributed resource allocation. In *IJCAI*, pages 361–366.

Zhang, G., Eddy Patuwo, B., and Y Hu, M. (1998). Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62.

Zhao, B., Xue, M., Zhang, X., Wang, C., and Zhao, J. (2015). An mas based energy management system for a stand-alone microgrid at high altitude. *Applied Energy*, 143:251–261.