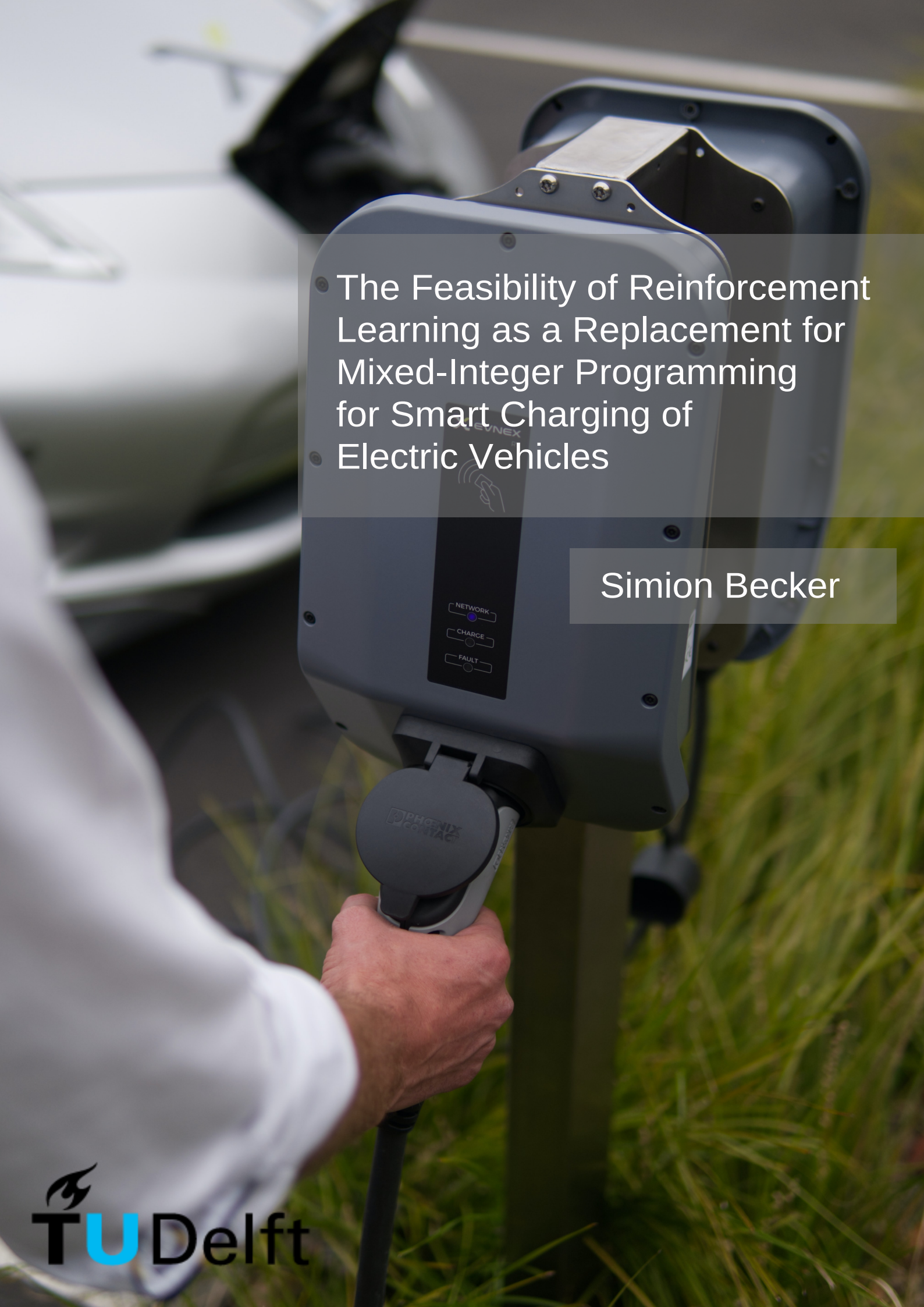

The Feasibility of Reinforcement Learning as a Replacement for Mixed-Integer Programming for Smart Charging of Electric Vehicles

Simion Becker

TU Delft

Cover image courtesy of Ed Harvey, https://unsplash.com/photos/SvDqCT8zmL0

Delft University of Technology

The Feasibility of Reinforcement Learning as a replacement for Mixed-Integer Programming for Smart Charging of Electric Vehicles

by

Simion Becker

to obtain the degree of Master of Science
in Sustainable Energy Technology
at the Delft University of Technology,
to be defended publicly on Monday March 28, 2022 at 10:00 AM.

Supervisors:
Daily supervisor: Dr. ir. Gautham Ram
PhD supervisor: Ir. Yunhe Yu
PhD supervisor: Ir. Ksenija Stepanovic

Thesis Committee:
Prof. dr. ir. Bauer
Dr. ir. Gautham Ram
Dr. ir. Jochen Cremer

Student Number: 5150345

An electronic version of this thesis is available at
http://repository.tudelft.nl

TU Delft Delft University of Technology

PREFACE

With a background in Theoretical Physics and Sustainable Energy Technology, the topic of Smart Charging with Reinforcement learning was new to me, but a great challenge that I enjoyed all the aspects of. The learning curve that I went through was immense and I am very thankful for the opportunity the DCE&S faculty has provided me with. This project marks the end of my academic career and the past 2 years at the Technical University of Delft have inspired me and created a lot of memorable memories that I will prosper for the rest of my life. The Covid-19 circumstances have certainly changed the thesis process, but since the meetings and feedback loops were arranged very well and efficiently online, ultimately it had little impact.

Throughout this research, I have gained new skills in Machine Learning and Reinforcement Learning and I have gained expert knowledge on the advantages and disadvantages of reinforcement learning for smart charging. From the start of my thesis I was given a lot of freedom to explore different opportunities, which has helped me to be creative and has sparked my intrinsic motivation to make this project a success.

However, of course, I could not have completed this research on my own. Therefore, I would like to express my deepest appreciation to my daily supervisor Yunhe Yu for providing me with expert knowledge and for her guidance throughout the past year. Yunhe motivated me when there were obstacles on the road, provided new insights, and was always available for questions. Also, I am extremely grateful to my bi-weekly supervisor Ksenija Stepanovic, who extended a great amount of assistance. In our bi-weekly meetings, due to her extensive knowledge and experience with Reinforcement Learning Ksenija provided unparalleled support which was instrumental in making progress. I'd also like to extend my gratitude to supervisor Dr. ir. Gautham Ram Chandra Mouli, since the completion of this study could not have been possible without his support and assistance. I always enjoyed our monthly meetings, in which Dr. Ir. Gautham Ram Chandra Mouli was always extremely quick in grasping the root of my challenges, provided ingenious suggestions, and showed profound belief in my abilities.

Also, I would like to extend my sincere thanks to my girlfriend and my parents, who have provided me with a lot of mental support throughout this thesis and have consistently motivated me to do my best. They were especially helpful during the time that I was struggling to fit all my tasks into a single week when trying to improve the performance of the Reinforcement Learning algorithms, being active as a strategy consultant for De Kleine Consultant, and while being in the process of looking for a job for after graduating. I am very thankful for their support and I am lucky to have them around.

Last but not least, I would also like to thank Prof. dr. ir. Bauer and Dr. ir. Jochen Cremer for sitting in the panel and providing critical and constructive feedback to my research. Thank you for taking the time to evaluate my thesis. I am very much looking forward to your evaluation.

Simion Becker
March 22, 2022
Amsterdam

ABSTRACT

As the world is currently actively trying to reduce the consumption of fossil fuels, large investments are done in renewable energy sources and ways are sought after to electrify fossil fuel-intensive sectors. In line with these developments, the number of electric vehicles requiring access to the electric power grid has exploded putting increased pressure on the grid. One way to decrease the congestion in the grid is to make use of smart charging schedules for electric vehicles, with the objective of reducing peak demand and preventing the overloading of cables and transformers while reducing the cost of charging for electric vehicle owners.

The recent increase in the availability of real-life data has allowed the in-depth study of smart charging dynamics on a large scale through modeling and simulation. Mathematical optimization is a method that is often used to generate smart charging plans in state-of-the-art smart charging applications. However, while mathematical optimization can be very effective, as the objective function expands and more parameters are taken into account, the optimization becomes more complex and therefore require faster and smarter optimization algorithms. In addition, the recent availability of a vast amount of real-life data sets has made efficient data handling more important.

Machine learning is known to be an effective way of introducing Artificial Intelligence to smart charging algorithms. The use of reinforcement learning algorithms, a subset of machine learning could help overcome the disadvantages of mathematical optimization as trained algorithms are generally fast when predicting outcomes and have the potential to be accurate at the same time.

Therefore, the purpose of this work is to research the feasibility and additional benefit of machine learning to mathematical optimization-based smart charging algorithms. This is done through the development of end-to-end Q-learning and Double Deep Q Network reinforcement learning smart charging algorithms. The performance of both algorithms is evaluated on three separate case studies as well as on multiple different random cases and is compared to the charging performance of the average rate charging and mixed-integer programming algorithm benchmarks.

As a result, it becomes clear that for individual case studies the Q-learning and Double Deep Q Network agent are able to find cheap charging moments while charging the vehicle to 100% battery capacity without violating charging constraints. However, when testing the performance of the Q-learning and Double Deep Q Network agents it becomes clear that the average charging performance is significantly worse than using the method of mixed-integer programming as the algorithms do not learn to generalize well.

Finally, the advantages and disadvantages of replacing mixed-integer programming with reinforcement learning are discussed as well as some limitations and recommendations for future work and improvement are given.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AC	Alternating Current
AVR	Average Rate Charging
CC	Constant Current
CPO	Charge Point Operator
CPO	Central Processing Unit
CV	Constant Voltage
DNN	Deep Neural Network
DQN	Deep Q-Network
DDQN	Double Deep Q-Network
EV	Electric Vehicle
GPU	Graphical Processing Unit
HER	Hindsight Experience Replay
ML	Machine Learning
MILP	Mixed-Integer Linear Programming
MINLP	Mixed-Integer Non-Linear Programming
MIP	Mixed-Integer Programming
NN	Neural network
OSCD	Orchestrating Smart Charging in mass Deployment
PV	Photovoltaic
PER	Prioritizied Experience Replay
LP	Linear Programming
LR	Linear Regression
RL	Reinforcement learning
SOC	State of Charge
SVR	Support Vector Regression
TTL	Time-To-Leave

1 INTRODUCTION

As the world is currently actively trying to reduce the consumption of fossil fuels and the output of GHG emissions in the atmosphere, electric vehicles (EVs) are becoming more and more popular. According to the International Energy Agency, the transport sector is responsible for roughly 25% of the total energy consumption and therefore EVs can contribute heavily to the decarbonization of road transport [1]. In 2030 the number of EVs is even expected to rise up to 125 million [2]. Unfortunately, this transition to EVs also brings along some challenges. To start with, as the number of EVs that require access to the grid is expanding, more public charging infrastructure needs to be installed to meet the increasing charging demand. As a consequence, the charging of EVs is expected to put the network under increased pressure. In addition, the demand for electricity is rising due to electrification, and the share of fluctuating renewable energy on the grid is increasing, making it harder for grid operators to match supply and demand [3].

One way to decrease the grid congestion is to increase the capacity of the grid, however, grid expansion is extremely complex and requires a huge time and monetary investment. Alternatively, demand-side management solutions can be implemented to improve the match between electricity supply and demand. Current EV charging schemes are not smart, since EVs are mostly charged immediately after arrival. As a consequence, this leads to sub-optimal conditions as EVs are charged "blindly" and do not take the grid constraints, electricity prices, and the presence of other EVs into account. Therefore, the implementation of smart charging of EVs is considered a viable option to reduce the peak demand and prevent the overloading of cables and transformers [4].

1.1 SMART CHARGING FUNDAMENTALS

The recent increase in the availability of real-life data has allowed the in-depth study of smart charging dynamics on a large scale through modeling and simulation. The goal of smart charging is the scheduling of the charging process of EVs in a "smart" way. However, the definition of "smart" can differ for every case and heavily depends on which parameters are taken into account in the smart charging algorithms. On a small scale, a smart charging algorithm can be applied to a single EV charging station, while on a large scale multiple nodes with each multiple charging stations can be considered.

To illustrate smart charging, one can consider the following example of having an EV owner plug an EV into a residential charger when returning home from work around 6-7 PM. However, around this time electricity prices are generally relatively high, and charging at a later moment at night would lower the total charging cost for the consumer while reducing peak demand. A smart charging algorithm implemented in the back-end office of the Charge Point Operator (CPO) or in the energy management system at home could coordinate the charging of the connected EV, ultimately communicating the optimal time of charging to the EV based on current and future electricity prices.

There are two main classes of charging algorithms; uncontrolled and controlled charging. When charging happens uncontrolled, EVs are free to access the grid, and therefore the charging of an EV is not restricted under certain circumstances. However, with controlled charging, the opposite is the case, since the charging of EVs is tightly scheduled. Smart charging can be considered a sub-set of controlled charging. A further dimension in the classification of smart charging algorithms is whether the algorithm runs offline or online. When a smart charging algorithm runs offline, the optimal scheduling is based on EV data known beforehand. With online smart charging the charging schedule is based only on current data and past data from EVs that have already reached the charging station. Therefore, with online smart charging, there is no information available about future EVs arriving at the charging station [5]. While offline smart charging results can be effective, in practice, EV data is generally not available before the EV arrives. Therefore, online smart charging is more suitable for real-life appli-

cations. Often, uncontrolled or offline charging is used as a benchmark to judge the performance of smartly controlled online charging algorithms [5].

A third dimension in the classification of smart charging algorithms is whether the algorithm is built centralized or decentralized. While centralized smart charging algorithms are based on a single aggregator that looks at one or multiple objective functions for the complete system, decentralized smart charging algorithms consider the objective function(s) of a local node or charging point [6]. It is also possible that centralized and decentralized algorithms each operate on a different hierarchical level and are combined in a single model, this is called hierarchical control. However, when combining a centralized smart charging algorithm with a decentralized algorithm, one needs to ensure that both the global objective of the complete system does not clash with the objective of the smaller subset of the complete system. In particular, decentralized algorithms are often preferred over centralized algorithms when taking local factors such as local loads and residential photovoltaic (PV) energy generation into account.

While the smart charging example mentioned above is a simple example of a smart charging algorithm, many different approaches have been applied to charge EVs in a smarter way [7, 8, 9]. Usually, more sophisticated online controlled charger algorithms make smart use of multiple information sources and work in an integrated way to reach the objective function of the algorithm. The performance of these state-of-the-art algorithms is often judged on cost reduction for the economical objective function and for technical objective functions the algorithms are generally judged on peak reduction and network loss reduction [10, 11, 12, 13, 14].

1.2 MATHEMATICAL OPTIMIZATION AND MACHINE LEARNING

Mathematical optimization is a method that is often used to generate optimal charging plans in state-of-the-art smart charging applications. This method is applied in many different industries, including science and engineering, and is a way of finding the best possible solution to a decision problem given an objective function and a set of constraints. Different classes of optimization problems exist and are generally based on whether the objective function is linear, non-linear, convex, or quadratic and whether the constraints are linear or non-linear, and whether the decision variables are discrete or continuous [15].

While there are different optimization methods worth diving into, Mixed-integer Programming (MIP) is discussed in this section since it is one of the most used optimization methods and it will be one of the benchmarks used in this thesis [16]. MIP is a class of optimization in which the objective function is linear or non-linear, the constraints are linear and the variables are both continuous and discrete. When considering subclasses of MIP, a MIP problem is classified as mixed-integer linear programming (MILP) when the objective function and constraints are linear and is classified as mixed-integer non-linear programming (MINLP) when the objective function and/or constraints are non-linear. An algorithm that is often used to solve MIP optimization problems is the Branch-and-Bound algorithm. The algorithm systematically goes through the search space X which contains all possible answers to the optimization problem by creating a tree structure. In this tree structure branches are created by splitting up the tree in smaller subsets (S_1, S_2 ... S_n) of the search space, and rules are used to bound the search space. When there is a possible better solution $\hat{x}'$, the solution is stored for comparison to the current best solution $\hat{x}$. If no solution in S is better than $\hat{x}$, the corresponding subset is pruned and removed from list L. The best possible solution $\hat{x}$ based on the objective function $f(x)$ is returned after the tree has been explored and there are no subsets left in L [17].

While mathematical optimization can be very effective, as the objective function expands and more parameters are taken into account, the optimization becomes more complex as parameters are often interrelated and therefore require faster and smarter optimization algorithms. The recent availability of a vast amount of real-life data sets has made efficient data handling more important and the use of Artificial Intelligence (AI) is increasingly more often considered for improvement of the functionalities of current smart charging algorithms. machine learning (ML), in particular, a subset of AI, is known to be an effective way of introducing AI to smart charging algorithms. The advantage of machine learning

is that it allows the study, creation, and application of algorithms that learn from input data to output predictions [18]. However, the success of an machine learning algorithm depends highly on the quality of the input, since the higher the quality of the input data the better the output will be.

Within machine learning there are three main categories; supervised learning, unsupervised learning, and reinforcement learning (RL). While supervised learning works with labeled data to train the ML algorithm to provide the correct output, unsupervised learning does not make use of labels. In unsupervised learning, the algorithm learns to recognize patterns in data through clustering, since the data is unlabeled. Furthermore, a relatively new machine learning technique that has gained a lot of attention over the past few years is reinforcement learning. In reinforcement learning, an ML model is trained through trial and error with a reward function that aims to maximize the cumulative award [19].

Another important distinction in machine learning that can be made is the difference between regression and classification. While regression algorithms in machine learning concern the prediction of one or more quantitative values based on the input data, classification algorithms predict one or more labels based on the input. In general, supervised learning makes use of both regression and classification algorithms, while unsupervised learning makes use of clustering or dimensionality reduction algorithms. In contrast to supervised and unsupervised learning, reinforcement learning algorithms are decision-making algorithms. An overview of the different types of machine learning algorithms is provided in figure 1.2.1. Within these overarching categories, there are many different machine learning algorithms available and some are more suitable for certain applications than others. The most common supervised and unsupervised machine learning algorithms include K-nearest Neighbours, Decision Tree's, Random Forest algorithm, Neural networks (NN), and Support Vector Machines. Often used reinforcement learning algorithms are Q-learning, Deep Q-networks (DQN), and actor-critic reinforcement learning. The rationale for the selection of the algorithms under study in this thesis will be further elaborated on in chapter 4.

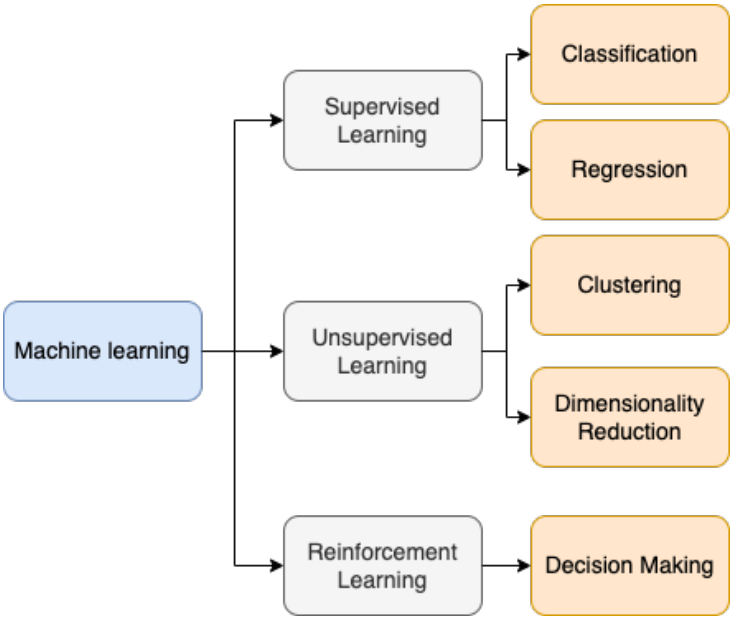

Figure 1.2.1: Different types of machine learning

1.3 RESEARCH MOTIVATION

It has been shown that MIP optimization algorithms are very effective in creating smart charging schedules since they are a good tool for achieving objectives while taking the grid, node, charger, and EV constraints into account [20]. Unfortunately, despite these optimization algorithms performing relatively well, for certain cases, a choice often has to be made between a fast computational time and a

high accuracy, since a higher accuracy leads to slower computational times. A computational time of multiple minutes limits the feasibility of the algorithm in real-life and slows down the research process due to long simulation times. Especially when implementing the optimization algorithm in an offline environment, the optimization has been proven to run slow. On the other side, a low accuracy is undesirable for future smart charging algorithms as true optimization is not achieved.

The implementation of machine learning could help solve this problem as trained algorithms are generally fast when predicting outcomes and have the potential to be accurate at the same time. Furthermore, while there is a significant amount of fundamental research on combining ML with optimization, little to no research has been focused on the potential of machine learning algorithms to either replace or work alongside current state-of-the-art LP/MILP smart charging algorithms. Therefore, a more thorough understanding of the feasibility of the application of reinforcement learning as a replacement for smart charging mathematical optimization algorithms will fill this research gap and could ultimately lead to a better performance of smart charging algorithms.

1.4 RESEARCH OBJECTIVES

The main objective of this study is to develop a fast and accurate machine learning model for smart charging and to evaluate its feasibility and output. Therefore, the main research question is the following:

What is the feasibility and benefit of reinforcement learning to using traditional LP/MILP optimization for smart charging of electric vehicles?

The following three sub-questions are introduced:

- *What is the charging performance and cost reduction potential of reinforcement learning algorithms compared to traditional LP/MILP optimization for smart charging of electric vehicles?*
- *What is the speed of the reinforcement learning smart charging algorithms compared to traditional LP/MILP optimization for smart charging of electric vehicles?*
- *What are the main advantages and disadvantages of reinforcement learning algorithms with respect to flexibility, scalability and the safeguarding of constraints compared to traditional LP/MILP optimization for smart charging of electric vehicles?*

The main contributions of this work are:

1. A literature review of the application of machine learning algorithms in combination with optimization and an overview of Q-learning and Deep Q-network reinforcement learning state-of-the-art approaches for smart charging
2. Development of a Q-learning reinforcement learning model for smart charging with the objectives of cost reduction and fully charging the vehicle
3. Development of a Double Deep Q-network (DDQN) reinforcement learning model with Prioritized Experience Replay (PER) for smart charging with the objectives of cost reduction and fully charging the vehicle
4. Evaluation of the charging performance, cost reduction, and speed of the developed reinforcement learning smart charging algorithms in comparison to LP/MILP and average rate charging

1.5 THESIS OUTLINE

This thesis exists of nine chapters and the content is organized as follows. First, in chapter 2 a review of the literature available on the main approaches regarding the combined fields of mathematical

optimization and machine learning is provided. Subsequently, chapter 3 explains the fundamental theory behind the Q-learning and DDQN reinforcement learning algorithms and covers a number of state-of-the-art approaches. In chapter 4, the methodology used in this work is elaborated on and the different steps required to reach the research objectives are laid out. Furthermore, the specific Q-learning and DDQN algorithm setup is discussed in chapter 5 and the specific case studies and training procedures used for evaluation are presented in chapter 6. Next, in chapter 7 the trained algorithms are evaluated based on different performance metrics for these case studies. Finally, in chapter 8 the evaluation results are explained and analyzed in more detail after which in chapter 9 the main conclusions and recommendations are discussed.

2 LITERATURE REVIEW

The purpose of this chapter is to provide a review of the literature available on the main approaches, algorithms, and research results regarding the combined fields of smart charging, optimization, and machine learning. First section 2.1 provides an overview of the three different approaches identified in literature regarding the integration of optimization and machine learning. Subsequently, in subsections 2.2 until 2.4 relevant research on each approach to combining ML and optimization is discussed.

2.1 OPTIMIZATION AND MACHINE LEARNING

Despite the fact that the integration between optimization and machine learning is in an early stage of development, different reviews have been written that cover multiple approaches for combining optimization with machine learning [16, 21]. In both reviews, the main goal of combining optimization with ML has been recognized to be to speed up time-consuming optimization computations. While [21] identifies three main approaches to combining ML and optimization, [16] bundles two approaches together coming up with the following two main categories of approaches: (**1**) End-to-End learning and (**2**) Optimisation augmented by ML. Based on the review of [21], the distinction can be made between three different approaches by adding a third main approach to the list: (**3**) ML to improve optimization input. This leaves us with the following three approaches:

1. End-to-End learning
2. Optimisation augmented by ML
3. ML to improve optimization input

To start with, End-to-End learning is described as the approach that trains an ML model to predict solutions that the optimizer would normally output by being trained on the optimizer inputs. This method has the potential to be significantly faster than the original optimization algorithm in real-time since the learning phase of the ML model has already happened offline in an earlier stage [16]. Furthermore, the second main approach to combining ML and optimization described by [16] is to use an ML model to augment the optimization algorithm to increase its speed. Finally, the third approach entails the use of ML to improve the input data to the optimizer. While this is not likely to significantly improve the computational time of the algorithm, it can be effective in improving the accuracy of the optimization algorithm. Recent research findings on these three approaches are discussed in the subsections below.

2.2 END-TO-END LEARNING APPROACH

In contrast to [21], [16] takes the approach of *End-to-End learning* a bit further and divides it up into two sub-approaches: (**1**) *learning with constraints* and (**2**) *learning on graphs*. With learning on constraints, the input data that is fed to the ML model is a vector of different problem instances each containing inherent knowledge of the constraints that can be learned by the ML model. The second approach, learning on graphs, is a method to learn solutions from optimization algorithms by representing the problem on a graph. Both approaches can be tackled through supervised learning and reinforcement learning.

An example of research that was successful in applying end-to-end learning through learning on constraints, is the work by [22] on combining ML and mathematical optimization to predict the optimal production of offshore wind parks through supervised learning. The authors trained a Linear Regression (LR), Neural Network, and Support Vector Regression model (SVR) on the optimized solutions of

a MIP algorithm to replace the slow MIP algorithm. As a result, they found that ML models and the SVR are effective in producing fast and accurate values close to the optimized values.

In addition, another research applies End-to-End learning to smart charging by training different ML models on the output of a dynamical programming model that computes the optimal charging scheduling for 17 EVs, taking gasoline consumption into account [23]. As a result, it was found that out of multiple ML models, the deep neural network (DNN) algorithm performed the best resulting in solutions very close to the global optimum.

Specifically in the field of smart charging reinforcement learning has been identified as an effective way to build a functioning EV scheduling algorithm [19, 24, 25]. An example of research on a decentralized SC algorithm that takes renewable energy into account is the research by [25] who have built an online Q-learning reinforcement learning model for the optimization of the revenue of an EV charging station connected to a renewable energy source. As a result, the total costs were reduced by 40-80% compared to an algorithm that makes random decisions. More complex functionalities such as the request for ramping capacity and the total available power on the grid were not taken into account.

With regard to learning on graphs, Graph Neural Networks can be effective in modeling a set of nodes and their individual relationships [26]. It can be used to identify patterns and classes that were not openly visible. As an example, in the research by [27], a graph neural network was trained for the quadratic assignment problem. By providing the graph neural network with the input and corresponding solved instances, it was able to generalize and produce results on new inputs from the same distribution.

2.3 OPTIMIZATION AUGMENTED BY ML APPROACH

The second approach is the approach of optimization augmented by ML. This approach can be split up into three sub-approaches: (**1**) learning methods to configuration, (**2**) learning methods to branching and (**3**) learning methods to cutting planes [16, 28].

To start with, the first approach is an approach in which an ML model is trained to output the right configuration parameters for the optimizer, based on data on previously successful parameters. Choosing the right parameters from the start can significantly reduce the computational time. Examples of optimizer parameters that could require careful configuration are parameters of pre-solving operations and the step size of the algorithm. Furthermore, in contrast to approach one, approaches two and three occur during the optimization itself as lower-level decisions of the optimizer are replaced by ML models ultimately improving the total algorithm computational time. The second approach of learning methods to branching is defined as learning the choice of variables to branch on, which can dramatically change the size of the branch-and-bound tree, hence the solving time. Finally, the third approach of learning methods to cutting plane selection entails the selection of planes to cut from the search space to improve the speed of the optimization algorithm.

An example of learning methods to configuration is the research by [29], which uses ML to classify whether Mixed-Integer Quadratic Programming should be linearized or not, since linearizing the MIQP can speed up the problem-solving time. As a result, this research showed that the best performing ML algorithms are the Support Vector Machine and the Radial Function with an accuracy of around 80-90 % in predicting whether a MIQP should be linearized or not.

With respect to learning methods to branching, the research performed by [30] shows the feasibility of this approach. In the research, a framework is built for ML models to learn how to speed up the process of branching. This is done by first observing the branching decisions and then training a ranking ML model to mimic the branching decisions in a faster way. Results show that significant gains in computational time were achieved compared to the original branching method. Furthermore, in a recent paper, it was shown that branching governed by ML can be effectively done by learning a deep neural network to make the right variable selection decision [31]. In this research, the authors were able to speed up the process by a factor of 2-10.

As a final example, the selection of the right cutting planes based on an ML model has been applied

for quadratic semi-definite outer approximation via neural networks by [32]. In this research, each cut is ranked on the improvement of the objective to train the ML model to output the best cuts. As a result, the research demonstrates that their proposed cutting method is computationally light and easy to integrate.

2.4 ML TO IMPROVE OPTIMIZATION INPUT APPROACH

The final main approach is the approach of using ML to improve the input to the mathematical optimizer. This approach is different than the first two approaches, since it focuses on the improvement of the input values to optimization algorithms, instead of fundamentally changing the optimization algorithm itself. In particular, this method is of added value for input values that are currently based on inaccurate approximations or that are hard to describe through functions and therefore are not included in the optimization algorithm. An example of research that uses ML to improve the input of the optimizer is a work in which non-linear battery behavior is modeled with the use of neural networks and subsequently used in the optimizer [33]. The authors use a large dataset of uncontrolled charging sessions to train a regression model to estimate the maximum power that can be provided to the EV based on features such as the current state-of-charge (SOC) of the vehicle. Subsequently, the optimization algorithm takes the estimate and alters the charging capacity allocation accordingly. As a result, the mean SOC is increased significantly by up to 21% compared to the algorithm that assumes there to be no specific charging profile. Another example of this approach is research by the same authors that uses linear regression to predict the departure time of EVs, which results in a mean absolute prediction error of 141 minutes [34]. More examples of input values that have been approximated by ML models to improve the optimization are the EV idle time to improve prioritization of EVs and user classification to improve the smart charging actions [35, 36]. An overview of the different approaches and sub-approaches to combining machine learning and optimization can be found in figure 2.4.1

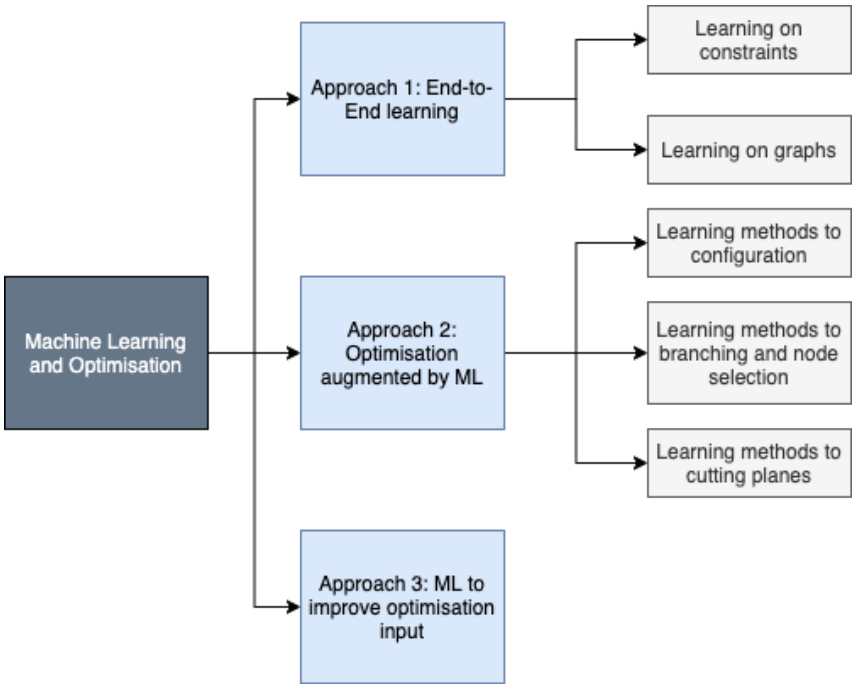

Figure 2.4.1: An overview of the different approaches and sub-approaches identified in literature to combining machine learning and optimization

3 REINFORCEMENT LEARNING FOR SMART CHARGING

Before the models can be discussed in further detail it is important to understand the fundamental theory behind the reinforcement learning algorithms used in this thesis for smart charging. This chapter is organized as follows. First, in section 3.1 the fundamentals of Q-learning and the available literature on Q-learning and smart charging is discussed. Next, in section 3.2, an introduction is provided to Deep Q-Networks. Subsequently, in section 3.3 the recent state-of-the-art Q-learning and DQN reinforcement learning methods are covered. Finally, in section 3.4 and in section 3.5 the recent literature on the application of Q-learning and DQN on smart charging is discussed.

3.1 Q-LEARNING FUNDAMENTALS

Q-learning is a model-free reinforcement learning algorithm that makes use of an offline trained Q-value function, to teach an agent how to make decisions that support the objective function. An episode is considered a complete sequence of time steps with a corresponding state and action, which ends with a terminal state. For illustration purposes, if Q-learning was used for a game of PAC-MAN, a single episode could be set to be the exact position of the PAC-MAN within the maze (state) and the next possible movements (actions) until the PAC-MAN either eats all of the dots or gets eaten by a colored ghost (the terminal state). The state s of the agent, which is the PAC-MAN in this case, is a representation of the environment that exists of a set of state-values describing the situation that the PAC-MAN is in. In PAC-MAN state-values could for example be the x coordinate of the PAC-MAN, the y coordinate of the PAC-MAN, and the distance to the colored ghost. In order to teach the PAC-MAN to survive, a positive reward can be given to the PAC-MAN for eating all dots and a negative reward can be given if the PAC-MAN gets eaten by the colored ghost.

In Q-learning, it is the goal of the agent to maximize the cumulative discounted reward in a chain of actions that map between subsequent states. For a single step, within one episode, the algorithm starts in state s and takes an action a which brings the agent to the new state s'. In every step, the decision which action to take depends on the Q-values stored in the Q-table, in which every Q-value is linked to a particular action and state. Whether the agent uses the Q-table or takes a random action is determined by the epsilon-greedy policy, which depends on the value of ϵ. According to the epsilon-greedy policy, in every time step, a random value between 0 and 1 is computed. If the random value is larger than the ϵ value between 0 and 1, a random action is assigned. However, if the randomly generated value is larger than ϵ, the Q-table is exploited by searching for the maximum Q-value $Q(s, a)$, based on the current state s that the agent is in by taking the corresponding action. Through the decay of ϵ during the training process, the policy allows for sufficient random exploration at the start to make sure that different actions are experimented with to discover optimal actions, while towards the ending lower ϵ values allow the exploitation of the Q-table.

After an action is taken, the agent interacts with the environment to determine the new state s' and the next state-action pair $Q(s', a')$ is determined. Next, the old Q-value $Q(s, a)$ is replaced by a new value based on the reward function and stored in the Q-table under the corresponding specific state and action position. The value of this new Q-value stored in the Q-table $Q(s, a)$ is determined by the bellman equation below:

$$\text{Q(s,a)} = (1\text{-}\alpha)\ \text{Q(s,a)} + \alpha\ (\ \text{R} + \gamma\ \text{Q(s',a')}\ ) \tag{3.1}$$

There are a number of different variables in the Bellman equation which are described as follows:

- The reward R: Since it is the goal of the Q-learning algorithm to maximize the cumulative reward, the reward R is a value determined by a reward function in every time step to measure how well the action has led to the desired performance based on the objective function.

- The learning rate α: The learning rate α is the weight that determines to which degree the future Q value of the next state s' is taken into consideration as opposed to the current Q-value in determining the updated Q value.
- The discount factor γ: The discount factor γ discounts the Q-value corresponding to the next state s' and is used for determining the current Q-value.

A schematic overview of the Q-learning process is provided in figure 3.1.1

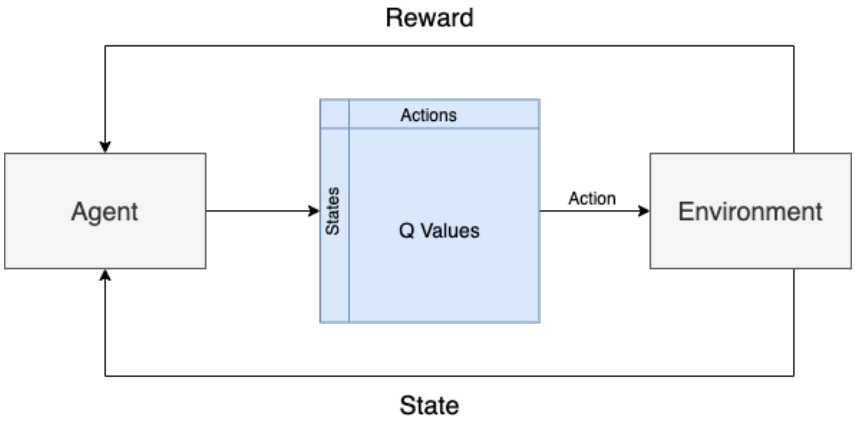

Figure 3.1.1: A schematic overview of the main components of the Q-learning algorithm including the agent, environment, and Q-table.

After a significant amount of episodes, the Q-values in the Q-table converge as the learning curve flattens. Subsequently, after all the training episodes have been run the algorithm comes to an end and the Q-table can be used as a look-up table to find the appropriate action for every state that the agent is in. The pseudocode for the algorithm is found below in algorithm 3.1.

Algorithm 3.1: Pseudocode of the Q-learning algorithm

Input: Hyperparameters
Output: $Q(s,a), \forall S, a \in A$

```
Initialize Q(s,a), ∀S, a ∈ A
for each episode do
    Initialize State
    for each time step do
        Select a based on S, derived from Q(s,a) depending on ε, otherwise select random
         action a
        Take action a, observe reward R and s' from environment
        Replace Q(s,a) ← (1−α) Q(s,a) + α ( R + γ Q(s',a') )
        Go to next state, s ← s'
    end
end
```

3.2 DEEP Q-NETWORK FUNDAMENTALS

The Deep Q-network algorithm was first developed by DeepMind in 2015 and combines both reinforcement learning and deep neural networks [37]. In contrast to tabular Q-learning, DQN reinforcement learning trains neural networks to predict Q-values based on the state-space input. However, similar to tabular Q-learning, an action a is chosen corresponding to the maximum Q-value depending on ϵ. The chosen action has an effect on the environment and provides the next state s' and corresponding reward R to the agent as can be seen in figure 3.2.1.

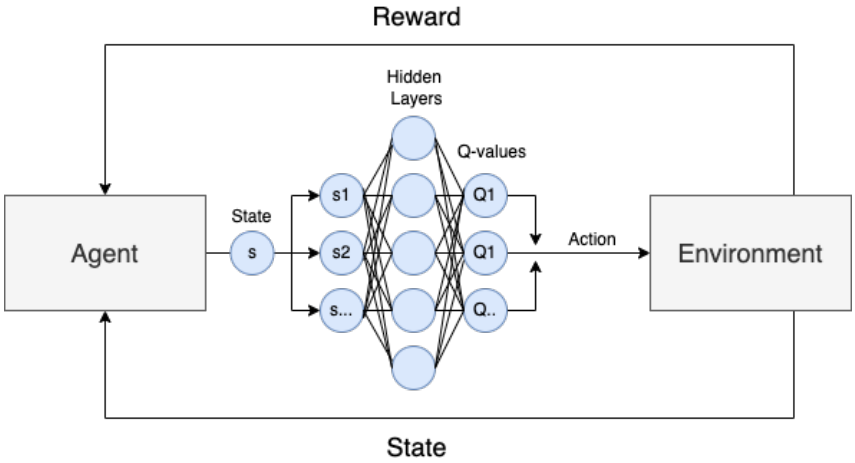

Figure 3.2.1: A schematic overview of the main components of the DQN algorithm including the agent, environment, and deep neural network.

In the process of training, the neural network is trained in every episode through supervised learning. However, since this sequence is highly sequential, in order to prevent a highly oscillating or diverging output of the neural network, an experience replay memory D is used. Therefore, after an action has been selected, the action a, the current state s, reward R, and next state s' are stored together in a replay memory D for training. The replay memory has a fixed size M_{size} and in every time step, the oldest value is replaced by the most recent value to document new experiences. However, instead of training the neural network on all stored transition (s, a, R, s') in the replay memory, a small batch of training data (minibatch) is selected with minibatch size B_{size}. This trick is effective in getting rid of the sequential bias. Next, the Q-values Y_j for supervision that the neural network is trained on are calculated according to the second part of the Bellman equation $R + \gamma Q(s', a')$.

Another issue that the DQN algorithm deals with is that since there is no lookup table, both $Q(s, a)$ and $Q(s', a')$ would be produced by the same neural network while being closely correlated since $Q(s, a)$ follows $Q(s', a')$. Therefore, a second neural network called the target network is introduced to stabilize the learning process of the agent. The main neural network is used to predict $Q(s, a)$, while the target neural network is used to predict $Q(s', a')$. After a number of episodes, the target neural network is updated with the weights of the main neural network depending on the target update frequency, as shown in figure 3.2.2:

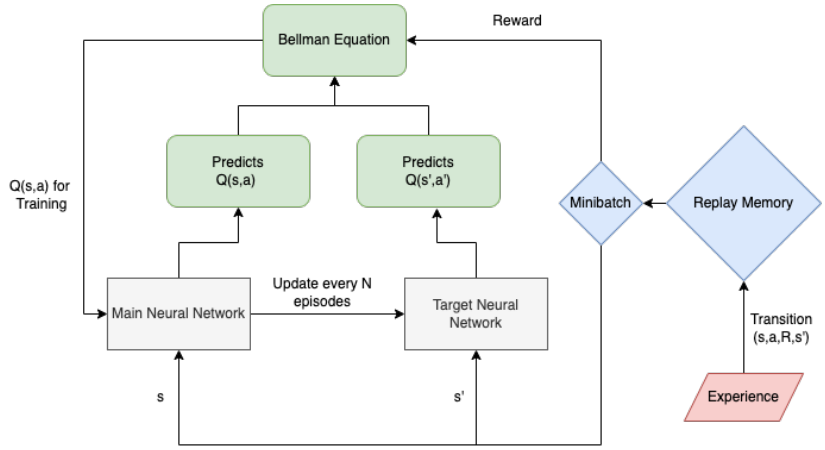

Figure 3.2.2: A schematic overview of the interaction between the replay buffer, main neural network, and target neural network in the DQN algorithm.

Because of the use of neural networks, in contrast to tabular Q-learning the algorithm works for continuous state-space values. The algorithm does not allow continuous action-space, since every

separate continuous Q-value output still responds to a discrete action. However, the neural network allows both more states and actions to be added since the size of the lookup table is no longer a computational restriction, which can lead to more accurate results. The pseudocode for the algorithm is found below in algorithm 3.2:

Algorithm 3.2: Pseudocode of Deep Q Network algorithm

Input: Hyperparameters
Output: Action-value function $Q(s,a), \forall S, a \in A$

Initialize main neural network action-value function $Q(s,a)$
Initialize target neural network action-value function $Q(s,a)$
Initialize replay buffer D
for *each episode* **do**
    Initialize State
    for *each time step t* **do**
        Select $a = max_a Q(s,a)$, depending on ϵ, otherwise select random action a
        Take action a, observe reward R and s' from environment
        Store transition (s,a,R,s') in replay memory D
        Go to next state, $s \leftarrow s'$
        Sample minibatch of transitions from replay memory D
        Set $y_j = \begin{cases} r_j & \text{if t is last time step} \\ r_j + \gamma Q(s',a') & \text{if t is not last time step} \end{cases}$
        Take gradient descent step on $(y_j - Q(s,a))^2$
    end
    if *episode number = target update frequency* **then**
        Update target network parameters
    end
end

3.3 STATE-OF-THE-ART

Since the Q-learning algorithm is relatively robust, there are little to no additional approaches mentioned in literature that can significantly improve its performance, other than, for example, reward shaping, table initialization, and prioritized sweeping [38]. However, in contrast to Q-learning, there are different approaches that have been developed for DQN reinforcement learning to improve the learning of the agent.

To start with, an alteration to the DQN algorithm that is often applied to improve its stability is called the Double Deep Q-network. DDQN is a method of decoupling the maximization of the predicted future Q-values by the target neural network from the prediction of future Q-values by the target neural network. These two are coupled since the maximum Q-value chosen by the target network directly depends on Q-values provided by the target network. The problem with this is that as a consequence the Q-values are unstable since the agent chooses the wrong maximum Q-value by overestimation of the Q-values. Therefore, the goal of DDQN is to improve the stability of the learning process by having the final action decision depend both on the Q-values of the main neural network and on the Q-values of the target neural network [39]. The epsilon-greedy policy is still used to determine whether to explore the consequences of a random action or to exploit the neural network. However, instead of only choosing an action based on the maximum value of the Q-values predicted by the target neural network, the index of the maximum value of the main neural network is used to select the "maximum" Q-value from the Q-values predicted by the target neural network. Therefore, the second part of equation 3.1 can be rewritten in the following way:

$$\mathrm{R} + \gamma\ \mathrm{Q(s',a')} \Rightarrow \mathrm{R} + \gamma\ \mathrm{Q(s',}\ \underset{a'}{\mathrm{argmax}}\ \mathrm{(Q(s',a'))}) \tag{3.2}$$

In equation 3.2 above it can be seen the argmax function selects the index of the main neural network after which this index is provided to the target neural network to select the right Q-value (corresponding to action a').

Another commonly used method to improve the learning of the DQN agent is to use Prioritized Experience Replay. PER is used to improve the learning of the agent by increasing the speed up the learning process [40]. In contrast to uniform sampling for the replay memory, PER lets the reinforcement learning agent choose samples for training the neural network based on the magnitude of their temporal difference error. Tuples stored in the replay memory with a higher temporal difference have a higher error, hence can lead to higher learning compared to other tuples.

Furthermore, there are different methods developed to deal with sparse rewards. In reinforcement learning, a reward is called sparse when it is only given to the agent in a small handful of events. For example, in PAC-MAN a sparse reward would be to only reward the agent once all the dots have been eaten in the game. The opposite of a sparse reward is a dense reward in which the agent receives a reward in the majority of the events. Due to the limited occurrence of sparse rewards, it can be very difficult for RL agents to learn anything since feedback on the agent's actions is very limited. However, different methods have been developed to improve the learning under sparse rewards. An example of such a method that is effective for sparse binary reward is Hindsight Experience Replay (HER), which works particularly well for binary rewards [41]. HER is a technique that also allows learning from failed experiences that do not lead to the final sparse reward by acting as if the failed performance was actually the goal. As a result, the agent receives feedback from the sparse reward more often improving the learning of the agent.

Finally, another approach for improving the performance of the DQN algorithm is related to the problem of unsuccessful exploration of the RL agent. Due to unsuccessful exploration, the agent can become stuck in a non-ideal local optimum. To prevent this, different methods have been developed that can help solve the exploration problem and ultimately increase the learning of the agents. One of these methods is a curiosity-driven approach to the reward function [42]. In this approach the agent is motivated to explore new state-action pares through a separate curiosity reward function which provides a higher reward for actions leading to the least frequently visited states. Moreover, there are different exploitation-exploration algorithms other than the epsilon-greedy policy that can be used to tweak the ratio between exploration and exploitation such as Thompson sampling, Upper Confidence Bound (UCB), and Boltzmann exploration [43].

3.4 Q-LEARNING AND SMART CHARGING

There are several papers applying Q-learning reinforcement learning to smart charging that have been published in the past few years [19, 44, 25, 45]. In order to have an idea of the current status of research on the use of Q-learning for smart charging, this section will cover some main approaches and corresponding results in literature.

Different objectives can be identified within the application of Q-learning for smart charging. These objectives can differ from building operation cost reduction and charging cost reduction to power consumption reduction, EV SOC increase, PV self-consumption improvement, peak load reduction, or a combination of these goals. To start with, an example of the use of Q-learning for smart charging is research focused on reducing the cost of a smart building with a V2G station attached to the building [44]. The authors developed a Q-learning model for a smart building that takes PV production, energy demand, energy storage, and a vehicle-to-grid station into account without the knowledge of future values. The state-space consists of EV state-of-charge, energy demand, and time in days. The action space consists of four actions; buy energy, sell energy, charge battery, or discharge the battery. As a result, the proposed Q-learning algorithm was able to significantly reduce the average daily cost of the building operation compared to a number of benchmarks.

While the previous example takes the possibility of an EV being connected to a smart building into account, it doesn't focus primarily on the minimization of the EVs charging cost. An example of an effective Q-learning smart charging model that focuses solely on charging EVs, is a model that takes

the time until the vehicle leaves, the customer's willingness to pay for a 100% charged vehicle, the energy price, and the renewable energy production as states [25]. The objective is to maximize the profitability of the charging station. The agent has two options for charging a vehicle, a slow charging low current option and a fast-charging high current option. As a result, when training 40 times on 90 different days, compared to random charging the model was able to increase the charging station's profit by 40-80% .

A final example of a research that takes charging cost reduction a step further is a research that focuses on the minimization of the EVs' energy cost by adding a V2G ability to the model and by taking market bidding into account [45]. In this work, an agent is developed that learns to determine the right energy selling bid price and energy buying bid price for the electricity market, while ensuring sufficient battery capacity. As a result, the overall average energy cost was near zero.

3.5 DQN AND SMART CHARGING

Since Deep Q learning is considered a better tool for more complex state-action spaces than Q-learning, the use of the DQN algorithm has been a topic of research within the field of smart charging [19, 46, 47, 48, 49]. As is the case for the literature on Q-learning smart charging, the literature on DQN smart charging can also be segmented based on the objective of the smart charging model.

To start with, recent research has focused on implementing a DDQN model with HER that maximizes PV self-consumption and EV state of charge at departure at the same time [46]. The state space included the PV production, building load demand, EV charging demand, time until EV departure, and SOC. As a result, with the use of a binary sparse reward for completing charging the model was able to charge the vehicle to a SOC of more than 80% in 33% of the time, with an execution time of 2.16 seconds.

Another research has expanded on this by adding the additional objective of reducing EV charging cost [48]. The authors have built a DDQN model which charges and discharges an EV in response to the hourly electricity prices. By modeling the electricity price, current energy level, remaining energy, and remaining charging time as states in the state space and multiple different charging and discharging powers as actions, they were able to create 30% cheaper charging schedules than uncontrolled charging with an execution time of 5.2 ms. The reward function consisted of a reward depending on the monetary benefit of charging or discharging and a delayed reward relative to the power left in the battery at the end.

Building on the previous work, another research takes it a step further by adding battery degradation cost to the reward function with the same goal of reducing charging cost [50]. To motivate the agent to charge the EV a sparse reward is provided at the final time step that is inversely related to the leftover SOC. As a result, their proposed approach was able to reduce the charging cost by 77.30% compared to uncontrolled charging.

A final example is a state-of-the-art DQN approach that makes use of Prioritized Experience Replay [47]. In contrast to the previously mentioned works, the authors discard monetary objectives and focus on decreasing peak loads and the load variance by charging during load valleys. In addition, this model takes multiple vehicles into account at the same time and doesn't rely on any predictions or user input data. The features of the state are both EV-specific and collective features and the action space is discrete and exists out of the three actions; don't charge, charge at half power, and charge at full power. Ultimately, the model's performance came near to the optimal optimization strategy with a decrease of 65% in the load variance compared to uncontrolled charging.

4 THESIS METHODOLOGY AND OBJECTIVES

This section covers the methodology used in this thesis to reach the research objectives stated in chapter 1. First, in section 4.1 the decision rationale for the chosen approach based on the literature review is discussed. Next in section 4.2 the different steps required to reach a solution to the main objective are stated. Finally, in section 4.3 the exact evaluation criteria of the developed solution is elaborated on in more detail.

4.1 FOCUS AND CONTRIBUTION OF THE THESIS

To start with, the approach that has been chosen for this thesis is approach End-to-End learning. There are multiple reasons why End-to-End learning is chosen over the approach of optimization augmented by ML and the approach over ML to improve optimization output. To start with, End-to-End learning allows fast and accurate output without the support of the original MIP optimizer. This allows for simplification since a standalone model is created instead of a model that exists of multiple algorithms. Furthermore, the alternative of optimization augmented by ML requires a deep understanding of branching, cutting, and configuration algorithms and while it has the potential to improve the speed of the model, it does not tackle fundamental bottlenecks of MIP models caused by the Branch-and-Bound Algorithm. Finally, the approach of using ML to improve optimization input is less sufficient to answer the research question, since while it has the potential to improve the accuracy, it is not expected to significantly improve the speed of the existing MIP smart charging algorithm.

Within the field of smart charging, end-to-end reinforcement learning is widely used for smart charging applications. Therefore, the decision has been made to use reinforcement learning algorithms in this thesis. Within smart charging, reinforcement learning has been proven to be capable of creating smart charging plans. While there are many different reinforcement learning algorithms, this thesis will focus on two of the most applied reinforcement learning algorithms for smart charging; Q-learning and Double Deep Q-network reinforcement learning [19]. Alternatively, reinforcement learning algorithms such as the Deep Deterministic Policy Gradient, Asynchronous Advantage Actor-Critic Algorithm, and Soft Actor Critic have the advantage of allowing a continuous action space in contrast to the Q-learning and the Deep Q-network algorithm. However, a continuous action space is not required in this work making the Q-learning and Double Deep Q-network sufficient.

Although there are quite some recent studies on the application of Q-learning and DDQN for smart charging, the research on the feasibility of reinforcement learning as a replacement for traditional optimization methods remains limited. The majority of existing research focuses on the performance evaluation of the RL agents around uncontrolled charging which makes it hard to evaluate the charging performance in comparison to alternative state-of-the-art optimization methods such as LP/MIP. Furthermore, in contrast to previous studies, this work aims for an extensive evaluation of the RL agents' performance through the assessment of non-trivial metrics such as the RL execution speed and constraint violation. These metrics have been previously assessed only to a very limited extent, but are essential to grasp the potential of RL for smart charging applications. In addition, only a few works in literature research the ability of smart charging reinforcement learning to generalize and perform well on arbitrary cases, which is essential for future real-life applications.

4.2 SOLUTION DEVELOPMENT

In order to complete the research objectives discussed in chapter 1, several sub-objectives can be formulated, which are described in the section below:

1. **Development of a Q-learning and Deep Q-network model for smart charging:**

a) Case study development
b) Algorithm set-up
c) Reward engineering
d) Hyperparameter tuning

2. **Evaluation of the charging performance, cost reduction and speed of the proposed reinforcement learning smart charging algorithms:**
   a) Evaluate the charging performance, cost reduction, and execution speed for three specific cases both prior to and after training the agent on multiple different cases in comparison to the average rate charging (AVR) benchmark and the MIP smart charging benchmark
   b) Evaluate the generalization performance of the algorithms by computing average charging performance, cost reduction, and execution speed for 10,000 different cases and compare to the average rate charging benchmark
   c) In-depth analysis and discussion of the reinforcement learning algorithms performance in comparison to LP/MIP smart charging algorithms

The exact evaluation criteria that will be used to evaluate the performance, cost reduction, and speed of the reinforcement learning algorithms are discussed in the next section. A visual overview of the solution development is described in figure 4.2.1, of which the first two components have been completed through the literature review in chapter 2. Furthermore, the simulation and modeling of the reinforcement learning algorithms are done in Python and the Keras Tensorflow library is used for Deep Q-learning in the Deep Q-network approach.

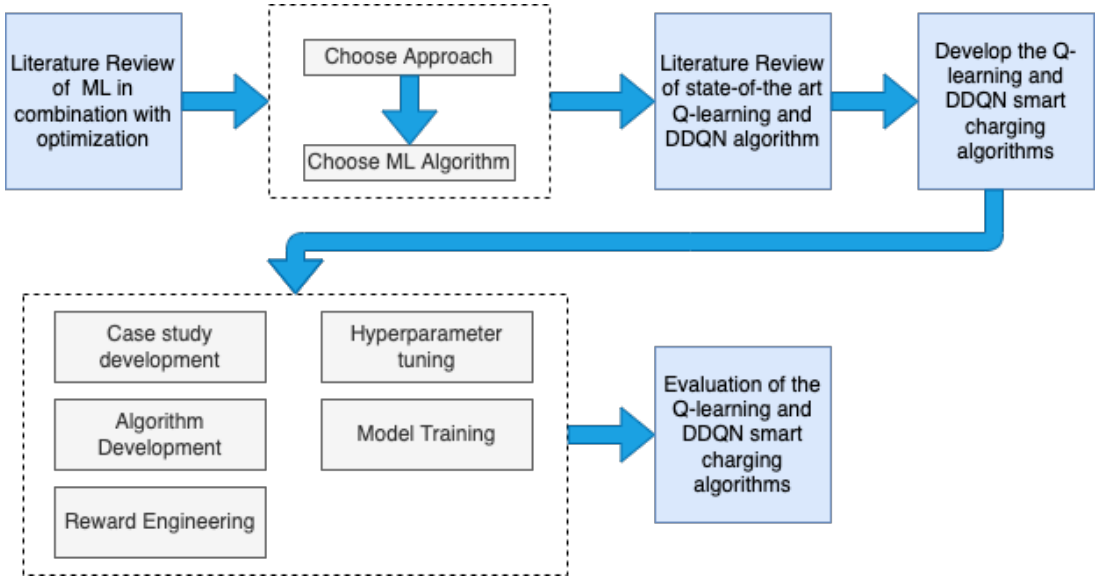

Figure 4.2.1: A visual overview of the solution development

4.3 EVALUATION CRITERIA

While the section above has covered the main overarching themes of the evaluation criteria, in this section the exact framework for evaluation of the RL agents and the corresponding criteria are covered in more detail:

1. **Evaluation of a single case study:**
   - Occurrence of a constraint violation
   - Final SOC reached by the RL agent

- Total EV charging cost in Euro per kWh for the RL agent
- Relative percentage EV charging cost difference in comparison to the AVR charger
- Execution speed in seconds per episode

2. **Evaluation of the average performance on multiple different episodes:**
   - Percentage of episodes in which constraints are not violated by the Rl agent
   - Average final SOC reached by the RL agent
   - Average execution speed in seconds per episode
   - Percentage of episodes that reach the maximum possible SOC
   - Percentage of charging sessions that reach the maximum possible SOC
   - Average charging cost difference relative to AVR for all charging sessions that reach max possible SOC
   - Percentage of charging sessions that are cheaper than AVR and reach max possible SOC
   - Average charging cost difference relative to AVR for all charging sessions that are cheaper and reach max possible SOC
   - Average EV charging cost per episode in Euro/kWh for the RL agent
   - Average EV charging cost per episode in Euro/kWh for the AVR agent
3. **Analysis of the output data for different inputs:**
   - Relative percentage charging cost difference for all sessions that reach full battery capacity for different vehicle leaving times in multiple episodes
   - Final SOC for the different vehicle leaving times in multiple episodes
   - Relative percentage charging cost difference for all sessions that reach full battery capacity for different battery arrival capacities in multiple episodes
   - Final SOC for the different battery arrival capacities in multiple episodes

Points 1, 2, and 3 will be discussed in the result section in chapter 7, while the further analysis of the output and the significance of the research is discussed in chapter 8.

5 SMART CHARGING REINFORCEMENT LEARNING

This chapter discusses the specifications of the Q-learning and DDQN reinforcement learning models developed for the application of smart charging. This chapter is organized as follows. First, in section 5.1 the overarching system structure of the reinforcement learning model for both Q-learning and DDQN is explained, after which in section 5.2 the Q-learning model is discussed in detail including the action space, state space, and reward engineering. Subsequently, in section 5.3 the action space, state space, and reward engineering for the DDQN model are elaborated on. Finally, in section 5.4 an overview of the hyperparameters for both reinforcement learning models is provided.

5.1 SYSTEM STRUCTURE

This section covers the fundamentals behind the system structure used for the development of the Q-learning and DDQN smart charging algorithms. The system structure consists of a few main components including an Energy Management System, an AC grid, an EV charger, a building load, and a solar PV installation with an inverter. In this system, the energy management system can be considered to be the agent that governs the bidirectional power exchange with the AC grid, and the unidirectional power exchange with the charger, building load, and Solar PV installation. The power generated by the solar PV installation is first used for the building load, after which any excess energy is used for the charging of the EV, before being supplied to the AC grid. An overview of the system structure is provided in figure 5.1.1 below.

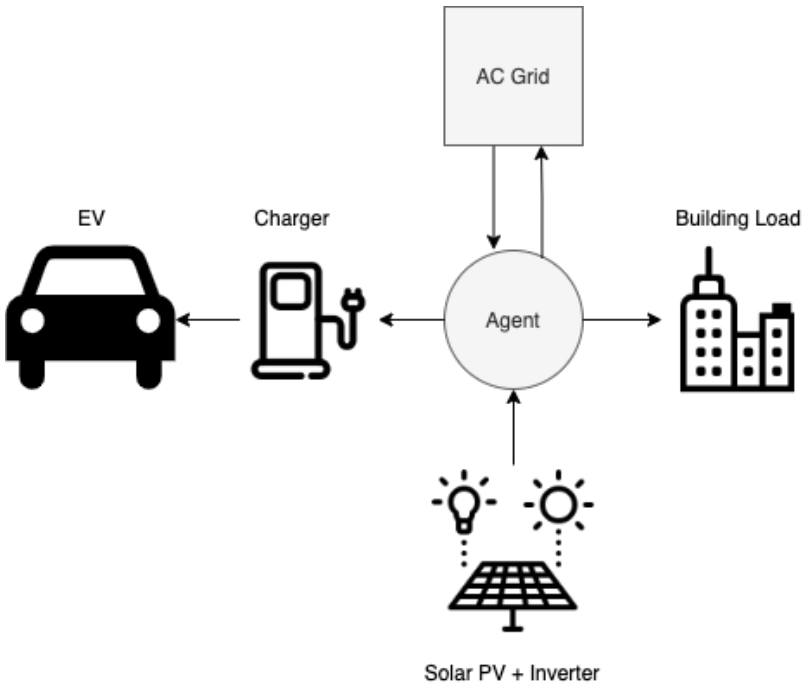

Figure 5.1.1: The system structure used for the reinforcement learning algorithms

5.1.1 Data collection and model parameters

The RL models use electricity prices that are taken from the 2018 Day-Ahead Market (DAM) in the Netherlands. The selling price of electricity is set to 95% of the buying price. Furthermore, both the

PV power and load are based on typical diurnal fluctuating patterns and are simulated for the length of a year including seasonal variations for the generated PV power. The original PV power data was converted to 1kWp and then multiplied with the PV installation size [51]. The rated PV power is set to 20 kWp representing 80 solar panels of 0.25 kWp and the yearly energy consumption of the building is set to 70000 kWh. The load data is based on a standardized load profile from the Netherlands in 2018 [52]. Furthermore, the time horizon for the optimization is 12 hours, with a step size of 5 minutes. In addition, the EV owner has a variable integer value leaving time between 3 and 12 hours, which is known beforehand by the agent. Therefore, a single episode consists of 12 hours and a total of 144 time steps of 5 minutes.

Despite the algorithm optimizing window being fixed for a period of 12 hours, the agent is trained to output non-zero current values after the vehicle has left. Therefore, this optimization window allows the agent to optimize for different vehicle leaving times as the vehicle can leave at any given time between 3 and 12 hours in whole hours. An overview of the different time-related model parameters is given in table 5.1.1.

Symbol	Parameter	Value	Fixed/Variable
ΔT	Time step	5 Minutes	Fixed
T	Time Horizon	12 hours	Fixed
ΔT_{Total}	Total time steps	144 hours	Fixed

Table 5.1.1: The main time-related model parameters for the smart charging algorithm

Next to the time-related model parameters, there are multiple charging, grid, and EV parameters defined for the smart charging model. The decision has been made to keep some parameters fixed to minimize the complexity of the model. An overview of these parameters is provided in table 5.1.2.

Symbol	Parameter	Value	Fixed/Variable
p_{max}^{PVr}	Rated PV Power	20 kWp	Fixed
p^{load}	Yearly Energy Consumption	70000 kWh	Fixed
TTL	Time-to-leave	3-12 hours	Variable
V_{nom}	Nominal Voltage	230 V	Fixed
$\eta_{charging}$	Charging Efficiency	0.97	Fixed
$p_{max}^{charger}$	Max charging power charger	11.04 kW	Fixed
p_{max}^{EV}	Max charging power EV	22.08 kW	Fixed
ϕ_{Phase}	Phase	Three-phase	Fixed
B^{max}	Total Battery Capacity	50 kWh	Fixed
B^{min}	Minimum Battery Capacity	0 kWh	Fixed
$B^{Arrival}$	Arrival Battery Capacity	10-30 kWh	Variable

Table 5.1.2: The hyperparameters for the smart charging algorithm

As can be seen in the table above, all parameters are fixed except for the time-to-leave and arrival battery capacity. These parameters can be set to be variable during training to improve the generalization ability of the agent.

5.1.2 Main assumptions

There are some main assumptions on which both the Q-learning and Double Deep Q-network models are built upon. A list of these main assumptions is provided below:

1. **Perfect knowledge**: In this study, the assumption is made that there is 100% accuracy with respect to the knowledge of the future PV power, electricity buying and selling price, and load.
2. **Electricity selling price**: The electricity selling price is assumed to be equal to 95% of the buying price.

3. **Charging window**: It is assumed that the EV owner provides the agent with a charging window between 3-12 hours, by indicating within how many hours the EV should be charged. This information is communicated to the EV agent at the start of the charging window. Furthermore, it is assumed that the EV owner unplugs the EV at the end of the charging window.

4. **No CC/CV Charging**: It is assumed that the EV battery does not follow a CC/CV charging plan.

5. **No Overcharging**: It is assumed that overcharging is not possible, even if the algorithm outputs a charging action after being fully charged.

5.1.3 Objectives

In contrast to mathematical optimization, an objective function in reinforcement learning is not defined as a loss function. However, instead, the reward function is used to indirectly achieve the objective of the agent. The main objective of the smart charging algorithm in this work is twofold, making it a multi-objective model:

1. To charge the electric vehicle to the highest possible SOC before the EV-owner intends to leave

2. To minimize the money spent in EUR/kWh on electricity for the charging of the EV

Here the highest possible SOC is defined as the max possible SOC possible if the agent were to charge at full power in every step for the duration of the charging window, which differs depending on the case under consideration.

5.1.4 Environment

The environment consists of the equations that describe the system dynamics and provide the information required for updating the state space and calculating the reward. The environment is consulted after every time step and depends on the action taken in the step before. When considering the state space, all states are updated in every time step. The power delivered to the vehicle, the battery capacity, and import and export power values are also computed in every time step and are used to update the environment. The corresponding equations in the environment are described below:

$$P_t = I_t^{e+} \times V_{nom} \times \phi_{phase}, \qquad t \in T \tag{5.1}$$

$$B_t = P_t \times \eta_{charging} \times \frac{\Delta T}{60}, \qquad t \in T \tag{5.2}$$

$$S_t = \frac{B_t - B^{min}}{(B^{max} - B^{min})}, \qquad t \in T \tag{5.3}$$

$$P_t^{diff} = (P_t / \eta^{Charger}) + P_t^{load} - P_t^{PV} = P_t^{G(imp)} - P_t^{G(exp)}, \qquad t \in T \tag{5.4}$$

$$P_t^{G(imp)} = \{P_t^{diff} | P_t^{diff} > 0\}, \qquad t \in T \tag{5.5}$$

$$P_t^{G(exp)} = -1 * \{P_t^{diff} | P_t^{diff} < 0\}, \qquad t \in T \tag{5.6}$$

5.1.5 Constraints

The reinforcement learning agent is subjected to three technical constraints which can not be violated during the charging process:

1. Alternating current (AC) charging standards (International Electrotechnical Commission 61851) come with a lowest possible set point of current supplied to the vehicle by the charger of 6 A.

$$(I_t^{e+} = 0) \text{ or } (6 \leq I_t^{e+} \leq 16), \qquad t \in T \tag{5.7}$$

2. The power provided to the EV can not exceed the max charging power capacity of the EV charger of 11.04 kW

$$0 \leq P_t \leq 11.04 \text{ kW}, \qquad t \in T \tag{5.8}$$

3. The agent is not allowed to supply current to the vehicle when the vehicle has left or when the vehicle has been charged to 100% SOC to prevent overcharging.

The first constraint is automatically satisfied through the limitation of the action space. The second constraint is ensured by the limit on the first constraint since it has a maximum value of 16 A. Finally, the third constraint is implemented in the reward function through the constraint violation penalty described below.

5.2 Q-LEARNING MODEL

This section covers the specifications of the Q-learning model. First, subsection 5.2.1 and 5.2.2 cover the action and state space, after which in subsection 5.2.3 the Q-learning reward function is discussed.

5.2.1 Action space

The actions that the agent can take for the charging currents to the EV are discretized and exists of a current of 0 A or a current between 6-16 A. The reason for discretization is to limit the size of the Q-table, as a too-large size can not be stored in the memory of the computer. An overview of the actions possible is provided in the equation below.

$$a_t = \{0, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}, \qquad t \in T \tag{5.9}$$

5.2.2 State space

Furthermore, in order to capture the situation that the EV is in, different state values are collected from the environment every time step to describe the state of the system after an action has been executed. The state space of the Q-learning model includes the state of charge of the EV SOC_t, the normalized (until the vehicle leaves) market clearing price for buying electricity in the time step under consideration $C^{buy}_{norm,t}$, the number of hours left until the vehicle leaves TTL^{Now}_t, the normalized number of steps left until the until vehicle leaves that have a lower normalized market clearing price than the current one N^{lower}_t, and the original number of hours left until the vehicle leaves $TTL^{original}_t$.

$$S_t = (SOC_t, C^{buy}_{norm,t}, TTL^{Now}_t, N^{lower}_t, TTL^{original}_t), \qquad t \in T \tag{5.10}$$

The state values are discretized to restrict the size of the Q-table since bigger more complex state spaces were not possible due to application memory limitations. Given this action and state space, the q-learning table size amounts to roughly 6 million different states multiplied by 11 different options for the charging actions. A more detailed overview of every state is provided in table 5.2.1 below.

Symbol	State	Unit	Discretization
SOC_t	State of charge of EV	-	21 steps between 0 and 1
$C^{buy}_{norm,t}$	Normalized (until vehicle leaves) market clearing price for buying electricity in the time step under consideration	-	21 steps between 0 and 1
TTL^{Now}_t	Number of hours left until the vehicle leaves	Hours	10 steps between 3 and 12
N^{lower}_t	Normalized number of steps until the vehicle leaves that have a $C^{buy}_{norm,t}$ lower than the current $C^{buy}_{norm,t}$	-	21 steps between 0 and 1
$TTL^{original}_t$	Original number of hours left until the vehicle leaves	Hours	10 steps between 3 and 12

Table 5.2.1: The state space symbols, description, unit, and discretization level for the Q-learning model

5.2.3 Q-learning reward function

As mentioned before, a reward function is used in every time step to measure how well the action has led to the desired performance. The reward function consists of the following three components that collectively contribute to reaching the objectives of the smart charging algorithm:

$$\text{Reward}_t = \text{R}^{cost}_{t,\text{normalized}} + \text{R}^{SOC}_t - \text{R}^{constraint}_{t,\text{normalized}}, \qquad t \in T \tag{5.11}$$

1. Cost reward: To start with, the cost reward $\text{R}^{cost}_{t,\text{normalized}}$ depends in the money spent on electricity to charge the vehicle in equation 5.12. In this equation R^{cost}_t is the current electricity buying

price, which is multiplied by the energy provided to the vehicle in kWh in that specific time step. Subsequently, R_t^{cost} is normalized to a value between 0 and 10 for non-zero current actions. For non-zero current actions, the cost spent on electricity increases for higher current and as a result, the cost reward decreases. However, if the zero current action is chosen, the reward is set to 12 in order to discourage charging if not necessary. The equations that describe the cost reward and the normalization are provided below:

$$R_t^{cost} = C_t^{buy} * P_t * (\Delta T/60), \qquad t \in T \tag{5.12}$$

$$R_{t,\text{normalized}}^{cost} = \begin{cases} \overline{R}^{cost} - (\overline{R}^{cost} - \underline{R}^{cost}) \frac{|R_t^{cost} - R_{t,max}^{cost}|}{|R_{t,max}^{cost} - R_{t,min}^{cost}|}^{0.6} & \text{if } I_t^{e+} \neq 0 \\ 12 & \text{otherwise} \end{cases}, \qquad t \in T \tag{5.13}$$

The normalization of R_t^{cost} in equation 5.13 is based on the the upper normalization parameter $\overline{R}^{cost}$ and the lower normalization parameter $\underline{R}^{cost}$. This charging costs are normalized since every episode can have different electricity prices. Furthermore, in equation 5.13 $R_{t,min}^{cost}$ is defined as the lowest possible cost reward when the electricity price C_t^{buy} is the highest value for the episode and P_t is equal to the maximum possible power value when the current provided is 16A. Furthermore, $R_{t,max}^{cost}$ is defined as the highest cost reward when the electricity price C_t^{buy} is the lowest value for the episode and P_t is equal to the lowest non-zero power output when the current is 6 A . Both $R_{t,min}^{cost}$ and $R_{t,max}^{cost}$ are computed again for every episode. After R_t^{cost} is normalized relative to $R_{t,max}^{cost}$ and $R_{t,max}^{cost}$ it is taken to the power 0.6 to shape the reward function and normalized relative to $\overline{R}^{cost}$ and $\underline{R}^{cost}$.

2. Charging reward: To motivate the agent to charge the vehicle to a state of charge of 100%, the agent is provided with a single high reward of 1500 if the vehicle has receives a state of charge of 100% before the EV owner intends to leave:

$$R_t^{SOC} = \begin{cases} 1500 & \text{if SOC = 1 when } TTL_t^{Now} = 0 \\ 0 & \text{otherwise} \end{cases}, \qquad t \in T \tag{5.14}$$

Since R_t^{SOC} is a sparse reward and can only be provided once, it is not normalized.

3. Overcharge penalty: Whenever the vehicle is full and power is provided to the vehicle, a negative penalty is added to the cumulative reward. The equations that describe the constraint violation penalty reward and the normalization are provided below:

$$R_t^{constraint} = I_t^{e+}, \qquad t \in T \tag{5.15}$$

$$R_{t,\text{normalized}}^{constraint} = \overline{R}^{constraint} - (\overline{R}^{constraint} - \underline{R}^{constraint}) \frac{|R_t^{constraint} - R_{t,max}^{constraint}|}{|R_{t,max}^{constraint} - R_{t,min}^{constraint}|}^{0.6}, \qquad t \in T \tag{5.16}$$

Just like for the cost reward the magnitude of the current value equal to $R_t^{constraint}$ determines the final normalized reward. Therefore, in equation 5.16, $R_t^{constraint}$ is normalized between the upper normalization parameter $\overline{R}^{constraint}$ and the lower normalization parameter $\underline{R}^{constraint}$. In this equation $R_{t,max}^{constraint}$ is the maximum possible current of 16A and $R_{t,min}^{constraint}$ is the minimum possible non-zero current of 6A. Therefore, the higher the current the higher the penalty for overcharging.

The normalization of the rewards used in Q-learning requires careful tuning. Normalization is done in this way to improve the learning of the agent. The normalization allows re-scaling of the reward between an upper and lower bound value facilitating control over the value to achieve a desired trade-off with the other rewards. In addition, the power of 0.6, reshapes the rewards into a curve with a gradient to improve the feedback to the agent. The constraint violation penalty has been set to a value

much lower than the worst possible reward to ensure that the agent does not find a local optimum while still violating the overcharging constraint. Furthermore, $\overline{R}^{cost}$ was not set too high to encourage the charging of the vehicle and receive the SOC reward $\overline{R}^{cost}$ of 1500 for charging the vehicle to 100%. Table 5.2.2 shows the final upper and lower normalization parameters that lead to optimal results.

Symbol	$\overline{R}^{cost}$	$\underline{R}^{cost}$	$\overline{R}^{constraint}$	$\underline{R}^{constraint}$
Value	10	0	500	550

Table 5.2.2: Model parameters for the reward engineering of the Q-learning model

5.3 DDQN MODEL

This section covers the specifications of the DDQN model. First, subsection 5.3.1 and 5.3.2 cover the action and state space, after which in subsection 5.3.3 the Q-learning reward engineering is elaborated on.

5.3.1 Action Space

Just like with Q-learning, the actions that the agent takes are discretized and the charging currents that the agent can provide are 0 A or a value between 6-16 A. In theory, the action space could be made less discrete for DDQN reinforcement learning, since the output of the neural network is simply a Q-value for each output. However, this would require a neural network with an extremely large amount of outputs, which significantly increases the required training time for a decent prediction. Therefore, for simplicity's sake, the action space is kept discrete as can be seen in equation 5.17 below:

$$a_t = \{0, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, , 16\}, \qquad t \in T \tag{5.17}$$

5.3.2 State space

Furthermore, in order to describe the current situation that the EV is in in every step, eight different state values are collected from the environment. While the state space for Q-learning was kept to a limited amount to reduce the size of the Q-table, DDQN allows a more extensive state space. In order to enhance the input to the neural network, the input values are normalized between 0 and 1, making the state-values unitless. The total state space used for the DDQN model S_t is described below:

$$S_t = (SOC_t, TTL_t^{Now}, C_{norm,t}^{buy}, N_t^{lower}, C_{t,average}^{buy}, C_{t,five}^{buy}, TTL_t^{Binary}, SOC_t^{Binary}), \qquad t \in T \tag{5.18}$$

Similar to Q-learning the environment updates the state space after an action occurs. The definition and corresponding unit of every state are provided in table 5.3.1 below. When comparing the state space of DDQN to the state space used for Q-learning we find that the states $C_{t,average}^{buy}$, $C_{t,five}^{buy}$, TTL_t^{Binary} and SOC_t^{Binary} have been added to the state space. The reason for this is that these statesvvimprove the amount of information regarding the price trajectory and the moment the vehicle leaves. $C_{t,average}^{buy}$ is defined as the average normalized electricity buying price until the vehicle leaves and remains constant throughout the episode, while $C_{t,five}^{buy}$ is equal to the average normalized electricity buying price of the past five steps. The decision has been made to use the electricity buying price of the past five steps since five steps is sufficient to indicate a recent transition in the buying price. Furthermore, TTL_t^{Binary} and SOC_t^{Binary} are binary values indicating a value of one when the vehicle has left and when the SOC is 1 and zero otherwise.

Symbol	State
SOC_t	State of charge of EV
TTL_t^{Now}	Normalized number of hours left until the vehicle leaves
$C_{norm,t}^{buy}$	Normalized (until vehicle leaves) market clearing price for buying electricity based on prediction
N_t^{lower}	Normalized number of steps (until vehicle leaves) that have a C_t^{buy} lower than the current C_t^{buy}
$C_{t,average}^{buy}$	The normalized average relative electricity buying price until the vehicle leaves
$C_{t,five}^{buy}$	The average normalized electricity buying price of the past five steps
TTL_t^{Binary}	Binary value indicating a value of one, once the EV has left and zero otherwise
SOC_t^{Binary}	Binary value indicating a value of one, once the vehicle has reached an SOC equal to 1 and zero otherwise

Table 5.3.1: The state space symbols, description, unit, and discretization level for the DDQN model

5.3.3 DDQN reward function

For DDQN, the reward function also exist out of three components as can be seen in equation 5.19 below:

$$\text{Reward}_t = \text{R}_{t,\text{normalized}}^{cost} + \text{R}_{t,\text{Mdense-norm}}^{SOC} - \text{R}_{t,\text{normalized}}^{constraint}, \qquad t \in T \tag{5.19}$$

1. Cost reward: The cost reward $\text{R}_{t,\text{normalized}}^{cost}$ is defined the same as for Q-learning. The cost reward R_t^{cost} depends on the current electricity buying price multiplied by the energy provided to the vehicle in kWh and subsequently R_t^{cost} is normalized to a value between 0 and 10 if the current action is non-zero. The reward is set to 12 if the current action is zero. The equations that describe the cost reward and the normalization are provided below:

$$\text{R}_t^{cost} = C_t^{buy} * P_t * (\Delta T/60), \qquad t \in T \tag{5.20}$$

$$\text{R}_{t,\text{normalized}}^{cost} = \begin{cases} \overline{R}^{cost} - (\overline{R}^{cost} - \underline{R}^{cost}) \frac{|\text{R}_t^{cost} - \text{R}_{t,max}^{cost}|}{|\text{R}_{t,max}^{cost} - \text{R}_{t,min}^{cost}|}^{0.6} & \text{if } I_t^{e+} \neq 0 \\ 12 & \text{otherwise} \end{cases}, \qquad t \in T \tag{5.21}$$

   In the above equation also $\text{R}_{t,min}^{cost}$ and $\text{R}_{t,max}^{cost}$ are defined in the same way as for the Q-learning model. $\text{R}_{t,min}^{cost}$ represents the worst possible cost outcome in every episode and $\text{R}_{t,max}^{cost}$ the best possible cost outcome. The values of $\overline{R}^{cost}$ and $\underline{R}^{cost}$ are given in table 5.3.3.

2. Medium-Dense charging reward: In contrast to the reward for Q-learning, a more dense charging reward $\text{R}_{t,\text{Mdense-norm}}^{SOC}$ is proposed. To motivate the agent to charge the vehicle to a state of charge of 100%, 20 SOC targets $N_{SOC}^{targets}$ have been set up in steps of 0.05 between 0 and 1. This reward is considered medium-dense since the agent does not receive a reward at every time step. Instead, the agent receives a reward every few time steps, for which the charging power determines the number of time steps it takes to reach a target. As can be seen in equation 5.22, every time the vehicle reaches a SOC target the average cost in Euro per kWh is calculated to reach that SOC target. The average cost in Euro per kWh is calculated by dividing the sum of the cost in all steps N leading to a charging target by the total gain in battery capacity ΔB. After the agent reaches a charging target, $\text{R}_{t,\text{Mdense}}^{SOC}$ is reset to zero again:

$$\text{R}_{t,\text{Mdense}}^{SOC} = \frac{\sum_{n=1}^{N} \text{R}_{t,n}^{cost}}{\Delta B}, \qquad t \in T \tag{5.22}$$

$$\text{Weight} = \begin{cases} \overline{R}^{SOC}_{Mdense} - (\overline{R}^{SOC}_{Mdense} - \underline{R}^{SOC}_{Mdense}) \frac{|\mathrm{R}^{SOC}_{t,\text{Mdense}} - \mathrm{R}^{SOC}_{t,max}|}{|\mathrm{R}^{SOC}_{t,max} - \mathrm{R}^{SOC}_{t,min}|}^{0.6} & \text{if SOC = a target} \\ 0 & \text{otherwise} \end{cases}, \quad t \in T \tag{5.23}$$

$$\mathrm{R}^{SOC}_{t,\text{Mdense-norm}} = 100 \times \text{Weight}, \quad t \in T \tag{5.24}$$

As can be seen in equation 5.23, the average cost in Euro per kWh spent on electricity to reach a charging target is normalized relative to $\mathrm{R}^{SOC}_{t,max}$ and $\mathrm{R}^{SOC}_{t,min}$ once a SOC target has been reached. Here $\mathrm{R}^{SOC}_{t,max}$ is equal to the maximum possible electricity price in Euro/kWh for every specific episode and $\mathrm{R}^{SOC}_{t,min}$ is equal to the minimum possible electricity price. Subsequently, the value is normalized between the upper normalization parameter $\overline{R}^{SOC}_{Mdense}$ and the lower normalization parameter $\underline{R}^{SOC}_{Mdense}$ to compute the weight. Finally, the weight is multiplied with a reward value of 100 to motivate the agent to charge the vehicle to 100% as cheap as possible. Therefore, the lower the average cost in Euro per kWh to reach the charging target the higher the weight and the higher the reward.

3. Overcharge penalty: Similar to the Q-learning model, the agent receives a negative penalty when the vehicle or has left and power is provided to the vehicle. The equation that describes the constraint violation penalty reward and its corresponding normalization is provided below:

$$\mathrm{R}^{constraint}_{t} = I^{e+}_{t}, \quad t \in T \tag{5.25}$$

$$\mathrm{R}^{constraint}_{t,\text{normalized}} = \overline{R}^{constraint} - (\overline{R}^{constraint} - \underline{R}^{constraint}) \frac{|\mathrm{R}^{constraint}_{t} - \mathrm{R}^{constraint}_{t,max}|}{|\mathrm{R}^{constraint}_{t,max} - \mathrm{R}^{constraint}_{t,min}|}^{0.6}, \quad t \in T \tag{5.26}$$

Also for DDQN, $\mathrm{R}^{constraint}_{t,max}$ is the maximum possible current of 16A and $\mathrm{R}^{constraint}_{t,min}$ the minimum possible non-zero current of 6A. Therefore, the higher the current the higher the penalty for overcharging. The values of $\overline{R}^{constraint}$ and $\underline{R}^{constraint}$ are provided in table 5.3.3.

4. Alternative rewards: To motivate the agent to charge the vehicle to a state of charge of 100%, also a sparse charging reward and a dense charging reward could be effective. For the sparse charging reward, a delayed final reward is provided to the agent once the vehicle has left. A weight factor of 1 is multiplied with the difference between the full capacity of 50 kWh and the final capacity squared. This encourages the agent to charge the vehicle to maximize the total discounted reward as can be seen below.

$$\mathrm{R}^{SOC}_{t,sparse} = \text{Weight} \times (B^{max} - B^{final})^2, \quad t \in T \tag{5.27}$$

Furthermore, a dense charging reward was considered that has 1000 SOC targets $N^{targets}_{SOC}$ in small steps of 0.001 between 0 and 1. It can be considered dense since it is able to provide a reward in every step. Every time the vehicle reaches a SOC target the agent is rewarded with a charging reward equal to the inverse of the electricity buying price at that moment times the number of SOC targets covered by delivering a certain amount of current in that time step. This encourages the agent to charge the vehicle and reach SOC targets when prices are low. Since every charging action leads to the agent reaching a number of SOC targets, the agent receives feedback in every step making it a dense reward. The reward function is shown below:

$$\mathrm{R}^{SOC}_{t,\text{Dense}} = \frac{1}{C^{buy}_{t}} \times N^{targets}_{SOC}, \quad t \in T \tag{5.28}$$

$$\mathrm{R}^{SOC}_{t,\text{Dense-norm}} = \overline{R}^{SOC}_{Dense} - (\overline{R}^{SOC}_{Dense} - \underline{R}^{SOC}_{Dense}) \frac{|\mathrm{R}^{SOC}_{t,\text{Dense}} - \mathrm{R}^{SOC}_{t,max}|}{|\mathrm{R}^{SOC}_{t,max} - \mathrm{R}^{SOC}_{t,min}|}^{0.6}, \quad t \in T \tag{5.29}$$

In the equation above $\overline{R}^{SOC}_{Dense}$ and $\underline{R}^{SOC}_{Dense}$ can be set to a value depending on the desired behaviour in relation to the cost reward. Furthermore, $\mathrm{R}^{SOC}_{t,max}$ and $\mathrm{R}^{SOC}_{t,min}$ are equal to the the best and worst possible $\mathrm{R}^{SOC}_{t,\text{Dense}}$ values in every episode.

However, with regard to the alternative rewards mentioned above, the decision was made to go forward with the medium-dense charging rewards as the sparse charging reward did not motivate the agent to charge the vehicle sufficiently and the dense charging reward severely alters the model-free nature of the DDQN model, since in every step the charging reward stands in direct relationship to the contradicting SOC reward. Table 5.3.3 shows the final normalization hyperparameters for the reward function of the DDQN model model that lead to optimal results.

Symbol	$\overline{R}^{cost}$	$\underline{R}^{cost}$	$\overline{R}^{SOC}_{Mdense}$	$\underline{R}^{SOC}_{Mdense}$	$\overline{R}^{constraint}$	$\underline{R}^{constraint}$
Value	10	0	0	1	15	25

Table 5.3.2: Model parameters for the reward engineering of the DDQN model

5.4 HYPERPARAMETERS

Furthermore, there are several hyperparameters that need to be tuned for both algorithms, which can be found in table 5.4.1. Some hyperparamaters have been determined through close study of different hyperparameter versions while others have been determined through trial and error. In this section the rationale behind a number of hyperparameters will be discussed.

Since the SOC reward can be seen as a delayed reward because it is not provided in every time step, the value of the learning rate α and discount factor γ are set to be relatively high. A high value of α and γ makes sure that the Q-value corresponding to the next state $Q(S', A)$ has more of a contribution since its weight is increased in the Bellman Equation. Ultimately, because of this, a delayed reward is translated through to all Q-values that are part of the same chain of state and actions values.

Symbol	**Hyperparamater**	**Q-learning**	**DDQN**
$N_{episodes}$	Total number of episodes prior to generalization	450K	22.5K
$N_{episodes}$	Total number of episodes after generalization	45M	90K
ϵ	Epsilon value	0.9	0.9
α	Learning rate	0.3	0.001
γ	Discount factor	0.999	0.999
ϵ_{min}	Minimum epsilon value	0.05	0.05
α_{min}	Minimum alpha value	0.25	0.25
$A_{epsilon}$	Initial plateau length	0.5	0.5
$B_{epsilon}$	Steepness of decreasing gradient	0.1	0.1
$C_{epsilon}$	Bottom plateau height	0.2	0.2
M_{size}	Replay Memory Size	-	50000
R_{size}	Minimum replay Memory Size	-	1000
B_{size}	Mini batch size	-	32
$T_{frequency}$	Target network update frequency	-	10
$N_{Neurons}$	Number of neurons in each layer	-	128
N_{Layers}	Number of layers in neural network	-	3
$Tr_{frequency}$	Training frequency	-	24 time steps

Table 5.4.1: Q-learning and DDQN reinforcement learning hyperparameters

Furthermore, both the values of α and ϵ were set to decay in Q-learning as the number of episodes gets closer to the final number of episodes. In DDQN, only ϵ was set to decay. The DDQN algorithm does not make use of an explicit α factor, since the future Q-value is predicted as a stand-alone value instead of combined with the old Q-value to be stored in the Q-table, which is the case for Q-learning. The value α for DDQN is represented in the neural network instead. For Q-learning a lower value of α decreases the degree to which a future Q-value is weighed into the current Q-value. This way as the number of episodes increases the Q-values become more robust and depend less on the future. A lower value of ϵ decreases the degree to which random actions are taken, allowing more exploitation of the

Q-table in Q-learning and the neural network in the DDQN model. The equation used to determine the ϵ decay rate is described below in equation 5.30 until 5.33 and exist of three components governed by the hyperparameters $A_{epsilon}$, $B_{epsilon}$, $C_{epsilon}$ that determine the shape of the ϵ decay:

$$\epsilon_{standardized} = \frac{(episode - A_{epsilon} \times N_{episodes})}{B_{epsilon} \times N_{episodes}} \tag{5.30}$$

$$\epsilon_{cosh} = cosh(e^{-1 \times \epsilon_{standardized}}) \tag{5.31}$$

$$\epsilon_{intermed} = 1.1 - (\frac{1}{\epsilon_{cosh}} + \frac{episode \times C_{epsilon}}{N_{episodes}}) \tag{5.32}$$

$$\text{Epsilon} = 1 + \frac{\epsilon_{intermed} - 1.1}{1.1 + 0.2} \tag{5.33}$$

A plot of the ϵ decay for different hyperparameters is provided below in figure 5.4.1 and shows how the shape of the curve differs for the different hyperparameters:

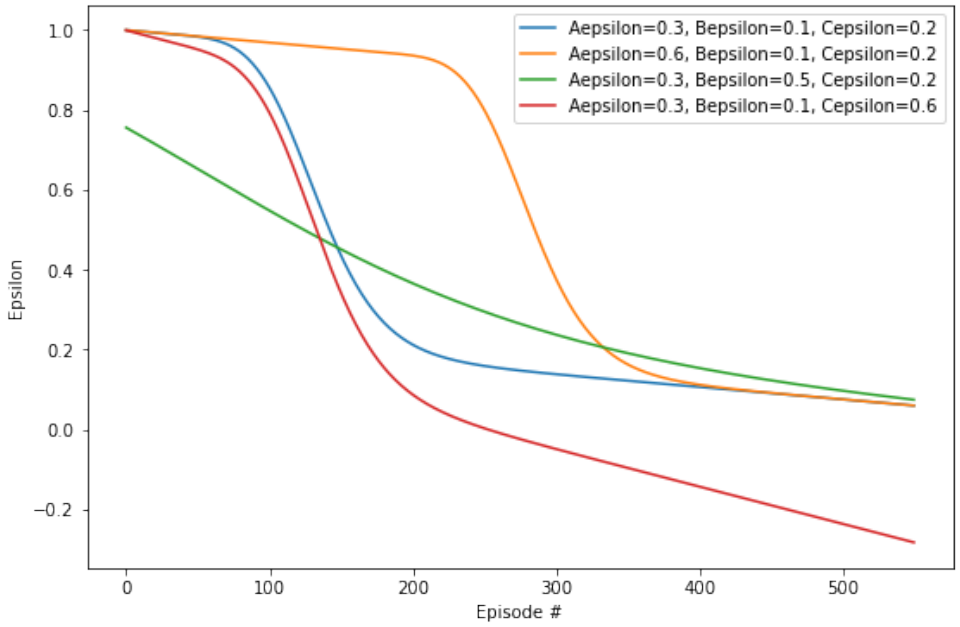

Figure 5.4.1: The influence of hyperparameters on the epsilon decay slope

Here $A_{epsilon}$ governs the length of the left tail until it starts decreasing, $B_{epsilon}$ determines the slope of the line and $C_{epsilon}$ determines the steepness of the left and right tail.

α is decayed with a different more simple equation. The equation used to determine the decay rate for α in Q-learning is described below in equation 5.34:

$$\text{Decay Rate} = \frac{\alpha_{min}}{\alpha_{current}}^{1/N_{episodes}} \tag{5.34}$$

5.4.1 Minibatch size and training frequency

As mentioned before, the minibatch is a sample of data taken from the replay buffer for the training of the neural network of the DDQN model. The minibatch size is set to 32, which means that 32 training data points are provided to the neural network with a single data point existing of the state space of 8 states and the corresponding q-values for every action. As the number of episodes increases, epsilon decreases according to equation 5.33 and the size of the minibatch is set to increase as well. Therefore the minibatch size starts with 32 but is increased to 640 increasing the stability of the model at the end

of the training cycle and making it less susceptible to noise.

Furthermore, in order to speed up the training time of the DDQN RL agent, the reward evolution and charging cost evolution over 890 episodes is studies for different training frequencies of the neural network. Lower training frequencies, allow for faster training speeds since the step of training the neural network with a mini batch takes up a relatively large amount of time. As can be seen in the figure below, figure 5.4.2 displays different training frequencies for the DDQN agent trained on multiple different cases and figure 5.4.3 for the DDQN agent trained on case study 2 only.

The top left figure shows that the final mean average reward is similar for the different training frequencies except for the training frequencies of 50 and 100 episodes for which the final reward is slightly less. The top right figure shows that for these two training frequencies the final mean cost is close to zero, which means that the EV is not charged at all, while the other frequencies have a mean average non-zero cost of around 2. It can again be observed that the costs is quite close to each other for the other training frequencies. However, when observing the mean average reward and cost evolution in figure 5.4.3 when training on case study 2 it becomes clear that only the highest frequency of training the neural network every 24 steps in a single episode is close to the performance of the frequency of training the neural network in every step. Therefore, for the DDQN model the decision was made to train the neural network every 24 steps, in contrast to in every step, significantly speeding up the training time while leading to a similar performance.

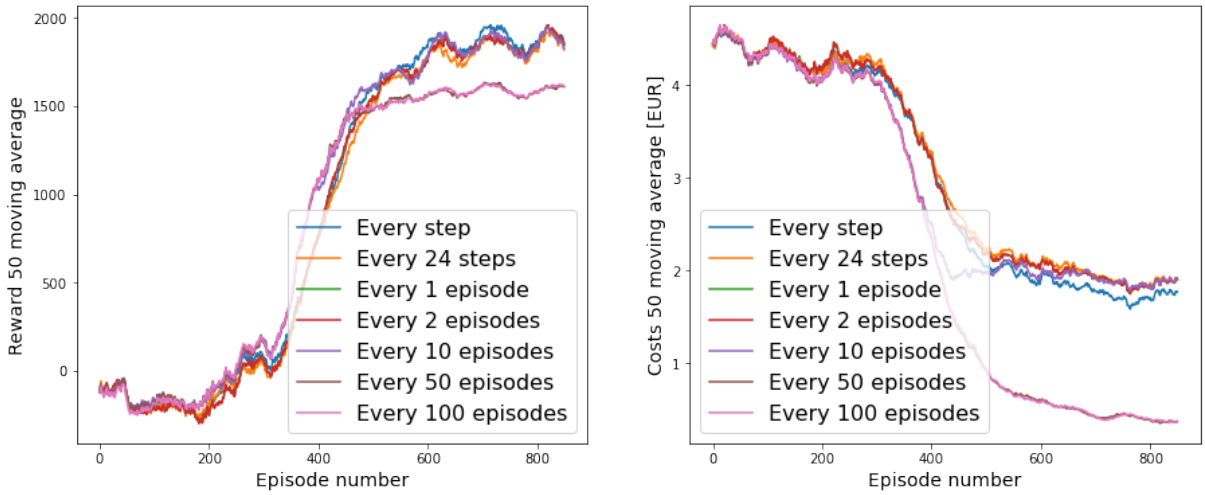

Figure 5.4.2: Plots of the reward evolution and charging cost trained on multiple different cases

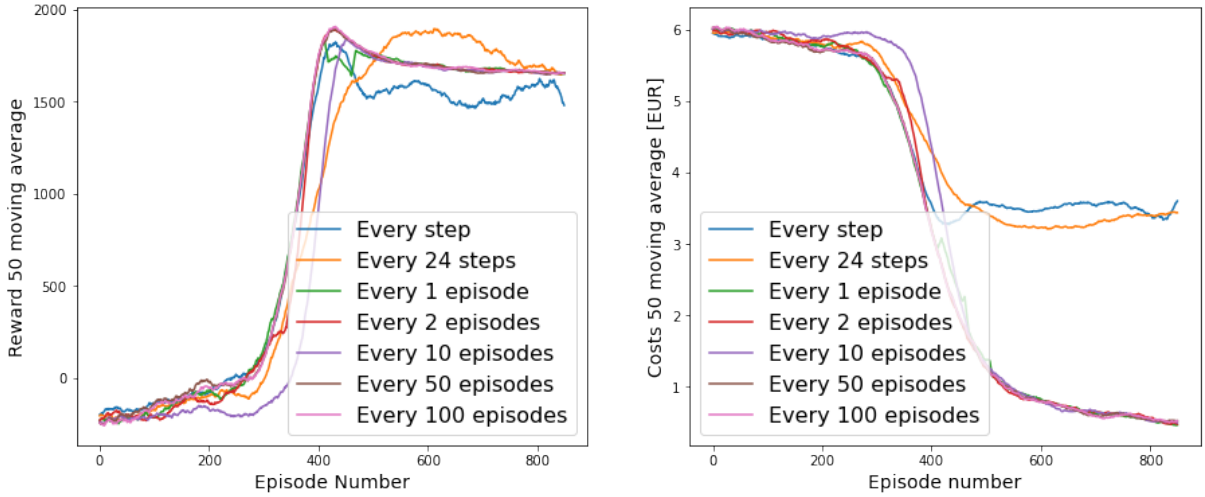

Figure 5.4.3: Plots of the reward evolution and charging cost trained on case study 2

6 TRAINING DATA AND CASE STUDIES

The purpose of this chapter is to discuss the training and testing procedure of the Q-learning and DDQN smart charging models. Training data is necessary to teach the ML models to output the right before while testing data has the function of evaluating the performance after the ML models have been trained. This chapter is organized as follows. First, in section 6.1 the case studies that are used for training and evaluation the Q-learning and DDQN model are discussed. Next, in section 6.2 the performance benchmarks for the evaluation of the RL models are elaborated on. Finally, in section 6.3 the exact training and testing data set-up is discussed in more detail.

6.1 CASE STUDIES

In order to consistently evaluate the performance of the reinforcement learning models, three different case studies are used to test the performance of the agent. These three specific case studies were selected because they differ from each other and therefore challenge the reinforcement learning agents in alternative ways. The exact specifications of these case studies are provided in the table 6.1.1 below.

	Case 1	**Case 2**	**Case 3**
Price Pattern	High/Low/High	High/Low	Mid/High/Low
TTL [hours]	12	12	3
Arrival Battery Capacity [kWh]	10	10	30

Table 6.1.1: Case study parameters for three different case studies

While the price pattern of case 1 challenges the reinforcement learning agents to prolong charging until roughly one-third of the time window when prices are low, in case 2 the vehicle can be charged most cheaply at the end of the time window. Furthermore, case 1 and case 2 have a vehicle leaving time of 12 hours and an arrival battery capacity of 10 KWh. Finally, case 3 represents an alternative corner case in which the electricity prices are lowest at the end of the time window. However, in case 3 the vehicle leaves after 3 hours, which requires the agent to charge the vehicle right from the start when the electricity prices are the lowest within the restricted charging window of 3 hours.

6.2 BENCHMARKS

As for the performance benchmarks, both the Q-learning and DDQN model are compared to the performance of the average rate charging scheme and the original MIP solver when testing on the individual case studies both prior to and after training on many different cases. When evaluating the average performance on multiple different cases, the Q-learning and DDQN performance are compared to each other.

6.2.1 Average rate charging benchmark

As a first benchmark, the charging cost, speed, and final achieved SOC of the reinforcement learning model are compared to that of average rate charging. With average rate charging the power provided to the EV is equal to the total energy required divided by the time available. This power is then converted to a current equal to or larger than the minimum non-zero charging current level of 6 A. Therefore, if the average required current by the vehicle in every time step is below 6 A, the charging current is set

to 6 A and the vehicle is fully charged before the EV owner intends to leave, as described in equation 6.1.

$$I_t^{e+} = \begin{cases} \frac{E_{asked}}{(T_{dep}-T_{arr})\times 230\times 3} & I_t^{e+} > 6 \\ 6 & I_t^{e+} \leq 6 \end{cases}, \qquad t \in T \tag{6.1}$$

6.2.2 MIP benchmark

As a second benchmark, next to the average rate charging benchmark, the performance of a simplified version of the MIP solver for the project Orchestrating Smart Charging in mass Deployment (OSCD) is evaluated for each case study [53]. As mentioned before in chapter 1, the MIP solver is a mathematical optimization smart charging algorithm with the goal of reducing the cost of charging EVs. Since the original OSCD algorithm allows more complex computations, the objective function has been set to match the objectives of the RL models:

$$\text{Min. } C^{opt} = C^p(B_{SOC}^{Arrival} + d - B_{SOC}^{Final}) + \Delta T \sum_{t=1}^{T}(P_t C_t^{buy} - P_t C_t^{sell}), \qquad t \in T \tag{6.2}$$

In the equation above, C^p has been set to 10 Euro per missing percentage of SOC, d represents the energy required to reach full capacity and B_{SOC}^{Final} is the battery SOC at the end of the time horizon. Furthermore, equations 5.1.4 until 5.1.4 describing the environment of the RL models are used as sub-functions for the MIP solver. In addition to the constraints mentioned for RL in section 5.1.4 and the equations 5.1.4 until 5.1.4 in the environment, a number of additional constraints are also formulated for the MIP solver:

$$P_t^{PV} \leq K_t^{PV} p^{PVr} p_t^{PV(fc)} \eta^{inv}, \qquad t \in T \tag{6.3}$$

When EV is connected:

$$B_t \geq B_{SOC}^{Arrival}, \qquad t \in T \tag{6.4}$$

$$B_t \leq \text{Min}\{d + B_{SOC}^{Arrival}, B^{max}\}, \qquad t \in T \tag{6.5}$$

$$B_t \geq B_{SOC}^{min}, \qquad t \in T \tag{6.6}$$

When EV is not connected:

$$I_t^{e+}, P_t, B_t = 0, \qquad t \in T \tag{6.7}$$

Finally, all other external factors related to the MIP solver such as the case-specific electricity buying and selling prices, the PV power, and building load have been set to be the same as for the RL model.

6.3 TRAINING AND TESTING

One of the most important steps that determine the performance of the RL agents is the way in which it has been trained. As part of the training process, first, a data set is created for training, after which it is split into 90% training data and 10% testing data. This guarantees that the evaluation results are not biased since the agent has no prior experience with the testing data.

To further clarify the training and testing set-up, figure 6.3.1 below shows an image portraying which trained models are tested on which case studies. First, both algorithms are trained and tested on the three individual case studies only (prior to generalization), which means that the agent is trained on the same case study for all episodes and has not been trained to generalize. However, ultimately, it is the goal to have the Q-learning and DDQN smart charging models perform well for arbitrary price patterns including a varying TTL value and arrival battery capacity value. Therefore, the agent is also trained on many different random case studies to assess the generalization performance of the agent.

To improve the generalization performance in every episode the agent is provided with a new price pattern, TTL value, and battery arrival capacity. Subsequently, the Q-learning and DDQN models trained in many different episodes (after generalization) are tested on the three individual case studies as well as on 10 thousand different cases.

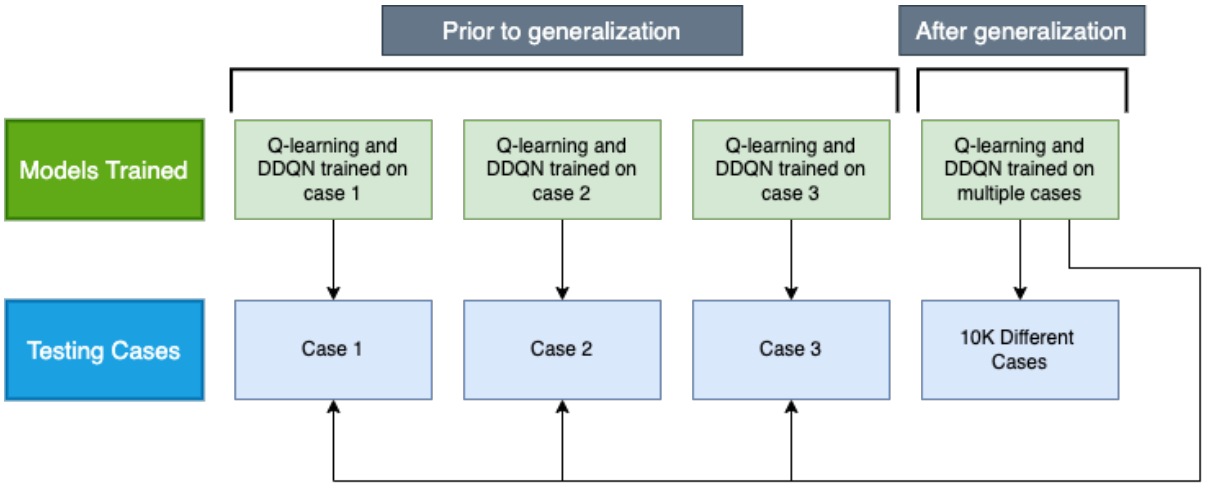

Figure 6.3.1: Overview showing which training and testing data the different Q-learning and DDQN models are trained on

6.3.1 Training Q-learning

When training the Q-learning agent on the three case studies only, a total of 450 thousand episodes were trained on. For all episodes, both the vehicle leaving time and arrival battery capacity are kept constant for every case study according to the previously mentioned values in table 6.1.1. The figure below shows the moving average reward and moving average cost every 1000 episodes prior to generalization on cases 1, 2, and 3.

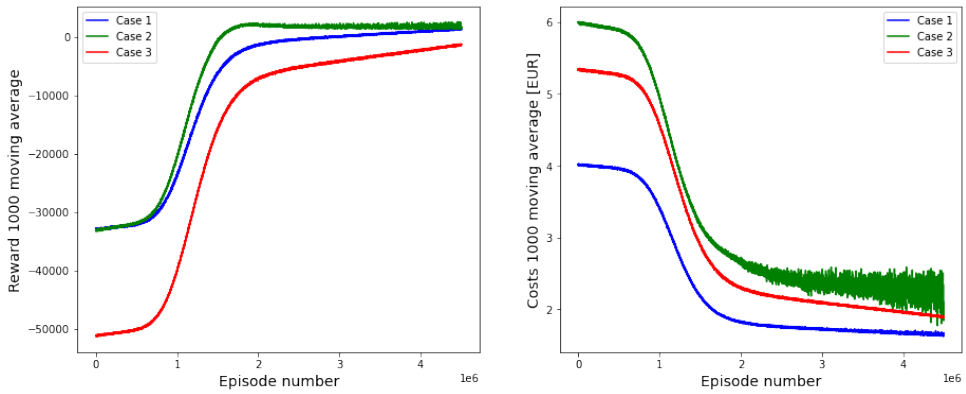

Figure 6.3.2: The increasing moving average reward and declining moving average cost every 1000 episodes for the Q-learning algorithm for 450K different cases on cases 1, 2 and 3

In the figure above it can be seen that the agent starts off with a negative reward, as it receives many punishments for overcharging, after which it slowly learns to stop overcharging and is able to achieve higher average rewards over time. Furthermore, the right plot demonstrates the decrease in the average EV charging cost over time as it decreases over the 450 thousand episodes. The final reward achieved in cases 1 and 2 is significantly higher than the final reward achieved in case 3. Furthermore, the lowest cost is achieved in case 1, while the highest cost is achieved in case 2. Therefore, if the vehicle has been charged to 100%, the right plot provides an indication of to which degree the objective of charging the vehicle as cheap as possible has been reached. However, it must be taken into account that the average charging cost per episode depends on two factors and can be misleading. The first is the price pattern

of the episode and the second is the final SOC reached in that episode. To start with, episodes with relatively low electricity prices inherently lead to lower charging costs. Therefore, it is not necessarily true that a low total charging cost is the result of smart charging. In addition, if a vehicle is not charged to 100% the plot can be misleading since not charging the EV to full capacity will always result in a relatively lower total charging cost than successful charging the EV to full capacity.

To test the Q-learning generalization performance, the Q-learning agent was trained on 45 million episodes under the hyperparameters provided in table 5.4.1. Every episode consists of a random price pattern in the year, with a random TTL value between 3-12 hours and an arrival battery capacity between 10-30 kWh in steps of 5 kWh. Figure 6.3.3 below displays the moving average reward and moving average cost every 10 thousand episodes over a total of 45 million episodes.

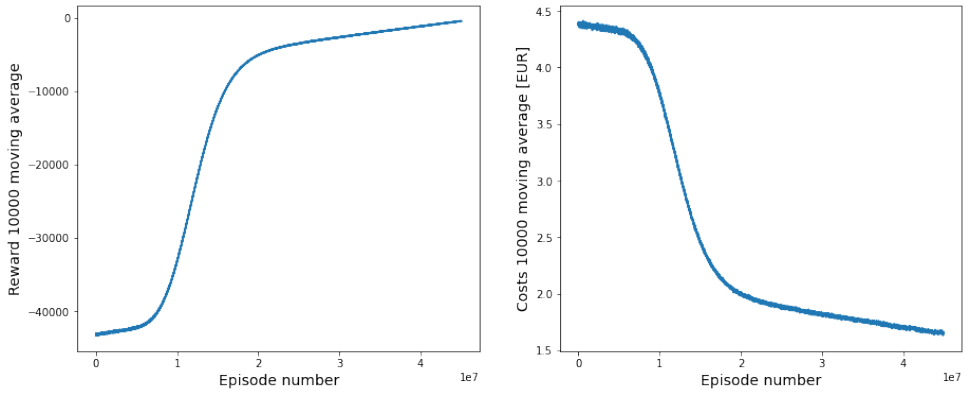

Figure 6.3.3: The increasing moving average reward and declining moving average cost every 10,000 episodes for the Q-learning algorithm for 45 million different cases

From this figure, it becomes clear that the Q-learning agent learns how to increase the reward over time and that the reward function successfully stimulates the agent to lower the costs of charging the EV. After training, the moving average reward is close to 0, as is the case for the individual case studies and the moving average total EV charging cost per episode is between 1.5 and 2 Euro after 45 million episodes. However, this again depends highly on the price pattern of the episodes and the final SOC.

6.3.2 Training DDQN

When training the DDQN agent on the three case studies an amount of 45 thousand episodes are trained on. The number of episodes trained on for the DDQN agent is significantly less due to the relatively slower training speed compared to Q-learning. Just like for the Q-learning agent, both the vehicle leaving time value and arrival battery capacity is kept constant for every case study. The figure below shows the evolution of the moving average reward and EV charging cost for every episode prior to generalization on cases 1, 2, and 3. The figure demonstrates that the DDQN agent is able to learn from the reward function since the reward increases over time for every case study.

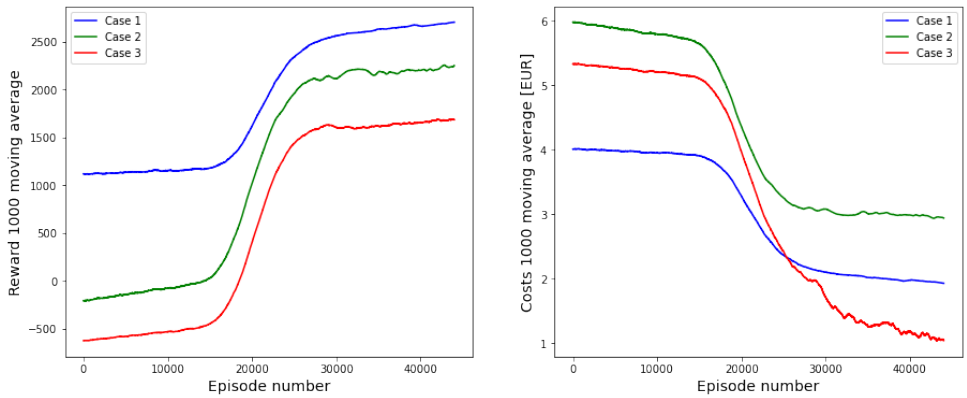

Figure 6.3.4: The increasing moving average reward and declining moving average cost every 1000 episodes for the DDQN algorithm for 45 thousand different episodes on cases 1, 2 and 3

In contrast to the Q-learning model, the DDQN agent reaches significantly different rewards for every case with the agent reaching the highest reward for case 1. This is the case since the weight in the reward function for reaching a SOC target for the DDQN algorithm is normalized and therefore depends highly on the price pattern in relation to the maximum and minimum price in every case under consideration. Therefore, while the final reward is higher for case 1 than in case 2, it does not necessarily mean that the vehicle is charged "smarter" in case 1 than in case 2.

Furthermore, from the right plot, we can see that the agent is effective in reducing the average EV charging cost over the 45 thousand episodes. What stands out in the plot is the relatively low final charging cost for case 3. As will be further discussed in chapter 7 this is a result of bad charging performance and an example of how the charging cost plot can be misleading since not charging the EV to full capacity inherently leads to low charging cost. vspace2mm
In order to test the generalization performance of the DDQN agent, the agent was trained on 180 thousand episodes under the hyperparameters provided in the table 5.4.1. Figure 6.3.5 below displays the evolution of the moving average reward over 180 thousand episodes and the moving average EV charging cost evolution over 180 thousand episodes.

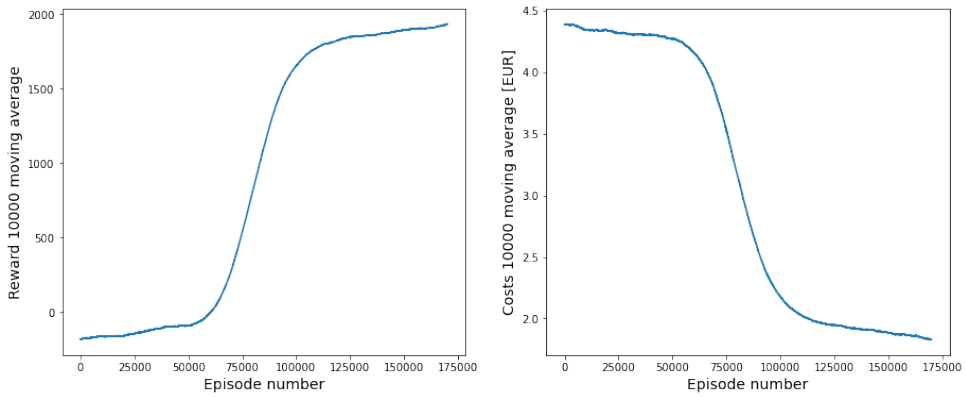

Figure 6.3.5: The increasing moving average reward and declining moving average cost every 1000 episodes for the DDQN algorithm for 180 thousand different cases

From this figure, it becomes clear that the agent improves its generalization performance over time, as the moving average reward increases. Furthermore, after training, the right plot shows that the moving average total EV charging cost per episode is slightly lower than 2 euro.

7 PERFORMANCE RESULTS

Since it is the goal of this research to assess the feasibility of reinforcement learning as a replacement for mathematical optimization for smart charging, the most important point of interest is the extent to which the Q-learning and DDQN agents are able to perform ideal charging behaviour. Furthermore, while the plots of the evolution of the moving average reward and EV charging cost give an indication of the performance, exact performance benchmarks are required to be able to assess the degree to which the objectives of smart charging are reached. Therefore, the purpose of this chapter is to discuss the evaluation results of the Q-learning and DDQN algorithm for the smart charging case studies as well as their generalization performance. This chapter is organized as follows. First, in section 7.1 until 7.3 the results of the performance of the RL models on the previously mentioned three case studies are provided, both prior and after generalization. Finally, in section 7.4 the average performance of the RL agents on multiple cases is analyzed in more detail.

7.1 CASE STUDY 1

To start with, case study 1 is a case with a price pattern that starts off with high electricity prices that decrease and then ramp up again with a TTL value of 12 hours and an arrival battery capacity of 10 kWh. When considering the price pattern corresponding to case study 1 in figure 7.1.1 below it becomes clear that an ideal charging schedule would be to not charge the EV for the first 40 time steps, but to completely charge the EV to a SOC of 100% between roughly time step 40 and 80 when the electricity prices are relatively low. When taking a closer look at the charging performance of the different algorithms in figure 7.1.1, after the RL agents have continuously trained on the same case, we can see that the MIP charger charges exactly in this time period, while the Q-learning agent and DDQN agent do not. Furthermore, the Q-learning agent's performance is the closest to the MIP performance, while the DDQN agent fails to show ideal charging behaviour since it charges the EV at the start of the episode when prices are relatively high. In addition, it can be seen in figure 7.1.1 that all agents are able to successfully charge the vehicle to 100% before the vehicle intends to leave at the end of the episode.

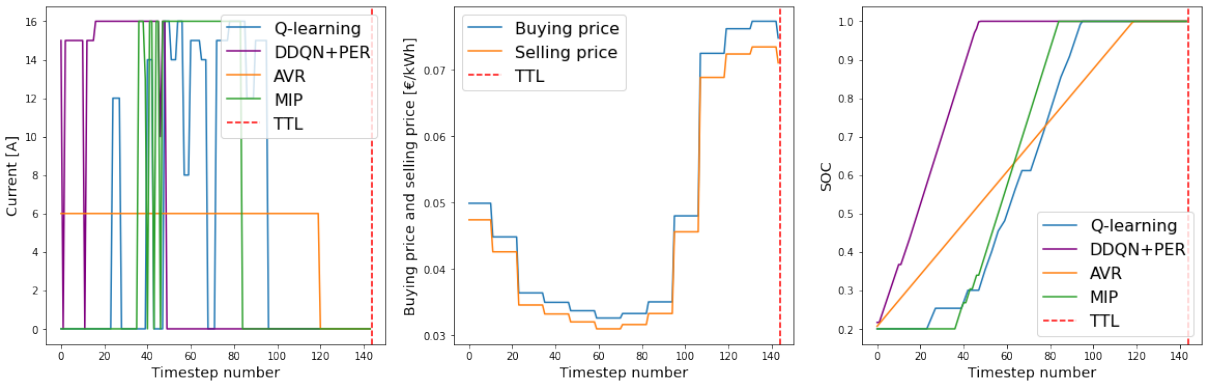

Figure 7.1.1: Plot of the charging current, electricity buying and selling prices and the EV SOC of the RL agents, AVR charger and MIP prior to generalization on case 1

After having discussed the general charging performance, more quantitative insight into the charging performance can provide further detail and improve the understanding of the RL agents' potential. Therefore, table 7.1.1 below provides an overview of a number of quantitative metrics related to the charging performance of the RL agents in comparison to the AVR and MIP benchmarks for case 1.

To start with, these metrics show that there is no occurrence of a constraint violation both prior and after generalization, which means that the constraint penalty is effective in preventing overcharging.

Furthermore, it can be noted that prior to generalization the total EV charging cost for the Q-learning agent is almost just as low as the EV charging cost of the MIP agent, both with a relative percentage EV charging cost difference of -18.46% and -21.08 % to the average rate charging schedule. Surprisingly, in contrast to the Q-learning agent, the DDQN agent performed significantly worse than the Q-learning agent with an average EV charging cost of 0.0428 Euro/kWh and a relative percentage EV charging cost difference of -2.36%. However, both agents perform better than the average rate charging schedule.

Case 1 - Prior to generalization	**Q-learning**	**DDQN**	**AVR**	**MIP**
Occurrence of a constraint violation	No	No	No	No
Final SOC reached by the RL agent	1.0	1.0	1.0	1.0
Total EV charging cost [Euro/kWh]	0.036	0.0428	0.044	0.034
Relative percentage EV charging cost difference in comparison to AVR charger [%]	-18.46	-2.36	0	-21.08
Execution speed [seconds/episode]	0.0171	0.2845	-	0.060

Table 7.1.1: Smart charging results of the RL agents and the AVR and MIP benchmarks prior to generalization on case 1

Figure 7.1.2 shows the charging performance for the RL agents after generalization. These results demonstrate that the Q-learning and DDQN agents charging performance on case 1 remains relatively unchanged after training on many different episodes. This means that the training on many different episodes instead of on case study 1 only has hardly improved the charging performance of the agent one case 1. A possible explanation for this is that when training on many different episodes, the RL agents can not explore all possible options in every single episode and therefore are less successful in finding the optimal charging moments. Furthermore, minor differences in the charging performance include the Q-learning agent and DDQN charging slightly earlier than prior to generalization and the DDQN agent charging at a constant current of 16A in contrast to the previously fluctuating current levels. Nevertheless, both RL agents remain successful at reaching a SOC of 100% before the end of the episode.

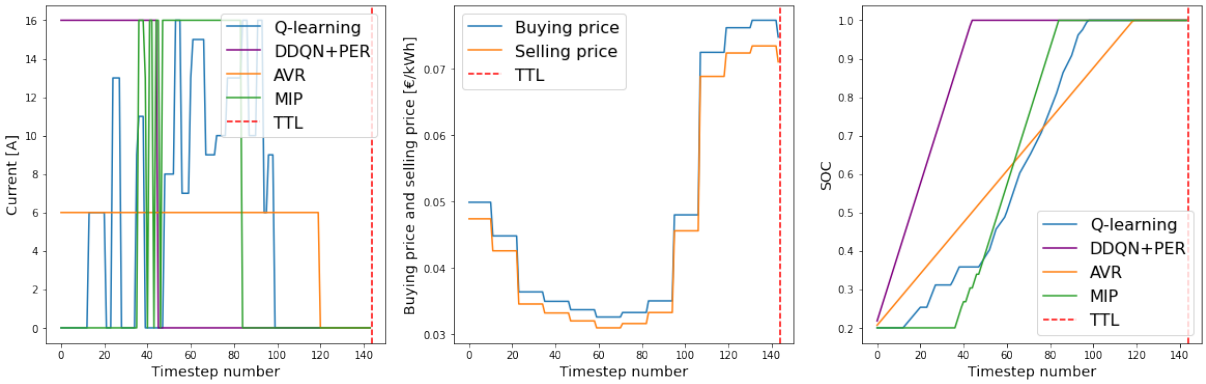

Figure 7.1.2: Plots of the charging current, electricity buying and selling prices and the EV SOC of the RL agents, AVR charger and MIP after generalization on case 1

When considering the quantitative performance in table 7.1.2, both the Q-learning and DDQN agent's performance after generalization is worse than when training on case 1 only. This is in line with expectations since it is expected that training and testing on a single episode provides more opportunity for the agent to explore the cheapest charging moments than when training on many different cases.

On another note, with regard to the execution speed of the agents in case 1, the execution speed in both tables show that the Q-learning agent is significantly faster than the DDQN agent and roughly 4 times faster than the MIP solver. However, the DDQN agent is a lot slower than the MIP solver. This can be explained by the fact that the DDQN agent uses its neural network to predict Q-values to base its actions on, while the Q-learning agent uses its Q-table as a lookup table which is significantly faster.

Case 1 - After generalization	**Q-learning**	**DDQN**	**AVR**	**MIP**
Occurrence of a constraint violation	No	No	No	No
Final SOC reached by the RL agent	1.0	1.0	1.0	1.0
Total EV charging cost [Euro/kWh]	0.037	0.043	0.044	0.034
Relative percentage EV charging cost difference in comparison to AVR charger [%]	-16.26	-1.69	0	-21.08
Execution speed [seconds/episode]	0.0324	0.2775	-	0.060

Table 7.1.2: Smart charging results of the RL agents and the AVR and MIP benchmarks after generalisation on case 1

7.2 CASE STUDY 2

When comparing case study 2 to case study 1, it must be noted that case 2 has been set to have the same TTL value of 12 hours and an arrival battery capacity of 10 kWh. However, the price pattern starts with a high price after which it decreases near the end of the episode challenging the RL agents to delay charging to the end of the episode.

Close analysis of the plots of the charging performance prior to generalization in figure 7.2.1 reveals that both the Q-learning and DDQN agent successfully delay charging of the EV. Just like for case study 1, the MIP simulation results show ideal charging behaviour as it charges the vehicle in time step 100 to 144 when prices are lowest. However, while the DDQN agent charges most of the EV between the 60th and 120th-time step, the Q-learning agent charges most of the EV at the end of the episode when prices are the lowest outperforming the DDQN agent. Therefore, just as for case study 1 the charging performance of the Q-learning agent more closely mimics the charging performance of the MIP simulation than the DDQN agent.

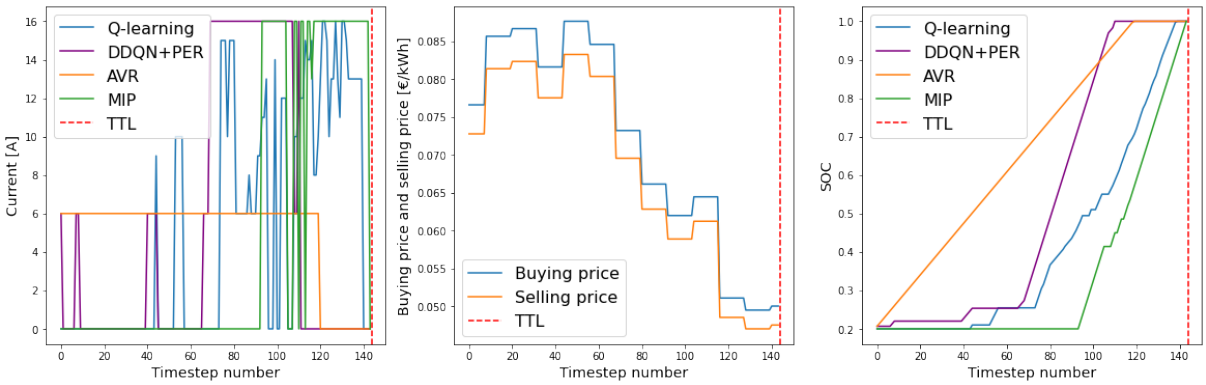

Figure 7.2.1: A plot of the charging current, electricity buying and selling prices and the EV SOC of the RL agents, AVR charger and MIP prior to generalization for case 2

Analysis of the quantitative performance metrics in table 7.2.1 shows that no constraints are violated and that the Q-learning and DDQN agent both have a lower total EV charging cost in Euro/kWh than the average rate charger. Nevertheless, the MIP simulator still outperforms both the Q-learning and DDQN agent with the EV charging cost difference relative to the average rate charger being -27.18% for the MIP simulation, -19.13% for the Q-learning agent and -10.84% for the DDQN agent. However, it is important to correctly interpret these results. It is expected that the relative cost difference is higher in case study 2 than in case study 1, since the AVR charger blindly charges from the start and therefore in this case it charges throughout the times steps with relatively high electricity prices, increasing the relative cost difference. Therefore, the relative cost difference does not only depend on the charging performance of the agent but also depends on the price pattern. Price patterns with higher prices at the start of the time window naturally lead to higher cost differences if cheaper charging opportunities are at the end of the charging window.

Case 2 - Prior to generalization	Q-learning	DDQN	AVR	MIP
Occurrence of a constraint violation	No	No	No	No
Final SOC reached by the RL agent	1.0	1.0	1.0	1.0
Total EV charging cost [Euro/kWh]	0.064	0.070	0.079	0.057
Relative percentage EV charging cost difference in comparison to AVR charger [%]	-19.13	-10.84	-	-27.18
Execution speed [seconds/episode]	0.0229	0.2741	-	0.0499

Table 7.2.1: Smart charging results of the RL agents and the AVR and MIP benchmarks prior to generalization on case 2

Taking the performance after generalization under consideration, as can be seen in figure 7.2.2 both the Q-learning agent and DDQN agent lose their ability to charge the vehicle at the end of the episode. In contrast to the DDQN's charging performance in case study 1, for case study 2 the charging performance is significantly changed after generalization as it charges most of the EV in the first half of the episode increasing the total EV charging cost. This suggests that after generalization case study 2 is a lot more challenging for the RL agents than case study 1. This can be explained by the fact that it is harder for an agent to learn to charge only at an end of an episode than earlier in an episode. The reason for this is that in order for the agent to find out through exploration that it is best to prolong charging to the end of an episode, a long sequence of zero current actions needs to follow each other up. The chances of this happening through random exploration are a lot smaller than for a sequence of non-zero actions. Therefore, case study 2 can be considered more challenging than case study 1, resulting in a large change in the charging performance. However, despite this, it can still be noted that the agent reaches the goal of charging the EV to a SOC of 100% both prior to and after generalization.

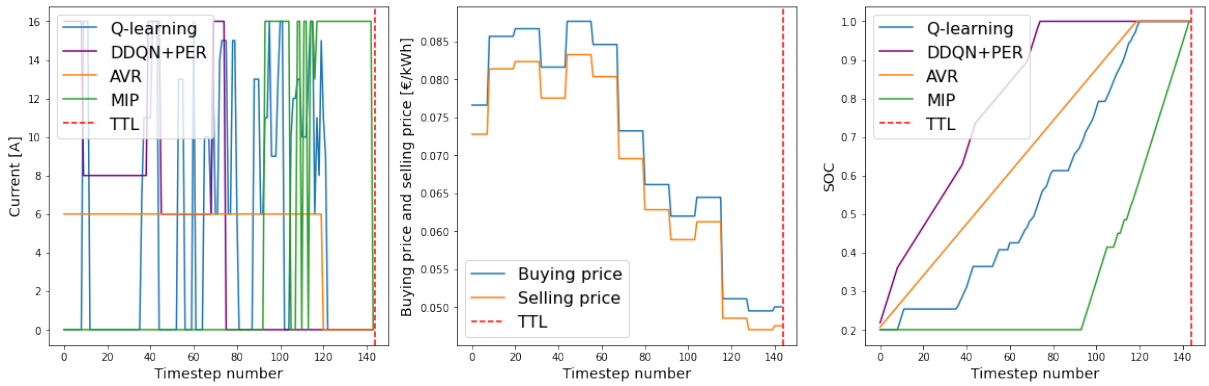

Figure 7.2.2: A plot of the charging current, electricity buying and selling prices and the EV SOC of the RL agents, AVR charger and MIP after generalization for case 2

When assessing the quantitative performance on case 2 of the RL agents after generalising on many episodes, it can be seen in table 7.2.2 that the total EV charging cost increases significantly due to charging at earlier time steps. As a result, the EV charging cost difference relative to the AVR charger of the DDQN agent even becomes 7.85%. These quantitative results also make it clear that for case study 2 the generalization has severe consequences for the charging performance of the RL agents. In a similar fashion to case study 1, the agents trained to generalize tend to charge earlier than prior to generalization.

On a final note, the execution speed of the Q-learning agent is again significantly lower than that of the DDQN agent and the MIP solver. The DDQN execution time is 0.2879 seconds, which is a lot higher than that of the MIP solver and that of the Q-learning agent.

Case 2 - After generalization	Q-learning	DDQN	AVR	MIP
Occurrence of a constraint violation	No	No	No	No
Final SOC reached by the RL agent	1.0	1.0	1.0	1.0
Total EV charging cost [Euro/kWh]	0.074	0.085	0.079	0.057
Relative percentage EV charging cost difference in comparison to AVR charger [%]	-5.9	7.85	-	-27.18
Execution speed [seconds/episode]	0.015	0.2879	-	0.049

Table 7.2.2: Smart charging results of the RL agents and the AVR and MIP benchmarks after generalisation on case 2

7.3 CASE STUDY 3

Finally, case study 3 can be considered a corner case with a TTL value of 3 and an arrival battery capacity of 30 kWh. The shorter TTL value compared to the other cases limits the flexibility of the agent to find relatively cheap moments to charge the EV.

As can be seen in figure 7.3.1, ideally the RL agents should charge the EV in the first 20 time steps and right after the peak in the electricity price around time step 30. When evaluating the charging performance of the RL agents and benchmarks prior to generalization for case 3, it can be seen that the MIP simulation results follow this ideal charging schedule. However, both the DDQN and Q-learning agent fail to identify the right charging moments, with the DDQN agent hardly charging the vehicle at all. The charging performance of the DDQN agent, in particular, stands out, since it only charges the vehicle to a final SOC of 0.616, while the other models successfully charge the vehicle to a SOC of 1 before the vehicle intends to leave after 3 hours.

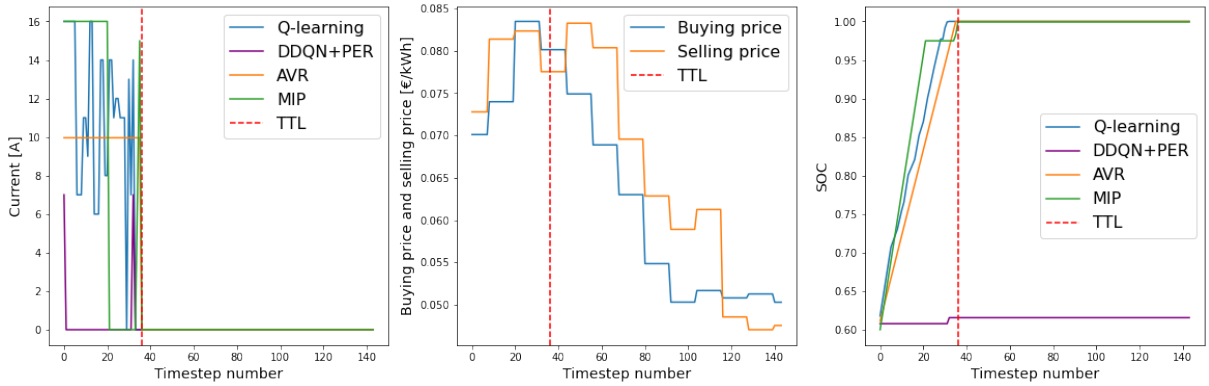

Figure 7.3.1: A plot of the charging current, electricity buying and selling prices and the EV SOC of the RL agents, AVR charger and MIP prior to generalization for case 3

When taking a closer look at the performance metrics for case 3 in table 7.3.1, one can find that also for this case the constraints related to overcharging and charging the vehicle once the vehicle has left are not violated and that the penalty in the reward function is effective. Furthermore, it can be seen that after training the charging costs of the Q-learning agent is higher than that of the AVR charger. This suggests that low vehicle leaving times are more challenging for the RL agents. Furthermore, it can be noted that the DDQN is total EV charging cost difference is -96.19% cheaper than the MIP solver. However, as mentioned before, this is misleading as the DDQN agent has not been charged to 100%.

Case 3 - Prior to generalization	**Q-learning**	**DDQN**	**AVR**	**MIP**
Occurrence of a constraint violation	No	No	No	No
Final SOC reached by the RL agent	1	0.616	1	1
Total EV charging cost [Euro/kWh]	0.082	0.077	0.079	0.71
Relative percentage EV charging cost difference in comparison to AVR charger [%]	2.75	-96.19	-	-9.63
Execution speed [seconds/episode]	0.0365	0.2817	-	0.0160

Table 7.3.1: Smart charging results of the RL agents and the AVR and MIP benchmarks prior to generalization on case 3

After generalization, figure 7.3.2 shows that Q-learning has shifted its charging to earlier time steps more closely mimicking the ideal charging behaviour. Furthermore, it can be noted that the DDQN agent now succeeds in charging the EV battery to 100%, while also charging the EV at earlier time steps. With regard to case study 3, the results show that in contrast to case studies 1 and 2 the generalization of both the Q-learning and DDQN agent by training on multiple cases has had a positive influence on the charging performance for case 3. This is caused by an overall increased tendency after generalization to charge the EV faster, which behaviour we also recognized in case study 1 and case study 2. Charging the EV faster limits the risk of not charging the vehicle to 100% and as a consequence losing out on the SOC reward.

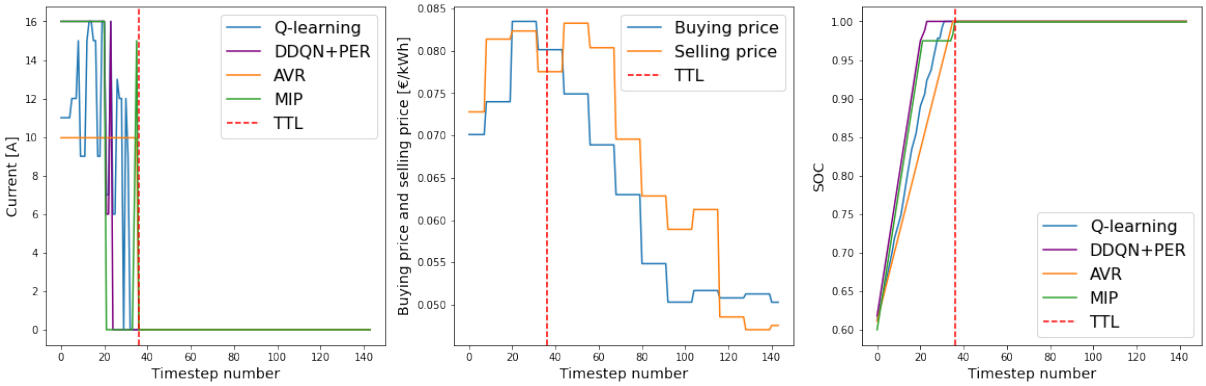

Figure 7.3.2: A plot of the charging current, electricity buying and selling prices and the EV SOC of the RL agents, AVR charger and MIP after generalization for case 3

This is reflected in the relative percentage EV charging cost difference in comparison to the AVR charger as can be seen in table 7.3.2 since it is -0.97% for the Q-learning agent and -2.71% for the DDQN agent after generalization. This means that through experience with many other cases the agent has received feedback through the reward function changing the output of the RL agents for case study 3.

Finally, compared to the other two cases, the MIP simulation execution speed is significantly lower for case 3 being roughly equally as fast as the Q-learning agent. The DDQN agent's execution speed is around 0.2816 seconds, which is significantly slower than the execution speed of the MIP solver in line with the execution speed for the other case studies.

Case 3 - After generalization	Q-learning	DDQN	AVR	MIP
Occurrence of a constraint violation	No	No	No	No
Final SOC reached by the RL agent	1.0	1.0	1.0	1.0
Total EV charging cost [Euro/kWh]	0.079	0.077	0.079	0.071
Relative percentage EV charging cost difference in comparison to AVR charger [%]	-0.97	-2.71	0	-9.63
Execution speed [seconds/episode]	0.0158	0.2816	-	0.016

Table 7.3.2: Smart charging results of the RL agents and the AVR and MIP benchmarks after generalisation on case 3

7.4 MULTI-EPISODE EVALUATION RESULTS OF THE REINFORCEMENT LEARNING AGENT

While the previous section showcases examples of the charging performance on specific case studies, it is more insightful to test the average performance of the Q-learning and DDQN models after generalization on 10,000 cases that the agents have not seen before. Each case study has a unique price pattern, TTL value and arrival battery capacity combination.

It must be noted that since the DDQN algorithm allows for continuous arrival battery capacity values, both the Q-learning and DDQN algorithms are tested on slightly different case studies. Nevertheless, the huge variety in data makes it an effective way of testing the average performance of both smart charging models.

Table 7.4.1 provides an overview of the average performance of the Q-learning and DDQN models on these random case studies. To start with, it can be seen that the Q-learning agent does not violate constraints in 96% of the charging sessions, which means that in 96% of the cases the EV is not charged when already left or fully charged. When comparing this metric to the DDQN agent, we find that in 90% of the charging sessions the DDQN agent does not violate constraints which is less. A possible explanation for this is that in the Q-learning model the penalty for overcharging is a lot higher than in the DDQN model. Furthermore, for DDQN the EV is charged to an average final SOC of 0.99, which is slightly higher than the final SOC for Q-learning of 0.94. These results demonstrate the ability of both the Q-learning and DDQN agents to learn to not violate the overcharging constraints through a penalty in the reward function and the ability of the agents to reach close to full SOC levels.

Metric	Q-learning	DDQN
Percentage of episodes in which constraints are not violated by the Rl agent [%]	96%	90%
Average final SOC reached by the RL agent	0.94	0.99
Average Execution Speed [seconds/episode]	0.0055	0.347
Percentage of episodes that reach max possible SOC [%]	85%	91%
Average charging cost difference relative to AVR for all charging sessions that reach max possible SOC [%]	-1.72%	-1.05%
Percentage of sessions that are cheaper than AVR and reach max possible SOC [%]	58.14%	46.69%
Average charging cost difference relative to AVR for all charging sessions that are cheaper and reach max possible SOC [%]	-6.2%	-8.61%
Average EV charging cost [Euro/kWh] for the RL agent	0.053 Euro/kWh	0.054 Euro/kWh
Average EV charging cost [Euro/kWh] for AVR	0.054 Euro/kWh	0.054 Euro/kWh

Table 7.4.1: Results for different metrics evaluating the average performance of the Q-learning and DDQN agents on multiple cases

Another metric of importance to evaluate the feasibility of the RL agent as a replacement for the MIP smart charging algorithms is the average execution speed. While the MIP execution speed for the case studies was between 0.016 and 0.060 seconds per episode, the Q-learning agent is significantly faster on average as it takes 0.0055 seconds on average for the Q-learning agent to execute an episode. The average execution speed of the DDQN agent is 0.347 seconds, which is remarkably slower than both the MIP and the Q-learning agent.

Furthermore, the evaluation results show that for Q-learning in 85% of the charging sessions, the EV is charged to the maximum possible SOC percentage while for the DDQN agent this is a higher percentage of 91%. Here, the maximum possible SOC percentage is defined as the maximum percentage reached if the agent were to charge with full power in every time step. However, turning to the average charging cost difference relative to AVR for all these sessions we find that both the Q-learning and DDQN agents perform quite poorly, with both only being -1.72% and -1.05% cheaper.

While the previous metric gives a good indication of the charging performance, it is also interesting to see what percentage of these sessions are actually cheaper. The evaluation of the performance on 10,000 different episodes shows that the Q-learning agent is cheaper in only 58% of the 85% of the cases that reach the maximum possible SOC with an average cost difference of -6.2%. Surprisingly, the DDQN agent is only cheaper in 46.69% of the cases in which the agent reaches the maximum possible SOC which is less but does have a higher average cost difference of -8.61 %. This means that despite the Q-learning agent being cheaper than the average rate charger on more cases that reach the final SOC than the DDQN agent, for these cases the DDQN agent is cheaper on average.

Finally, the multi-episode evaluation results show that when it charges the total average EV charging cost in Euro/kWh for the Q-learning agent and DDQN agent is less than the average rate charger. Moreover, it is the case that only the Q-learning agent is on average only 0.01 Euro/kWh cheaper than the AVR charger and that the DDQN agent is on average just as cheap. This shows the limited extent to which the agents are able to find the cheaper charger moments after generalization.

7.4.1 Q-learning performance analysis

In order to have a better understanding of the reason behind the charging performance of both algorithms, a closer look at the two goals of a cheap EV charging cost and a high as possible SOC for different vehicle leaving times and arrival battery capacities can provide better insight.

Figure 7.4.1 and 7.4.2, therefore, provide an overview of the relative charging cost difference percentage compared to the average rate charger for all sessions that reach a maximum charging target for different vehicle leaving times and for the different arrival battery capacities. Furthermore, the degree of completion is plotted for different vehicle leaving times and arrival battery capacities. The degree of completion is defined as the degree in percentage to which the agent was able to reach the maximum possible charging target.

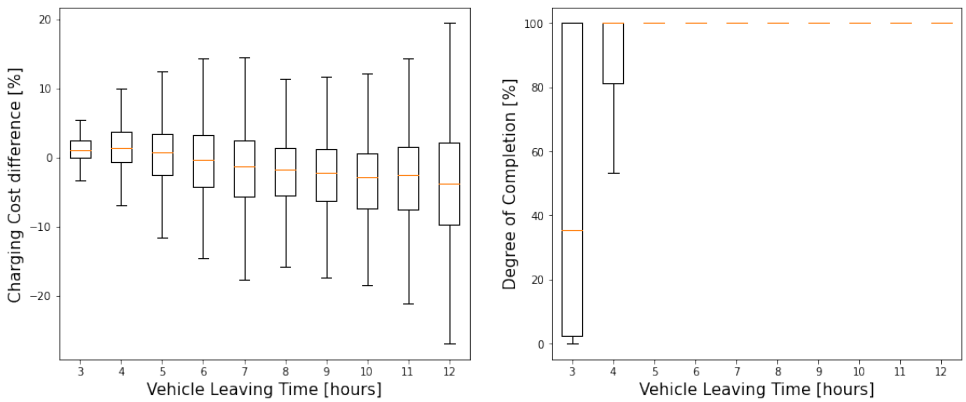

Figure 7.4.1: Box plots for the vehicle leaving time versus charging cost, discarding outliers

Given that the vehicle leaving time distribution is random and therefore can be considered to be an equal distribution for every vehicle leaving time, a clear trend can be identified, since on average the agent is more often less cheap for lower vehicle leaving times than for higher vehicle leaving times. Furthermore, a second trend can be identified, namely that the longer the vehicle leaving time the more likely the agent is able to provide a charging plan that charges the vehicle to a degree of completion percentage of 100%. Moreover, the length of the box plot whiskers for the charging cost difference is smaller for lower TTL values, which could indicate that a shorter time frame also allows less space for different charging costs.

In addition, in figure 7.4.2, it can be seen that the lower the arrival battery capacity of the EV, the more negative the charging cost difference is, but the harder it is for the Q-learning agent to charge the vehicle to 100% SOC. The latter is in line with expectations since as the required energy increases it becomes more challenging for the agent to find the right moments to charge to decrease costs while fully charging the vehicle.

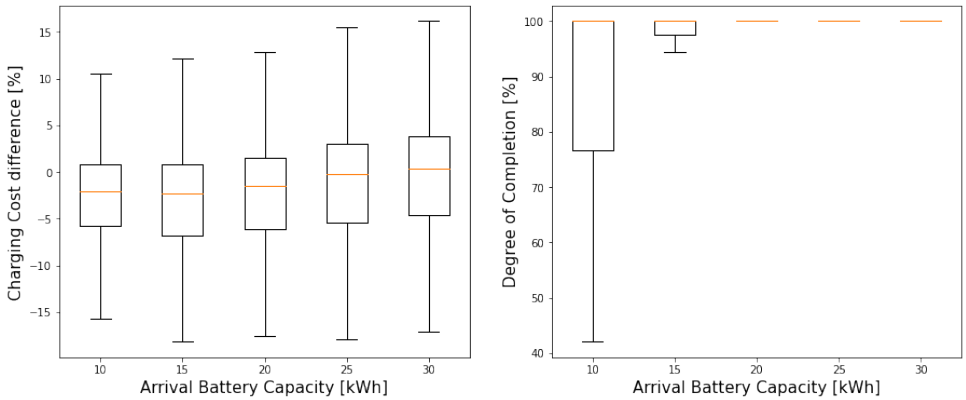

Figure 7.4.2: Box plots for the different battery arrival capacities versus degree of completion, discarding outliers

7.4.2 DDQN performance analysis

Figure 7.4.3 provides an overview of the relative charging cost difference percentage compared to the average rate charger and the degree of completion for different vehicle leaving times. There are a number of similarities worth mentioning when comparing the performance of the DDQN agent to that of the Q-learning agent.

To start with, the same trend can be found in the relationship between the charging cost difference and the vehicle leaving time, which strengthens the hypothesis that higher vehicle leaving times allow more flexibility and therefore higher positive and negative charging cost differences. Furthermore, also for the DDQN agent, it is the case that higher vehicle leaving times lead to a higher degree of completion. However, an important difference is that for a vehicle leaving time of 3 hours, the average degree of completion is significantly higher than for the Q-learning agent. This suggests that the DDQN agent is more successful in charging the vehicle to 100% for low TTL values.

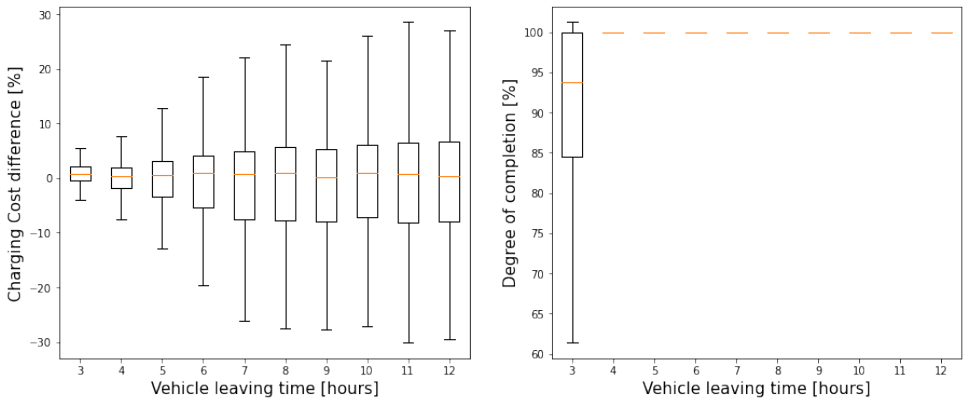

Figure 7.4.3: Box plots for the vehicle leaving time versus charging cost, discarding outliers

Figure 7.4.4 shows two plots of the relative charging cost difference percentage compared to the average rate charger and the degree of completion for different arrival battery capacities. In contrast to the limited number of arrival battery capacities, the DDQN agent is trained on any arrival battery capacity between 10 and 30 kWh.

Analysis of the left plot shows that the charging cost difference median is roughly the same for every arrival battery capacity with a total charging cost difference slightly higher or lower than 0%. When considering, the degree of completion for different arrival battery capacities, it can be seen that the median of the box plot is 100% for every arrival battery capacity. Both findings suggest that in contrast to the vehicle leaving time, the arrival battery capacity has little influence on the charging performance of the DDQN agent. This indicates that the DDQN model is less sensitive to different arrival battery capacities than the Q-learning agent. This could be a consequence of the more extensive state-space used for the DDQN model than for the Q-learning agent.

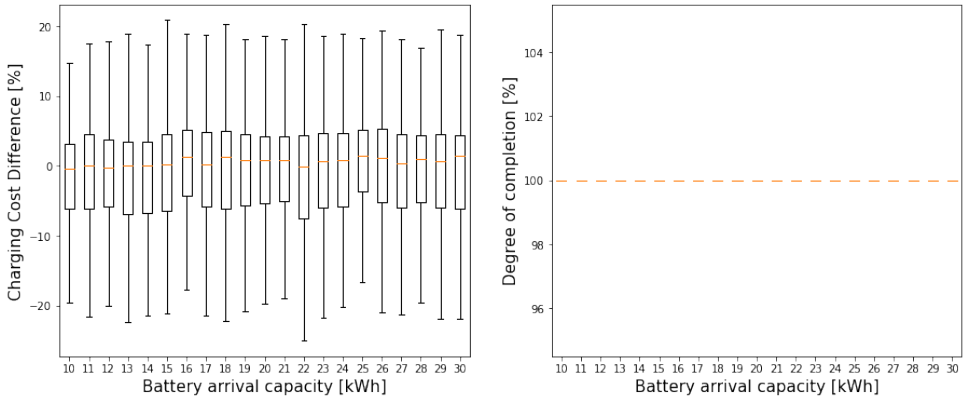

Figure 7.4.4: Box plots for the different battery arrival capacities versus degree of completion, discarding outliers

8 DISCUSSION

The purpose of this chapter is to discuss the meaning of the research results of the Q-learning and DDQN algorithms in more depth and to evaluate the feasibility of RL for smart charging. This chapter is organized as follows. First, in section 8.1 the performance of the RL agents is discussed in further detail and in section 8.2 the main advantages and disadvantages of reinforcement learning for smart charging are discussed. Finally, in section 8.3 the main shortcomings and limitations of this work are elaborated on.

8.1 THE PERFORMANCE OF THE RL AGENTS

The goal of this thesis was to assess the feasibility of reinforcement learning as a replacement for the MIP smart charging algorithm. The motivation for this research was that currently, MIP algorithms are effective in creating smart charging schedules, however, MIP slows down or becomes less accurate if its complexity increases. Therefore, in this work, the alternative of reinforcement learning for smart charging was researched for a simplified set-up to explore its feasibility. As for performance benchmarks, the charging performance and speed were evaluated in comparison to the original MIP smart charging algorithm and the average rate charging schedule.

The smart charging challenge in this work is multi-objective since it is the goal of the agents to both charge the vehicle as cheap as possible and to charge the vehicle to 100% SOC. These objectives can be considered inherently contradicting since it would be always the cheapest option to not charge the EV at all, however, this would not charge the vehicle to 100% SOC forcing the agent to make a trade-off. While in MIP this trade-off is made through a clearly defined objective function which the agent seeks to minimize, in reinforcement learning the careful process of reward engineering is used to steer the agent towards its objectives. The reward functions of the Q-learning and DDQN model in this work can be considered sparse and mid-dense limiting the interference between an immediate cost reward and the charging reward, restricting the need for a trade-off in each individual time step. Therefore, in theory, when considering the reward functions it should be possible for both objectives to be met. However, as the research results show, it can be hard to predict the effect of the reward function on the agent making it challenging to get the RL agents to output the desired performance.

8.1.1 Vehicle charging performance

To start with, the research results show that the contradicting multi-objective goal of charging the vehicle to 100% while saving cost was achieved relatively well with reinforcement learning. When considering the goal of charging the vehicle to 100%, the research results show that in two out of the three case studies prior to generalization the RL agents are successful in charging the vehicle to 100% and that after generalization the RL agents are successful in all three cases. This suggests that generalization can have a positive effect on the charging performance of the agents, with DDQN in particular, as it improves the chance of the agents reaching a full battery due to experience with a wide range of different cases.

In addition, the multi-episode evaluation results show that the average final SOC reached by the RL agent is higher than 0.9 for both RL agents, with the agents struggling mostly with lower vehicle leaving times. This trend suggests that shorter time frames are more challenging for the RL agents, bringing down the average final SOC. A possible explanation for this could be that as the vehicle leaving time gets closer to 3 hours, the chance of only having relatively high electricity prices until the vehicle intends to leave increases. In addition, it could be that because the algorithms have been trained on many different cases, the agents learn the trend that generally it is not a good idea to charge when electricity prices are high. As a result, the EV is not always charged to 100% for lower vehicle

leaving times, since lower vehicle times are often accompanied by relatively little space to find cheaper charging moments.

8.1.2 Cost-saving performance

The results of the cost-saving performance of the RL agent demonstrate four things. Firstly, the results prior to generalization provide evidence that RL can be used to find cheaper charging moments for individual smart charging case studies. However, as mentioned before, the cost-saving of the RL algorithms relative to the average rate charging highly depends on the cost pattern itself. Therefore, the highest charging cost difference relative to AVR is expected for cases in which the RL agent charges at the end of an episode that has have high electricity prices at the start and the lowest electricity prices at the end.

Secondly, the research results show that both prior and after generalization the Q-learning agent saves significantly more cost on the case studies than the DDQN agent for two out of three cases. Only for case study 3, the DDQN agent performs slightly better after generalization than the Q-learning agent. This could be caused by the fact that the training data differs slightly for Q-learning since in contrast to the DDQN model, the arrival battery capacity values are steps of 5 kWh between 10 and 30 kWh. Therefore, there are fewer different cases for the Q-learning algorithm to learn. Alternatively, the success of the Q-learning model can be explained by a better reward function since the sparse reward allows for less interaction with the cost reward than the mid-dense reward for DDQN.

Thirdly, the results of the cost-saving performance of the RL agents on multiple different cases suggest that on average after generalization the Q-learning agent and DDQN have a similar performance. Despite the fact that after generalization on average the Q-learning agent is cheaper than AVR in more cases, Q-learning has a lower cost difference compared to AVR than for the DDQN agent for all cases that reach the maximum possible SOC.

Finally, it is the case that in roughly half of the cases both the Q-learning and DDQN performance is worse than the average rate charging benchmark and is therefore definitely worse than the MIP output. This shows that while the individual RL agent can be effective in saving cost when trained and tested on individual case studies, for random cases the Q-learning and DDQN performance is unsatisfactory as the performance after generalization is insufficient to replace MIP as a tool for smart charging. This is strengthened by the average EV charging cost being only 0.001 Euro/kWh cheaper for Q-learning and DDQN not being cheaper at all. It can be concluded, therefore, that the proposed reward function in combination with the state space for both Q-learning and DDQN, for reducing the charging cost of the EV, in contrast to the SOC reward, does not motivate the agent sufficiently to find the cheapest charging moments for arbitrary cases.

8.1.3 Constraint violation and execution speed

Finally, when regarding the average performance results for the multi-episode evaluation, the percentage of cases in which the overcharging constraints are not violated is very high, indicating that the penalty in the reward function is effective in avoiding violation. This suggests that with even more training time, the percentage of cases in which the overcharging constraints are not violated could potentially be brought to the percentage of cases in which constraints are not violated to 100%. However, the constraint under consideration is relatively simple as it is decoupled from the value of the charging current output. The agent is either allowed to charge when the vehicle is not full or has not left yet and is not allowed to charge when the vehicle is full or has left. More complex constraints such as grid constraints would be directly related to the magnitude of the current output intervening with the cost reward. This would add an additional trade-off to be made by the agent based on the reward function and would require careful reward engineering.

Furthermore, we find that the execution speed is both fast for Q-learning and DDQN, as the execution speed is sub-second with the Q-learning speed being a lot faster than the DDQN execution speed. This difference in speed is due to the fact that the Q-learning algorithm uses a giant lookup table with Q-values which requires only a central processing unit (CPU) to search for the right state-action

combination, while the DDQN model uses a neural network requiring compute-intensive graphical processing unit (GPU) capacity. However, this does suggest that depending on the GPU computational power used for the DDQN output, the opportunity exists for a further decrease of the execution speed of the DDQN model. In addition, while both RL agents are guaranteed to have an execution speed in the same order for every case, the optimization speed of the mixed-integer programming optimization model differs depending on the type of case. Furthermore, the optimization time of the MIP is expected to explode as more and more nodes are added to a model for simulation, while the RL execution speeds have the potential to remain constant as long as the number of states, the time step, the time horizon and the and number of possible actions remain roughly the same.

8.2 THE FEASIBILITY OF RL FOR SMART CHARGING

While in the previous section the charging performance of the Q-learning and DDQN agent was discussed in further detail, this section is focused more on the discussion of the potential of RL for smart charging in relation to the current state-of-the-art MIP optimizer through an overview of RL's advantages and disadvantages.

8.2.1 Advantages of Q-learning and DDQN for smart charging

While RL utilization for smart charging is still at its initial stage, there are some advantages of RL for smart charging that can be recognized:

- **Computationally fast:** Reinforcement learning algorithms have the potential to be less computational heavy than traditional optimization methods, particularly for more complex cases than the ones tested in this work. This is due to the fact that the algorithms are pre-trained and therefore only have to output the solution.
- **Descriptive:** This research shows that RL is flexible since it is effective in processing the smart charging interactions including the charging process of the battery and constraints. Therefore, both the Q-learning and DDQN models allow for expansion of more complex smart charging dynamics including complex non-linear relationships, since in contrast to MIP it does not require extensive domain expertise to model the equations into the optimization model. For reinforcement learning the addition of complex non-linear relationships (e.g. the influence of temperature on battery charging and battery degradation) can be relatively easily added to the environment and trained on without a significant increase in the final execution speed after training.
- **Scalable:** The RL models proposed in this work have the potential to be scalable to larger problems while maintaining a relatively fast execution speed. When modeled as a decentralized algorithm, through extensive training the RL models can be trained to take other EVs and grid constraints into account. However, since the state-space, the number of steps, and number of actions can be kept the same, this is not expected to increase the execution speed.
- **No data labeling:** Another main advantage of using RL for smart charging is that in contrast to supervised learning methods, no large labeled data sets are required to train the ML agent. All data is "labeled" through the reward function due to experiences with the environment while training. This allows for model-free simulations mimicking situations for which training data is unavailable or hard to find such as for example data on the effect of rolling blackouts on EV smart charging. Furthermore, RL removes the presence of labeling bias, which is often the case for supervised learning problems since labeled data is not required for training.
- **No future information required:** Since RL allows for the design of the state-space, a state-space can be developed which does not contain information about the future. Even without the presence of information about the future such as a detailed prediction of future energy prices, RL algorithms can be taught to develop smart charging plans based on information about the past price trajectory. Therefore, while MIP smart charging algorithms require information about the future to reach optimality for smart charging plans, RL agents can in theory still learn ideal

charging behavior without future information with the right state space representation. This is because, after training, the RL agents have the ability due to past experience to make predictions about future actions based on the available input data.

8.2.2 Disadvantages of Q-learning and DDQN for smart charging

Now that the main advantages have been discussed, there are also some disadvantages worth mentioning for the replacement of mathematical optimization by reinforcement learning:

- **Large training time:** The main disadvantage of RL for smart charging is the large training time required for sufficient exploration to prevent the RL agent from ending up in local optima with a non-ideal charging output. As mentioned before, Q-learning has faster training times than DDQN due to the use of a Q-table. However, for DDQN the training time is a lot slower making it challenging for the DDQN agent to sufficiently explore the different options. The reason for this is that the target network is not frequently updated and therefore wrong for many episodes and that despite the use of PER recently experienced transitions are not always used for learning immediately. As a result, for DDQN in particular it is challenging to escape local optima also since rarely occurring newly found improvements can be treated by the neural network as outliers and are hard to get acknowledged by the neural network. While the use of prioritized experience replay improves the learning, still large training times are required to sufficiently explore the different charging options. In addition, these large training times significantly slow down the research process since even a minor change to the RL model would require complete re-training and delicate reconsideration of the hyperparameters involved.

- **Simulation environment dependence:** A second disadvantage is that the performance of trained RL models is highly dependent on the simulation environment and is therefore likely to perform less well in real-life environments. For example, the real charging dynamics of an EV battery are likely to be different than the simulation and therefore can confuse the agent resulting in wrong charging commands.

- **Complex reward engineering:** Since every single piece of idealized behavior related to the model objectives has to be reflected in the reward function, it becomes increasingly hard to keep track of the intermediate interactions between multiple parts of the reward function as the complexity of the model increases. Reward shaping, in particular, requires careful consideration, since despite it being used to make learning easier, it increases the complexity of multi-objective optimization in reinforcement learning as the relationships between different components of the reward function become non-linear and hard to follow. Ultimately, as more complex models are required for smart charging this means that it will become more challenging to reach optimality since instructing the agent what to do through the reward function becomes increasingly complex.

- **Simulation environment dependence:** Since many different hyperparameters are involved in the training process and the exploration of the reinforcement learning agent contains a random component it can be challenging especially for deep reinforcement learning to reproduce research results. An extensive list of hyperparameters, the exact code, and computational workload would be necessary to reproduce the same results limiting its applicability for research in academia preventing future deployment.

- **Safety constraints:** A final main disadvantage worth mentioning is that both Q-learning and DDQN have problems with safety if they were to be implemented in real-world applications. In contrast to mathematical optimization, because all constraints are inside the reward function it can not be guaranteed that constraints will be complied to. Depending on the type of constraint the RL output could be checked before actually outputting the current value, however, this is not ideal.

8.3 SHORTCOMINGS AND LIMITATIONS

With regard to shortcomings in the research, there are a few limitations to this approach. The main limitation of this research is the limited training time. Longer training times could potentially lead to a better charging performance after generalization for both algorithms. However, because of a lack of time, the number of episodes for the Q-learning and DDQN agents was restricted to 45M and 180K episodes.

In addition, for this work the decision was also made to create a simplified smart charging setup. A more complex model focused on the overall electricity cost involved with the cost for electricity to supply the load and PV Power would more closely reflect the state-of-the-art OSCD Mixed-Integer Programming optimization model. Therefore, unfortunately, the current results can not tell us anything about the degree to which the Q-learning and DDQN algorithm are able to take grid constraints and up and down ramping into consideration.

Furthermore, a final apparent limitation in this work is that the influence of multiple vehicles attached to the same node is not taken into account. This would have provided more in-depth insight into the feasibility of the RL models as a replacement for the OSCD MIP optimization model.

9 CONCLUSIONS AND RECOMMENDATIONS

This work aimed to assess the feasibility of reinforcement learning as a replacement for mixed-integer programming for smart charging of electric vehicles. This chapter is organised as follows. In section 9.1 the contributions of this work and the main findings are discussed. Next, in section 9.2 based on the main conclusions, some recommendations are provided for improvements of the RL smart charging models. Finally, in section 9.3 a number of recommendations for future work are mentioned.

9.1 CONCLUSIONS

In this thesis project research was done regarding the feasibility of a Q-learning and DDQN reinforcement learning smart charging model as a replacement for mixed-integer programming for smart charging of electric vehicles. The system structure of the charging cases under consideration in this work consists of a single node, a building loading, and a solar PV with a single EV attached to the charger. The objectives of the smart charging algorithms were to charge the electric vehicle to the highest possible SOC before the EV-owner intends to leave and to minimize the money spent in EUR/kWh on electricity for the charging of the EV.

To reach this research goal first a literature review was conducted on the application of machine learning algorithms in combination with mathematical optimization, after which an end-to-end reinforcement learning approach was chosen. Subsequently, an overview was provided of the Q-learning and Deep Q-network fundamentals and of corresponding state-of-the-art approaches for smart charging. Next, a Q-learning and Double Deep Q-network model with Prioritized Experience Replay were developed and evaluated. Finally, the charging performance, cost reduction, and speed of the algorithms were evaluated on multiple smart charging case studies as well as their generalization performance on random cases.

Through close analysis of the performance of the RL agents in comparison to the average rate charging and MIP benchmark it is possible to answer the aforementioned research questions stated at the beginning of this work.

1. *What is the charging performance and cost reduction potential of reinforcement learning algorithms compared to traditional LP/MILP optimization for smart charging of electric vehicles?*

   In conclusion, both the Q-learning and DDQN algorithm were able to reach the goal of charging the vehicle to 100% in the majority of the cases. This shows that the proposed sparse and mid-dense reward is very effective in forcing the agent to charge the EV to 100%. With respect to the cost reduction performance, this work shows that for the RL algorithms trained on the three case studies the algorithms are able to charge cheaper than AVR with the relative price difference ranging between -19.13% and 2.75% for Q-learning and between -10.84% and 7.85% for DDQN depending on the type of case. However, for all three case studies, the MIP smart charging model significantly outperforms the RL agents. Furthermore, after generalization, both algorithms do not significantly improve their performance on the case studies and are on average only slightly cheaper than the average rate charger. Therefore, the charging performance and cost reduction potential of the reinforcement learning algorithms perform worse than the traditional LP/MILP optimization models. In addition, the reinforcement learning models have a poor ability to generalize. It is important to highlight that the generalization ability is essential for smart charging as the agent should be able to perform well for different cases.

2. *What is the speed of the reinforcement learning smart Charging algorithms compared to traditional LP/MILP optimization for smart charging of electric vehicles?*

   Compared to the execution speed of generating a charging plan of traditional LP/MILP, the speed of the Q-learning algorithm is faster with an average speed on 10000 episodes of 0.006

seconds while the MIP speed is between 0.016 and 0.060 seconds on the case studies. The average execution speed for the DDQN agent is higher than the traditional LP/MILP with an average speed on 10000 episodes of 0.347 seconds. However, it must be noted that the execution speed for LP/MILP significantly increases as the number of equations, constraints, EVs, and nodes expands. Therefore, these results confirm the hypothesis that RL algorithms are fast and are expected to remain fast even after expansion of the algorithm as a result outperforming the speed of traditional LP/MILP.

3. *What are the main advantages and disadvantages of reinforcement learning algorithms with respect to flexibility, scalability and the safeguarding of constraints compared to traditional LP/MILP optimization for smart charging of electric vehicles?*

   To summarize, the Q-learning and DDQN algorithms satisfy the main requirements for creating charging plans for the smart charging of electric vehicles. As this research shows, the RL agents are well suited for multi-objective optimization which is often the norm for smart charging applications. In particular, the speed and flexibility of the RL algorithm in theory make room for scalable smart charging solutions. However, in practice compared to traditional LP/MILP optimization RL requires long training times and challenging "black box" reward and hyperparameter engineering. Furthermore, the Q-learning and DDQN algorithms can not guarantee the non-violation of constraints and are dependent on the training environment which makes them unsuitable for real-world applications with safety constraints. In addition, the state-action space of Q-learning and DDQN reinforcement learning is limited and the training time depends highly on the computational power available. Altogether, these disadvantages make the Q-learning and DDQN algorithms hard to reproduce and extremely time-consuming to iteratively improve for modeling and simulation purposes, which are both essential elements to make efficient progress in the development of smart charging algorithms.

9.2 RECOMMENDATIONS FOR IMPROVEMENT OF THE REINFORCEMENT LEARNING MODELS

As mentioned before in the limitation section, due to time constraints, the decision was made to focus on a simplified version of the more extensive OSCD smart charging algorithm. However, there are a number of alterations that can be made to the RL algorithms with the potential to improve its performance and to provide further insight into their feasibility. A list of recommendations for improvements is provided below:

- **Increase training time**
  The number of episodes on which the agent trains directly influences the exposure of the agent to different electricity patterns, vehicle leaving times, and battery arrival capacities throughout the year and therefore has the potential to improve the generalization ability of the algorithms. Therefore, significantly increasing the number of episodes for training could have a positive effect on the agent's degree of generalization and ultimately on the agent's performance. In particular, access to a more powerful GPU would allow faster training of the DDQN model, potentially leading to results outperforming the Q-learning model.

- **Alter state-space**
  The state space designed for both the Q-learning and DDQN model was based on both current research and trial-and-error. However, alternative state space representations are worth exploring. The state space could for example be expanded to include more information about the past and future price pattern, which has the potential to improve the agent's choice of actions.

- **Focus cost reduction on overall electricity cost**
  When regarding the reward functions used in this work, the cost reward was limited to the EV charging cost as this directly leads to the objective of reducing the charging cost reward for the EV. The generated PV power and building load were taken into consideration, however not directly used in the reward function. A suggestion for the improvement of the current model is to alter the

reward function to take the overall electricity cost into account for the reward function. This could allow "smarter" charging and further cost reduction by smartly taking PV power production into account.

- **Extend constraints to include grid constraints and regulation capacity constraints**
  The constraints focused on in this thesis were limited to the AC charging standards (IEC 61851) and the constraint that prevents charging when the vehicle has left or has reached 100% SOC. The OSCD smart charging algorithm also includes constraints related to the grid and up and down-regulation reserve capacity. An improvement to this work could include these constraints to better understand the ability of the RL algorithms to comply with these constraints.

- **Extend time window to 24 hours**
  In this model the time horizon was limited to 12 hours. An extension of this time horizon to 24 hours would allow for more flexibility for the agent to find cheaper charging moments and could therefore significantly reduce the overall charging costs. However, it must be noted that this also increases the time required to process a single episode as the number of actions taken in a single episode increases.

- **Expand vehicle TTL and battery arrival capacity values**
  For simplification purposes the vehicle leaving times were restricted to integer values between 3-12 hours and the battery arrival capacity to integer values between 10-30 kWh for DDQN and to values between 10-30 kWh in steps of 5 kWh for Q-learning. Evaluation of the charging results after training on more extensive data could contribute to a better understanding of the generalization ability of the RL agents.

9.3 RECOMMENDATIONS FOR FUTURE WORK

While a number of recommendations have been mentioned that have the potential to improve the current reinforcement learning models, a few suggestions can also be made for future work:

- **Test Alternative RL algorithms**
  The RL agents tested in this thesis were limited to Q-learning and DDQN. However, future research could expand the performance comparison of different RL algorithms by including other reinforcement learning algorithms such as Deterministic Policy Gradient, Asynchronous Advantage Actor-Critic Algorithm, and Soft Actor-Critic that can be used to reach the same objectives.

- **Optimisation augmented by ML approach**
  Future work could use ML algorithms to predict beforehand in which optimization cases the MIP optimizer is either slow or non-optimal and to use an ML model to speed up that specific case of optimization through smart configuration, branching, or through the introduction of the right cutting planes.

- **ML to improve Optimisation input**
  The approach of ML to improve optimization might prove to be an important area for future research. For example, an ML model could be used in future work to estimate non-linear relationships or to linearize non-linear functions such as the charging of the battery or more complex consumer preference cost functions. This could reduce the speed and complexity of a future more accurate version of the OSCD MIP smart charging algorithm.

Overall, the combination of the fields of machine learning and mathematical optimization, with reinforcement learning, in particular, is a promising and exciting field from which a hope new insights can be expected in the near future. Further research will point out how machine learning algorithms and mathematical optimization either separately or in combination can be used best in synthesis to leverage their independent advantages.

BIBLIOGRAPHY

[1] IEA, "Total final consumption (tfc) by sector, world 1990-2018," Accessed: Dec 21, 2022. [Online]. Available: https://www.iea.org/data-and-statistics/data-browser/?country=WORLD{&}fuel=Energyconsumption{&}indicator=TFCShareBySector

[2] P. Moriarty and D. Honnery, "Prospects for hydrogen as a transport fuel," *International Journal of Hydrogen Energy*, vol. 44, no. 31, pp. 16 029–16 037, Jun. 2019.

[3] A. Moreno-munoz, *Large Scale Grid Integration of Renewable Energy Sources*. Institution of Engineering and Technology, 2002.

[4] J. Lu and J. Hossain, *Vehicle-to-grid : linking electric vehicles to the smart grid*. Institution of Engineering and Technology, 2015.

[5] W. Tang and Y. J. Zhang, *Optimal charging control of electric vehicles in smart grids*. Springer, Accessed: Mar 3, 2022. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1331482

[6] A. J. Cheng, B. Tarroja, B. Shaffer, and S. Samuelsen, "Comparing the emissions benefits of centralized vs. decentralized electric vehicle smart charging approaches: A case study of the year 2030 California electric grid," *Journal of Power Sources*, vol. 401, pp. 175–185, Oct. 2018.

[7] C. P. Mediwaththe and D. B. Smith, "Game-Theoretic Electric Vehicle Charging Management Resilient to Non-Ideal User Behavior," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 11, pp. 3486–3495, Nov. 2018.

[8] Z. Xu, Z. Hu, Y. Song, W. Zhao, and Y. Zhang, "Coordination of PEVs charging across multiple aggregators," *Applied Energy*, vol. 136, pp. 582–589, Dec. 2014.

[9] M. Nour, S. M. Said, A. Ali, and C. Farkas, "Smart charging of electric vehicles according to electricity price," in *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, Feb. 2019, pp. 432–437.

[10] A. Ivanova, J. A. Fernandez, C. Crawford, and N. Djilali, "Coordinated charging of electric vehicles connected to a net-metered pv parking lot," in *IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, Sept. 2017.

[11] C. S. Ioakimidis, D. Thomas, P. Rycerski, and K. N. Genikomsakis, "Peak shaving and valley filling of power consumption profile in non-residential buildings using an electric vehicle parking lot," *Energy*, vol. 148, pp. 148–158, Apr. 2018.

[12] H. H. Eldeeb, S. Faddel, and O. A. Mohammed, "Multi-Objective Optimization Technique for the Operation of Grid tied PV Powered EV Charging Station," *Electric Power Systems Research*, vol. 164, pp. 201–211, Nov. 2018.

[13] Y. Kuang, Y. Chen, M. Hu, and D. Yang, "Influence analysis of driver behavior and building category on economic performance of electric vehicle to grid and building integration," *Applied Energy*, vol. 207, pp. 427–437, Dec. 2017.

[14] J. F. Franco, M. J. Rider, and R. Romero, "A Mixed-Integer Linear Programming Model for the Electric Vehicle Charging Coordination Problem in Unbalanced Electrical Distribution Systems," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2200–2210, Sept. 2015.

[15] R. Fachrizal, M. Shepero, D. van der Meer, J. Munkhammar, and J. Widén, "Smart charging of electric vehicles considering photovoltaic power production and electricity consumption: A review," *eTransportation*, vol. 4, May. 2020.

[16] J. Kotary, F. Fioretto, P. V. Hentenryck, and B. Wilder, "End-to-End Constrained Optimization Learning: A Survey." Mar. 2021. [Online]. Available: https://arxiv.org/abs/2103.16378

[17] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discrete Optimization*, vol. 19, pp. 79–102, Feb. 2016.

[18] D. Paper, *Hands-on Scikit-Learn for machine learning applications : data science fundamentals with Python*. Berkeley, CA: Apress, 2020.

[19] H. M. Abdullah, A. Gastli, and L. Ben-Brahim, "Reinforcement Learning Based EV Charging Management Systems–A Review," *IEEE Access*, vol. 9, pp. 41 506–4153.

[20] G. R. C. Mouli, M. Kefayati, and P. Bauer, "Integrated pv charging of ev fleet based on energy prices, v2g, and offer of reserves," *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 1313–1325, Mar. 2019.

[21] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, Apr. 2021.

[22] M. Fischetti and M. Fraccaro, "Machine learning meets mathematical optimization to predict the optimal production of offshore wind parks," *Computers & Operations Research*, vol. 106, pp. 289–297, Jun. 2019.

[23] K. L. López, C. Gagné, and M. Gardner, "Demand-Side Management Using Deep Learning for Smart Charging of Electric Vehicles," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 2683–2691, Feb. 2019.

[24] K. Valogianni, W. Ketter, and J. Collins, "Smart charging of electric vehicles using reinforcement learning," *AAAI Workshop - Technical Report*, pp. 41–48, Jan. 2013.

[25] S. Dimitrov and R. Lguensat, "Reinforcement learning based algorithm for the maximization of ev charging station revenue," in *2014 International Conference on Mathematics and Computers in Sciences and in Industry*, Feb. 2014, pp. 235–239.

[26] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[27] A. Nowak, S. Villar, A. Bandeira, and J. Bruna, "A Note on Learning Algorithms for Quadratic Assignment with Graph Neural Networks," Accessed: Mar 3, 2022 2017. [Online]. Available: https://arxiv.org/abs/1706.07450

[28] A. Lodi and G. Zarpellon, "On learning and branching: a survey," *TOP*, vol. 25, no. 2, pp. 207–236, Jun. 2017.

[29] P. Bonami, A. Lodi, and G. Zarpellon, "Learning a classification of mixed-integer quadratic programming problems," in *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, Jun. 2018, pp. 595–604.

[30] E. B. Khalil, P. L. Bodic, L. Song, G. Nemhauser, and B. Dilkina, "Learning to branch in mixed integer programming," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Feb. 2016, p. 724–731.

[31] V. Nair, S. Bartunov, F. Gimeno, I. Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang, R. Addanki, T. Hapuarachchi, T. Keck, J. Keeling, P. Kohli, S. I. Ktena, Y. Li, O. Vinyals, and Y. Zwols, "Solving mixed integer programs using neural networks," Accessed: Mar 3, 2022. [Online]. Available: https://arxiv.org/abs/2012.13349

[32] R. Baltean-Lugojan, P. Bonami, R. Misener, and A. Tramontani, "Selecting cutting planes for quadratic semidefinite outer-approximation via trained neural networks," Accessed: Mar 3, 2022. [Online]. Available: http://www.optimization-online.org/DB_FILE/2018/11/6943.pdf

[33] O. Frendo, J. Graf, N. Gaertner, and H. Stuckenschmidt, "Data-driven smart charging for heterogeneous electric vehicle fleets," *Energy and AI*, vol. 1, Aug. 2020.

[34] O. Frendo, N. Gaertner, and H. Stuckenschmidt, "Improving Smart Charging Prioritization by Predicting Electric Vehicle Departure Time," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–8, Apr. 2020.

[35] A. Lucas, R. Barranco, and N. Refa, "Ev idle time estimation on charging infrastructure, comparing supervised machine learning regressions," *Energies*, vol. 12, no. 2, Jan. 2019.

[36] D. Sun, Q. Ou, X. Yao, S. Gao, Z. Wang, W. Ma, and W. Li, "Integrated human-machine intelligence for EV charging prediction in 5G smart grid," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, Jul. 2020.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[38] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine Learning*, vol. 13, no. 1, pp. 103–130, Oct. 1993.

[39] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," Accessed: Mar 3, 2022. [Online]. Available: http://arxiv.org/abs/1509.06461

[40] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," Accessed: Mar 3, 2022. [Online]. Available: https://arxiv.org/abs/1511.05952

[41] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," Accessed: Mar 3, 2022. [Online]. Available: https://arxiv.org/abs/1707.01495

[42] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," Accessed: Jan 21, 2022. [Online]. Available: https://arxiv.org/abs/1705.05363

[43] M. A. Wiering and M. Van Otterlo, *Reinforcement learning*. Springer, 2012.

[44] S. Kim and H. Lim, "Reinforcement learning based energy management algorithm for smart energy buildings," *Energies*, vol. 11, Aug 2018.

[45] S. Najafi, M. Shafie-khah, P. Siano, W. Wei, and J. P. Catalão, "Reinforcement learning method for plug-in electric vehicle bidding," *IET Smart Grid*, vol. 2, no. 4, pp. 529–536, Dec. 2019.

[46] M. Dorokhova, Y. Martinson, C. Ballif, and N. Wyrsch, "Deep reinforcement learning control of electric vehicle charging in the presence of photovoltaic generation," *Applied Energy*, vol. 301, Nov. 2021.

[47] F. Tuchnitz, N. Ebell, J. Schlund, and M. Pruckner, "Development and Evaluation of a Smart Charging Strategy for an Electric Vehicle Fleet Based on Reinforcement Learning," *Applied Energy*, vol. 285, Mar. 2021.

[48] J. Lee, E. Lee, and J. Kim, "Electric vehicle charging and discharging algorithm based on reinforcement learning with data-driven approach in dynamic pricing scheme," *Energies*, vol. 13, no. 8, Apr. 2020.

[49] E. Mocanu, D. C. Mocanu, P. H. Nguyen, A. Liotta, M. E. Webber, M. Gibescu, and J. G. Slootweg, "On-line building energy optimization using deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 4, pp. 3698–3708, May. 2019.

[50] Z. Wan, H. Li, H. He, and D. Prokhorov, "Model-free real-time ev charging scheduling based on deep reinforcement learning," *IEEE Transactions on Smart Grid*, vol. 10, no. 5, pp. 5246–5257, Nov. 2019.

[51] G. R. C. Mouli, P. Bauer, and M. Zeman, "System design for a solar powered electric vehicle charging station for workplaces," *Applied Energy*, vol. 168, pp. 434–443, Apr. 2016.

[52] NEDU, "Verbuiksprofielen-profielen 2018," Accessed: Feb 28, 2022. [Online]. Available: https://www.nedu.nl

[53] OSCD, "Orchestrating smart charging in mass deployment." [Online]. Available: https://oscd.eu/