Master's Thesis 2022   30 ECTS
Faculty of Science and Technology

Deep reinforcement learning for vehicle-to-grid control of long-term parked EVs: A case study of Oslo airport Gardermoen

Filip Kristoff Ørn
Data Science

Preface

Dear reader, there are a couple of things about this thesis that you should be aware of. The first one is that all of the code used to test the algorithms can be found at the main gitlab page of this master [1]. The other thing that should be noted is that all the text in this thesis that is coloured in red, are clickable hyperlinks. The most important links to know about is the ones in the table of contents, and those in the header. The ones in the table of contents will send you to their respective sections, while the link in the header will send you back to the table of contents. The reason for this structure is so that you can easily find your way to the sections that you are interested in. Say you are in results and wonder how part of the methods were done. Then you can click the hyperlink in the header, go back to the table of contents, and from there navigate to the part of the methods section that interested you. I hope you find this feature as useful as I did, and without further ado here is the rest of the preface.

This thesis had a somewhat turbulent start. I was originally planning on setting up a communication program for data gathering from (and for performing vehicle-to-grid on) electric vehicles as my master project. Sadly the delivery, and setup, of the equipment needed to get this program running was delayed and so the topic of the thesis changed to what it is today, although the communication program was still continued as a separate smaller project (the code can be found at this gitlab page [2]). My current topic has always been something that I wanted to write about. Primarily because it let me work with reinforcement learning, which is something I have been wanting to get to know more about for quite some time. It has periodically been tough working with the master-project and the thesis. Particularly in the periods where nothing seemed to work and the fear of failure, and to disappoint the people i care for, was hanging like a dark cloud over me. It is still in these times of trial that we come out as stronger and better people and I am proud that I managed to see it through.

Still this project would not have been completed without the help of some very special people. Therefore I would like to express my gratitude to my master supervisor Heidi Samulesen Nygård. She provided me with much needed advice along the way and gave me the freedom and space that I needed to be able to write and complete my thesis. Additionally i would like to thank Thomas Martinsen for all the energy he has brought to our meetings and for giving me a good understanding about what work the project has been in need of. This gave me a feeling of purpose during my brief time in the project, that it would have been difficult for me to complete this master thesis without. I would also like to express how much i appreciate all the invaluable feedback and advice that Fadi Al Machot has given me, which helped my thesis get out of a hole that it at one point was stuck in. I am also very grateful for all the feedback Åshild Grøtan has given me on my thesis. Without her help the thesis would have been in a worse state than my room after this whole master-writing period. I would also like to thank Muhammad Tabish Parray for being a good conversation partner and for our many insightful exchanges of ideas and thoughts on how to solve our respective problems. Finally I would like to extend my gratitude to Nordpool for allowing me to use their data in my thesis.

It is somewhat odd be done after all the years of my master degree and look back at the years that have passed. I cannot help but to look back at it all and treasure the many fond

memories i have share with the many friends i have made at this university. Still i cannot help but to think that I do not know how i would have gotten through these five years of university had it not been for the support and comfort that my family have given me. It is difficult for me to find such rest and peace as in the chaotic days at home. I am very thankful for the many airplane tickets that you, Anne and Stein, have provided me with. It has granted me the privilege of travelling home often to be together with you and my siblings while living far away from home. I am particularly grateful for being allowed to see my siblings grow up to become the people and friends that I am very proud of. When i look back at it, it is these moments together with you that i cherish the most. That is why despite all the difficulties, and for many the sadness, that the Corona pandemic has brought with it, I am glad that it gave us the opportunity to be more together.

15th of May 2022, Ås
Filip Kristoff Ørn

Abstract

There is a growing demand for electricity that leads to costly expansions of the grid capacity having to be made. Additionally this increase in demand also leads to the introduction of unregulated energy resources that causes greater variations in the electricity prices. Solutions needs to be developed to reduce the stress on the energy grid and the variations in the prices. One such solution is to use the batteries in electric vehicles (EVs) as batteries for the grid, also known as vehicle-to-grid (V2G). This case-study reviewed the potential use of the batteries of the long-term parked EVs at the P10 parking garage at Oslo airport Gardermoen for such V2G.

To be able to realise the full potential of the EVs for V2G algorithms have to be developed that can optimize the charging and discharging of the EVs. This thesis looked at a type of reinforcement learning (RL) algorithms called double deep Q-learning (DDQL), Genetic algorithms (GA) and a simple algorithm as potential algorithms for optimizing these charging and discharging decisions. The DDQL was divided into two separate versions, one called mean-DDQL, that used the mean of the price data, and one called STD-DDQL that used the shape of the price data for optimization. The algorithms were tested for the Nissan Leaf EVs, that are a part of a very limited number of EVs that can work as grid batteries. The Nissan Leafs were optimized using the electricity price data from Nordpool for the years 2019, 2020 and 2021.

The tests showed that the GA was the best performing algorithm, followed by the mean-DDQL, the STD-DDQL and finally the simple algorithm. It was observed that the GA had long runtimes, but that this was not a problem for optimizing for the next 24 hours. Still for real-time optimization, or for optimization that needs to happen within seconds, then it was too slow and the DDQL algorithms were to be preferred.

It was further observed that if either of the DDQL algorithms were to be used then the mean-DDQL was a better choice than the STD-DDQL as it was more robust towards no-profit periods.

Using the optimal algorithm, GA, it was observed that under realistic conditions using the EVs for grid batteries would lead to an average earning of bellow 200 kr for the car owner for 2019 and 2020. For the year 2021 then the average profit was around 1500 NOK when using the GA for optimization. This showed that for a regular year then V2G used purely to earn a profit based on the daily price difference was low, and so the EVs should probably rather be optimized for power tariffs.

Contents

List of Figures

List of Tables

Acronyms

C-rate Charging rate of the battery, a metric for the charge rate relative to the battery capacity. v, vii, viii, 11, 12, 15, 33, 48–60, 63, 64, 68

DDQL Double deep Q-learner. iii, v, 21, 29–31, 33, 36, 38–40, 44, 45, 49–53, 58, 60–62, 64, 65, 68, 70

EV Electric Vehicle. iii–v, vii, viii, 1, 12, 13, 15, 26–28, 30, 31, 33–35, 37, 46, 48–60, 62–70, 79–87, 89, 91, 92

GA Genetic algorithm. iii, v, 4, 23, 29, 30, 33, 36, 41, 42, 44, 46–48, 53, 60–62, 64–66, 68–70

kW Kilowatt. vii, viii, 26, 49, 52, 53, 79–82

kWh Kilowatthours. vii, viii, 7, 15, 49, 50, 53, 54, 56, 57, 66, 79–87

mean-DDQL Double deep Q-learner with mean-based reward function. iii, vii, viii, 39–41, 44, 46–65, 68, 70, 84, 85

RL Reinforcement learning. iii, 4, 38

SoC State of Charge. 11–13, 15, 27, 28, 31, 33, 52, 53, 70, 90, 92

STD-DDQL Double deep Q-learner with standardization-based reward function. iii, viii, 39, 41, 44, 46–65, 68, 83–86

Introduction

Background

„The world is electric" is the slogan of the Norwegian transmission grid operator Statnett. With an expected global growth of 4% in 2022 [3] the worldwide electric power consumption this slogan is about to turn from dream into reality. This growing demand for electricity is due to the increase of energy-intensive industries, changes in service demand[3] and traditionally fossile-driven sectors turning electric.

The global transportation sector is one of these sectors that is turning electric and it saw from 2018 to 2019 a 40% year-on-year increase in electric vehicle (EV) sales[4]. This increase is in part a result of long-term policies aimed at converting the transportation sector into a zero-emission sector. Just in Norway the government has set itself a goal that all new cars that are bought will be electric vehicles (EVs) in 2025[5].

Such a net-zero transportation future as the Norwegian government envisions will require a major increase in the electric energy demand. According to Statnett this increase in demand will amount to a 12 TWh increase [6] from the current 139 TWh annual electric power consumption of Norway[7]. With this increase in power consumption there also follows an increased strain on the Norwegian power grid [6]. This strain is due to the EVs requiring much power when being charged, and so if many EVs are to charge at the same time then it might overload the grid [8]. The reason for this is that the grid has a limited capacity for power transfer and exceeding this limit for longer periods of time will lead to an overload that has the potential of shutting down parts of the grid. Such a blackout could happen if several homeowners return from work and all charge their EVs at a period of already high power consumption. Such a period is around 16/17 o'clock when most people come home from work in the same time period and start using several electrical appliances, such as the stove or the water heater, all at once. To combat these periodically large spikes in power consumption, and the potential risk of blackouts caused by the EVs, then costly expansions of the distribution grid will be required[9] which inevitably will result in a larger energy bill for everyday consumers.

In addition to putting additional strain on the grid, the increasing electric energy consumption will also further reduce the Norwegian power surplus, which is expected to shrink from 15 to 3 TWh by 2026 [10]. To combat this loss of surplus, 11TWh of wind farms was planned to have been constructed by 2021, and a total of 21 TWh of wind-farm energy is expected to be built by 2030 according to Norwegian watercourse and energy directorate (NVE)[11]. This expansion of the wind power capacity has not been without controversies, but additional electricity production has to be built and the increasing use of wind power is not unique in a European setting[12]. Several European states, among others Germany and Spain, are turning from their non-renewable power plants to unregulated renewable energy resources, such as wind and solar, to be able to meet their energy demands[13].

One consequence of this transition to unregulated renewables Europe, and the expansion of wind in Norway, is that one sees a power generation that is much more unpredictable, with larger fluctuations in the energy price[11]. The reason for this is the unregulated nature of renewable energy resources, where one cannot control when the wind blows or the sun shines. In essence this means that one has to adapt the consumption according to the production rather than to produce according to the consumption, as it has historically been[14]. This is not desirable, neither for the consumer nor the industry, as one cannot have people living in homes in the winter that are dark and unheated or factories that have to turn off their power during production because the wind is not blowing or the sun is not shining. Consequently there is a growing demand for solutions and technology that can store energy for later use and solutions to move consumption from periods with much pressure, to periods with little pressure on the grid [14].

According to NVE the EVs are a potential part of the solution to many of the problems listed above, and also to create a more stable renewable energy grid [15]. The EVs can potentially stabilize the grid by using their batteries to store energy when there is plenty of renewable energy production, and then pass it back when this production diminishes. Similarly they can be a flexible load that can stop charging/discharging from/to the grid when the grid capacity is about to be exceeded, so as to reduce the stress on the grid. So why are not the EV batteries already widely adopted within the energy grid? Two of the main obstacles that hinders their widespread use of EVs are **battery degradation** [15] and **societal factors**[16].

Battery degradation, as will be discussed in more detail in the Litium batteries-subsection, happens each time the EV charges/discharges. The more frequent the charge/discharge, the faster the degradation[17]. By NVEs calculations using an EV for vehicle to grid (V2G) once per day could halve the battery life of the EV battery by up to 50% [15]. How large this degradation actually is, and if V2G actually degrades the battery under all conditions[18], is an ongoing field of research . Nevertheless the potential of degradation makes consumers wary of using their car for V2G [16].

Amongst the **societal factors** Heuveln et al. [16] found that many consumers are reluctant to participate in V2G out of fear for the driving range ot their car, also known as range anxiety. It was also found that some users would participate in V2G ”Only at long-term parking lots, where it did not conflict with mobility”.

Motivation

This thesis is a part of a larger project where the battery degradation and societal factors can be remedied. The project is called Next to Gardermoen (NeX2G) [19] and is a collaborative research project between the Norwegian University of Life Sciences (NMBU), Statnett, OsloMet, Avinor, Elvia and Lyse. It is funded by the Norwegian research council[20] with a timeframe of 4 years from 2021 to 2024. NeX2Gs aim is to look at the possible use of the batteries in the EVs parked at the P10 parking garage at Oslo airport Gardermoen for V2G and as flexible loads to be used to help with balance the energy grid.

When using the EVs in the P10 parking garage then the charging and discharging will happen under controlled conditions aimed at reducing the potential degradation so that it does not affect the capacity in the battery [19]. The societal factors are also reduced as, due to flight information, the batteries can easily be charged back to the desired level [21].

To realise the potential of the EVs at Gardermoen for V2G, NeX2G has set up several sub-goals. Amongst these one is to find a way to optimize the use of the battery as an energy storage unit. It is this sub-goal that this thesis will take a closer look at.

Main objective:

The main objective of this thesis is to look at the potential of using a reinforcement learning algorithm to optimize the charging/discharging decisions for the EVs at Gardermoen. This will be done by first selecting one widely used reinforcement algorithm from the literature in addition to another suggested algorithm for testing the performance of the reinforcement algorithm. They will be tested on their ability to optimize V2G decisions for the relevant EVs based on the electricity price.

Subgoals:

The thesis of Hoff [22] reviewed the profitability of a V2G park at Gardermoen. The study did not take into consideration what the profits that optimally charging and discharging of the EVs can achieve. This thesis can hopefully provide an estimate of how much cost-savings that can be expected from V2G during a regular year, and if this profit can be ignored as Hoff suggested.

Algorithms in the literature

The group of algorithms called *Reinforcement learning* (RL) could potentially be used for V2G operations. The use of RL has already gained considerable interest in terms of its usage within the energy domain, particularly since 2018 [23]. Several applications have been found, such as for pricing the V2G operations for EVs for aggregators [24], optimizing the use of photovoltaic cells for EV charging [25] and EV charging scheduling [26], [27], [28]. Additionally several articles have reviewed the potential use of RL in V2G.

Dang et al. [29] and Shi et al. [30] proposed that a RL algorithm called ”Q-learning” could be used for optimizing the charging/discharging scheduling of EVs. The scheduling would be done by making the algorithm optimize the actions using a price signal. Using the algorithm as proposed has the benefit that the solution is guaranteed to converge to the optimal solution, as Q-learning has this convergence property [31].

Unfortunately there are some limitations to Q-learning as well. An important one is that Q-learning needs all the possible pairs of actions and states (the information about its surroundings) to be discrete. This leads to Q-learning becoming impossible to use if the space of possible state/action pairs grows [24]. For the case at Gardermoen this particular property makes Q-learning difficult to use as the electricity prices around Gardermoen is continuous, and so are all the possible state-action pairs.

A solution to overcome this issue is to combine Q-learning with a type of algorithms called *neural networks* that results in another type of algorithm called a deep Q-learner (DQL). DQL makes it possible to use a continuous state-action pair space. One property that is lost by using the neural networks is that convergence to an optimal solution is no longer guaranteed [32]. There are several articles such as [33], [34], [35] and [36] that have reviewed the use of DQL for V2G operation based upon electricity prices that are only known one hour in advance. All of these studies are from the past 3 years, and so this is very much an ongoing field of research.

A hole in this research, which this thesis will explore, is how the DQL will perform in the case where more prices than the next hour is known. In other words, how would the DQL perform in a case such as at Gardermoen where the prices are known at the most 36 hours in advance?

To further explore this hole another algorithm is needed as a standard that the performance of the DQL can be compared with. In their review of V2G technology Shariff et al. [37] proposed particle swarm optimization(PSO) and Genetic algorithm(GA) as algorithms to be used for V2G optimization. Similarly Tan et al. [38] pointed to PSO and GA as algorithms that aught to be used, but argued for RL as a possible candidate for V2G optimization. From these studies it was determined to use GA and a simple decision-making algorithm for testing the performance of the DQL-algorithm against.

Theory

The Norwegian power grid

The power grid in Norway is divided into three major parts, the transmission-grid (420-300kV), the regional-grid (132-33kV) and the distribution-grid (22kV -230V) [39], as can be seen in figure 1. The transmission grid is responsible for transporting the energy in between the regional grids or countries. The regional grid is the connection between the transmission grid and the distribution grid and also the connection point for power-intensive industries, while the distribution grid is responsible for sending out the energy to the end users[39].

As previously discussed the Norwegian power grid has a limited capacity for power transmission, and so the grid has to be built to be able to handle large spikes in energy consumption, also known as peak demand. The limit of the power transfer capacity is mainly a problem

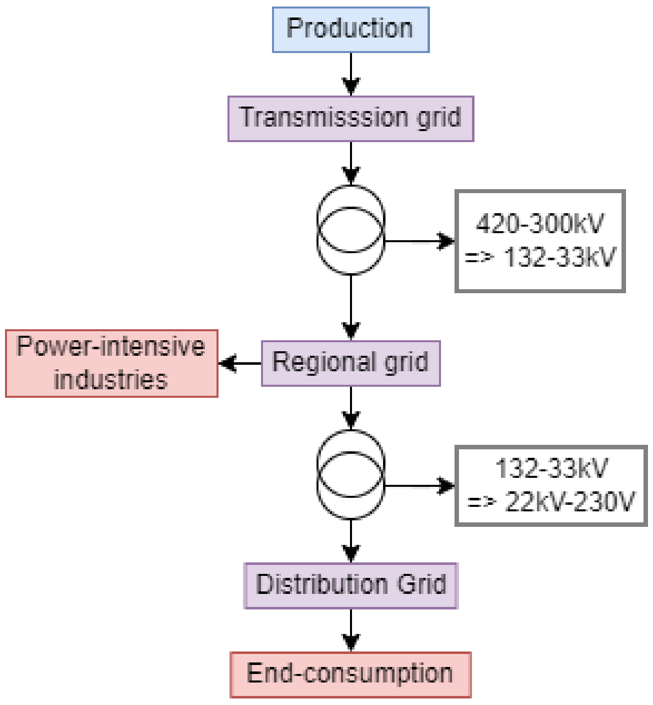


Figure 1: A schematic representation of the Norwegian power grid. The boxes in purple are the different grid levels. The red boxes are the electric-consumers and the arrows show which grid they draw their power from. The overlapping circles shows the transformer in between each grid and the white boxes besides them show the change in voltage.

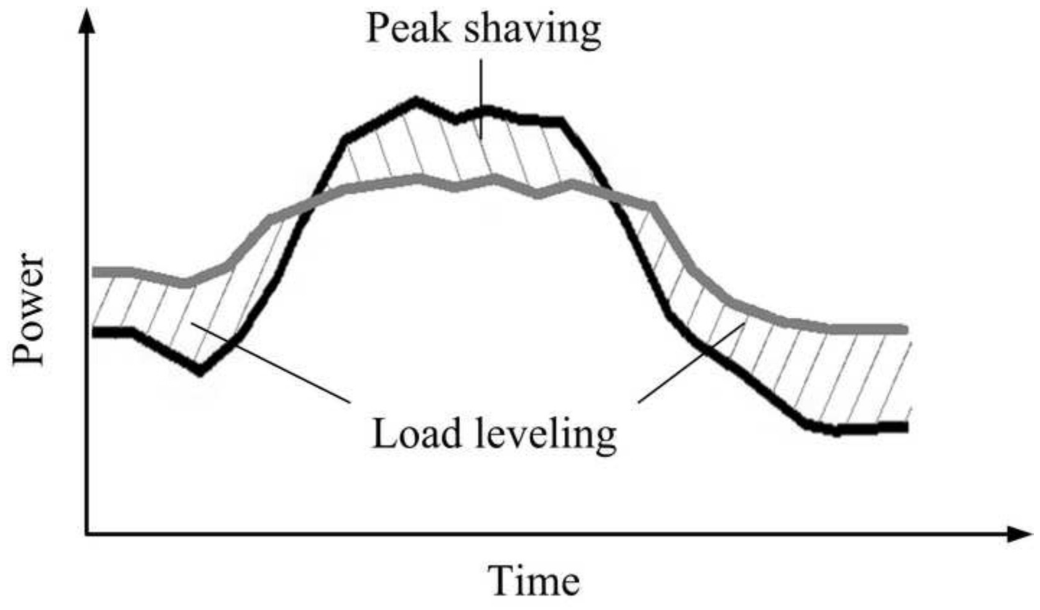


Figure 2: The goal of active power support. The image was fetched from [38] and is reused with permission.

in the distribution grid[8]. In a report from Energi Norge [8] it was found that expanding the capacity of the distribution grids does not always entail a large increase in cost. It is not particularly costly to build more capacity in grids that are already scheduled for replacement or upgrades, due to being old or worn. The major problem is upgrades in the capacity of power-lines with a long life-expectancy. Being able to reduce these peaks, so that power lines with long lifetime-expectancy will not have to be upgraded, can thus lead to large cost savings. As an example of the size of potential cost savings then it was reported that simply charging the EVs smarter, meaning to charge them in off-peak periods, could results in savings up to 11 billions. In other words there is a lot to gain by lowering or completely removing the peaks in consumption. This means that the objective of the algorithms reviewed in this thesis is to optimize with respect to ”Active Power Support”.

Active power support is the use of stored energy to flatten out the load curve of the grid when the energy demand is large. This is done by storing energy in the storage unit when there is little demand for energy and discharge all, or parts of this energy, when the demand is large. This results in the peaks in demand becoming flatter and the valleys being shallower. In essence one moves the power from one period to another [37], as can be seen in figure 2.

Profit maximization will be the metric to use when optimizing for active power support, as argued by Yong et al. [38]. This stems from the price of electricity mostly being representative for the demand, by optimizing for the price then one will optimize for active power support. To optimize for the price one wants to minimize the cost of charging the EVs, while the keeping the profits from selling the energy back to the grid is at a maximum, i.e. charge when there is little demand in the grid, and discharge back to the grid when there is a large demand.

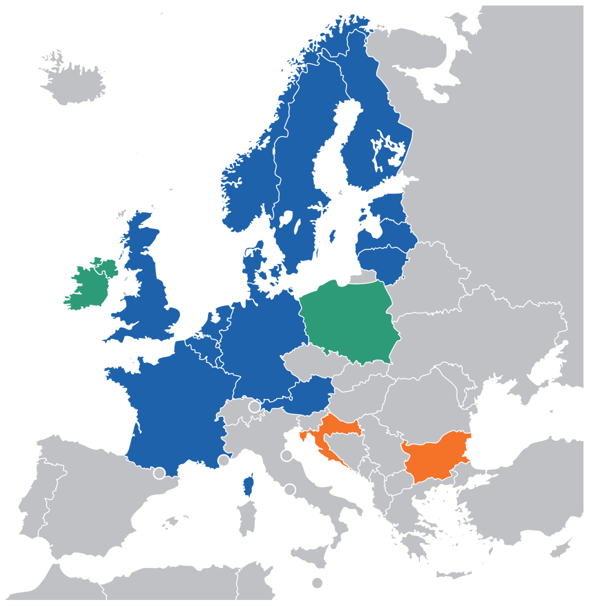


Figure 3: Nord Pool markets: The blue regions are current markets, the green are expansion markets, and the orange are serviced markets. Used under CC4, and can be found at [41]

The Norwegian electricity prices

The price data optimized for in this thesis is the elspot data fetched from NordPool for the Oslo-region. Nordpool is a day-ahead and intraday energy market in northern Europe where the price of electricity is set [40]. Countries that participates in Nordpool can be seen in figure 3. The prices per unit of energy for each hour within a day are called elspot prices, and are set each day at 12:00 o clock for the 24 hours of the next day.

This means that the price of electricity can only be known at a minimum of 12 hours, and at a maximum of 36 hours in advance for end-consumers with a spot-price agreement. The prices for the coming day are published at 13:00. Thus at 12 o clock only the next 12 hours are known, while at 13 o clock then one knows the next 36 hours.

One of the costs that comes in addition to the elspot price is the demand tariff. The demand tariff is an additional cost that is calculated by taking the largest electric consumption during a month of an energy consumer and multiply it by a set electric price (equation 1). The electricity price set for the demand tariff is typically more than 10 times larger than the average electricity price. As an example the average price for the 4th quarter of 2021, which was a period with a particularly high energy price, was 1,081 kr/kWh [42] while the demand tariff price for a business in the same period would have been 70-90kr/kWh, with the prices being larger in the winter than in the summer[43]. The goal of the demand tariff is to punish high peak demand by consumers and to give an incentive to spread the consumption over the whole day, rather than using all the electric equipment all at once. It is something that can be optimized for, but will not be used in the profit optimization of this thesis, although a possible way of optimizing for the tariff will be discussed in the discussion-section.

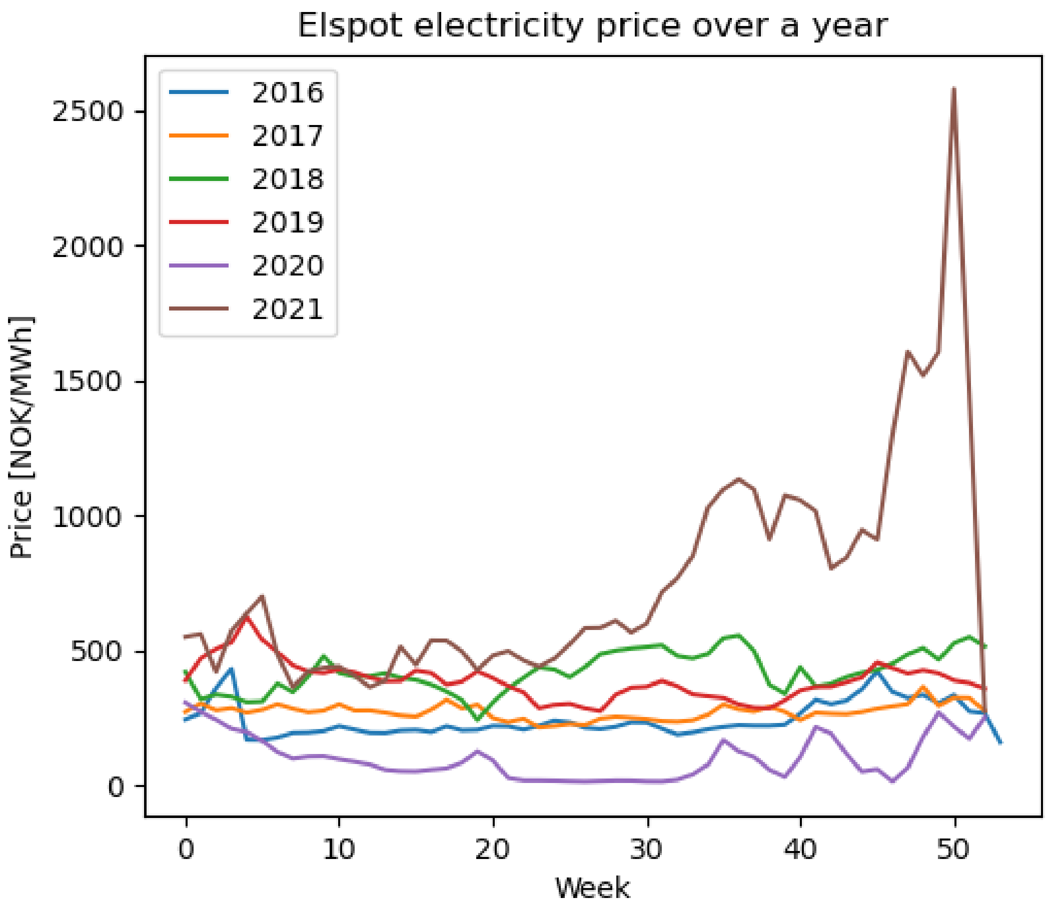


Figure 4: The variation in elspot prices from Nordpool over several years

$$Cost_{tariff} = Energy_{peak} * Price_{tariff} \tag{1}$$

The prices in Norway varies from year to year, as can be seen in figure 4, from season to season and from day to day. Regarding the seasonal trends then the energy price is typically cheaper in the summer and more expensive in the winter [44]. One reason for this seasonal difference can be attributed to the ice-melting in the spring. This melt-water amounts to to thirds of the water that fills up the water magazines and so high levels of water will reduce the price of electricity from hydro power plants [44]. As 88% of the Norwegian power production is hydro-power [45] the the reduction of price for hydro power will lead to low prices overall in Norway. The reason why the prices are high in the winter is due to a low inflow of water, the lack of melt-water and high consumption. The reason why the energy consumption is high in the winter is because 60% of the Norwegian electric consumption is tied to heating [46], this leads to a higher energy demand in the winter that increases the prices [47].

The price of electricity can also vary a lot from year to year, where the price is determined

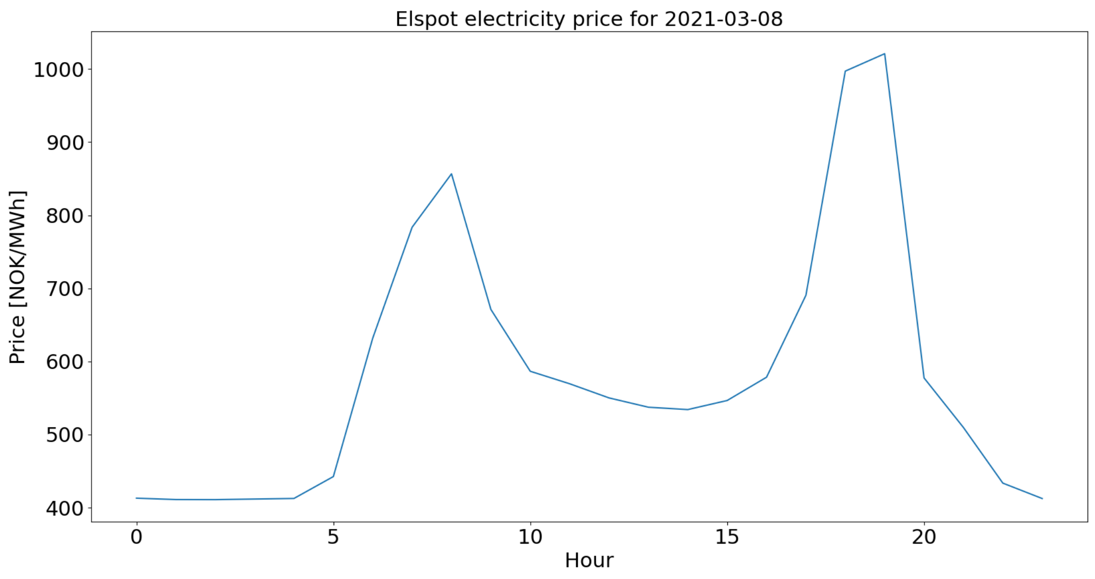


Figure 5: Elspot prices from Nordpool for the day 8th of March 2021, showing a good example of the regular pattern in the price data.

by factors such as the water level in the hydro power plants and the temperature [47]. If it is a particularly cold winter or it is a dry year, with little rainfall, then the price will be high because of increased consumption in the winter and low water levels in the magazines. An example of a dry year with a resulting increase in electric prices is the year 2018 that had a historical warm and dry summer that resulted in higher prices than in the rest of the year, as can be seen in figure 4. On the other hand then when it is a warm winter, and lot of rain throughout the year, then the prices will be lower.

From a day to day basis there is also generally a repetitive pattern to be seen. The energy price is typically higher during the morning around 7 - 12 o clock, i.e. when people wake up and turn on their electrical equipment, and in the evening when people get home (15-21 o clock) [48] as can be seen in figure 5. The height of these peaks vary from day to day and so does the shape of the price curve and which specific hour that has the largest value, although the price in the night is generally always lower than the price during the day.

Batteries

Batteries, also called ”galvanic cells”, are according to Britannica any ”class of devices that convert chemical energy directly into electrical energy” [49]. A battery is in its most general form built up by an anode (that is the part of the battery with the ability to release electrons during discharging), a cathode (the part that accepts electrons during discharging) and an electrolyte in between the anode and cathode (which ensures that the electrons go through the wire and hinders the build-up of net charge on both the cathode/anode as this would force the chemical reaction to halt)[49]. An illustration of a galvanic cell can be seen in figure 6. The chemical energy in the battery will not be converted before some conductive material connects the anode and the cathode. This will trigger the conversion reaction that will continue until either the chemical energy is depleted or the wire is removed [49].

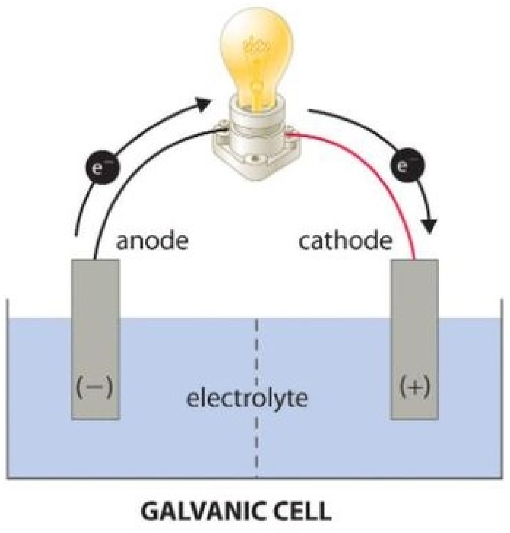


Figure 6: Shows the basic workings behind a Galvanic cell. Used under CC4, and can be found at [50].

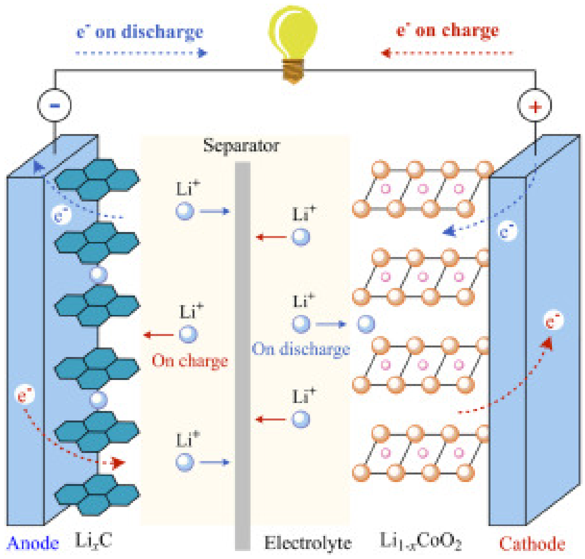


Figure 7: Shows the basic workings behind a Lithium ion battery similar to that found in the EVs. Used under CC4, and can be found at [51]

There are two main types of batteries, primary and secondary. The primary batteries are intended for one time use only, such as the ordinary AAA-batteries. Secondary batteries, like the Lithium-ion batteries used in cell-phones, are made to be rechargeable [52]. For this thesis only secondary batteries will be reviewed, specifically Lithium Nickel Cobalt Manganese Oxide (NMC) batteries, as these are the ones used in the relevant EVs. An image of the inner workings of such a lithium ion battery can be seen in figure 7.

Relevant metrics and formulas

The storage capacity of batteries in EVs is given in the units as kWh and is expressed as:

$$C = P * t \tag{2}$$

where C is the energy capacity, P the power which the battery will deliver and t the time over which the power can be delivered. To express how much of this power is currently available out of the total energy capacity the depth of charge (DoD) is often used, which is defined in equation 3 where C_{used} is the capacity used and C_{init} is the initial capacity.

$$DoD = \frac{C_{used}}{C_{init}} \tag{3}$$

In addition to DoD the state of charge (SoC) is also often used. State of charge is as described in equation 4:

$$SoC = (100\% - DoD) \tag{4}$$

C-rate is used to simplify comparisons across cars with different battery sizes and charging rates. The C-rate is as described in equation 5, where C_{rate} is the charge rate, $E_{charging}$ is the amount of energy that the car can charge in an hour and E_{total} is the total capacity of the battery. This metric is particularly important as a high C-rate will mean that a car can quickly charge or discharge its entire battery.

$$C_{rate} = \frac{E_{charging}}{E_{total}} \tag{5}$$

In addition to these formulas used for understanding the battery then the following statistical formulas will also be used: The the standard deviation formula and the standardization formula. A brief explanation of all these formulas can be found in Appendix II.

Battery efficiency

When considering the V2G operations, it is important to consider the efficiency of the battery, as this determines how much of the energy that one attempted to store that is actually available. A common metric of efficiency in batteries is the round-trip efficiency (RTE). The RTE is the percentage of charged energy that can later be discharged, as seen in equation 6.

$$RTE = \frac{E_{out}}{E_{in}} \tag{6}$$

The RTE is affected by many factors, among others current, temperature, internal resistance and SOC of the battery [53]. Additionally the efficiency varies from one battery type to the next, e.g. lead-acid batteries have an RTE of 60-70%, while lithium-ion batteries can have an RTE of up to 95%[54]. Yuliya Preger et al. [55] reported that the relevant NMC batteries had an RTE of 95% for a C-rate of 0.5 and a temperature of 25°C with the RTE decreasing with higher C-rates.

Battery degradation

Another important aspect of V2G is the battery degradation as this limits what sort of actions charging/discharging actions should can be taken. Battery degradation can be divided into calendar- and cycling-aging [56]. The calendar aging is aging that happens over time, independent of the usage, similar to a tire getting cracks around its edges over time. Cycling-aging is aging due to the usage of the battery, just as how driving will wear down the tires along the contact surface with the road. One result of both of these aging mechanisms is the loss of capacity in the battery. Out of the two only cycling aging will be considered in this thesis.

Cycling aging can be accelerated by several factors such as the frequency of cycling, the temperature, the DoD and the C-rates at which the battery is being charged/discharged [57]. It is also different between battery types as to how much cycling aging is being affected by the factors as suggested above. An example of this is a study by Peterson and Whitacre where they "observed little to no relationship between DoD and capacity fade" and also that "the dominant cell degradation method is not dependant upon [the] rate of discharge" for the A123 lithium iron phosphate batteries [58]. The iron phosphate batteries are in other words good for frequent charging discharging cycles, while for other batteries such as the Lithium Nickel Cobalt and Aluminium batteries (NCA) battery, it can be detrimental [59]. It is still an ongoing topic of research as to how much these factors influence the aging. For instance a study on similar NCA batteries by Uddin et al. [18] found that V2G operations under the right conditions can prolong the battery life of NCA batteries.

Taking a look at a study by Ecker et al. on NMC batteries, the decrease in the capacity of NMCs for different SoC-intervals was found, as shown in figure 8. From the graph one can see that for a choice of a SoC-interval close to 50% SoC would preserve the battery life of the NMC batteries the longest. Still it is common in the literature to operate with a larger SoC-interval span, such as the articles [60], [61], [62], [63] and [64] that uses the SoC range of 20% to 80% as a range where it is acceptable to do V2G operations within.

Relevant EVs and their battery

As of today there are only four EVs that support V2G, Mitsubishi Outlander PHEV Mitsubishi Eclipse Cross PHEV, the Nissan Leaf series and the Nissan e-NV200 [66]. Of all these only the Nissan leaf series will be reviewed. This is because the Nissan leaf amounts to 14.47% of the total EVs in Norway [67], which make them more numerous than the other V2G compatible

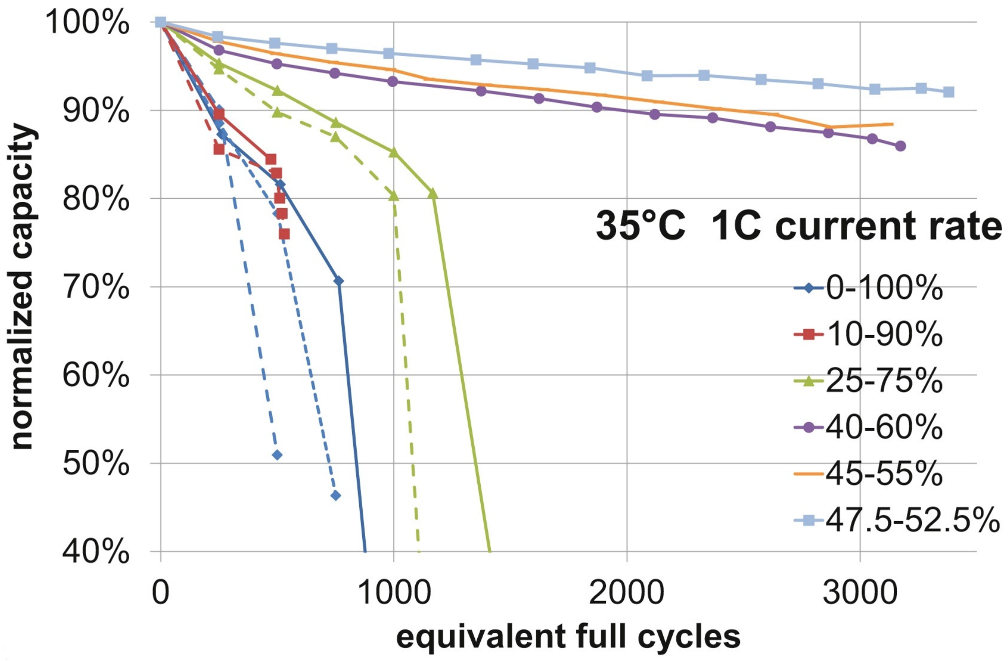
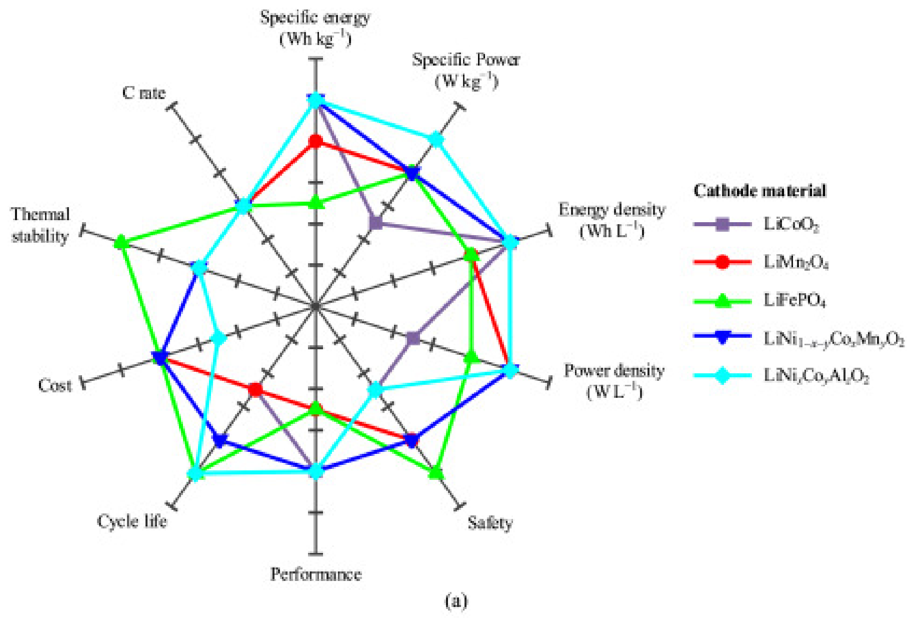


Figure 8: The decline in capacity compared with the cycles needed to get to that particular SoC. Each graph shows in what SoC range the battery was operating [65].

EVs.

In the Nissan leaf series there is a large span in battery capacity and some difference in the maximum charge acceptance(the maximum charge rate of the vehicle) as can be seen in table 1. It should be noted that amongst these models then only the 2013 or older models are eligible for V2G [68]. Additionally a point shall be made that not all the capacity listed in the table is available, and certain parts of the batteries are reserved to protect the batteries from charging too much, or completely depleting it. The available capacity of all the Nissan leaf models can be found at [69], but for the remaining part of this thesis it will be assumed that the available battery capacity is equal to that listed in the table.

There is a difference in the chemical composition between the first generation and second generation Nissan Leafs. The Nissan Leafs made after 2017 are considered second generation and those before are considered first generation. It is only the second generation Leafs, that uses NMC-batteries [71], while the first generation used a $LiMn_2O_4 + LiNiO_2$ chemical composition for their batteries[71]. As can be seen from figure 9, then this change in battery composition led to an increased performance for the EV batteries, longer cycling life and generally a higher energy density for the batteries. In addition to the difference between the batteries between the generations of the Leafs in terms of their chemical composition there is also a structural difference between them. Batteries from the second generation has a layered structure, while that of the first generation has a spinel structure [72]. This difference between the generations of batteries in terms of chemical composition and structure will not be considered, and all the batteries will be assumed to have the same structure and be NMC-batteries.



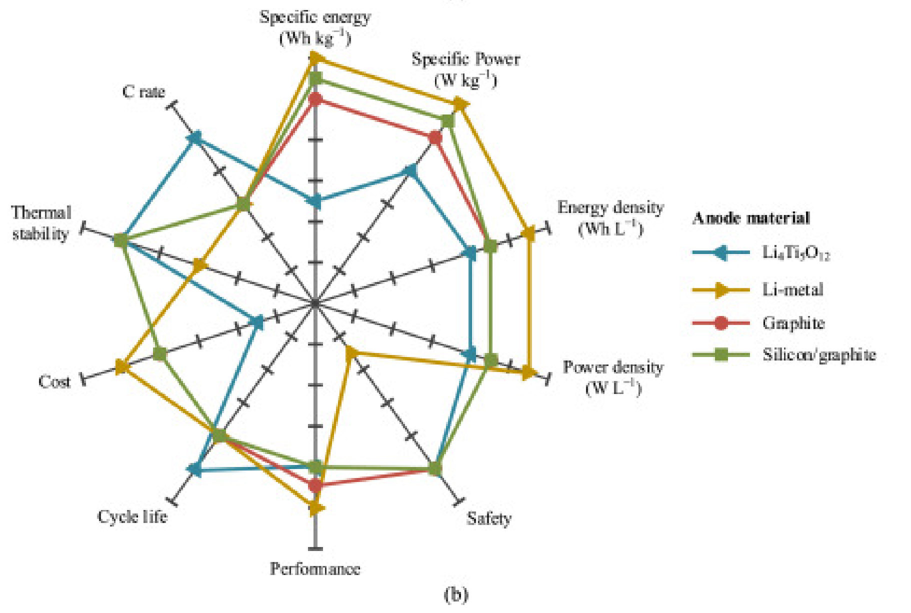


Figure 9: Overview over the different characteristics of the different cathode and anode materials in batteries and their properties. (a) Cathode material. (b) Anode material. Used under CC4, and can be found at [51]

Nissan model	Battery size [kWh]	Maximum power acceptance[kWh]	Charging time [h]	C-rate
Leaf (2011-2012)	24	3.3	7.3	0.14
Leaf S (2013-2016)	24	3.3	7.3	0.14
Leaf S (2013-2016)	24	6.6	3.6	0.28
Leaf (2017)	30	3.3	9.1	0.11
Leaf SL/SV (2016)	30	6.6	4.5	0.22
Leaf S (2018)	40	3.3	12.1	0.08
Leaf (2018)	40	6.6	6.1	0.17
Leaf e+ (2019)	62	6.6	9.4	0.11

Table 1: All the Nissan leaf models and some key properties about each car. The values for all cars but the 2019 version was found at [70] and the e+ was found at [69].

Charging power

EVs will not necessarily charge with a constant power for all SoCs, as suggested by table 1, but will rather vary with the current SoC [73]. In addition to this there are differences between the different car models and the cars in terms of at which SoC-intervals that they will have a stable power consumption when charging. According to [74] a test done on a Renault Zoe showed that it had a stable power consumption while charging up to 70% under fast DC charging. Charging above 70% would lead to the charging power starting to decrease. Wang et al. [73] had a somewhat different result when testing with a Nissan Leaf, where the power charging curve started to decline already from a SoC of 20% and onwards. In a test done on charging rate for different Nissan leaf models [75] then for the 24kWh version of the Leaf the charging curve was similar to that of [73], while for the 30kWh version the charging curve was stable up till 80%. For the 40 kWh model then the charging curve was only stable up to 60%. As there is a large variation as to which SoC-intervals that has a stable charging power it will be assumed that the charging rate will be equal for all SoC levels. It should, as already shown, be noted that in a real life situation then this assumption will not hold.

Machine learning

Machine learning (ML) is a subfield of artificial intelligence that studies algorithms that has the ability to improve after being provided data about the problem that it is going to solve. It can be divided into four major categories: **supervised**, **unsupervised**, **Semisupervised** and **reinforcement learning** [76] (figure 10).

Supervised learning is learning where one wants to find the relationship between a dataset and some labels. This is done so that future/new data can be labeled automatically. To do this one starts with providing labeled data to an algorithm and then try to make it learn the pattern that lead to this label[76]. An everyday example to illustrate how supervised learning works is to look at how children learn what a dog is. Initially the child will ask what animal it

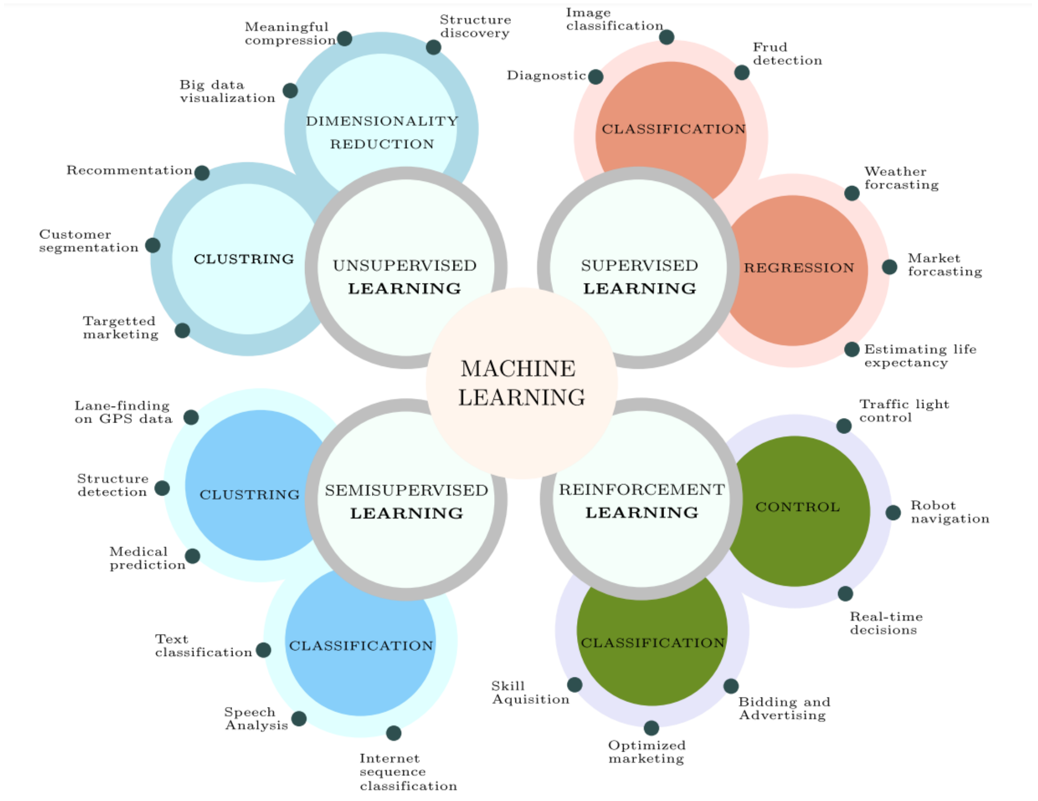


Figure 10: This figures gives a brief overview of the three major branches of machine learning. The image was fetched from [77] and is reused under CC4.

is looking at, then it will be told that it is a dog (the label) and the child will try to remember what it is about this animal that makes it a dog. Later the child might recall wrongly that the dog is a horse, whereupon it will be corrected. It will then try to figure out what it was about its logic that was wrong and try to form a new concept for what that makes a dog a dog. This will result in the child becoming better and better at finding out which animals that are dogs and which ones are not.

Unsupervised learning is different from supervised learning in that one only has data, but no labels, and tries to categorise data based on some predefined rule. In other words one tries to see if there are patterns in the data, without knowing if they actually exist[76]. This can be compared to poem-analysis where there are several answers as to what might be the message or theme of the poem. Uncovering the meaning might be helpful in understanding what the author could mean and what others probably would think when they read it, although there is possibly no pattern there at all.

Semisupervised learning is a combination of supervised and unsupervised learning that uses both labelled and unlabelled data for training [78]. In essence what the semisupervised algorithms do is to train an algorithm on already labelled data. This is then used to predict unlabelled data, and thus give them a label. This is useful as one can then let the semisupervised learning algorithms label data that would otherwise be too time-consuming or costly to label[79].

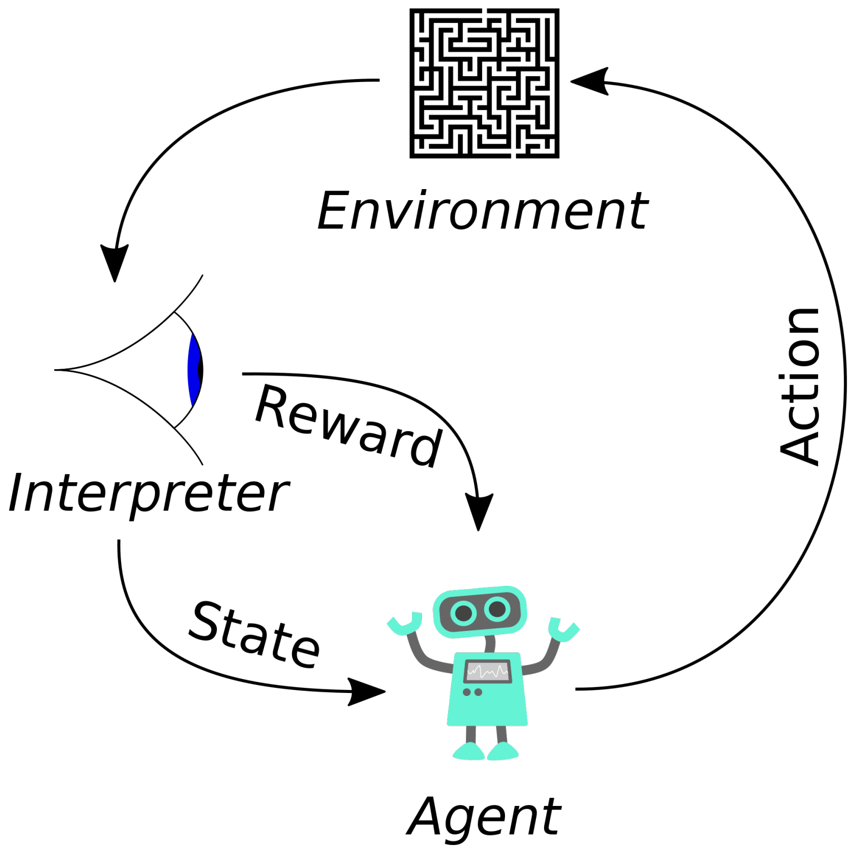


Figure 11: This figures shows the basic mechanic behind a reinforcement learning algorithm. The image was fetched from [80] and is reused under CC0.

Reinforcement learning

Sutton and Barto defined reinforcement learning as an algorithm that maps ”situations to actions so as to maximize a numerical reward signal”[81]. In simpler terms this means that reinforcement learning is any algorithm that makes a decision based upon whether that particular decision brings in long-/short-term rewards. An intuitive example of use-cases for reinforcement learning is the game Chess. In the game one obviously wants to receive the reward, i.e. win the game, but it is not clear how to do so. There could be several steps that are equally good and their value all depend upon the strategy. Still, one player eventually wins the game, so it is obvious that some actions are better than others. So how does one learn which actions that brought one closer to victory or loss? Trying to solve this problem is the goal of reinforcement learning. It is also this capability of learning a strategy to solve a problem that should in theory make reinforcement learning a good candidate algorithm for handling charging/discharging decisions of vehicles. This is because it could learn a good charging/discharging strategy based on its accumulated cost/income rather than a victory, as in chess.

A reinforcement learning algorithm can be generalized as an agent that works and lives in an environment that it can interact with, as shown in figure 11. From this environment it will receive a state, that is information about its environment. The agent then decides to perform an action based on this state [76]. To make this decision it will use a certain policy, which is a rule that maps the current state to the next action that it is to take. The policy will normally use some sort of value function to try to estimate what future rewards the agent can expect from taking a certain action in that particular state. With some combination of policy and value function the agent will choose an action. This will lead to the environment returning a new state and a reward for that particular action. The agent will then continue to repeat this process until it is forced to stop or it reaches a given reward-threshold. The rewards accumulated while the agents did its actions will then be used to train the agent to make better decisions the next time. One popular reinforcement learning algorithm is the Q-learning algorithm.

Q-learning

In statistics the Q-function is a function used to describe the probability of getting a value larger than the standard deviation. Similarly in reinforcement learning the Q-function is a function that estimates the long term reward of a given state-action (S,A)-pair [82] as shown in equation 7:

$$Q : S \times A \to \mathbb{R} \tag{7}$$

Generally one does not know beforehand which choices that are the best and what the optimal Q-values are. One way of finding these values is the Q-learning algorithm. The Q-values are estimated by performing actions and seeing how much reward those actions produce [24]. These accumulated rewards are used to update the Q-table, which is a table that handles the mapping from each state-action pair to their corresponding Q-value. The update of the Q-values in the Q-table is done by using the algorithm 8

$$Q(s_t, a_t) \leftarrow (1 - \alpha) \cdot Q + \alpha \cdot \delta \tag{8}$$

where $Q(s_t, a_t)$ is the Q-value of the action a_t given state s_t. α is the learning rate, i.e. how much each update will affect the previous value, Q is the current Q-value and δ is the Bellman equation as expressed in equation 9:

$$\delta(s_t, a_t) = r_t + \gamma maxQ(s_{t+1}, a) \tag{9}$$

Where r_t is the reward for doing that particular action a_t, γ is a weight to decide how much possible future rewards will influence the total reward of taking action a_t and $maxQ(s_{t+1}, a)$ is the maximum Q-value given the next step. As can be seen from the formula, then the earlier steps are tied to the later steps by the $maxQ(s_{t+1}, a)$-part of the equation. This ensures that the reward that is returned in the end will influence the earlier Q-values and guarantees the convergence of the Q-learning algorithm to the optimal Q-values given enough time [76].

When the Q-learner decides on which action to take then it does a combination of something called exploration and exploitation. In short exploration is to try out a random action to see which rewards it produce, while exploitation is to choose the action with the largest q-value, i.e. choose the action that so far is know to bring the greatest reward. A good combination of the two is important, as a pure exploration phase will make the algorithm try out needlessly many possible actions, while a pure exploitation phase will make the algorithm decide on a particular set of actions without having tried out other alternatives.

An issue with using Q-learning, as mentioned earlier, is that it is using a table to map all combinations of states and actions to their respective Q-values. This is unfeasible when the possible combination of actions and states are large or continuous [24]. To solve this issue one started to use deep neural layers to approximate the Q-values, and this algorithm is called Deep Q-learning (DQL). This brings in a trade off, as the convergence to the optimal Q-values are no longer guaranteed when using a deep neural network for estimating the Q-values and so choosing good parameters becomes just the more important. Before DQL is covered it is necessary to take a look at what neural networks are.

Neural networks

Neural networks in ML are algorithms that uses a set of one or several layers of something called neurons to map a set of input values to a set of output values. An example of such a neural unit is shown in figure 12. These neurons have been inspired by the neurons in the brain and works in a somewhat similar way, in that they both take in inputs, weigh them and then decides what signals to pass on. The ML-neurons takes in a set of inputs, multiply them by a set of weights and sum all the input/weight products. This sum is then passed on to an activation function that is used to transform the sum and is there to help the algorithm capture non-linear patterns [76].

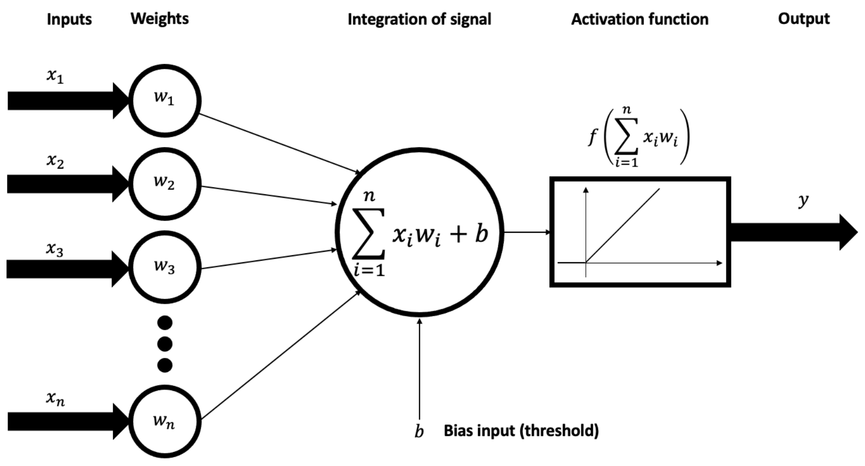


Figure 12: This figures shows the basic mechanic behind a neuron in a neural network. The image was fetched from [83] and is reused under CC4.

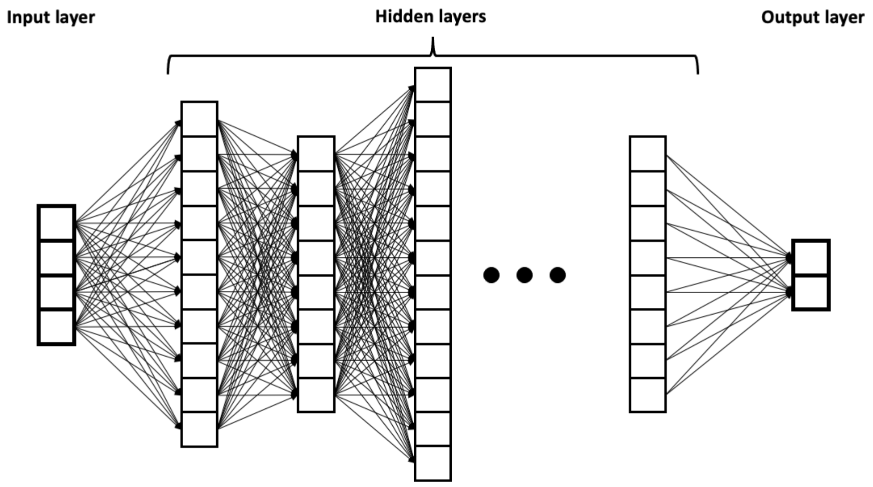


Figure 13: This figures shows the basic mechanic behind a neural network. The image was fetched from [84] and is reused under CC4.

This transformed sum is then passed on to another neuron or outputted out of the algorithm. A chain of such neurons passing a signal on is called a neural network, an example of such a network can be seen in figure 13. These neural networks consists of one output layer and several hidden layers. The hidden layers are all the layers in between the input and output layers. Having these hidden layers helps in finding structures in the data. To illustrate this suppose that one wants to predict if an image shows a dog or a cat, and one passes this image to a neural network. The neural network might then in its first layer find lines, in the second layer corners and edges, in the next one it might find shapes and in the last layer it will determine whether it sees a dog or a cat. In other words each layers finds structures in the input data and the above layers can put together these structures to find even more complex structures.

The neural network is trained by adjusting the weights of each individual neuron in the network [76]. To update the weights an updating algorithm is used. It works by first providing the algorithm with some input data, then it proceeds to pick up the output values from the neural network of the algorithm and then compares the output values with the true labeled data that it has been provided using a loss function. The loss function is used to evaluate how wrong/right the output values of the neural network was compared to the true values. The update function then uses the result from the loss function to see how much it must alter the weights.

DQL

Deep Q-Learning is similar to Q-learning in that it uses the bellman equation, albeit with a slight modification, as can be seen in equation 10.

$$\delta(s_t, a_t) = r_t + \gamma max Q^*(s_{t+1}, a) \tag{10}$$

Where one of the main differences between equation 9 and 10, beyond equation 10 having its Q-values calculated by a neural network, is that it uses a target network denoted as Q^*. This target network is used to predict the Q-values of the next state. The reason why it is needed is to stabilize the training of the neural network, as it can become unstable due to frequent weight updates. The target network is updated under some set of rules, one possible rule is to copy the weights from the main network over to the target network every n iteration. The rule that will be used in this thesis is the rule given in equation 11:

$$w'_{target} = w_{target}\tau + (1-\tau)w_{main} \tag{11}$$

where $w'_{target}$is the new target weights, w_{target} is the current target weights, w_{main} is the wights of the main network and τ is a coefficient to determine how quickly the target weights will be updated.

DQL uses, similarly to Q-learning, a combination of exploration and exploitation to improve its ability to predict Q-values. During its exploration phase it will gather samples and store them in a replay buffer to be used for training later:

$$S = (s_t, a_t, r_t, s_{t+1}) \tag{12}$$

here S is the sample in the replay buffer, s_t is the state, a_t is the action, r_t is the reward at time t and s_{t+1} is the reward at the timestep after t. The samples in the replay buffer is then used to train the main network using mean squared error (MSE) as a loss function:

$$L = \sum_{t=0} (\delta(s_t, a_t) - Q(s_t, a_t))^2 \tag{13}$$

where $Q(s_t, a_t)$ is the value it actually estimated and L is the loss. This loss function will be used to update the neural network so that it can better reproduce the the Q_{value}.

Double DQL

Another that is done to change to the DQL algorithm is to add an additional layer to the target network to make the learning more stable [85] so that it becomes something called a double DQL (DDQL). This change makes the bellman equation look like this:

$$\delta'(s_t, a_t) = \gamma max Q^*(s_{t+1}, argmax_a Q(s_{t+1}, a)) \tag{14}$$

$$\delta(s_t, a_t) = r_t + \delta' \tag{15}$$

Where the main difference from 10 is that instead of providing the target network with the action that was taken, it will instead be provided the action that comes from the action with the largest q-value that the main network would have predicted for the next time step.

Still even with the increased stability then the DQL and DDQL is still prone to something called catastrophic forgetting. Catastrophic forgetting is the abrupt loss of previously learnt knowledge, and it is a challenge for the performance of neural networks [86]. As new knowledge is learnt, i.e. the weights of the neural network is adjusted, then old knowledge stored in

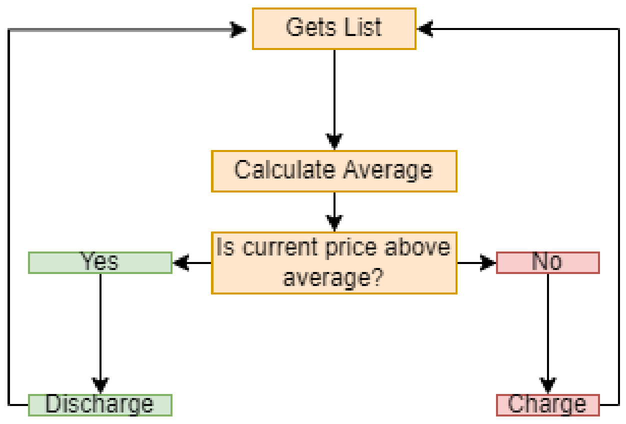


Figure 14: Flowchart of the logic behind the simple algorithm.

the previous wights are erased. This leads to sudden drops in the performance of the neural networks as it forgets how to solve tasks that it previously knew how to solve and is what that causes catastrophic forgetting.

Simple algorithm

The simple algorithm tries to solve this problem in a simple way that is easy to interpret. The goal of the simple algorithm is to have a baseline that both of the genetic algorithm and DQL algorithm needs to beat. This will be the baseline due to its low computational cost and ease of use. If it were to be of similar performance to the other models then it would therefore be a better choice. The flowchart for this simple algorithm can be seen in figure 14. The algorithm always start by being provided an array of price values on the form:

$$V_{price}(t) = (P_t, P_{t+1}, ..., P_{t+n}) \tag{16}$$

Where V_{price} is a vector with all of the prices that is to be used to make a decision. This array will then contain the n next hours of electricity prices, where t=1 will have the price for the current hour. Then the average is taken of this array. The average is then used to decide whether to charge or discharge. If the price is above the average, then it will discharge and if it is bellow then it will charge. After a decision has been made then the algorithm is provided a new array of electricity prices that has been shifted by t+1, such that P_t becomes the previous P_{t+1} and so on.

The advantage of using this algorithm is that it is simple to understand and easy to deploy. However it is so simple that it might make many sub-optimal decisions.

Genetic algorithms

Genetic algorithm is an optimization algorithm that mimics the evolution in nature [38] and will convergence to the global maximum [87] given enough time and a proper stopping criterion. It works as described in figure 15. It is easy to implement, but has the major downside that it will potentially spend much time on converging to an optimal solution, thus it is unsuited for real-time decision making [88]. The figure and all the steps will be explained using the car scheduling problem at hand, i.e. when to charge/discharge so as to accumulate the most profit?

To solve this problem the GA does the following: Firstly it creates several chromosomes, shown as "initiation" in the figure. A chromosome in GA is a set of numbers that specify a proposed solution. So for the charging/discharging problem then a chromosome would be a vector where each element would designate whether to do a charging or discharging action. All elements in the chromosomes are created randomly when they are initialized so as to cover as many possible scheduling orders as possible.

After this has been done then all of the created chromosomes are passed to a fitness function for evaluation. The fitness function provides each algorithm with a score that signifies how good the performance was for that particular combination. There are several fitness functions that could be used, all depending upon what one wants to achieve. One such fitness function could be the accumulated profit by performing all the steps in the chromosome.

After the fitness function has been performed it is tested whether or not the stopping criterion has been reached. This stopping criterion is there to ensure that the algorithm stops and does not keep on trying to optimize forever. There are many possibilities for a stopping criterion. One could be to stop when the best score reaches a particular value or to stop after a predetermined amount of steps/time.

If the stopping criterion has not been reached then the selection phase happens. In this phase then the n chromosomes with the best fitness score will be selected and used to develop new solutions. This resembles the natural selection that is seen in nature, where only the organisms with the genes that gives them an upper edge on the others organisms will survive.

After this step follows the crossover step. In this step the selected chromosomes gets parts of their solution swapped with another algorithm. One heuristic for swapping is as shown in figure 16 to select one point in the middle of a chromosome and swap one of the pieces with the corresponding piece in another selected chromosome. This is akin to the sexual reproduction seen in many organisms, where random parts of the genes are interchanged between organisms. It is not guaranteed that the resulting chromosome is better than its parents, but it is more likely that solutions with good chromosomes will produce a good or even better solution than two bad chromosomes.

After the crossover step comes the mutation step, where there is a predetermined chance that one or several random changes will be done to the chromosome. This is similar to the random mutation one sees in nature where changes in the genome may result in new traits.

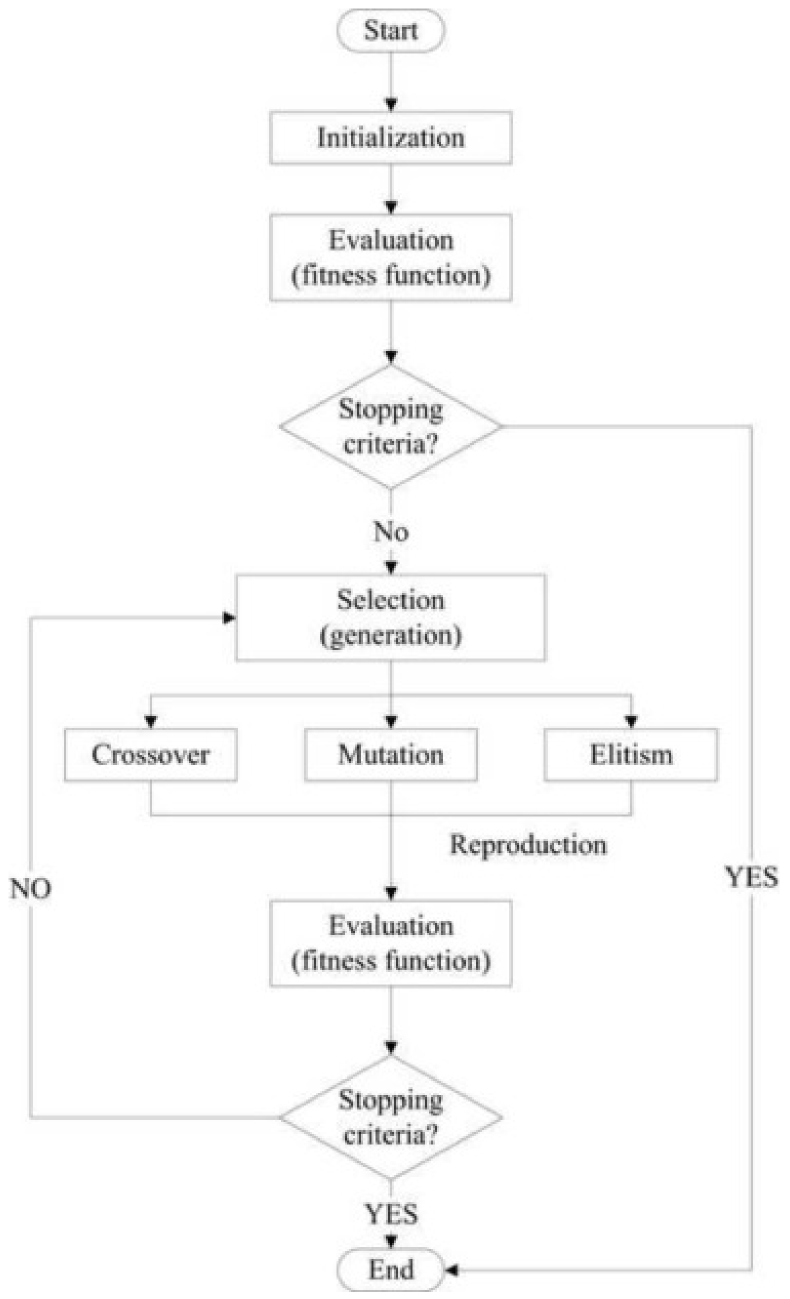


Figure 15: Shows how the genetic algorithm works on a conceptual level. The image was fetched from [38] and is reused with permission.

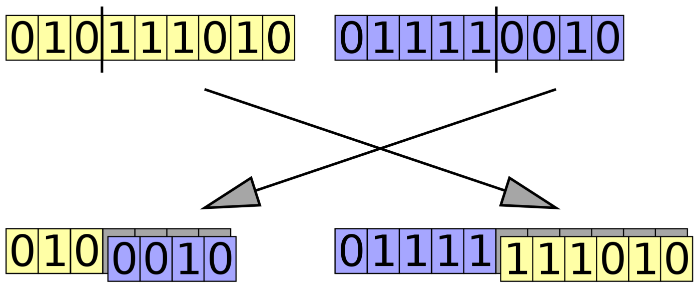


Figure 16: Shows how a crossover step in the genetic algorithm works. There are several approaches to doing this crossover. In this figure then an approach called single point crossover is used. In the single point crossover step then a random point within one of the arrays is selected and the part of the genome up to this point is interchanged with the corresponding part of the other genome. The image was fetched from [89] and is reused under CC3.

Following the above steps it is also possible to implement an optional elitist step, which is to preserve some of the best performing chromosomes without modifying them and using copies of them in each future step to ensure that the resulting chromosomes will be equally good or better than the previous ones. This speeds up the convergence of the algorithm, but with the downside that it might converge to a local optima.

After this final optional step then there is a test for the stopping criterion. If this criterion is met then the whole algorithm will output its optimized decision, while if it does not hold the cycle will repeat, starting with the selection criterion.

Method

Oslo airport at Gardermoen

The NeX2G project aims at using the EVs parked at the P10 parking garage at Oslo Airport Gardermoen for V2G operations. The P10 garage can be seen in figure 17 and is one of two parking lots at Gardermoen that have EV-charging, the other one being the P11 garage. At the time of writing this thesis then there were no chargers that supported V2G at either of the parking garages, but as a part of the project there are plans of installing 5 V2G-capable chargers at the P10 garage.

In the P10 parking house there are 379 parking spaces with EV charging. These 379 parking spaces are divided over 4 charging strips that each can support 64kW of charging [90]. All of these charging strips are connected to a fuse box that again is connected to a transformer. It is from this same transformer that the rest of Oslo airport get their power.

The power transfer over the charging strips is controlled. If there are too many EVs drawing power from the charging strips at the same time, or there is drained too much power from the airport as a whole, the charging power of the EVs will be reduced to a minimum level. If that

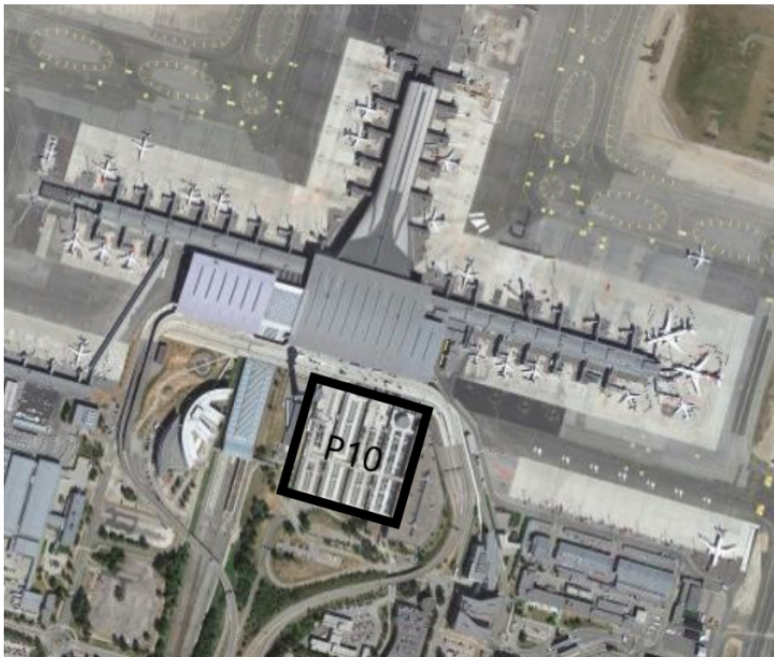


Figure 17: Air photo of Oslo airport at Gademoen. The black rectangle shows the P10 parking lot. This image has been taken from [90] and is used with permission.

does not sufficiently cut the power consumption then the EVs that are charging will be put in a queue-system.

All of the previous information about Gardermoen is rephrasing of the information from the thesis of Rasmus Gåsemyr Tveitane. His thesis should be consulted for a more in depth overview over the parking garages and Oslo airport at Gardermoen. His thesis can be found at [90].

Availability of cars

The parking times of the EVs were analysed, as this will influence what sort of assumptions that can be made about the test cases. Suppose that an EV enters the P10 parking lot with a SoC of 10% and would need 9 hours to charge to the 100% SoC level that it is expected to leave with. This means that if the car is to be picked up within those 9 hours it is impossible to do any V2G, as the EV would have to be constantly charging. On the other hand, if the EV were to be picked up after more than 9 hours then it is a question of how long it would be parked before being picked up. If the EV is to stay parked for 12 to 14 hours, then one can in theory do V2G. Still, one might be more interested in optimizing the charging intervals to minimize the charging cost and the average demand in the grid.

Therefore a distinction has to be made between parking times that allows for V2G and which that only allow charging. For this thesis the limits have been set such that when it is 24 hours left then it is only possible to charge. This limit was chosen such that an EV would be able to use the night to optimize for the lower electricity price during the night, to lower the energy cost of the charging overall.

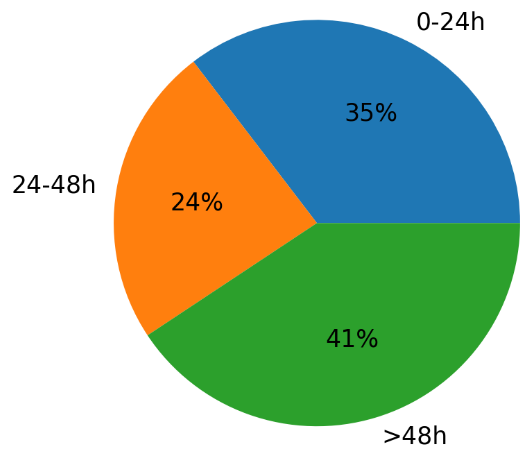


Figure 18: The amount of cars by their parking time. Based on data from the P10 parking lot in the time period from 21st of March 2021 until the second of February 2022.

The distribution of parking times for the P10 parking garage was studied to see how many cars that would be available for V2G. Readings from the individual charging stations for each EV at P10 were reviewed for the period from 21st of March 2021 until the 2nd of February 2022. It was found that a total of 65% of the EVs parked at P10 can do V2G under these conditions (figure 18). This makes it valid to use the limit of 24 hours as there are several cars that can do V2G under these constraints. It should be noted though that 37% of the 65% of EVs that can do V2G are parked for 24-48 hours. This means that under the constraint of 24 hours they might have everything from 1 hour to 24 hours that can be used for V2G. This also means that these EVs will have a varying ability to do V2G under the current constraints.

General assumptions and constraints

Before the testing could be done several assumptions had to be made. One assumption was that the power consumption of the EVs is not restricted by any outer power consumption constraints. Such constraints are limitations in power transfer capabilities in the cables connected to the EV chargers. This assumption does not hold for the P10 garage. Nevertheless this was assumed to not be the case in this thesis so that the algorithms that have been chosen could be tested purely on their ability to optimize V2G based on the electricity price, rather than their ability to even out the power consumption for a given amount of EVs on the same power-line.

It was also assumed that the chosen efficiency does not vary due to temperature, SoC or charging currents. This is despite these being factors that affect the RTE as described in the efficiency subsection. Additionally it was assumed that the EVs would be charged/discharged with their maximum acceptable charging-rate and that this charging-rate would be the same for all SoC values for the battery, even though this is not the case in reality. It was also assumed that the battery do not experience any degradation in the SoC ranges used in the testing. The reason for these assumptions were to simplify the models.

Concerning the price data then it was assumed that the price when selling is equal to the price in the Nordpool data. This assumption is made because the P10 garage is connected to the rest of the Oslo airport grid, and so sending energy back to this grid means that the airport will not purchase that electricity from the grid. Thus the EV ”sells” its energy for the same price as in Nordpool back to the airport.

It was also assumed that the EVs that are optimized for a whole year will be parked the entire time, and will not be picked up in this period. This assumption is made because it will simplify the algorithm testing. In a real life implementation it can be thought that the EVs would be handed over to a charging-optimization algorithm 24 hours prior to being picked up. With such a solution then it would be as if the EVs was going to stay on the airport for a whole year, all up until the last 24 hours before it is picked up.

The maximum amount of price-data that each algorithm received each day was limited to a maximum of 24 hours. This follows from the explanation in the On the price data subsection.

The reason why 24 was set as a maximum and not 36 was to avoid having overlapping regions for the GA. This would happen if the GA optimized for the next 36 hours every day at 13 o clock. Then there would be an overlap of the last 12 hours of the 36 hours known each day when the GA were to make a new optimization.

For the DDQL and the simple algorithm a maximum limit of 12 hours is the most realistic upper limit for the price data that they will have available. This is because they can only take in a fixed amount of hours each time they make a decision. If they were to use more hours than 12, such as say 14 hours ahead, then at 12:00 each day when only the next 12 hours are known then they would not have access to the last 2 hours that they would need. This could be amended by using some sort of predictive algorithm to fill out these values or using some other rule such as setting the last 2 of the 14 hours to 0 or the mean of the remaining 12. Still to test such solutions is beyond the scope of this thesis and so it is set that realistically the simple algorithm and the DDQL can only have access to a maximum of 12 hours ahead. Still both the simple algorithm and the DDQL was tested for having access of everything from 2 up to 23 hours ahead. This was done so as to both see what the effects would be of having access to everything from a very few hours up more hours than they would realistically have.

Setup of the testing

The testing of the different algorithms were conducted as shown in figure 19 where the actual code that was used can be found at the main gitlab page for this thesis at [1]. First an environment was set up. In this environment the algorithms would decide what actions to take, receive a reward based on this action and get the prices of the next n hours. The reward that it got for taking its chosen action, in addition to the action it took, was then stored. The environment would then output the next n prices to the algorithm so that it could make take a new action. And so the process would be repeated all until the given algorithm reached a predetermined stopping point.

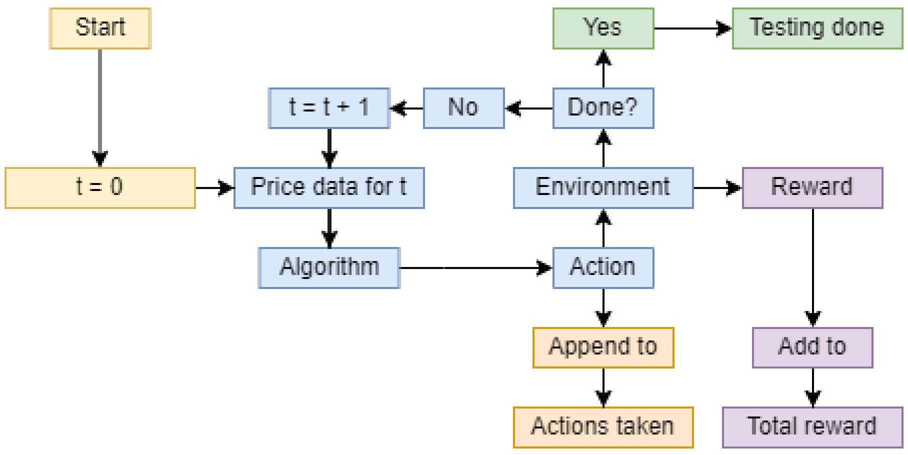


Figure 19: Overview of the testing system. The boxes in blue is the cycle that does the testing. The yellow boxes is the start that initiates the testing cycle.

It was different from algorithm to algorithm how they prepared for the testing phase. For the DDQL it would first have a training phase before the testing phase. For the GA it would have its training and testing phase simultaneously, while the simple algorithm did not use any sort of training phase at all. How the tests and training were run for each algorithm can be found in their respective methods subsections.

The price data that was used in the testing would ideally have been the electricity prices from Avinor rather than the prices from Nordpool, to get the best representation of the true cost of electricity for Oslo airport Gardermoen. Unfortunately these are confidential and could therefore not be used. For this reason the elspot prices from Nordpool were used. The Oslo-region was chosen to be used as the price-zone due to its close proximity to Gardermoen.

The price data that was available stretched back several years. It was decided that the price data that were to be used should be the price data from 2019 to 2021. This should be enough to cover a large span of prices, from the unusually high prices of 2021, to the low prices of 2020.

Each algorithm was tested for the years 2019, 2020 and 2021. The simple algorithms and the different implementations of the DDQL algorithms were additionally tested for varying window sizes, while the GA was only tested for a window size of 24. A ”window” is the name of the vector of the upcoming prices for each hour, and the size of the window refers to the length of the vector. A window of 24 will thus be a vector with the next 24 hours, where the first hour is the current hour. In addition to variations in the window sizes then each algorithm was also tested for each of the V2G compatible Leafs in table 1.

A summary of some of the main differences in testing regimes can be fount in table 2. As can be seen from the table then both the simple algorithm and the DDQL were both tested for the same interval of window sizes. The reason why there is such a large span in the window sizes that was tested is that both DDQL and the simple algorithm uses functions that are sensible to the window size, as will be further discussed in their separate sections. Additionally it was expected that the GA would have an advantage of using the largest possible window size. This is why it was used a window size of 24 for the GA.

Variable	**What that was tested**
EVs tested	All the 7 V2G compatible Nissan leaf models
Years	2019, 2020, 2021
Models	Simple algorithm, DDQL (2 seperate models), GA
Window size	2-23 (Simple algorithm and DDQL), 24 (GA)

Table 2: An overview over what that was tested for.

In addition to the window sizes then it was also different from algorithm to algorithm as to what extent they altered the data in their windows. For the simple algorithm and the GA then they would receive the data unaltered. The DDQL on the other hand had its window transformed before being used for training or decision-making. The transformation varied depending upon the reward function that was used and will therefore be discussed primarily in the DDQL subsection of the methods section. The transformations were done as it improved the profit of the DDQL algorithm.

It was also determined to test for two scenarios with different constraints. The first scenario is called "naive scenario" while the other is called "realistic scenario". A simple summary of the two scenarios can be found in table 3.

Constraint	Naive scenario	Realistic Scenario
SoC interval	0-100%	20-80%
Charging efficiency	100%	95%
Discharge efficiency	100%	95%

Table 3: A brief summary of the key constraints in each of the scenarios

Naive scenario

This scenario assumes no loss at either charging or discharging. Furthermore the battery was allowed to fully charge and discharge. This scenario is an unrealistic view of the problem, but serves as a theoretical maximum amount for the profit that could be earned over a year. It can also be used to compare what effect the constraints in the realistic scenario has on the performance of the different algorithms.

Realistic scenario

This scenario assumes that there will be a loss related to the charging and discharging of the EV. The loss in this scenario is based on the RTE for the car battery, and the RTE value is assumed to be 90 %. The efficiency that has been reported was up to 95% for the NMC batteries, still this more conservative value of 90% was chosen for the RTE. This was to compensate for potential losses in charging, and other potential losses such as varying efficiencies due to temperature and charging rate. From the RTE of 90% then the charging and discharging efficiencies were both set to 95%, which was determined by taking the square root of the RTE value: $\sqrt{0.9} = 0.95$. The charging and discharging values might realistically vary, but have been set to be equal to one another to simplify the testing. In addition to the other constraints the lower limit for the battery was set 20% SoC and the upper to 80%, as suggested in the literature to reduce the potential battery degradation.

No-profit regions

The maximum profit that can be made during a day by performing one charging and discharging operation will be when one charges at the minimum price and discharges at the maximum price of that particular day. With the realistic scenario and a RTE of 90%, only 90% of the purchased electricity can actually be discharged. This means that for a given day the maximum price times the RTE needs to be larger than the minimum price for V2G to be profitable. This means that certain days will not be profitable in terms of V2G in the realistic scenario due to the efficiency chosen. The formula for deciding whether a day is profitable or not is expressed in equation 17 where R is a variable that is positive if profit can be made and negative if a profit cannot be made. P_{max} is the maximum price while P_{min} is the minimum price during the given day.

$$R = \begin{cases} 1 & P_{max} \cdot 0.9 > P_{min} \\ -1 & P_{max} \cdot 0.9 \leq P_{min} \end{cases} \tag{17}$$

How many days that will have a negative R-value, i.e. they will not be profitable, will vary from year to year and in terms of the chosen RTE value. It should be noted that the amount of days that are no-profit may be either larger or smaller than the value found by using equation 17. The prices could potentially be larger because the maximum price sometimes comes before the minimum price during a day. In those cases then the difference in price between the smallest price at the beginning of the day and the maximum price during that day might be so small that the day becomes a no-profit day. The number of no-profit days might also be lower as this way of categorizing data does not take into accounts that the algorithms might charge energy on a minimum one day, and then discharging it on another day. In other words it does not consider that one day could be no-profit, but the difference across two days might be. Nevertheless it is used as a metric as it gives a simple and rough estimate of the difference in profitability for the days in the different years.

Percentage of absolute profit

A Unit was needed to measure the impact of no-profit days on the overall profitability of a week. This is why percentage of absolute profit was made. It is given in equation 18, where day is the current day of the week, P_{day} is the amount of profit that can be earned during a day, by making only one charging and discharging decision and n is the amount of days in a week.

$$P_{absprofit} = \frac{\sum_{day=1}^{n} ||P_{day} < 0||}{\sum_{day=1}^{n} ||P_{day} < 0|| + \sum_{day=1}^{n} P_{day} > 0} \tag{18}$$

The formula for P_{day} can be found in equation 19, where $min()$ and $max()$ are functions that finds the minimum and maximum values during a day respectively and the $efficiency$ is the RTE for the given case and t is the current day that is in question.

$$P_{day} = max(P_t) * efficiency - min(P_t) \tag{19}$$

Percentage of absolute profit is a measure of how large the weekly loss, due to performing V2G on a no-profit day, is compared to the combined value of the absolute value of all the losses made in that week due to no-profit days, and all the profit made on days with profit. This equation will return a decimal value. If the value is between 0 and 0.5 then it means that a net profit that week was made. If the value is in between 0.5 and 1 then it means that the losses due to no-profit periods will dominate and so unless the no-profit periods are avoided then that week will be a net loss. Percentage of absolute profit, as no-profit days has its limitations, as it does not consider the effects of performing several V2G operations in one day, but serves as a rough estimate to get an overview of the impact of no profit-days on the overall profitability of the week.

Environment

The environment is a system that all the algorithms need to be able to receive feedback on their actions, and get new information about the state of the EV that they are interacting with. The environment is supposed to be a representation of the EV and the electricity market. So it has access to both the Nordpool prices and some of the properties of the EV it is supposed to represent such as SoC of the battery and C-rate.

The environment is made as a python class, this was done to make it simpler to use by the different algorithms. It works as shown in the flowchart in figure 20. The environment works by one of the algorithm first requesting an action to be taken in the environment. This action is either to charge, discharge or do nothing. Then the environment decides what the reward for that particular action shall be. If the algorithm requests a discharge, then the reward will be positive, if it is a charge, then it will be negative and if it takes no action then the reward will be zero. The reason why the charging reward is negative is to present to the algorithm the cost of doing the charging action.

When a chosen reward is determined then the environment will return a vector on the form as shown in equation 20. In the return vector then s is the state after taking the decided action. The state will be given as a vector containing the prices of the n upcoming hours and the current SoC level. The r' is the modified rewards which is only relevant for the DDQL. d is a boolean variable telling the algorithm if it has reached a predetermined limit. In the case of the testing system then this will tell the algorithm it is done when the whole year has been tested. In the case of the training of the DDQL then this will tell the algorithm it is done when the DDQL has trained in the whole training sample. c is the SoC of the battery after the action has been taken, it was provided as a separate output to simplify the coding of the GA. r is the true reward (figure 20) of the current action.

$$R = (s, r', d, c, r) \tag{20}$$

The environment has built into it that it has an upper and a lower limit, such that no more can be charged than the upper limit and no more than the lover limit can be discharged. For the naive scenario the upper limit is set to a SoC of 100% and the lower to 0%, while for the realistic scenario then it is set to 80% and 20% respectively. If the environment is asked to charge and the battery is almost full, then it will charge up to the upper limit, and return the

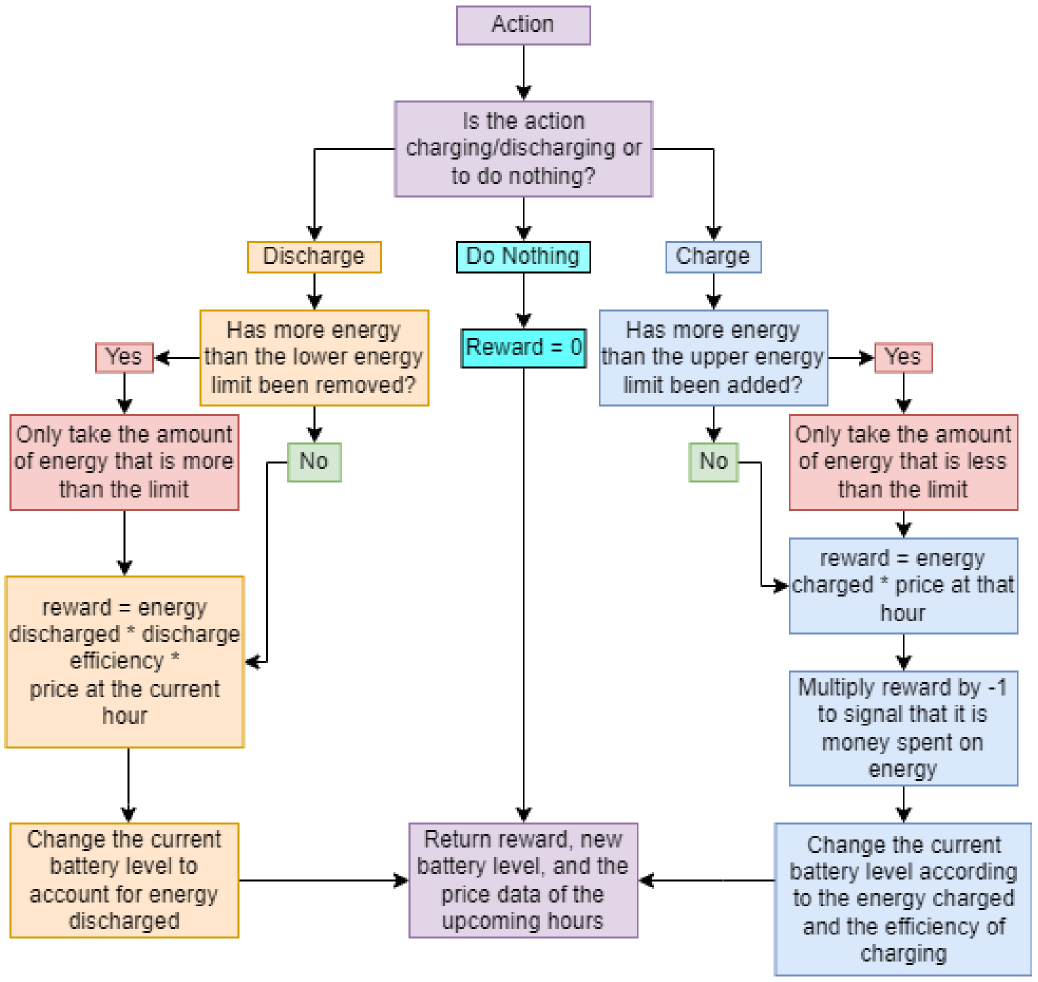

Action
Is the action charging/discharging or to do nothing?
Discharge
Do Nothing
Charge
Has more energy than the lower energy limit been removed?
Reward = 0
Has more energy than the upper energy limit been added?
Yes
Yes
Only take the amount of energy that is more than the limit
No
No
Only take the amount of energy that is less than the limit
reward = energy charged * price at that hour
reward = energy discharged * discharge efficiency * price at the current hour
Multiply reward by -1 to signal that it is money spent on energy
Change the current battery level to account for energy discharged
Return reward, new battery level, and the price data of the upcoming hours
Change the current battery level according to the energy charged and the efficiency of charging

Figure 20: Shows how the environment will respond when an action is taken. From the figure it can be seen that the reward from charging is not multiplied by the efficiency such as for the discharging. That is because the loss when charging is only measured as the energy lost on its way from the charger to the battery. Still one pays for all the energy that leaves the charger even though not all the energy reaches the battery. This means that the efficiency is only multiplied with the amount of battery that the EV is charged with, not with the reward.

reward for charging up to the limit. The same applies for discharging.

In addition to the limits then the environment can also handle varying efficiencies for charging and discharging separately. A lower efficiency than 1 for charging implies that of the energy purchased, then only a percentage of the purchased energy will reach the battery. For the discharging the mechanism is similar in that of all the energy that one discharges, then only a percentage equal to the efficiency is actually sold to the grid. The loss is as described in the efficiency subsection and is attributed to losses within the battery such as to internal resistance. Efficiencies in the charger is not taken into consideration even though this will reduce the efficiency even further.

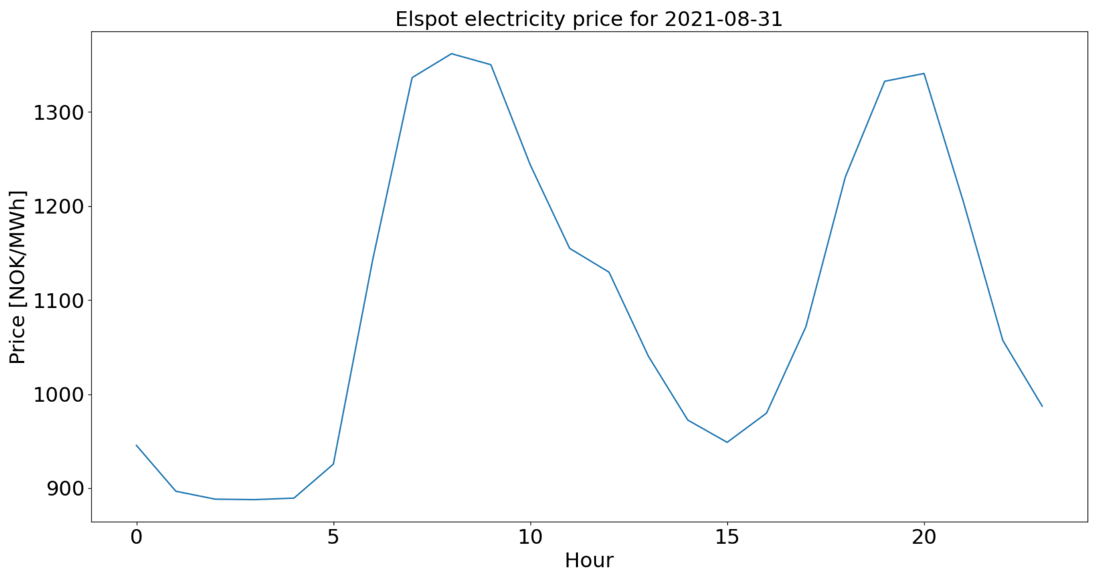


Figure 21: Elspot prices from Nordpool for the day 31th of August 2021, showing a price curve with clearly defined peaks and valleys.

Strategies for optimization

As mentioned earlier then the goal of this thesis is to optimize for profit, and any profit that can be made is based upon the difference in prices. This implies that the larger the difference between the highest and lowest price during a day then the more income there potentially is for each algorithm. This seems like a simple problem to solve, i.e. buy when it is at the cheapest and sell when it is at the most expensive. One could use figure 21 as a good example to illustrate this. A strategy could be to buy energy for the interval between 0 and 6 o' clock and sell between 6 and 12, purchase again between 12 and 17 and sell again between 17 and 22 o'clock. This seems fairly easy to do, and as something that could and should be easily automated, but the price curve is not always as easy to make a strategy for. A good example of how it can be difficult to find a good strategy can be found in figure 22, where such a simple and good strategy is not as apparent.

In addition to this variance in the price one also has to consider the charge/discharge rate. Suppose that one had an EV with a large discharge rate, such that one could charge/discharge the whole battery over one hour. In this case then one could make a very simple algorithm to solve for charging/discharging. The car would then charge at all the points where the price would go from decreasing to increasing, relative to its previous point, and would discharge at all the points when the price goes from increasing to decreasing. for figure 21 this would mean that it would charge at 3 and 15 o clock and discharge at 8 and 20 o' clock. This sort of strategy would change if the charging/discharging rates is such that it does not manage to fill the whole battery in 1 hour, as this would mean that the battery would have to be charged over a larger interval of time. Thus the charging/discharging rates are important for which strategies to pick and this means that an algorithm has to take this into consideration.

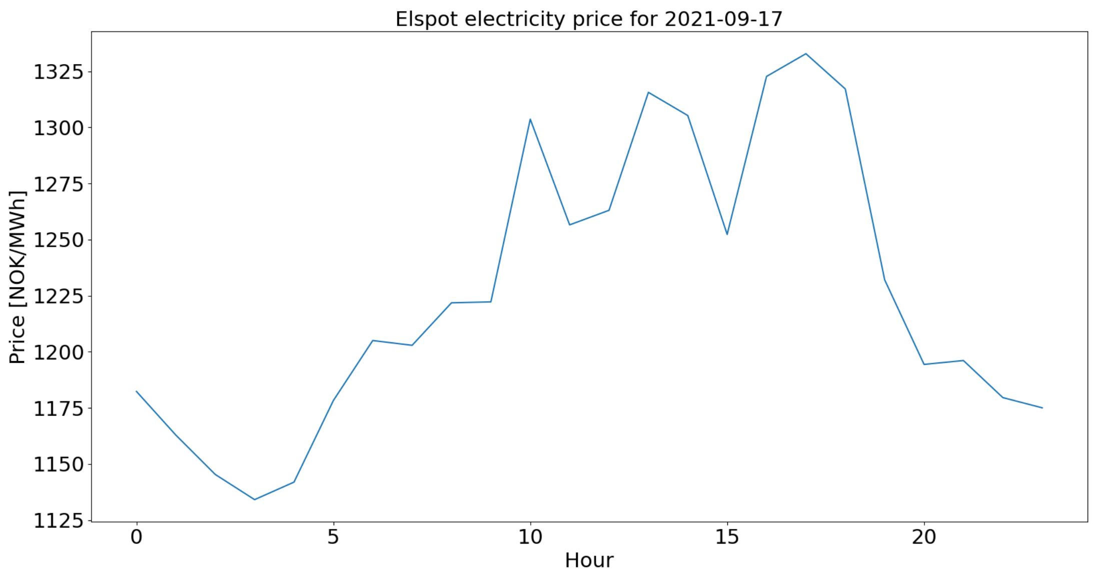


Figure 22: Elspot prices from Nordpool for the day 17th of September 2021, showing a price curve with a less clearly defined shape.

It is in other words not a clearly defined strategy that is bound to always give the best solution to the profit optimization problem. This means that this problem most likely can not be solved by a very simple algorithm. The simple algorithm used in this thesis was made to test how a very simple algorithm, with only one strategy, would compare with the DDQL that has the abilitiy to make a strategy according to the data that it is provided and the GA, that has the ability to change its strategy depending upon the price data that it is provided.

Algorithms used

Here the setup of the different algorithms will be discussed, in addition to the parameters that were used in each algorithm. The subsection will start with covering the simple algorithm, then the DDQL and finally the GA.

Simple algorithm

The simple algorithm works as described in the pseudocode of algorithm 1. It starts by being provided a window of price-data. Based upon this it calculates the average of the window and makes a decision based upon this average to charge/discharge/do nothing. If the current price is above average, it will charge, if it is bellow it will discharge and if it is the same value as the average then it will do nothing. The environment will respond to the action taken by the simple algorithm by returning the prices of the next hours and the reward of that particular action. The reward and the action taken will be stored in an array for later. After all of these steps have been performed it is checked if the end of the testing data has been reached. If it has been reached then the testing stops. If not then it will start by picking the next action and staring the loop again.

Algorithm 1 Simple algorithm

```
Inputs:
    Env                                              ▷ Environment to optimize for
Outputs:
    TotReward                                        ▷ The accumulated reward
    Actions               ▷ Set of actions that result in the optimal solution
Function SimpleAlgorithm
 1: TotReward ← 0
 2: Actions ← EmptyArray
 3: Done ← False
 4: PriceWindow ← Env.GetStart                       ▷ Gets the first price window
 5: while not Done do
 6:     ← PriceWindow.FirstElement
 7:     if Average(PriceWindow) > CurrentPrice then
 8:         Action ← Charge
 9:     else if Average(PriceWindow) = CurrentPrice then
10:         Action ← Do nothing
11:     else
12:         Action ← Discharge
13:     PriceWindow, reward', Done, Charge, reward ← Env.Step(Action)
14:     TotReward ← TotReward + reward
15:     Actions.append(Action)
16: return TotReward, Actions
```

The size of the window will affect how much the simple algorithm responds to new price data, as the average will depend upon the data points that is within the window. Take the extreme example of using a window size of 2: When the algorithm takes an action and moves the window by one price point ahead in time then the new datapoint might be much larger than the previous two. This would greatly increase the average in the new window compared to the previous two prices in the window. If the window size is 24 and one moves the window one step this effect will be much smaller because of the number of elements in the window. In other words the different window sizes needs to be tested to see which size that is ideal for the current issue at hand. For some price curve it could for instance be better to have a small window so that the average changed often. An average that changes often could lead to the simple algorithm taking fewer or more actions, thus leading it to perform better. This is the reason why for each year and each EV then the simple algorithm was tested for 22 different window sizes, ranging from 2 to 23.

DDQL

The DDQL algorithm was tested by first letting it explore the first 1500 hourly prices of each year, similar to how it was done in the study by Kiaee [36]. One difference between this thesis and the one of Kiaee is that this thesis does not train on any more than these 1500 hourly prices. In the works of Kiaee then the algorithm was first trained on the first 1500 hourly prices before the algorithm was tested on the next 500 hourly prices. Afterwards the prices that was used for training was moved by 500 points, such that the algorithm now got trained on all the hourly prices from the 500th hour to the 2000th hour. After this the process was repeated. Going by such a procedure might have improved the algorithm on future data. Yet the issue of catastrophic forget-fullness was observed in the algorithm and so the DDQL was not able to learn new information after a certain point, as will be further discussed in the discussion section. This led to the simplification of the algorithm where only the first 1500 hourly prices were used. More prices could have been used, but the 1500 was selected to see how well the algorithm could generalize to new data based upon a small subset of them.

The DDQL was set to explore the training data. Afterwards it was trained on the moves it took in the exploration phase and then tested on the whole 1500 hourly prices to see how well it had learned to optimize for the prices. If the model performed better than the previous models then it would be stored for future use. If the model performed worse than their predecessors for 6 times in a row then the testing would be stopped and the model would be tested on the whole year, to see its actual performance. The pseudo-code outlining the actual code used to both train and test the DDQL can be found in algorithm 2.

For the neural network used to approximate the neural network, then a two layered network, with one output layer, was used. The first layer used 100 neurons, and the ReLu activation function, while the second layer used 120 neurons and a ReLu activation function. The output layer consisted of 3 output neurons that all used a linear activation function. The full implementation can be found at [1].

Moving on to the reward function then this is one of the more difficult parts of RL , as what one rewards will determine what sort of behaviour one sees. One great example of how the choice of a poor reward function can lead to unexpected behaviour is the case of [91]. There a RL agent was trained to drive trough a track in a game by picking up rewards along the track, where the rewards would reappear after a certain amount of time. The plan would was that the agent would follow the rewards along the track and become increasingly better at the game. Unfortunately the agent ended up with driving around in circles to pick up the reappearing rewards, because this lead to it accumulating more reward, short term, than actually completing the track.

To try to find an optimal reward function then several possible reward functions were initially tested. It was tested as suggested by [36], [33], [34] and [30] to use the prices directly as reward, where to charge would result in a negative price and to discharge would result in a positive price. It was also attempted to use the reward function of [82] and [35]. All these reward functions showed poor performance when compared with the simple algorithm, and so two new reward functions were made to try to achieve a better performance. To the greatest

Algorithm 2 DDQL

```
Inputs:
  Env                                          ▷ Environment to optimize for
Outputs:
  TopReward                          ▷ The accumulated reward of the DDQL
  BestActions            ▷ Set of actions that result in the optimal solution
Function TrainAndTestDDQL
 1: Initialize replay buffer B
 2: Initialize DDQL-model Q with random weights
 3: Initialize target DDQL-model Q* with random weights
 4: TopReward ← $-\infty$
 5: BestModel ← EmptyObject
 6: Done ← False
 7: $\epsilon$ ← 1
 8: while TotReward has been altered in the last 5 iterations do
 9:     PriceWindow ← Env.GetStart            ▷ This will reset the environment
10:     while not Done do
11:         with probability $\epsilon$ do: action ← Random.action
12:         otherwise do: action ← argmax(Q(PriceWindow))
13:         PriceWindowNew, reward', Done, Charge, reward ← Env.Step(action)
14:         B.append((PriceWindow, Action, reward', PriceWindowNew, Done))
15:     for abc in B do
16:         update the Q by using equation 13 and the adam-optimizer
17:         update the Q* by using equation 11
18:     TotReward, Actions ← TestQInEnv(Q, Env)
19:     if TotReward > TopReward then
20:         TopReward ← TotReward
21:         BestModel ← Q
22: return BestScore, Charge, Actions
```

extent of the author of this thesis' knowledge these two reward functions has not been used before for reinforcement learning. The DDQL algorithm using the first one will hereby be called mean-DDQL while the DDQL using the other one will be called STD-DDQL. On a conceptual level they both work as shown in figure 23. They try to reward charging when the electricity price is low and sell it back when the electricity price is high, but they are two different approaches at solving this issue. A explanations of the two reward functions is to follow.

Mean reward

The mean reward works by finding the difference between the average price in the given window and the price when an action is taken. This difference is used as a reward, as can

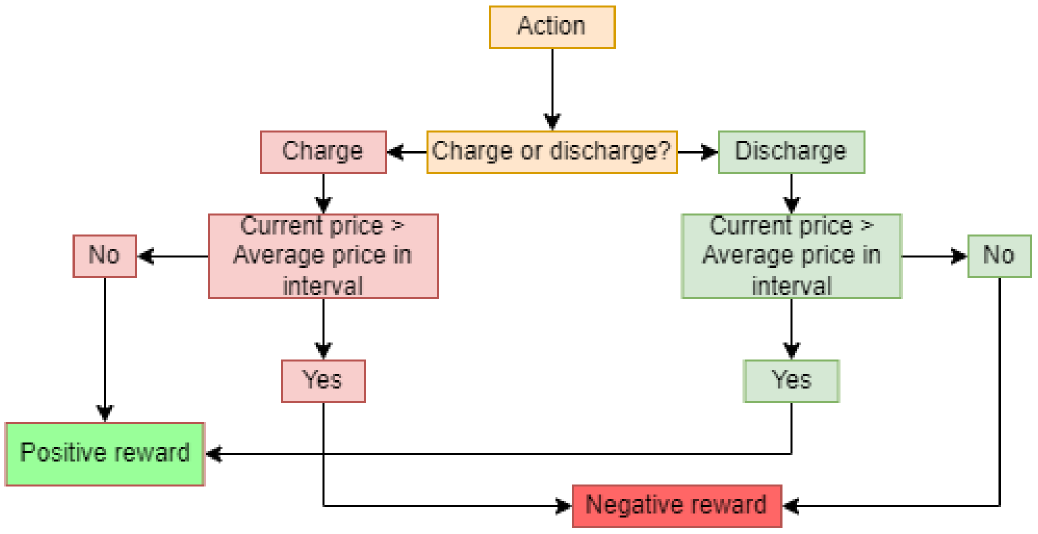

Action
Charge or discharge?
Charge
Discharge
Current price > Average price in interval
Current price > Average price in interval
No
No
Yes
Yes
Positive reward
Negative reward

Figure 23: Shows the concept behind the reward functions of the DDQL-algorithm. The principle behind both of these reward functions is: They will provide a negative reward, i.e. a punishment, if the algorithm sells when the electricity is cheap during a day. If the algorithm on the other hand sells when it is expensive or purchase when it cheap then it will be rewarded with a positive reward.

be seen in equation 21. In this equation then $R1$ is the mean reward, w is a weight that is -1 if the action was charging, 1 if it was discharging and 0 if the action was to do nothing, μ is the average price in the given window and P_{action} is the price at the time of performing the action. From the equation one can see that if the price is larger than the mean then this difference will be positive, if it is not then it will be negative. This difference will then be turned negative or positive, depending upon w, i.e. the action, following the rules as shown in figure 23.

$$R1 = w(P_{action} - \mu) \tag{21}$$

It was observed that the performance of the mean-DDQL improved when the mean of the input window was subtracted from each element in the input window. Because of this all the input windows of the mean-DDQL was transformed by subtracting the mean.

STD reward

Standardized (STD) reward works similar to that of reward function 1. The main difference is that instead of using just the distance from the mean then one also divides this by the standard deviation, as can be seen in equation 22. In that equation then all variables are the same as in equation 21, with the only difference being that σ is the standard deviation, as is shown in the appendix. The reason why this reward function was chosen is that it could be that this reward function is better at finding and charging at the peaks and valleys than the mean-algorithm. This is because the STD reward will give a larger value to prices further away from the mean than those close to it depending upon their relative size in accordance with the standard deviation. This standardizing of the data could improve its capability to generalize to new data and to make it better than the mean reward function at spotting peaks and valleys.

$$R2 = w(\frac{P_{action} - \mu}{\sigma}) \tag{22}$$

The input window to the STD-DDQL was standardized. This was because the STD-DDQL had its performance improved if the input window to the model was standardized.

Both the mean-DDQL and the STD-DDQL was provided windows with varying sizes. This was because both the mean-DDQL and the STD-DDQL have a reward function that is sensitive to the window size, the mean-DDQL because it uses the mean, STD-DDQL because it uses both the mean and the standard deviation of the window. It is thus not obvious which window sizes that will lead to the best performing algorithm, and so several window sizes from 2 to 23 was tested to compare the variation in performance due to window size.

Genetic algorithm

The genetic algorithm was set up as described with pseudo-code in algorithm 3 and it is an implementation of the concept presented in figure 15. The goal of the algorithm is to find the optimal scheduling for the next 24 hours. To do this then a variable called *BestScore* is initialized. This variable is used to store the best score that is achieved through optimization. The variable is set to negative infinity initially so that any score will be an improvement to this score. Afterwards then a set of genomes are created. The genomes will be of the same length as the price data that it will optimize for, and it will be create an equal amount of genomes to the *PopSize*. For the realistic-scenario then the *PopSize* was set to 100 while for the naive-scenario then it was set to 40. This was done because of long runtimes, as will be further discussed in the discussion section.

After these variables have been initialized then the while loop starts on line 3. The while loop will run until the *BestScore* has not been updated in the last x iterations, i.e. that the GA has not been able to improve in x interactions. For the realistic case then this x will be set to 10, while for the naive case this x will be set to 4. The difference in the parameters is due to the runtimes as will be further discussed in the discussion section.

In the while-loop the GA will start by selecting the genomes that performed the best in the environment. For the realistic scenario this was set to the 10 best, while for the naive test it was set to 2, due to long run-times. Then the genome will move on to loop through the different genomes and do a crossover of the genomes. This is done through a single point crossover, i.e. a random point within one of the arrays is selected and the segment from the start of one of the genomes up to the random point is swapped with the other genome.

Afterward the a random element in each genome that had had a crossover will be mutated. For this implementation this was done so that a random element within the genome was selected and had an 80% probability of being mutated, i.e. to have its value replaced by a random value.

Algorithm 3 GA

```
Inputs:
    PriceData                         ▷ Price data to optimize for
    PopSize                           ▷ Population size to use for genome
    Env                               ▷ Enviroment to test the solution in
Outputs:
    BestScore                         ▷ The score of the optimized solution
    Charge                            ▷ The charge in the battery after the optimization
    Actions                           ▷ Set of actions that result in the optimal solution
Function OptimizeGA:
 1: BestScore ← $-\infty$
 2: Genomes ← GeneratePop(PriceData.Length, PopSize)
 3: while BestScore has been altered in the last 10 iterations do
 4:     Genomes ← TopGenomes(Genomes, Env)          ▷ Selects the best genomes
 5:     NewGenomes ← EmptyArray
 6:     for index in [0 to Genomes.length - 1] do
 7:         Gen1 ← Genomes[index]
 8:         Gen2 ← Genomes[index+1]
 9:         for step in [0 to PopSize//Genomes.Length] do
10:             NewGen1, NewGen2 ← Crossover(Gen1, Gen2)
11:             NewGen1 ← Mutate(NewGen1)
12:             NewGen2 ← Mutate(NewGen2)
13:             NewGenomes.Add(NewGen1, NewGen2)
14:     if BestScore < BestGenome(NewGenomes) then
15:         BestScore ← BestGenomeScore(NewGenomes)
16:         Charge ← BestGenomeCharge(NewGenomes)
17:         Actions ← BestGenomeActions(NewGenomes)
18:     Genomes ← NewGenomes
19: return BestScore, Charge, Actions
```

The resulting new genomes would replace the previous genomes. If the new genomes had a genome that scored better than the previous best score, then it would replace the previous best score.

The GA was only provided windows with a size of 24 and it would only do the optimization for the upcoming day at 24:00 o clock. The choice of window size for GA is a trade-off between time and better optimization. With a larger window size then the GA can make better long-term optimization strategies, but by increasing the window size one potentially increase the runtime of the GA. Still 24 hours was chosen as this is the amount of hours of price data that is known ahead of time at 24:00, which is the time when the GA shall optimize for the next day. By choosing this window size and time to make the optimization then one avoids that two periods that was optimized for would be overlapping. Overlapping periods could make the GA better at making long term strategies, but is not guaranteed to happen.

To avoid having to consider the potential effects of overlapping regions the current choice of window size and time to make the optimization was chosen.

Results and discussion

Average performance in each scenario

This subsection will review how well each algorithm performed in the naive and realistic scenario. For both the naive and realistic scenario then the GA spent 1.5-2 hours on optimizing for one Nissan leaf model for a whole year. The Simple algorithm used around 5 minutes, while the mean-DDQL and STD-DDQL spent around 1.5 hours each on training and optimizing for the same Leaf model. The simple algorithm and mean-/STD-DDQL both spent their respective time on training/optimizing for all the 22 windows.

Naive scenario

In the figure 24 the average profit for the studied algorithms in the naive scenario is shown. The different DDQLs and the simple algorithms appear to all have close to the same profit and have a higher profit than the GA for all the years. It can also be seen that there was generally the most profit to make in 2021 and the least to make in 2020.

The average percentage price difference between the peak- and valley-price during a day was tested for each year. This amounted to 17% for the year 2019, 30% for 2020 and 32% for 2021. Additionally the average price for each year was found and it amounted to 387 NOK/MWh for 2019, 101 NOK/MWh for 2020 and 761 NOK/MWh for 2021.

Discussion

It is interesting that the simple algorithm, the mean-DDQL and the STD-DDQL performed better than the GA. The reason why the GA performed the worst across all the years is that it was used a small *PopSize* of 40 instead of the originally planned 100. Additionally the amount of iteration with no improvement of *BestScore* needed for the GA to stop optimizing was set to 2 instead of 10. This was done because of long run-times for the GA, where the GA initially used up to 12 hours on finding the optimal charging/discharging actions for a Nissan leaf model for a whole year. This was far longer than the more acceptable 1.5 hours that the run-time was brought down to. In other words the performance of the GA could have been increased so that it would have been better than the other three algorithms, but with the cost of longer run-times.

When reviewing the naive scenario then it is surprising that the simple algorithm and the mean-DDQL and STD-DDQL had such a similar performance. One would think that there are differences in the way they function, such that there would be a noticeable

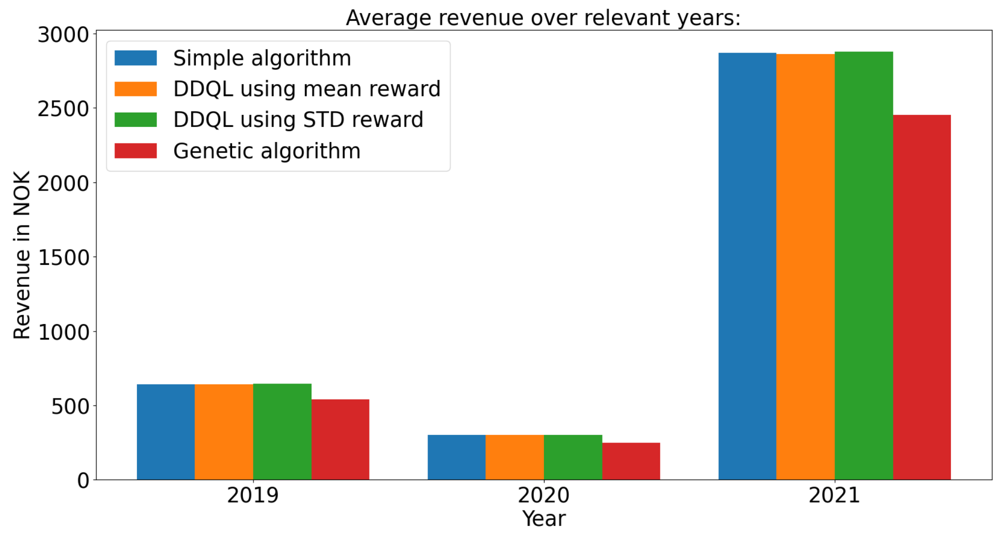


Figure 24: Shows the average earnings for each algorithm over the three relevant years for the naive scenario. The profit of each algorithm is grouped by their respective years.

difference in their profit. Additionally one would think that the simple algorithm would perform considerably worse than the DDQL algorithms, as it does not have any mechanisms built into it for handling optimization for peaks, other than the size of the window it will use. This shows that in the naive scenario then optimizing for price was not a complex problem, as even the simple algorithm could solve it well, and it could be solved by simple rules.

That the year 2021 is more profitable than the year 2019 and 2020 was to be expected considering that the average prices of electricity had increased that year. Compared to previous years then these high prices was an anomaly, and was caused by several factors such as a low level of water in the water magazines, a cold winter and higher electricity prices in Europe[92]. One possible reason for why the profit would increase is that when the prices increases then a profit can be made on the difference between two prices. Additionally with an increasing average price then the profit will be greater as can be seen in the difference between 2019, 2020 and 2021. To explain how the increasing average price affects the profit then suppose that the differences in price between the peak and valley during a day is a fixed percentage across different years, let this number be 20% and the average price for year a is 10NOK/MWh and 100NOK/MWh for year b. In this case then the profit that can be made each day is 0.2*10NOK/MWH = 2 NOK/MWH and 0.2*100NOK/MWh = 20NOK/MWh for year a and b respectively. This means that with an increasing average price then the profits will increase.

This is a simplification, as the percentage differences between the peaks and valleys during a day varies within the year and also across the different years. Additionally the shape of the price curve during each day in a year will also affect the profitability of a year, not just the percentage difference. Still an increasing average price generally means that an increased profit can be made by V2G, assuming that the percentage difference in price does not decrease to compensate for the increase in average price. The year 2020 illustrates how the increase in average price will influence profitability. In 2020 the average percentage difference between the peak and the valley was almost twice as large as for 2019, and still the algorithms earned around twice the amount for 2019 as in 2020.

Realistic scenario

The average profit for the algorithms in the realistic scenario is shown in figure 25, while barplots for each separate EV for the realistic case can be found in the appendix. The GA had the largest profit of all the algorithms over all the three years, with the simple algorithm having the lowest profit for the year 2019 and 2021. The mean-DDQL was the algorithm with the second largest profit in 2019 and 2020, but in the year 2021 STD-DDQL performed the second best.

When comparing figure 24 and 25 it can be seen that under the constraints set in the realistic case then the algorithms makes approximately half the profit as in the naive case. Additionally there are a larger variation in the profits made by the simple algorithm, the mean-DDQL and the STD-DDQL for each year in the realistic case than in the naive scenario. The largest change in variation from the naive to the realistic scenario is in the year 2019.

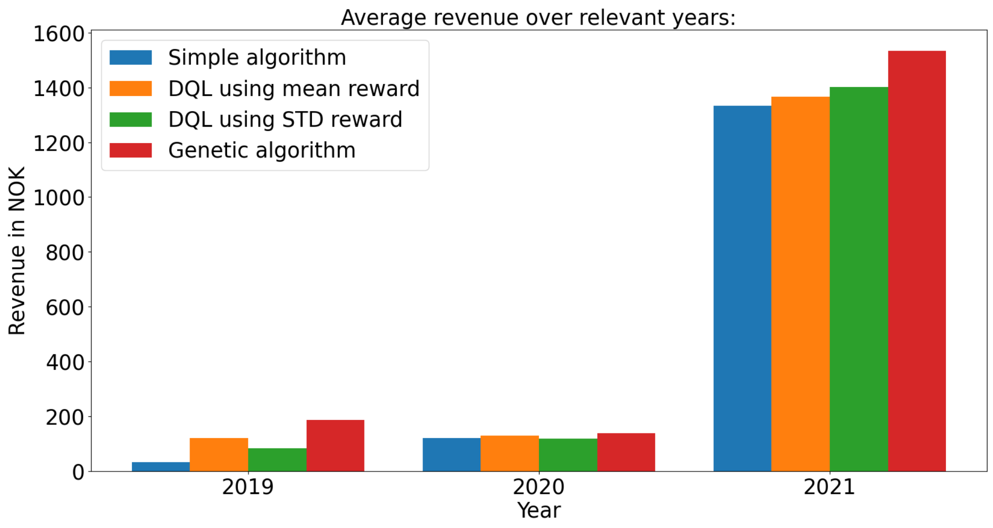


Figure 25: Shows the average earnings for each algorithm over the three relevant years for the realistic scenario.

Discussion

As can be seen in figure 25 then the GA algorithm has a larger profit than the other algorithms. This is somewhat surprising considering the performance of the GA in the naive case. Parts of the reason is that due to shorter runtimes for the GA in the realistic scenario then a larger *PopSize*, the amount of best selected genomes chosen and a longer threshold for stopping could be chosen. They were sett to a 100, 10 and 10 respectively in the realistic scenario. As these are made larger the GA will for most of the time converge towards the optimal solution which is why it outperforms the other algorithms as these are increased. The reason why these could be made larger for the realistic case than the naive case is most likely due to the GA saving much time on the no-profit days. This is because once it figures out it should do nothing in these periods, then the algorithm quickly stops.

The variation in profits of the simple algorithm, the mean-DDQL and STD-DDQL for the year 2019 is of considerable interest. The variation that is apparent for 2019 in the realistic case was not there for the naive case. Additionally it is larger than for the year 2020 and 2021 in both the realistic and naive case. This suggest that the year 2019 is different from the other years, and that it is this difference that causes this performance to vary from the other years in each of the scenarios.

When reviewing the average percentage difference between the peak and the valley, found in the naive case, then the difference for the year 2019 is half as that of 2020 and 2021. This would suggest that there could potentially be many no-profit periods for this year and that these periods could be what causes the difference in behaviour between the year 2019, and 2020 and 2021. To check this the amount of no-profit days for each year was found.

No-profit regions

Under the constraints set in the realistic scenario the amount of no-profit days can be seen in table 4. From the table one can see that the year 2019 is the year that has the most no-profit days, with 28% of its days being no-profit, which is approximately twice that of 2020 and 2021. Additionally one can see that 2020 has almost 30% more no-profit days than 2021.

Year	No-profit days
2019	101
2020	54
2021	41

Table 4: An overview over the amount of no-profit days for each year under the constraints set in the realistic scenario.

Discussion

The amount of no-profit days appear to follow the same pattern as the average percentage difference between the peaks and valleys. As the no-profit days amounts to over a quarter of the yearly days of 2019 then it is reasonable to think that they will have an effect on the variations in profit for the different algorithms. That the simple algorithm, mean-DDQL and

STD-DDQL all handles the input windows differently speaks in favour of the no-profit regions strongly affecting the variations seen for 2019. The simple algorithm for instance cares only whether the current price is above/bellow the average while the STD-DDQL cares more about what the shape of the price curve is. In total this makes it seem unlikely that they will handle the no-profit regions in the same way.

Still it is worth considering how much the amount of no-profit days actually affects the algorithms. The effect of having more no-profit days could for instance be that the overall profit for each algorithm would decrease without the amount of variation between the algorithms changing to any considerable degree. Additionally there is a limit to how different the actions that the algorithms can take most likely will be. As briefly discussed in the "Strategies for optimization"-section, then parameters such as C-rate could influence which strategies that are optimal. This means that for certain C-rate values the mean-DDQL and the STD-DDQL could end up picking the same strategy for making profit, as they have the ability to themselves decide for a strategy, unlike the simple algorithm. The performance of the simple algorithm and the mean-DDQL and STD-DDQL was therefore compared to see what effect the C-rate had on the profit that each algorithm managed to make.

Performance of algorithms across EV types

The following subsections will show comparisons of the performance for the simple algorithm, the mean-DDQL and the STD-DDQL for the realistic scenario for each individual year. The GA will be omitted in these comparisons as it performed the best for all the EVs across all the years in the realistic scenario and so is not so interesting to consider when reviewing the variation in the other algorithms. Additionally the GA was not tested for several window sizes and so reviewing it as a boxplot does not make sense.

The comparisons was done both in terms of their ability to make a profit, but also shows the variation in profit due to the choice of window size. The subsections will consist of boxplots that shows the variations in profit that was due to the window size for each algorithm. Such boxplots were made for each year and so each year will be presented with a general description of what that can be seen in the boxplots. They will be presented by increasing order. Finally when all the figures have been shown and described then there follows a discussion about these results.

The description that all the figures have in common will be shared here to make the figure description shorter for each one: For each figure then a consistent coloring scheme is used, where blue is the color of the simple algorithm, orange is the color of the mean-DDQL and green is the color of the STD-DDQL. The boxplots are grouped first by which car that was optimized for and then by which algorithm that did the optimization. This was done to simplify the comparison across the different cars.

All the boxplots has: A star, that symbolizes the score due to using a window size of 12. A triangle that points downward, that represents the maximum score of all the algorithms using a window size less than 12. The upwards pointing triangle symbolizes the minimum score of all the scores for algorithms using a window size above 12. The reason for using these symbols is to show more clearly how increasing the window size will affect the profit of the simple algorithm and the mean-DDQL/STD-DDQL. It also shows how the small window size profits are compared to the large window sizes. Additionally as the algorithms would not be able to use a window size larger than 12 then it is interesting to review how well the algorithm would be able to do on such a window size or smaller.

In addition to the symbols on the boxplots then the window size that yielded the largest profit is shown above each boxplot as numbered between 2 to 23 and has the same color as the algorithm and the boxplot it belongs to.

2019

for all the Nissan Leafs then the mean-DDQL has the highest profit of the three. This is followed by the STD-DDQL and then with the simple algorithm having the lowest profit across all the EVs in 2019 (figure 26).

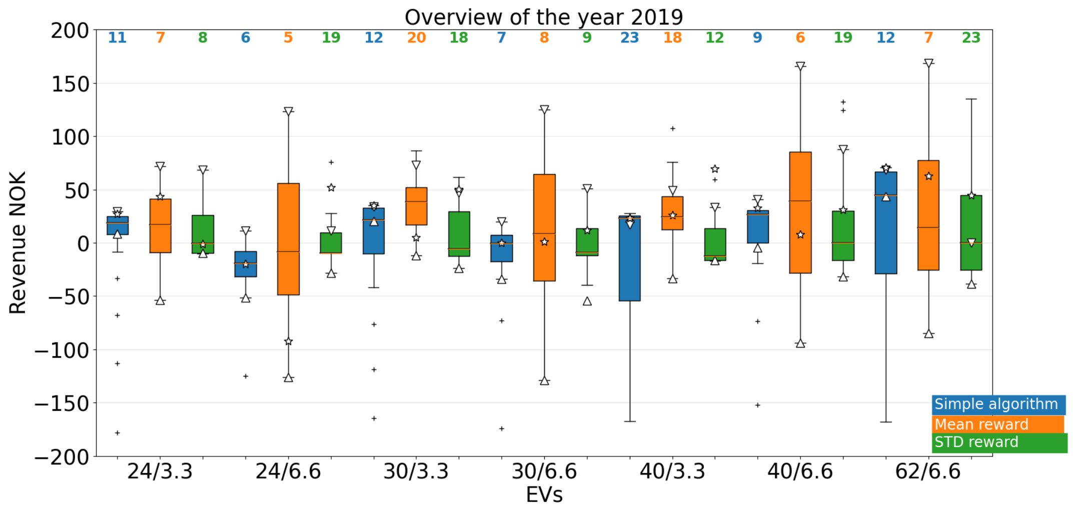


Figure 26: Boxplots showing the profit for the simple algorithm and DDQL with mean and STD based reward for the year 2019. It can be seen that the mean-DDQL generally has the largest variation in its profit for the C-rates above 0.15 of all the algorithms. Additionally the mean-DDQL vary from those EVs with a C-rate bellow 0.15 and those above.

From figure 26 it can be seen that the simple algorithm and the mean-DDQL used a smaller window size for cars with C-rates larger than 0.15, than those smaller than 0.15. Examples of this is the 24kWh/6.6kW vs the 24kWh/3.3kW EV. One exception from this pattern is the car 24/3.3 vs the 62/6.6 for the mean-DDQL. There the best performing model uses a window size smaller for the 62/6.6 than that of the 30/6.6 Leaf, even though the 30/6.6 has a larger C-rate than the 62/6.6. It can furthermore be seen that the simple algorithm and the mean-DDQL have the profit that varies the most with window size. For the mean-DDQL then the cars with higher C-rates than 0.15 had more variation in the profit due to window size than the cars with lower C-rates. This is with the notable exception of 62/6.6 that has a larger variation than the 30/3.3 which has the same C-rate, but less variations in the profit. Additionally, the mean-DDQL has more variation in its profit due to the window size than the STD-DDQL.

As it is a clear difference in behaviour that appear to depend on the C-rate, it was decided that the EVs would be categorized into two groups, the high C-rate which is all the EVs with a C-rate equal to or larger than 0.15 and low C-rate which is all the EVs with a C-rate bellow 0.15. Generally this means that all Nissan Leaf models with a max accepted charge rate of 6.6 kW, excluding the 62kWh Nissan leaf, will be considered as high C-rate EVs while all the others will be considered low C-rate EVs.

2020

The boxplots that compares the profit of the simple algorithm and the mean-DDQL/STD-DDQL in 2020 can be seen in figure 27. In the figure increasing battery size and charge rate appear to lead to increasing profits for all algorithms. Additionally the best window sizes for the mean-DDQL is generally higher than for the simple algorithm and STD-DDQL. The simple algorithm and the mean-reward DDQL also have smaller window sizes for the best performing models for the high C-rate EVs than for the low C-rate ones. The simple algorithm has less variation than the mean-/STD-DDQL due to the window size.

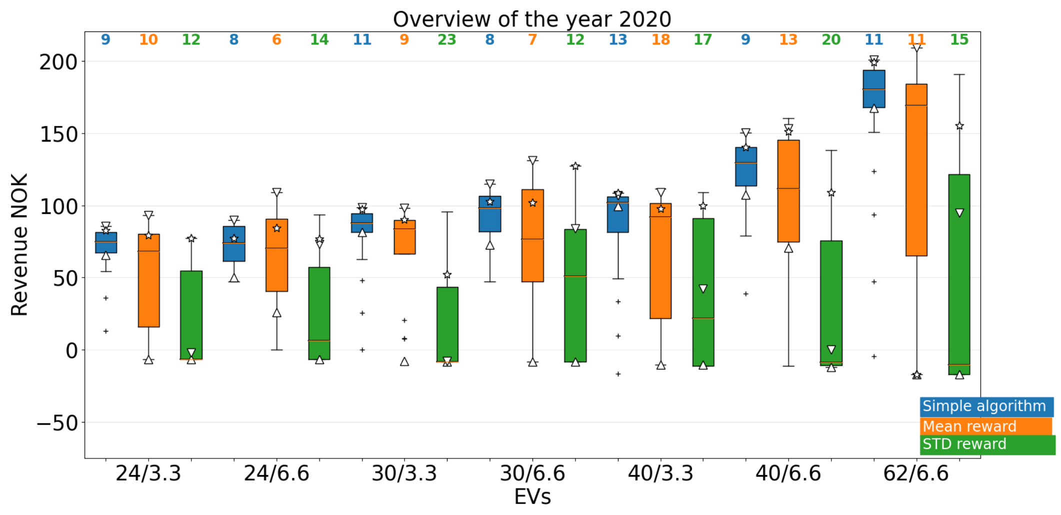


Figure 27: Boxplots showing the profit for the simple algorithm and DDQL with mean and STD based reward for the year 2020. A pattern of increasing profits due to higher battery capacity can be seen.

Generally the profit of the best performing algorithms is closer to one another across all EVs for 2020 than in 2019. Additionally the worst-performing window sizes for each algorithm for all Leaf models have a higher score in 2020 than in 2019. It can also be seen that most of the window sizes for the mean-DDQL leads to profits closer to the best score than for the STD-DDQL that has most of its window sizes performing closer to the worst performing window.

2021

In figure 28 it can be seen that the largest profit for each EV was close to the same across every algorithm. Additionally which algorithm that has the largest spread in its profit is varying from algorithm to algorithm, with no apparent pattern as to which one that has the largest spread. It can also be seen that all the high C-rate EVs resulted in smaller window size for all algorithms compared to the low C-rate EVs. In addition to this then the higher C-rate appear to have a larger variation in profit compared with window size, independent of algorithm.

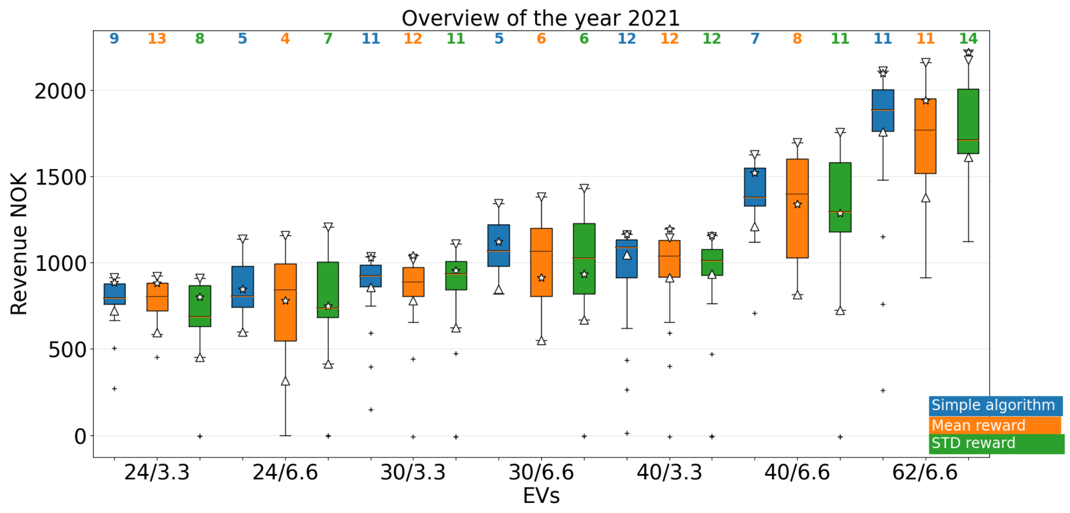


Figure 28: Boxplots showing the profit for the simple algorithm and DDQL with mean and STD based reward for the year 2021.

Discussion

When comparing figure 26, 27 and 28 it is apparent that the profits for each algorithm vary more from one EV to the next in 2019 than in 2020 and 2021. The difference across the years further strengthens the case that this variation is due to no-profit periods. Additionally for 2019 the mean-DDQL has the greatest spread in all its profits due to the window size, yet it performed the best for all Leafs in 2019. This indicates that the mean-DDQL is sensitive to the window size that it receives, but that it will make good decisions once a proper window size has been chosen.

The mean-DDQL spread in profit is also very pronounced for the high C-rate EVs, compared to the low C-rate EVs for 2019. It could be argued that this difference comes purely from the high C-rate Leafs using a charging rate of 6.6 kW. This could artificially make the variation appear larger from the low C-rate to the high C-rate Leafs. An example of this can be seen for the 62/6.6 Leaf that appear to have a much larger variation for all the windows tested than the 30/3.3 Leaf that has the same C-rate. As the 62/6.6 has a 6.6 kW charging rate it can charge/discharge twice the amount of its battery as the 30/3.3 Leaf. This makes the 62/6.6 EV able to charge more energy during a valley and discharge more at one of the peaks, resulting in a larger profit. As they have the same C-rate then in the case of the mean-DDQL they should not be any different. This because the training and testing of the DDQL-algorithms only works with SoC and C-rates rather than the actual battery capacity and charge rate and the total profit is first found after the training and testing.

A counter argument to that the difference is solely due to a larger charging rate can be seen when comparing 2019 with the year 2020 and 2021. There the difference in between the different charging rates are at the most around 50% larger (look at the 40/3.3 compared with the 40/6.6 for 2021), while for 2019 it is approximately 100% larger(look at the 24/3.3 compared with the 24/6.6). This difference does not seem like it can be attributed to a difference in average electricity cost levels as the year 2021 has a higher average price compared to 2019, while still having a smaller difference. In addition to this it can also be seen that the difference in profit for the best window is similar across each algorithm in the year 2020 and 2021, while they have different average prices. Still the difference across the high and low C-rate EVs for each algorithm are due to the difference in C-rate, but the for the year 2019 a factor seems to increase this difference. This strengthens the idea that the variations seen in the year 2019 is due to the amount of no-profit days.

Of course there are some additional arguments against this idea. One such argument is that it is not just the amount of no-profit days that determines the possible profits that can be made, but also how the shape of the price curve is within each day, and to what extent the algorithms can use this to make a profit. It could be that the difference seen in profits for 2019 is due to 2019 having a unique price curve, such that one needs a very special strategy to make any profit during a day. For instance it could be that even though the no-profit days are numerous, the loss they incur is small compared to their overall profit. Before this could be tested then it had to be studied how the mean-DDQL and STD-DDQL responded to the no-profit periods. It could for instance be that neither of them performed any charging or discharging operations in the no-profit regions, so that none of them actually were affected by the no-profit periods. If this was the case then it would have to be reconsidered what that caused the variations in profits seen in between the mean-DDQL and the STD-DDQL in 2019 for the realistic case.

Decision making

In the following figures samples of the decision making performed by the mean-DDQL and STD-DDQL, using the best performing window size, will be shown. The periods was picked out as representative samples for the remaining years, and shows the general behaviour of the the mean-DDQL and STD-DDQL. Only the behaviour of the DDQL algorithms are shown, but plots for the GA, simple algorithm and some additional plots for the mean-DDQL and STD-DDQL can be found in the appendix. A sample from the 24 kWh and 6.6kW EV was chosen as representative samples for the high C-rate EVs, while the sample from the 30 kWh 3.3 kW EV was chosen as representative samples for low C-rate EVs.

First the high C-rate plots will be shown followed by the low C-rate. The decision making plots are ordered as follows: The cyan bars shows the actions taken, if the bar is downward then it means that the EV is charging and if it points upwards then it means that it is discharging. The blue graph shows the price that has been standardized so that it will fit within the plot. The magenta graph sows the SoC of the EV, where a number of 0.8 equals 80% SoC and a number of 0.2 equals a SoC of 20%. The red transparent regions in the plots represents regions where no profits can be made. These are regions that all the algorithms ought to avoid charging or discharging in.

Decision-making for high C-rate EVs

When comparing the decisions for the mean-DDQL(figure 29) and STD-DDQL (figure 30) it can be seen that the mean-DDQL performs no discharging actions in the no-profit periods, that can be seen as the red regions. It should be noted that it wants to perform charging actions, but is not able to do this as the battery is already full. This pattern of the mean-DDQL not perform discharging actions in the no-profit regions is a mostly common pattern across the year 2019, 2020 and 2021. The STD-DDQL on the other hand performs charging and discharging in the no-profit periods (figure 30) and this is a general pattern across the year 2019, 2020 and 2021.

Discussion

Both the mean-DDQL and the STD-DDQL appear to generally make good charging and discharging decisions. They choose to charge at the valleys and discharge at the peaks, which is a good strategy for the shape of the price curve. Still they occasionally make sub-optimal decisions.

The mean-DDQL for instance does for the date 2019-01-15 performs a charging operation when it would have been better to wait to charge until 2019-01.16. Additionally it does not perform a full discharging operation at the peak of 2019-01-16, but it performs it in steps, where each discharge is at one of the peaks. In this case the STD makes a better choice and discharges mainly at the first and larger peak. On the other hand the STD-DDQL makes a worse decision than the mean-DDQL in that it decides to discharge all its battery on the first

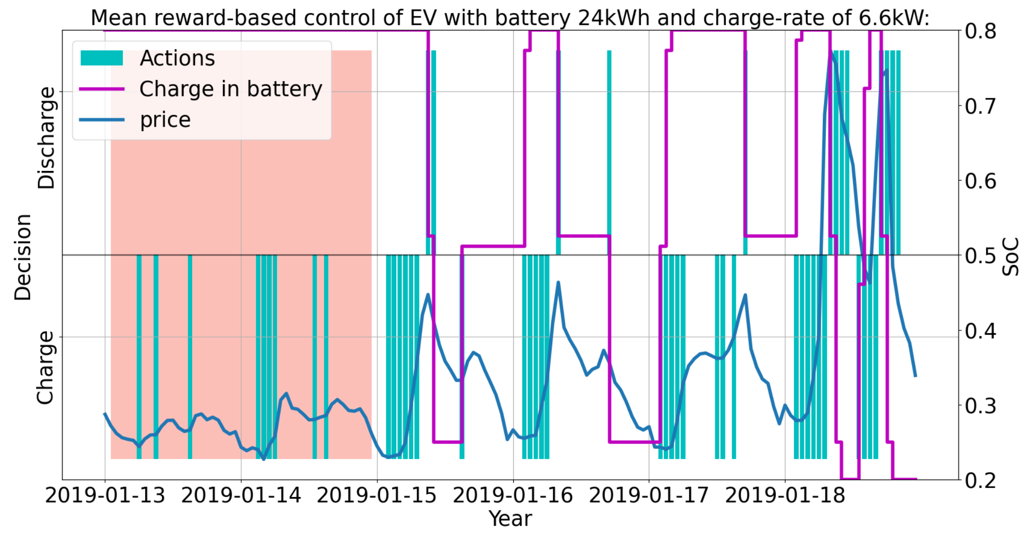


Figure 29: This figure shows the actions taken by the mean-DDQL in the period 2019-01-13 to 2019-01-18 for the EV having a battery of 24 kWh and a charging rate of 6.6 kW.

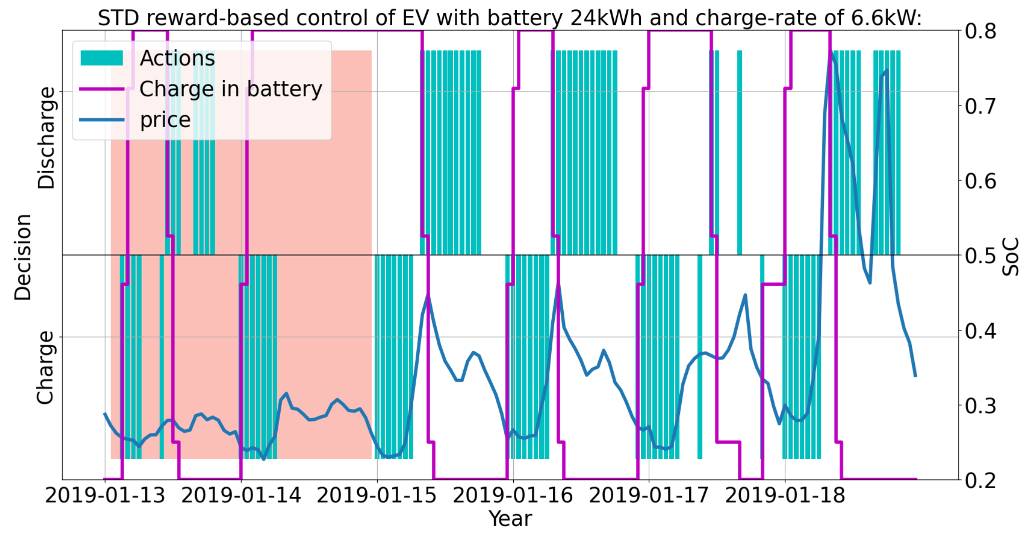


Figure 30: This figure shows the actions taken by the STD-DDQL in the period 2019-01-13 to 2019-01-18 for the EV having a battery of 24 kWh and a charging rate of 6.6 kW.

peak during 2019-01-17.

The mean-DDQL appear to make its V2G-decisions based on how long into the future the prices are growing or decreasing for the high C-rate EVs. The logic behind the charging/discharging appear to be the following: If the prices are decreasing in the whole window, then it is a peak, and it will discharge. Oppositely if they are increasing in the whole window then it is a valley and the mean-DDQL will charge. This could explain why the mean-DDQL discharges for the date 2019-01-16 (figure 29) in two steps, even though it would be more optimal to discharge on the first peak. This is because after it has discharged at the first peak then it saw that the prices were rising at the bottom of the window. This leads it to believe that another peak might be coming and so it stopped discharging. Then when it discovered that the graph was actually decreasing after the second peak it decided to discharge what it had left of its battery at the second peak. This is interesting as the mean-DDQL seemed to have adopted by itself the strategy proposed earlier that the EVs with a high C-rate should only discharge at the peaks and charge at the valleys.

Similarly it appears from figure 30 that the STD for high C-rate EVs tries to do charging at the peaks and discharging at the valleys. This sometimes leads to the STD-DDQL selecting good peaks and valleys, such as in 30 where it manages to hit the valleys and peaks of the date 2019-01-15 up to 2019-01-17 accurately. While it at other times leads to sub-optimal decisions such as on the second peak on the date 2019-01-17 where it discharges the whole battery on the first peak. This seems to be more of an occasional error since the STD-DDQL otherwise selects the peaks for charging and the valleys for discharging. Still it should be noted that the periods where the STD-DDQL wants to perform actions are larger than the ones for the mean-DDQL which indicates that the STD-DDQL has not understood as good as the mean-DDQL what that constitutes a peak. Nevertheless the STD-DDQL appear to do a good job at performing charging and discharging on the peaks and valleys, but with the most major disadvantage that it will be inclined to perform actions in no-profit regions.

From this it appears as if the mean-DDQL and the STD-DDQL have a similar performance for the normal periods, but where the main difference is that the mean-DDQL handles the no-profit periods better than the STD-DDQL. This further strengthens the case that it is the no-profit periods that causes these variations in performance.

Decision-making for low C-rate EVs

The mean-DDQL performs more actions in the no-profit regions (a sample is shown of this in figure 31) than it did in the same periods for the high C-rate Leaf. In terms of the profit regions it makes the same amount of full charging and discharging decisions, only using more steps with the low C-rate than with the high C-rate Leafs.

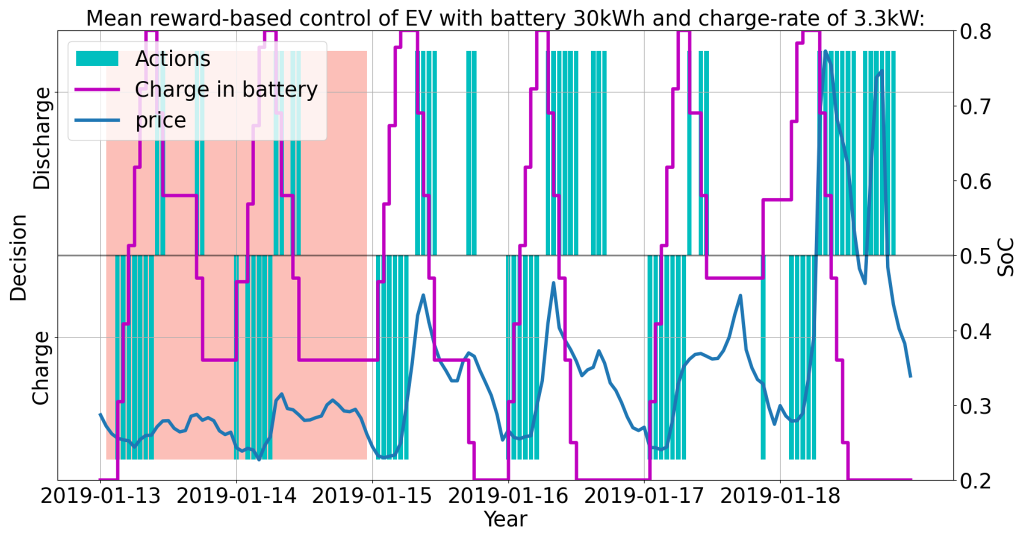


Figure 31: This figure shows the actions taken by the mean-DDQL in the period 2019-01-13 to 2019-01-18 for the EV having a battery of 30 kWh and a charging rate of 3.3 kW.

Oppositely the STD-DDQL occasionally avoids discharging in the no-profit periods, such as in figure 32. Still the behaviour is not as consistent as the mean-DDQL were for the high C-rate. There were seen several cases where the STD-DDQL would charge and discharge during the no-profit periods. A sample of this can be seen in the appendix.

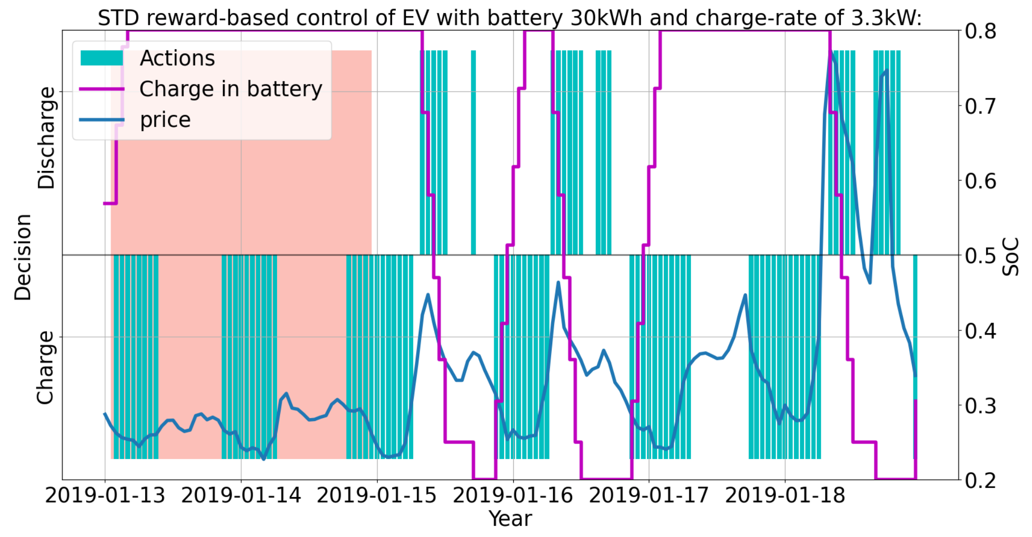


Figure 32: This figure shows the actions taken by the STD-DDQL in the period 2019-01-13 to 2019-01-18 for the EV having a battery of 30 kWh and a charging rate of 3.3 kW.

Discussion

The way both the mean-DDQL and STD-DDQL operates changes when optimizing for low C-rate EVs, as can be seen in figure 31 and 32. The STD-DDQL started to exhibit a behaviour more similar to that of the mean-DDQL for the high C-rate EVs. It would start to avoid the no-profit periods, and generally perform less charging/discharging operations. A good example of this can be seen in figure 32 where it for the date 2019-01-17 decided to skip a peak for then to discharge at the one the next day. Additionally it decides to charge during 2019-01-17 for then to save it for the peak at 2019-01-18, thus showing that it can be able to make long term decisions. In that case it would have earned a larger profit if it had discharged at the peak in 2019-01-17 and charged in the valley of 2019-01-18 though.

Despite the mean-DDQL performing more actions in the no-profit periods it still manages to perform better than the STD-DDQL for the year 2019 (figure 26). It could be that this is because the mean-DDQL on average performed more frequent, and/or more profitable actions in the profitable periods, and so performed better than the STD-DDQL. It could also be that the periods where the STD-DDQL performed both charging and discharging actions in the no-profit periods were the periods with the largest potential loss, and that performing actions in the other no-profit regions did not entail any major change in profits.

For the STD-DDQL it is not clear why it decided to avoid the no-profit regions for the low C-rate EVs while charging and discharging in the same periods for the EVs with a high C-rate. This could be linked to the way that the STD-DDQL is rewarded, as will be further discussed in the ”Properties of the DDQL algorithm” subsection. It could be that in periods without two clearly defined peaks then it believes it will not be profitable to discharge. Such a logic would make some sense when considering figure 32. The reason why this behaviour has changed from the high to the low C-rate could be because a lower C-rate would imply that the charging and discharging decisions have to be made over a longer period of time. This is unlike the high C-rate where it is more important to match the peaks with the valleys in the price curve. This would mean that for the STD-DDQL then it would have to look for good charging/discharging regions rather than good charging and discharging points. This could lead to the behaviour currently seen.

Discussion of the decision making

It appear that in the high C-rate case then the mean-DDQL was better at avoiding the no-profit periods than the STD-DDQL. This could be what lead it to have a considerable better performance than the STD-DDQL. For low C-rate EVs then this was somewhat opposite, with the STD-DDQL occasionally preferring to avoid the no-profit regions, while the mean-DDQL mostly did not. Despite this the mean-DDQL managed to perform better than the STD-DDQL.

As these are somewhat opposing results it questions the impact of the no-profit days for overall profitability for periods such as weeks or months. Suppose that the potential loss of charging at a valley, and discharging during a peak in a no-profit day is far lower than the potential profit of the other days. In that case then avoiding the no-profit days is of little importance, and so an algorithm can to some extent ignore the no-profit days of a week. Still the fact that the mean-DDQL for the high C-rate EVs mostly avoids the no-profit periods begs the question of how much doing V2G in the no-profit periods affect the overall profitability.

Effect of no-profit periods

It was therefore tested for each week during the year 2019, 2020 and 2021 how many days of no-profit that there were during that particular week. Then the percentage of absolute profit was calculated for each week.

Afterwards it was plotted how much the difference were between the profits earned by the mean-DDQL and STD-DDQL for the given week. This was done so that it could better be seen what impact the no-profit days had on the profit-making ability of each of the DDQL algorithms. The comparison of the variations in profit was plotted in two separate plots, one for the high C-rate and one for the low C-rate. The resulting figures can be seen in figure 33, 41 and 42. Although only figure 33 is shown here and the remaining ones are a part of the appendix.

For the plots comparing mean-DDQL and STD-DDQL then if the graph is above the 0 then the profit difference is in favour of the mean-DDQL, while if it is bellow zero then it is in favour of the STD-DDQL. This means that the higher the graph was in either positive or negative direction, the larger the profit was in favour of either algorithm. The height of the graph is given in the units NOK/MWh to simplify the comparisons across the Leafs. This means that the height of the graph is not given in percentage difference, but as the actual difference in profit, and so larger peaks means larger differences in profitability. The plot bellow shows a comparison of the low C-rate EVs and follows the same structure as the plot for the high C-rate EVs. The figures are shown bellow:

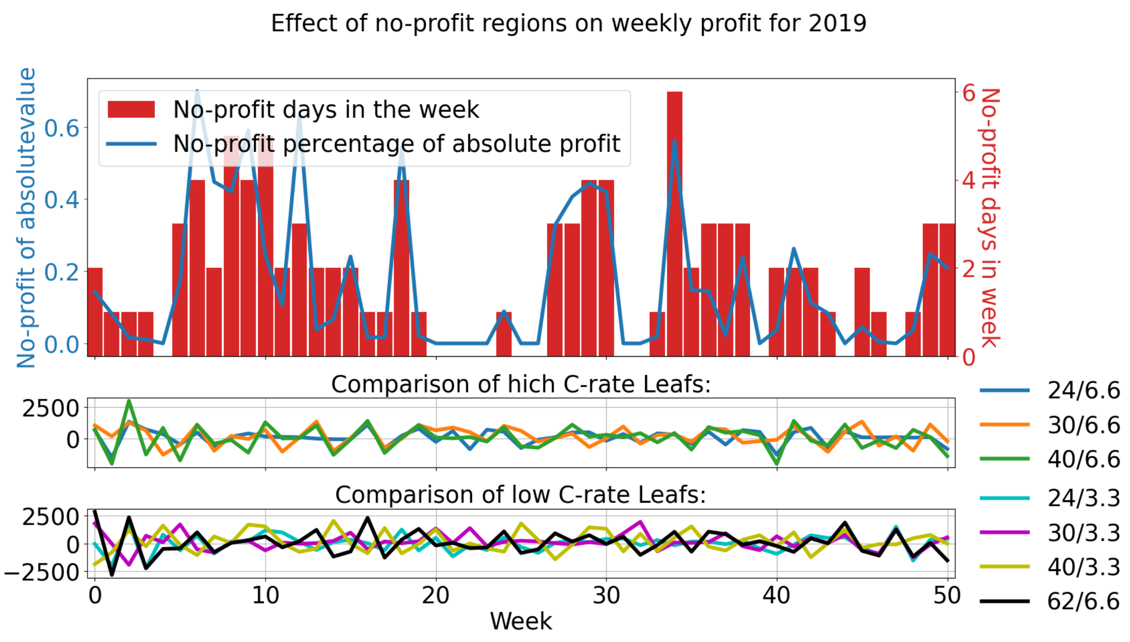


Figure 33: The uppermost plot shows the number of days with no-profit within the given week as red bars. The blue graphs in this plot shows how much of the total value of a week that the losses due to V2G in no-profit regions amounts to. Bellow is a graph that shows the difference between the profit of the mean-DDQL and the STD-DDQL for the high C-rate Leafs. The graph at the bottom shows a the difference between the profit of the mean-DDQL and the STD-DDQL for the low C-rate Leafs. Which leaf that equals what graph can be seen to the right in the figure.

One can see from the upper-most plot that for most of the weeks then an increasing number of no-profit days leads to the losses due to performing actions in the no-profit periods increasing too. This is not the case for all the weeks as the week 10 has a lower loss than week 9, even though week 10 has more no-profit days. Furthermore it can be seen that for several weeks, such as week 12 and 18, the no-profit days lead to a larger loss than the profit made during profit-days. This can be seen as the blue line is above 0.5 on the axis to the left.

Discussion

The graphs for the high C-rate leafs appear to have peaks at the major spikes in the percentage of absolute profit, such as for week 6 and 34. Still for most of the peaks then the effect is negated by being followed by a valley. This could indicate that the high C-rate does not actually perform better in this region. It could also be that these peaks and valleys are due to for instance the STD-DDQL charging at the end of one week, causing that week to have a smaller profit, for then to sell it at the next week, leading to the next week having a larger profit. This could explain why certain regions (week 10-20) are having cyclic peaks and valleys. Still it can be seen that for week 25 - 50 there are several weeks where the mean-DDQL makes more profit than the STD-DDQL in the weeks were there is a larger loss relative to profit. Examples of this can be seen in week 25-29 and 33-34. Additionally for the 24/6.6 and 30/6.6 EVs then the mean-DDQL performs better in the weeks 8-9 where the possible loss is larger than the profit for the weeks. All in all this strongly suggests that the mean-DDQL has a better ability than the STD-DDQL to perform well in these no-profit periods and that these no-profit periods affect the overall profitability. This means that with more no-profit periods then the mean-DDQL will perform better than the STD-DDQL.

Still it seems like it is in the days from week 20-50 with few no-profit days that the mean-DDQL makes the most profit compared with the STD-DDQL. This is not always consistent though, as one of the periods where the mean-DDQL performs the best compared to the STD-DDQL is in the week 35-39 where the loss amounts to close to 50% of the profit of the same week. Additionally the periods where mean-DDQL makes a profit during the weeks 20-50 is mostly in the regions with no-profit days, so although the difference in profit is smaller for the weeks with many no-profit days, they are still so many that they together makes up a large difference. This further strengthens the idea that the no-profit regions are the cause of the difference in profit seen for the high C-rate EVs.

When it comes to the low C-rate EVs then they do not appear to have a clear pattern as with the high C-rate EVs. Still for most of the weeks they tend to have their peaks above 0 indicating that the reason why the mean-DDQL manages to perform better for 2019 is that it on average makes better charging and discharging decisions than the STD-DDQL.

General discussion

This section will first discuss the properties observed in the GA and simple algorithm. Afterwards there will be a subsection that discusses why the GA is a better choice than the other algorithms for V2G optimization, and why it is still worth to consider the properties of the DDQL for V2G optimization. A discussion will then follow that highlights the different behaviour of mean-DDQL and STD-DDQL that was observed and why mean-DDQL was found to be a more robust algorithm than the simple algorithm and STD-DDQL, particularly against no-profit days, and should be used before these two algorithms. Afterwards there is a subsection about issues encountered during the training of the DDQL-algorithms followed by a subsection discussing the ability of the DDQL algorithms to generalize to new data. Following up on this is a general discussion of the size of the profit for V2G and how costs, such as power tariffs, can be optimized for in the already existing models.

General properties observed in GA and the simple algorithm

It was observed that the runtime was a great limitation for the GA and by cutting down on the runtime it inevitably led it to perform worse than the other algorithms. Still in the realistic case it was seen that it performed better than the other algorithms. This can most likely be attributed to that the GA will converge towards the optimum solution given enough time and a proper stopping criterion. This could have led it to find a better solution, on average, than the other algorithms and thus perform better for the realistic case. Additionally it is not bound by its a chosen strategy as the other algorithm which gives it an edge when there is a large variation in the shape of the price curve.

The other algorithms does not have the property of convergence as the GA does, and so they will only perform as well as the V2G strategy that they choose. Suppose that the mean-DDQL chooses a strategy of only charging when the electricity price is at the lowest during a day, and then sell it back to the grid when it is at the most expensive. In this case it will only work well if the price curve has two clear peaks and otherwise a price close to the average, meaning that charging at the peaks and discharging at the valleys is the only way to make a profit. If the price curve has many smaller peaks and valleys on the other hand then this is not a good strategy. This is because more profit could potentially be made by trying to charge at all of these valleys and discharge at all the peaks, if the difference between the peaks and valleys were large enough. In essence this means that the mean-DDQL, STD-DDQL and simple algorithm is prone to only perform well on the data that fits the strategy they have chosen.

As the strategy of the simple algorithm is very general it is understandable that it is doing the worst of the four in the realistic case. This because its logic of always charging when the price goes bellow average and discharging when the price goes above average results in it making many decision that do not result in long term earnings. It can for instance end up depleting its battery right before a peak comes, simply because the points right before the peak is also above the mean.

This issue means that the simple algorithm will be prone to perform badly in regions where the distance between a local peak and an even higher peak is just above the window size, such that the algorithm will discharge itself, directly followed by a charging procedure at the same price as it just discharged at. The most important issue with the simple algorithm though is that due to its frequent charging and discharging then it will perform badly when the price curve is such that there are few combinations of peaks and valleys that results in a positive profit. Despite all its drawbacks it performs comparatively close to that of the other DDQL algorithms for the year 2020 and 2021, showing that in the years with fewer no-profit periods then the simple approach could be used. This falls in line with the results from the naive scenario as with there being less no-profit periods then the realistic scenario starts to resemble more the naive scenario where there were little differences in between the simple and DDQL algorithms.

DDQL vs GA

The GA is consistently better (figure 25) than the other algorithms in the realistic scenario. Still, it has some notable drawbacks, the most important one being long run-times. For optimizing the charging and discharging patterns for each EV the GA used all from 1.5 hours to 12 hours to complete the calculations for one full year for one EV. This time depended on what the stopping condition was set to and other parameters, such as *PopSize* and the rate of mutation. Still a possible runtime of 12 hours is much longer than the DDQL that used 1.5 hours to train itself on 22 different window sizes for one reward function. The DDQLs has the added benefit that if they are trained with representative training sets using optimal parameters and reward-functions then most of the time that it spends is tied to training. Once it has trained then it can be used to make much quicker decisions on new unseen data, a benefit the GA does not have. This is again one of the major drawbacks of GA, i.e. that when it optimizes then it does not learn anything about the new data and so it has to start from scratch each time it receives new data that it shall optimize for.

In the scenario of a 24 hour optimization then the time-usage of the GA is not an issue. Even if the GA uses a full 12 hours to optimize for a whole year, then this equals to merely 12h/365days per year = 1.97 minutes on average per calculation for optimizing for the next 24 hours. This is not long when one wants to optimize for the next day, and so the short runtime of the DDQL is not important. This makes the GA appear as the most appealing algorithm to use for active power support for a full day optimization. Still this ability of the DDQL to make quick, and generally accurate, decisions relative to the GA makes it interesting to review. This because it can have its use in real-time regulation, such as frequency regulation, where spending 1.97 minutes on average for deciding the actions for the next 24 seconds will be too long. The possibility that the behaviour seen in the DDQL when optimizing for the price data could bear resemblance to that of the short time-frame predictions is why the properties of the DDQL algorithms was reviewed and is dedicated a considerable amount of space in the results section and this discussion section.

Properties of the DDQL algorithms

Reviewing the properties of the DDQL then the difference in between the mean-DDQL and STD-DDQL in robustness to no-profit periods stems from what decisions the different reward functions promote. The mean-DDQL will generally reward prices far away rather than those close to the mean, while the STD-DDQL will mainly reward any decision that is taken at a local extreme point. For the mean-DDQL the algorithm will decide to do nothing in periods when an action will result in a reward close to the mean. The STD-DDQL on the other hand does not have this capability as the mean-DDQL to compare the absolute distance from the mean, and so it cannot see or reward the potential profit to be made. The reason for this is that all its input prices are standardized, and so they will not contain any information as to how far away from the average price that the prices lie, only how far away from the mean they are in terms of the standard deviation. This results in it mainly rewarding any action taken in a peak or valley, thus it will frequently discharge in low-value periods, as can be seen as the frequent charging and discharging.

The behaviour so far described highlights some of the drawbacks and perks with the mean-DDQL and STD-DDQL. The mean-DDQL has the perk of being robust to no-profit regions for high C-rate EVs and to value large rewards over smaller ones. This is also a possible drawback, as what that is far from the mean can be different from year to year. Suppose that the algorithm is trained on data where the difference from the mean is large relative to other average years. Then the mean-DDQL might ignore profit regions, because it considers the profit regions to be no-profit based on its previous training data. This can be amended by mainly training on data where the peaks are lower, such that it will learn what the lower limit for the profitability for an action is. A downside of such a training strategy is that the mean-DDQL might prioritize to only perform actions right above the mean, or that it might learn some heuristic that will not prioritize performing any action at peaks/valleys higher/deeper than the peaks that it has previously encountered. Another possible issue with the mean-DDQL is that it will learn less from smaller rewards which means that it could be prone to only optimize for the larger rewards. On the other hand then this could also be considered a perk as it would result in fewer actions taken by the mean-DDQL which means less degradation for the EV battery.

In regards to the STD-DDQL then one of its major perks is that it will be effective in finding peaks and that this behaviour should be easy to generalize to new data, as the input data is standardized. This ability to generalize will most likely only apply to new price data with a similar shape on its price curve to the price data that the STD-DDQL has trained on. The drawback is of course, as shown earlier, that it is less robust to no-profit regions as it will then try to charge during the peaks and discharge during the valleys even though no profits can be made.

The properties of the mean-DDQL and STD-DDQL makes it so that for years with many no-profit periods then the mean-DDQL is the better choice, while for years with very few no-profit periods then the mean-DDQL is perhaps to be preferred. Still as the difference between the algorithms was relatively small in the years with few no-profit days (2020 and 2021), but large for the year with many (2021), then it makes a strong argument for using the mean-DDQL rather than the STD-DDQL for all years.

Generalization to other years

It has not been tested how well the algorithm for each year generalized to other years than the one they were trained on. Reviewing the profit for each chosen window sizes for each algorithm across the different years might yield some insights into how smooth such a transition would go though.

It can be seen from the figure 26, 27 and 28 that for the mean-DDQL then the lower the C-rate then the larger the window size that is required. Generally it can be seen that for the mean-DDQL the variation in window size for the best model for one type of EV was approximately the median value of the best window sizes ± 2 for low C-rate EVs. For the high C-rate EVs then there is a variation of the median ± 1.

This implies that the mean-DDQL should generalize well from one year to the next for high

C-rate EVs. Of course it is not guaranteed that it will perform well. This because its ability to generalize also depends upon what sort of structures it manages to learn from the training data and to what extent these structures are common across the different years. Still if the algorithm prefers the same window size across the different years then it is a good indication that there is a general strategy for optimizing for price that is common across the years that the algorithm could learn.

Moving on to the mean-DDQL used on a car with a low C-rate then it is more uncertain as to how well the algorithm generalizes from one year to the next due to the variations in the optimal window size. A possible reason for why is that as the C-rate decreases then the mean-DDQL has to charge and discharge over a larger-region and pay more attention to the shape of the price-curve rather than just when it was a peak or a valley. This implies that it would require a larger window size to properly catch the shape of the curve to see in which regions that it is profitable to charge or discharge. As the shape of the curve can vary from year to year and from day to day then the window size needed to capture these shapes will also vary with which shapes that are the most prevalent. Suppose that one shape of the price curve could be fully captured with a window size of 10, while another one might require a total of 20 in window size. Then if the size 10 shape dominates one year, it would make more sense to use a shape 10 window, while if it the next year was a size 20 then using a window of size 20 would be better. This would mean that an algorithm trained on one year would have difficulties being used in another year.

In relation to the STD-DDQL then it is not a clear pattern in which window sizes it prefers as the best across the different cars and years, but it seems to generally prefer a window size above 12. This could be because it uses the shape of the price curve to detect peaks and valleys and so it needs a large window size to understand the shape of the price curve. Such a large window size would not be possible to use in the real world, unless some alterations are made to the price data as described in the methods section. Still the variations seen in STD-DDQL implies that the STD-DDQL could generalize worse than the mean-DDQL on new data, but this has not been tested and so such conclusions cannot be drawn.

It should be noted that as both of the algorithms only was trained on the first 1500 hourly price data out of a total of 8760. By training on these first 1500 hourly prices they still perform comparatively well compared to the GA, suggesting that they have the ability to generalize the new data.

Issues encountered

During this work then much time was bound up in trying to use different reward function and schemes for implementing the DDQL. As briefly discussed in the methods section then the reward functions as suggested by [36], [33], [34], [30], [82] and [35] were all tested, but not used as all these reward functions showed poor performance when compared with the simple algorithm. What was observed in the cases for [33], [34], [30] that used a reward scheme as suggested by Kiaee [36] was that the DDQL did not want to do any charging, it only wanted to do discharging or no action at all. The reward function that Kiaee suggested worked by providing a positive reward when discharging, equal to the price of the electricity, and a charge would result in a negative reward, equal to the price of electricity. As the DDQL algorithm used in this thesis only wanted to do discharging it seems like it saw that discharging was the only action that resulted in a reward and so it tried to avoid charging as it saw that it only resulted in a negative reward.

Kiaee and [33], [34], [30] have all added a term that potentially could solve this issue. This term was set up so as to punish the algorithm if it did not charge the battery up to a certain level at the end of each day. A similar scheme as this was tried in this thesis by summing up all the accumulated rewards during a day and provide that reward back to the algorithm at the end of the day. This did not result in the algorithm wanting to charge, and so the scheme of Kiaee was abandoned.

It was also attempted to use the reward as suggested by Dang [82]. Dang suggested, similar to Kiaee, to reward the algorithm when discharging by using the price directly as a reward, but if the EV was charging then it would be punished by rewarding it with the value 1/price. The score of the algorithm did not improve with this scheme either, and the DDQL algorithm used in this thesis still wanted to only perform discharging or to do nothing, suggesting that it still saw that discharging would bring the most profit.

Still the studies by [36], [33], [34], [30], [82] and [35] highlights the use-case of DDQL when there is much price uncertainty. Such as demand-response situations, where only the price of the next hour is known. This means that the DDQL might still find its use-case at Gardermoen if such a V2G control scheme is implemented.

In addition to the issues related to the reward functions in the literature then after the current reward functions were adopted it was discovered that the model is prone to the catastrophic decay, that is that the accuracy gets better to then suddenly decrease. This happens over time so when the peak score is reached then the training of the algorithm should be stopped. If not stopped then it would experience a sudden drop in profit, and this decrease in profit would continue to decrease over time and ending in a zero profit over time. This drop in profit could also be due to over-fitting. This issue with the profit dropping over time was why the tested scheme of Kiaee was not adopted. The scheme was to train on 1500 test samples, test for the next 500 samples, for then to move the test samples 500 samples on, such that the new 1500 training samples would contain the previous 500 test samples.

Economy

In the naive scenario it can be seen in figure 24 that on average the simple algorithm, the mean-DDQL and STD-DDQL earn around 600 NOK by doing V2G in 2019. When the EV is performing under more realistic conditions then the best performing GA earns on average acoss all EVs 200 NOK per year, with all the other algorithms earning bellow this. This is a relatively small price for potentially degrading the battery. Additionally considering that a Wallbox V2G charger costs approximately 62 000NOK [93] then V2G purely done to earn a profit based on the difference in price is difficult to justify.

The amount of profit earned during the year 2021 shows more promise for the use of V2G with around 1600NOK on average across all EVs over a year by using the GA for optimization. Still 2021 is considered to be an extreme year in terms of the energy prices in Norway, and even if the algorithm were to earn 1600 NOK each year then it would take 62 000 NOK / 1500 NOK/year = 41 years. Which is a long time to pay down the cost of the charger considering that the same amount of money could have been invested with a better interest rate. Still Aasbøe's Master's thesis from 2021 reported that the cost of the chargers are expected to drop to 10 000NOK by 2030 [21] reducing the time for it to pay itself back to 6.67 years.

What has not been considered when looking at the payback period time is the possible profit of power tariffs. Assuming a power tariff of 90NOK/kWh and an EV being able to shave off 6.6kWh per month off the monthly peak consumption, then one would see a saving of 594 NOK per month and a total of 7128 NOK per year. So assuming the extreme prices of 2021, charging under realistic conditions, the highest power tariff prices, and a charger purchased with the prices from 2030, then one is on a down-payment of a little bit above a year. This is of course making many assumptions about optimal conditions, and so the payback period would anyways be considerably longer.

All in all the cost of V2G purely for earning a net income does not seem appealing if one have to invest in the chargers due to the small price difference under the years other than 2021, further strengthening the point argued for in the thesis of Hoff [22] that V2G purely for making a profit due to the variations in electricity prices in Norway is a factor that can be ruled out when considering the profitability of V2G from parked cars.

As both Tamara and Hoff argues for the reduction in power tariffs being what might make V2G profitable for the EV owner the following alteration to the environment is suggested to make the proposed models able to learn how to minimize the power tariffs: If the power tariff is 90 NOK/kWh, then 90 NOK/kWh should be added on top of the prices of each hour where the power consumption is expected to rise above the current peak consumption of that month. If a new peak is reached sometimes during a month then this peak will become the current peak of the month. This ensures that only the hours that have the potential of becoming new peaks will be discharged at, avoiding unnecessary discharging and saving the battery. At the start of each month then the current power peak could be set as the average of the prices in the previous month. For this scheme to work there needs to be accurate predictions of the power consumption of Gardermoen, or the relevant building that one wants to reduce the peak power of, for the next 24 hours.

Optimizing for the price curve does not guarantee that one optimize for active power support. Generally the price curve is supposed to represent the combination of supply and demand, so a low price should mean low demand or much production. Still the electricity prices are set on a day-ahead basis, which means that the prices will be set independently of the actual power consumption on the grid during the day. The prices are still mostly representative of the daily consumption so optimizing for the price will generally lead to a lower peak power consumption over-all, but it does not guarantee it. This is why to fully optimize for active power support one should introduce a combination of a power tariff scheme and price curve optimization.

The coordination of the V2G in several EVs needs to be addressed if the algorithms used in this thesis is scaled so as to include several EVs. Suppose for instance that the EVs are optimized for the price curve and they see a valley in the price curve and all decides to charge all at once. This scenario could lead to a peak power consumption in the P10 parking garage, which is opposite of the goal of active power support. This issue needs to be resolved if this solution is to be scaled. One solution could be to optimize one EV at a time for the predicted power consumption. Then the predicted power consumption could be adjusted according to the decisions chosen by the EV, and passed on to the next EV for choosing its decisions.

Conclusion

In this master thesis it was studied how two variants of the reinforcement learning algorithm called DDQL would compare with a simple algorithm and a GA for optimizing V2G for EVs at the P10 parking garage at Oslo airport Gardermoen. It was found that the GA was better than the DDQL and the simple algorithm at making charging/discharging decisions for the 24 hours intervals. This means that for the use-case at Gardermoen then GA, or a similar algorithm, is more suited than the two DDQL-algorithms and the simple algorithm. Still if a V2G control scheme is implemented where the prices will vary from one hour to the next then the DDQL might become relevant for Gardermoen as the literature has shown.

It was found that the GA had long run-times, which was not an issue for optimizing for the next 24 hours, but was making it too slow for performing charging/discharging actions in a real-time-setting. This rendered DDQL as the preferred choice in such a situation. In a real-time scenario the DDQLs also have an advantage over the simple algorithm in being more resistant to no-profit periods.

In the process of making the DDQL capable of performing V2G then two DDQL models were made, called mean-DDQL and STD-DDQL. As the DDQL can be used in real-time situations, where quick decision-making is needed, then both of these DDQL-algorithms were reviewed to see how well each performed under different conditions. This was also done to fill the hole in the literature on how the DQL would perform when optimizing for V2G while having access to future data. They were tested in two different scenarios, one naive and one realistic scenario. It was found that for the realistic scenario when optimizing for an EV with a C-rate higher than 0.15 then the mean-DDQL was more robust towards no-profit regions than the STD-DDQL. For EVs with a lower C-rate than 0.15 then the mean-DDQL had a larger profit than the STD-DDQL for the year 2019, that had many no-profit days, but this difference in profit did not seem to be linked with the ability to avoid the no-profit days. This difference rather appeared to be linked with the mean-DDQL overall performing better charging/discharging decisions.

The STD-DDQL performed better in the year 2021 than the mean-DDQL, which was the year with the fewest no-profit periods. This implies that the STD-DDQL could perform better than the mean-DDQL in years with few no-profit regions. Still the average profits of STD-DDQL for all EVs was much more similar to that of the mean-DDQL in the year 2021 than in the year 2019. It was also found that the mean-DDQL for EVs with a C-rate higher than 0.15 most likely is better than the STD-DDQL at generalizing to new data, although this has not been tested for.

The mean-DDQL is to be preferred over the STD-DDQL in real-time situations because of its robustness to no-profit periods, and due to its comparatively similar profits in periods with few no-profit days to the STD-DDQL.

Finally in terms of the subgoal then it was seen that for the year 2019 and 2020 then the average profitability for doing just V2G using GA was bellow 200 kr for the EV owner. Which is small relative the potential savings from optimizing for power tariffs. For the year 2021 then the profit had risen to around 1500 NOK when using the GA for optimization, which makes V2G much more interesting. Still this was an exceptional year in terms on price, so the point argued for by Hoff that V2G purely for making a profit due to the variations in electricity prices in Norway is a factor that can be ruled out when considering the profitability of V2G from parked cars is further strengthened.

Future work

One cannot guarantee that real life V2G will be similar to those conditions tested for in this thesis. Some issues that might arise is that the RTE efficiencies vary a lot from car to car or that the maximum charge rate varies much in between a SoC of 20 and 80%. To be able to test this thesis in real life, make more accurate models and test the accuracy of the current models in a real life setting then more data is needed and a system to test V2G decisions on a real EV, As a part of this thesis then such a system was set up, and the basic concepts underlying this system can be found in the appendix. A further work of this master should then be to use this system that has been set up to gather the needed data and test the algorithms used in this thesis in a real life situation.

One reward function that partially worked for the DDQL, but not as good as the current reward functions, was the scheme by Brock et al. [35]. They used the change in the value of the energy stored in the battery as a reward function. This means that if the price was decreasing then the algorithm would receive a punishment equal to the decrease in price times the energy in the battery. If the energy price was increasing then oppositely the algorithm would get a reward. This algorithm was studied later in the thesis process, but could potentially perform better than the current DDQL-reward functions as its reward could be made less dependent on the window size. This is why a potential future work could be to see if this reward function could be adapted to make the DDQL perform better.

Additionally it could be checked if this problem could be transformed into a supervised learning problem. This could be done by letting the GA predict which actions to take and then train a supervised learning algorithm on the choices of the GA. This would be done to see if it can learn to make the same decisions just with much shorter decision-making time than the GA on new data.
It should also be tested over the same window sizes for previous years to ensure that the effects seen in 2019, 2020 and 2021 is consistent across the different years. Particularly the resistance of the mean-DDQL to no-profit periods.

It would also be interesting to see what effect it would have to use previous prices in the window provided to the DDQL algorithms to see if this would improve their performance. This could be useful for the algorithm so that it knows whether the current point that it is looking at was large/small compared to the previous prices and not just the future.

As already mentioned then it has not been studied how the algorithms generalize from one year to the next. This is thus something something that could be interesting to study, to see if the mean-DDQL actually makes good predictions on new years.

It has not been tested for, given the optimal window size, how much the DDQLs and the GA will vary in their profit when trained and tested several times. This is something that aught to be studied to see if this is something that would affect the results in this thesis.

References

[1] Filip Kristoff Ørn. *Algorithm testing and data analysis.* URL: https://gitlab.com/Raidg/algorithm-testing-and-data-analysis.

[2] Filip Kristoff Ørn. *Communication program for V2G.* URL: https://gitlab.com/Raidg/communication-program-for-v2g.

[3] IEA. *Electricity Market Report - January 2022.* URL: https://www.iea.org/reports/electricity-market-report-january-2022. (fetched: 16.04.2022).

[4] IEA (2020). *Global EV Outlook 2020.* URL: https://www.iea.org/reports/global-ev-outlook-2020.

[5] Samferdselsdepartementet. *Norge er elektrisk.* June 10, 2021. URL: https://www.regjeringen.no/no/tema/transport-og-kommunikasjon/veg_og_vegtrafikk/faktaartikler-vei-og-ts/norge-er-elektrisk/id2677481/.

[6] Vegard Holmefjord Anders Kringstad. *Et elektrisk Norge – fra fossilt til strøm.* URL: https://www.statnett.no/globalassets/for-aktorer-i-kraftsystemet/planer-og-analyser/et-elektrisk-norge--fra-fossilt-til-strom.pdf. (fetched: 03.11.2021).

[7] Statnett. *Increased power consumption and plans for new industry generates need for more power production.* URL: https://www.statnett.no/en/about-statnett/news-and-press-releases/news-archive-2021/increased-power-consumption-and-plans-for-new-industry-generates-need-for-more-power-production/.

[8] Ingrid Bye Løken. *Strømnettet i et fullelektrisk Norge.* URL: https://www.energinorge.no/contentassets/74f33e5598d64578bda89c1fa864e83a/rapport---stromnettet-i-et-fullelektrisk-norge.pdf. (fetched: 16.04.2022).

[9] Pöyry - Menon Economics. *Vurdering av atferdsvirkemidler som kan bidra til reduksjon av effekttopper.* URL: http://publikasjoner.nve.no/eksternrapport/2019/eksternrapport2019_03.pdf. (fetched: 03.11.2021).

[10] Statnett. *Kraftig forbruksvekst og industrialiseringsplaner gir behov for mer kraftproduksjon.* URL: https://www.statnett.no/om-statnett/nyheter-og-pressemeldinger/nyhetsarkiv-2021/kraftig-forbruksvekst-og-industrialiseringsplaner-gir-behov-for-mer-kraftproduksjon/. (fetched: 16.04.2022).

[11] Jonas Skaare Amundsen og Ingrid Bjørshol Holm Gudmund Bartnes. *Kraftmarkedsanalyse 2018 - 2030.* URL: https://publikasjoner.nve.no/rapport/2018/rapport2018_84.pdf. (fetched: 16.04.2022).

[12] IEA. *Renewable electricity growth is accelerating faster than ever worldwide, supporting the emergence of the new global energy economy.* URL: https://www.iea.org/news/renewable-electricity-growth-is-accelerating-faster-than-ever-worldwide-supporting-the-emergence-of-the-new-global-energy-economy.

[13] European comission. *Paris Agreement.* URL: https://ec.europa.eu/clima/eu-action/international-action-climate-change/climate-negotiations/paris-agreement_en. (fetched: 16.04.2022).

[14] Anders Kringstad. *Kraftmarkedsanalyse 2018 - 2030.* URL: https://www.statnett.no/globalassets/for-aktorer-i-kraftsystemet/planer-og-analyser/2018-Fleksibilitet-i-det-nordiske-kraftmarkedet-2018-2040. (fetched: 16.04.2022).

[15] Magnus Buvik Hallgeir Horne and Jarand Hole. *Smarte ladesystemer og Vehicle-to-Grid.* 2019. URL: https://publikasjoner.nve.no/faktaark/2019/faktaark2019_09.pdf.

[16] Koen van Heuveln et al. "Factors influencing consumer acceptance of vehicle-to-grid by electric vehicle drivers in the Netherlands". In: *Travel Behaviour and Society* 24 (2021), pp. 34–45. ISSN: 2214-367X. DOI: https://doi.org/10.1016/j.tbs.2020.12.008. URL: https://www.sciencedirect.com/science/article/pii/S2214367X20302519.

[17] Matthieu Dubarry, Arnaud Devie, and Katherine McKenzie. "Durability and reliability of electric vehicle batteries under electric utility grid operations: Bidirectional charging impact analysis". In: *Journal of Power Sources* 358 (2017), pp. 39–49. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2017.05.015. URL: https://www.sciencedirect.com/science/article/pii/S0378775317306365.

[18] Kotub Uddin et al. "On the possibility of extending the lifetime of lithium-ion batteries through optimal V2G facilitated by an integrated vehicle and smart-grid system". In: *Energy* 133 (2017), pp. 710–722. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2017.04.116. URL: https://www.sciencedirect.com/science/article/pii/S0360544217306825.

[19] Thomas Martinsen. *Nettbalansering fra store parkeringsanlegg og næringsbygg - NeX2G.* URL: https://www.nmbu.no/prosjekter/node/43105. (fetched: 16.04.2022).

[20] The Research Council of Norway. *Network balancing from large parking facilities and commercial buildings.* URL: https://prosjektbanken.forskningsradet.no/en/project/FORISS/320825?Kilde=FORISS&distribution=Ar&chart=bar&calcType=funding&Sprak=no&sortBy=date&sortOrder=desc&resultCount=30&offset=0&Prosjektleder=Ivar+Berg.

[21] Tamara Nørreskov Aasbøe. *Importance of drivers and barriers for V2G?* URL: https://nmbu.brage.unit.no/nmbu-xmlui/handle/11250/2765758.

[22] Ole Hoff. *Lønnsomhetsanalyse av V2G-park.* URL: https://static02.nmbu.no/mina/studier/moppgaver/2019-Hoff.pdf.

[23] A.T.D. Perera and Parameswaran Kamalaruban. "Applications of reinforcement learning in energy systems". In: *Renewable and Sustainable Energy Reviews* 137 (2021), p. 110618. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2020.110618. URL: https://www.sciencedirect.com/science/article/pii/S1364032120309023.

[24] Dawei Qiu et al. "A Deep Reinforcement Learning Method for Pricing Electric Vehicles With Discrete Charging Levels". In: *IEEE Transactions on Industry Applications* 56.5 (2020), pp. 5901–5912. DOI: 10.1109/TIA.2020.2984614.

[25] Marina Dorokhova et al. "Deep reinforcement learning control of electric vehicle charging in the presence of photovoltaic generation". In: *Applied Energy* 301 (2021), p. 117504. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2021.117504. URL: https://www.sciencedirect.com/science/article/pii/S0306261921008874.

[26] Daniel O'Neill et al. "Residential Demand Response Using Reinforcement Learning". In: *2010 First IEEE International Conference on Smart Grid Communications*. 2010, pp. 409–414. DOI: 10.1109/SMARTGRID.2010.5622078.

[27] Xinan Wang, Jianhui Wang, and Jianzhe Liu. "Vehicle to Grid Frequency Regulation Capacity Optimal Scheduling for Battery Swapping Station Using Deep Q-Network". In: *IEEE Transactions on Industrial Informatics* 17.2 (2021), pp. 1342–1351. DOI: 10.1109/TII.2020.2993858.

[28] Zhiqiang Wan et al. "Model-Free Real-Time EV Charging Scheduling Based on Deep Reinforcement Learning". In: *IEEE Transactions on Smart Grid* 10.5 (2019), pp. 5246–5257. DOI: 10.1109/TSG.2018.2879572.

[29] Qiyun Dang, Di Wu, and Benoit Boulet. "A Q-Learning Based Charging Scheduling Scheme for Electric Vehicles". In: *2019 IEEE Transportation Electrification Conference and Expo (ITEC)*. June 2019, pp. 1–5. DOI: 10.1109/ITEC.2019.8790603.

[30] Wenbo Shi and Vincent W.S. Wong. "Real-time vehicle-to-grid control algorithm under price uncertainty". In: *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. 2011, pp. 261–266. DOI: 10.1109/SmartGridComm.2011.6102330.

[31] Francisco S Melo. "Convergence of Q-learning: A simple proof". In: *Institute Of Systems and Robotics, Tech. Rep* (2001), pp. 1–4.

[32] Balázs Varga, Balázs Kulcsár, and Morteza Haghir Chehreghani. *Deep Q-learning: a robust control approach*. 2022. DOI: 10.48550/ARXIV.2201.08610. URL: https://arxiv.org/abs/2201.08610.

[33] Aparna Kumari et al. "SV2G-ET: A Secure Vehicle-to-Grid Energy Trading Scheme Using Deep Reinforcement Learning". In: *International Transactions on Electrical Energy Systems* 2022 (2022).

[34] Jaehyun Lee, Eunjung Lee, and Jinho Kim. "Electric Vehicle Charging and Discharging Algorithm Based on Reinforcement Learning with Data-Driven Approach in Dynamic Pricing Scheme". In: *Energies* 13.8 (2020). ISSN: 1996-1073. DOI: 10.3390/en13081950. URL: https://www.mdpi.com/1996-1073/13/8/1950.

[35] Eli Brock et al. *An application of reinforcement learning to residential energy storage under real-time pricing*. 2021. DOI: 10.48550/ARXIV.2111.11367. URL: https://arxiv.org/abs/2111.11367.

[36] Farkhondeh Kiaee. "Integration of Electric Vehicles in Smart Grid using Deep Reinforcement Learning". In: *2020 11th International Conference on Information and Knowledge Technology (IKT)*. 2020, pp. 40–44. DOI: 10.1109/IKT51791.2020.9345625.

[37] Samir M Shariff et al. "A State of the Art Review of Electric Vehicle to Grid (V2G) technology". In: *IOP Conference Series: Materials Science and Engineering* 561.1 (Oct. 2019), p. 012103. DOI: 10.1088/1757-899x/561/1/012103. URL: https://doi.org/10.1088/1757-899x/561/1/012103.

[38] Kang Miao Tan, Vigna K. Ramachandaramurthy, and Jia Ying Yong. "Integration of electric vehicles in smart grid: A review on vehicle to grid technologies and optimization techniques". In: *Renewable and Sustainable Energy Reviews* 53 (2016), pp. 720–732. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.09.012. URL: https://www.sciencedirect.com/science/article/pii/S136403211500982X.

[39] Knut A. Mæhlum Lars; Rosvold. *overføringsnett*. November 7 2021. URL: https://snl.no/overf%5C%C3%5C%B8ringsnett.

[40] NordPool. *Day-ahead market*. URL: https://www.nordpoolgroup.com/en/the-power-market/Day-ahead-market/.

[41] Ssolbergj. *File:European Nord Pool markets.svg*. 2 September 2019. URL: https://commons.wikimedia.org/wiki/File:European_Nord_Pool_markets.svg.

[42] Magne Holstad. *Tidenes høyeste strømpris i 4. kvartal*. URL: https://www.ssb.no/energi-og-industri/energi/statistikk/elektrisitetspriser/artikler/tidenes-hoyeste-strompris-i-4.kvartal.

[43] Elvia. *Nettleiepriser og effekttariff for bedrifter i Oslo og Viken*. URL: https://www.elvia.no/nettleie/alt-om-nettleie/nettleiepriser-og-effekttariff-for-bedrifter-i-oslo-og-viken/.

[44] Finn R. Førsund. *Hydropower Economics*. 2nd ed. Springer, 2015, pp. 21, 35.

[45] Energy Facts Norway. *ELECTRICITY PRODUCTION*. URL: https://energifaktanorge.no/en/norsk-energiforsyning/kraftproduksjon/.

[46] ELVIA. *Hva bruker mest strøm?* URL: https://www.elvia.no/smart-forbruk/forbruk-og-sparing/hva-bruker-mest-strom/. (Hentet: 03.11.2021).

[47] Ine Oma Aslak Mæland. *Slik virker det: Hvorfor går strømprisen opp?* URL: https://www.statkraft.no/nyheter/nyheter-og-pressemeldinger/arkiv/2021/slik-virker-det-hvorfor-gar-stromprisen-opp/.

[48] Forbrukernorge. *Når er strømmen billigst?* URL: https://www.forbrukernorge.no/nar-er-strommen-billigst/.

[49] B. Schumm. *battery*. March 5, 2021. URL: https://www.britannica.com/technology/battery-electronics.

[50] the UC Davis Library. *Galvanic Cells*. Sep 18, 2019. URL: https://chem.libretexts.org/Courses/University_of_California_Davis/UCD_Chem_002C/UCD_Chem_2C_(Larsen)/Text/02%5C%3A_Electrochemistry/2.01%5C%3A_Galvanic_Cells.

[51] Wei Liu, Tobias Placke, and K.T. Chau. "Overview of batteries and battery management for electric vehicles". In: *Energy Reports* 8 (2022), pp. 4058–4084. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2022.03.016. URL: https://www.sciencedirect.com/science/article/pii/S2352484722005716.

[52] R.M Dell og D.A.J. Rand. *Understanding batteries.* The Royal Society of Chemestry, 2001.

[53] Rengui Lu et al. "Analysis of the key factors affecting the energy efficiency of batteries in electric vehicle". In: *World Electric Vehicle Journal* 4.1 (2010), pp. 9–13. ISSN: 2032-6653. DOI: 10.3390/wevj4010009. URL: https://www.mdpi.com/2032-6653/4/1/9.

[54] Dae Kyeong Kim et al. *HANDBOOK ON BATTERY ENERGY STORAGE SYSTEM.* URL: https://www.adb.org/sites/default/files/publica%tion/479891/handbook-battery-energy-storage-system.pdf.

[55] Yuliya Preger et al. "Degradation of Commercial Lithium-Ion Cells as a Function of Chemistry and Cycling Conditions". In: *Journal of The Electrochemical Society* 167.12 (Jan. 2020), p. 120532. DOI: 10.1149/1945-7111/abae37. URL: https://doi.org/10.1149/1945-7111/abae37.

[56] Arpit Maheshwari, Michael Heck, and Massimo Santarelli. "Cycle aging studies of lithium nickel manganese cobalt oxide-based batteries using electrochemical impedance spectroscopy". In: *Electrochimica Acta* 273 (2018), pp. 335–348. ISSN: 0013-4686. DOI: https://doi.org/10.1016/j.electacta.2018.04.045. URL: https://www.sciencedirect.com/science/article/pii/S0013468618307771.

[57] Kenza Maher and Rachid Yazami. "A study of lithium ion batteries cycle aging by thermodynamics techniques". In: *Journal of Power Sources* 247 (2014), pp. 527–533. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2013.08.053. URL: https://www.sciencedirect.com/science/article/pii/S0378775313014018.

[58] Scott B. Peterson, Jay Apt, and J.F. Whitacre. "Lithium-ion battery cell degradation resulting from realistic vehicle and vehicle-to-grid utilization". In: *Journal of Power Sources* 195.8 (2010), pp. 2385–2392. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2009.10.010. URL: https://www.sciencedirect.com/science/article/pii/S0378775309017443.

[59] Matthieu Dubarry, Arnaud Devie, and Katherine McKenzie. "Durability and reliability of electric vehicle batteries under electric utility grid operations: Bidirectional charging impact analysis". In: *Journal of Power Sources* 358 (2017), pp. 39–49. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2017.05.015. URL: https://www.sciencedirect.com/science/article/pii/S0378775317306365.

[60] W.V.H. Hasaranga et al. "A Fuzzy logic based battery SOC level control strategy for smart Micro grid". In: *2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB).* 2017, pp. 215–221. DOI: 10.1109/AEEICB.2017.7972416.

[61] Madhuresh Gupta, Soumyakanti Giri, and S. Prabhakar Karthikeyan. "Impact of Vehicle-to-Grid on Voltage Stability - Indian Scenario". In: *2018 National Power Engineering Conference (NPEC).* 2018, pp. 1–5. DOI: 10.1109/NPEC.2018.8476749.

[62] Lav Agarwal, Wang Peng, and Lalit Goel. "Probabilistic estimation of aggregated power capacity of EVs for vehicle-to-grid application". In: *2014 International Conference on Probabilistic Methods Applied to Power Systems (PMAPS).* 2014, pp. 1–6. DOI: 10.1109/PMAPS.2014.6960592.

[63] Wei Shi et al. "Research on the Time Factor of Li-Ion Battery for Vehicle-to-Grid (V2G) Implementation and Charging Station Application". In: *MEMS, NANO and Smart Systems*. Vol. 403. Advanced Materials Research. Trans Tech Publications Ltd, Feb. 2012, pp. 3364–3370. DOI: 10.4028/www.scientific.net/AMR.403-408.3364.

[64] Diyun Wu, Chunhua Liu, and Shuang Gao. "Coordinated control on a vehicle-to-grid system". In: *2011 International Conference on Electrical Machines and Systems*. 2011, pp. 1–6. DOI: 10.1109/ICEMS.2011.6073710.

[65] Madeleine Ecker et al. "Calendar and cycle life study of Li(NiMnCo)O2-based 18650 lithium-ion batteries". In: *Journal of Power Sources* 248 (2014), pp. 839–851. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2013.09.143. URL: https://www.sciencedirect.com/science/article/pii/S0378775313016510.

[66] Abhay Barnard Britto and Kevin Krannich. *What is vehicle-to-grid (V2G) technology?* URL: https://www.elvia.no/nettleie/alt-om-nettleie/nettleiepriser-og-effekttariff-for-bedrifter-i-oslo-og-viken/.

[67] Lasse Edvardsen. *Elbilstatistikk*. URL: https://elbilstatistikk.no/.

[68] Rachel Cross. *WHICH VEHICLES ARE COMPATIBLE WITH NUVVE'S V2G?* URL: https://nuvve.com/faq-items/which-vehicles-are-compatible-with-nuvves-v2g/.

[69] Elbil.no. *Nissan LEAF e+ 62 kWh (2019)*. URL: https://elbil.no/elbiler/nissan-leaf-e-62-kwh-2019/.

[70] chargehub. *Find Out the most Optimal Charging Solution for your Home According to the Electric Vehicle you Drive*. URL: https://chargehub.com/en/find-the-right-charging-station-power.html.

[71] Marklines. *Nissan LEAF Teardown: Lithium-ion battery pack structure*. Dec 6, 2018. URL: https://www.marklines.com/en/report_all/rep1786_201811.

[72] Nissan. *Electric vehicle lithium-ion battery*. URL: https://www.nissan-global.com/EN/INNOVATION/TECHNOLOGY/ARCHIVE/LI_ION_EV/.

[73] Min Wang et al. "Research on Charging Load Characteristics of EVs Based on Actual Charging Power". In: *IOP Conference Series: Materials Science and Engineering* 752.1 (Jan. 2020), p. 012012. DOI: 10.1088/1757-899x/752/1/012012. URL: https://doi.org/10.1088/1757-899x/752/1/012012.

[74] Elektromobilität. *battery*. June 23, 2020. URL: https://www.oeamtc.at/tests/elektromobilitaet/#feldtest-renault-zoe-38560909.

[75] Mark Kane. *Let's Look At Fast Charging Curves For Popular Electric Cars*. July 02, 2018. URL: https://www.oeamtc.at/tests/elektromobilitaet/#feldtest-renault-zoe-38560909.

[76] Sebastian Raschka Vahid Mirjalili. *Python machine learning. Machine learning and deep learning with python, scikit-learn, and TensorFlow*. 2nd ed. Packt, 2017.

[77] Tariq M. Khan and Antonio Robles-Kelly. "Machine Learning: Quantum vs Classical". In: *IEEE Access* 8 (2020), pp. 219275–219294. DOI: 10.1109/ACCESS.2020.3041719.

[78] Kaifeng Gao et al. “Julia language in machine learning: Algorithms, applications, and open issues”. In: *Computer Science Review* 37 (2020), p. 100254. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2020.100254. URL: https://www.sciencedirect.com/science/article/pii/S157401372030071X.

[79] Mohamed Farouk Abdel Hady and Friedhelm Schwenker. “Semi-supervised Learning”. In: *Handbook on Neural Information Processing*. Ed. by Monica Bianchini, Marco Maggini, and Lakhmi C. Jain. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 215–239. ISBN: 978-3-642-36657-4. DOI: 10.1007/978-3-642-36657-4_7. URL: https://doi.org/10.1007/978-3-642-36657-4_7.

[80] Megajuice. *File:Reinforcement learning diagram.svg*. URL: https://commons.wikimedia.org/wiki/File:Reinforcement_learning_diagram.svg.

[81] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning. An Introduction*. MIT Press, 2017.

[82] Qiyun Dang, Di Wu, and Benoit Boulet. “A Q-Learning Based Charging Scheduling Scheme for Electric Vehicles”. In: *2019 IEEE Transportation Electrification Conference and Expo (ITEC)*. 2019, pp. 1–5. DOI: 10.1109/ITEC.2019.8790603.

[83] BrunelloN. *File:Example of a neural network’s neural unit.png*. URL: https://commons.wikimedia.org/wiki/File:Example_of_a_neural_network%5C%27s_neural_unit.png.

[84] BrunelloN. *File:Example of a deep neural network.png*. URL: https://commons.wikimedia.org/wiki/File:Example_of_a_deep_neural_network.png.

[85] Hado van Hasselt, Arthur Guez, and David Silver. “Deep Reinforcement Learning with Double Q-Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 30.1 (Mar. 2016). URL: https://ojs.aaai.org/index.php/AAAI/article/view/10295.

[86] James Kirkpatrick et al. “Overcoming catastrophic forgetting in neural networks”. In: *CoRR* abs/1612.00796 (2016). arXiv: 1612.00796. URL: http://arxiv.org/abs/1612.00796.

[87] David Greenhalgh and Stephen Marshall. “Convergence Criteria for Genetic Algorithms”. In: *SIAM Journal on Computing* 30.1 (2000), pp. 269–282. DOI: 10.1137/S009753979732565X. eprint: https://doi.org/10.1137/S009753979732565X. URL: https://doi.org/10.1137/S009753979732565X.

[88] Anita Thengade and Rucha Dondal. “Genetic Algorithm – Survey Paper”. In: *IJCA Proc National Conference on Recent Trends in Computing, NCRTC* 5 (Jan. 2012).

[89] Josep Panadero. *File:Computational.science.Genetic.algorithm.Crossover.Cut.and.Splice.svg*. URL: https://commons.wikimedia.org/wiki/File:Computational.science.Genetic.algorithm.Crossover.Cut.and.Splice.svg.

[90] Rasmus Gåsemyr Tveitane. *Fleksibilitet i parkerte elbiler ved næringsbygg : en casestudie av Oslo lufthavn Gardemoen*. URL: https://nmbu.brage.unit.no/nmbu-xmlui/handle/11250/2830528.

[91] Jack ClarkDario Amodei. *Faulty Reward Functions in the Wild*. December 21, 2016. URL: https://openai.com/blog/faulty-reward-functions/.

[92] *Derfor er strømprisen høyere i år enn i fjor*. Feb 2, 2022. URL: https://www.energinorge.no/fagomrader/strommarked/derfor-er-stromprisen-hoyere-i-ar-enn-i-fjor/.

[93] *Wallbox Quasar*. 19 April 2022. URL: https://wallbox.no/nettbutikken/wallbox-quasar-bidireksjonal-dc-lader/.

Additional data for the realistic scenario

Profit for each EV

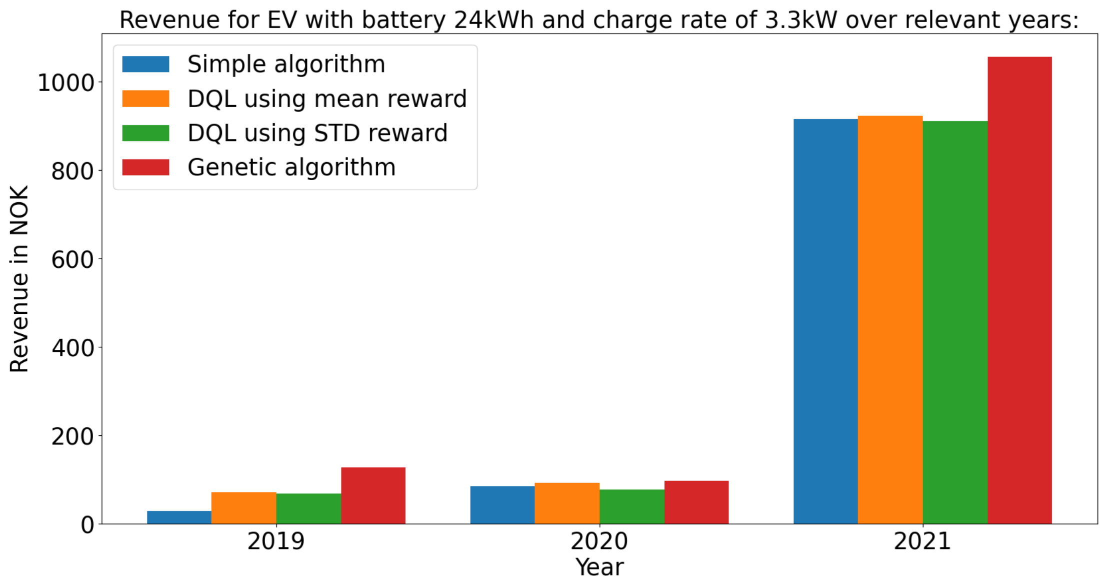


Figure 34: Shows the profit for each algorithm over the three relevant years for the 24kWh 3.3 kW EV.

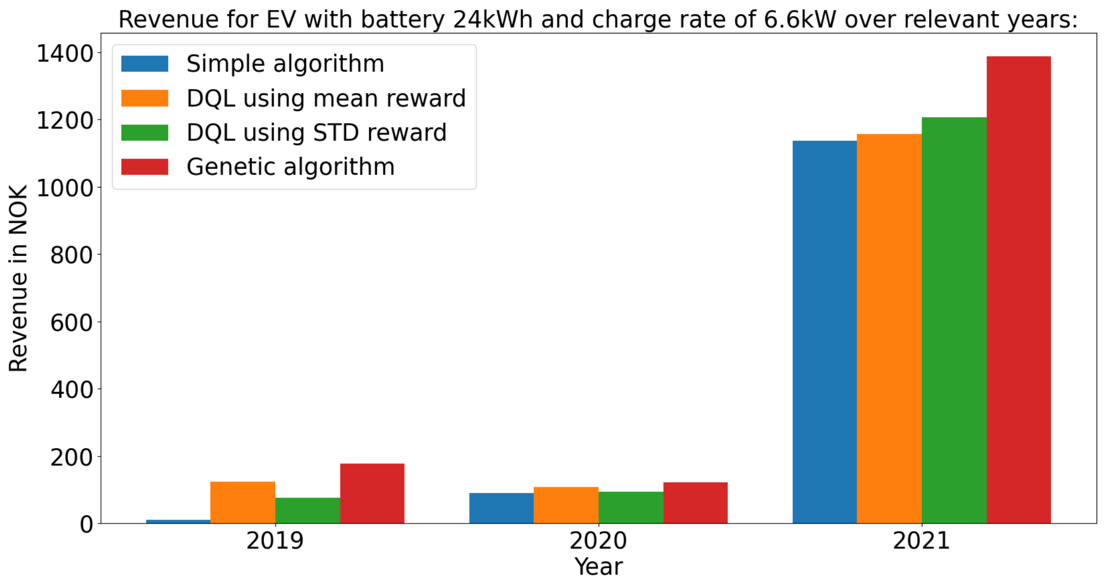


Figure 35: Shows the profit for each algorithm over the three relevant years for the 24kWh 6.6 kW EV.

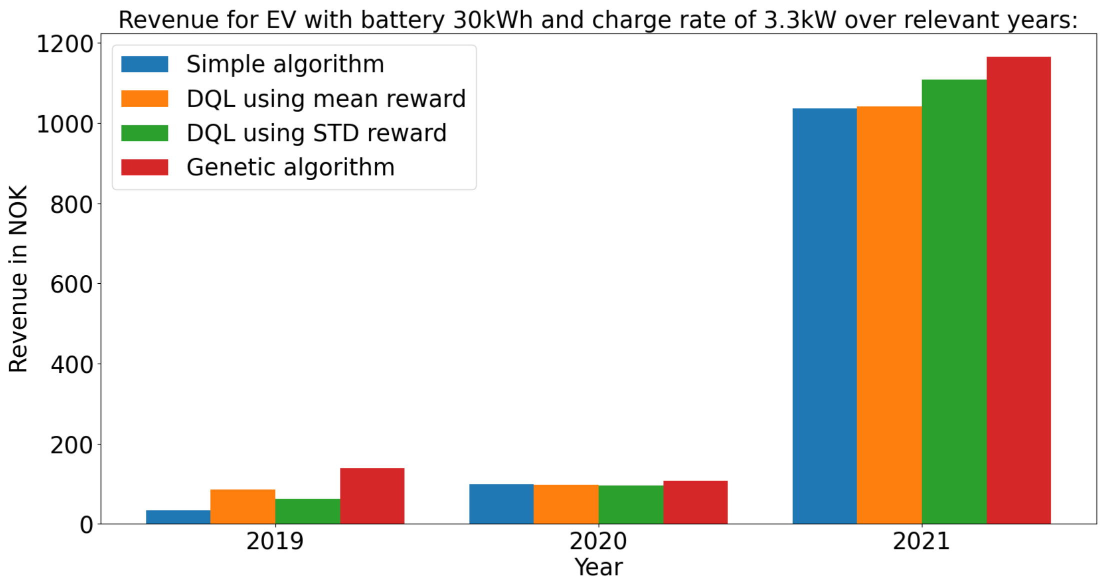


Figure 36: Shows the profit for each algorithm over the three relevant years for the 30kWh 3.3kW EV.

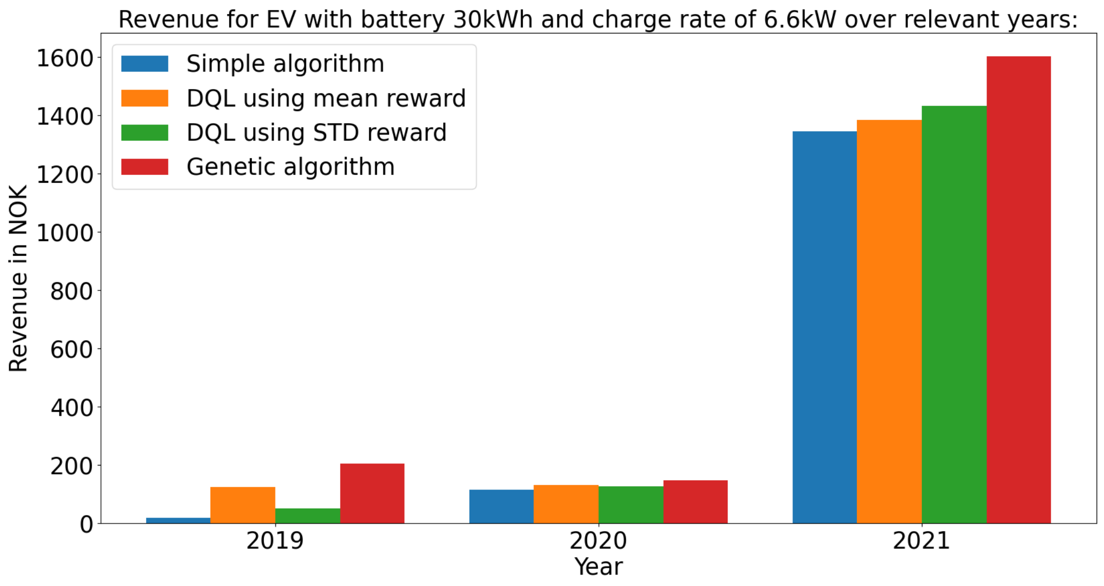


Figure 37: Shows the profit for each algorithm over the three relevant years for the 30kWh 6.6 kW EV.

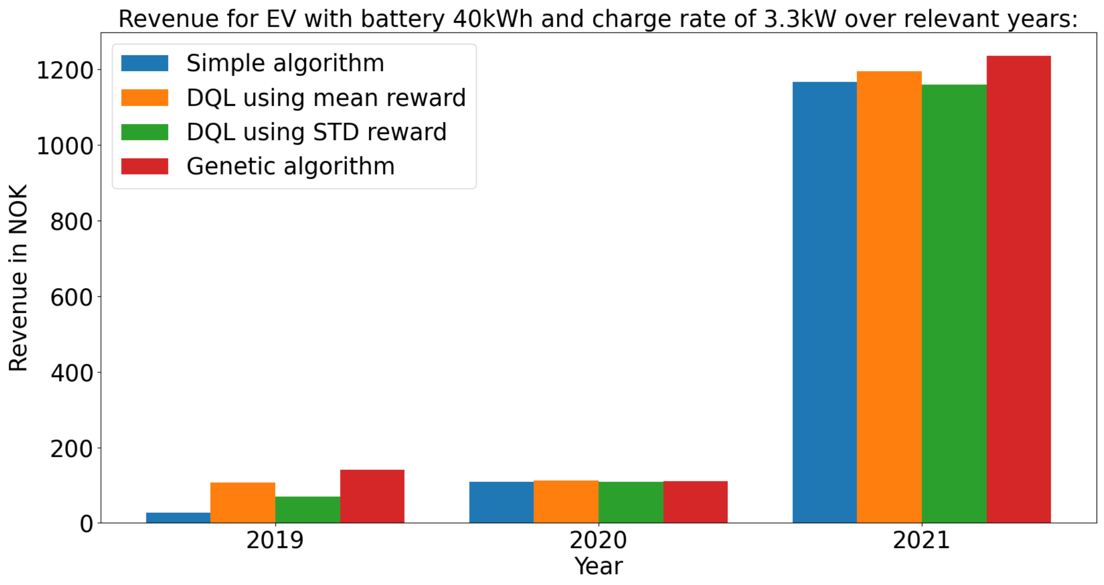


Figure 38: Shows the profit for each algorithm over the three relevant years for the 40kWh 3.3 kW EV.

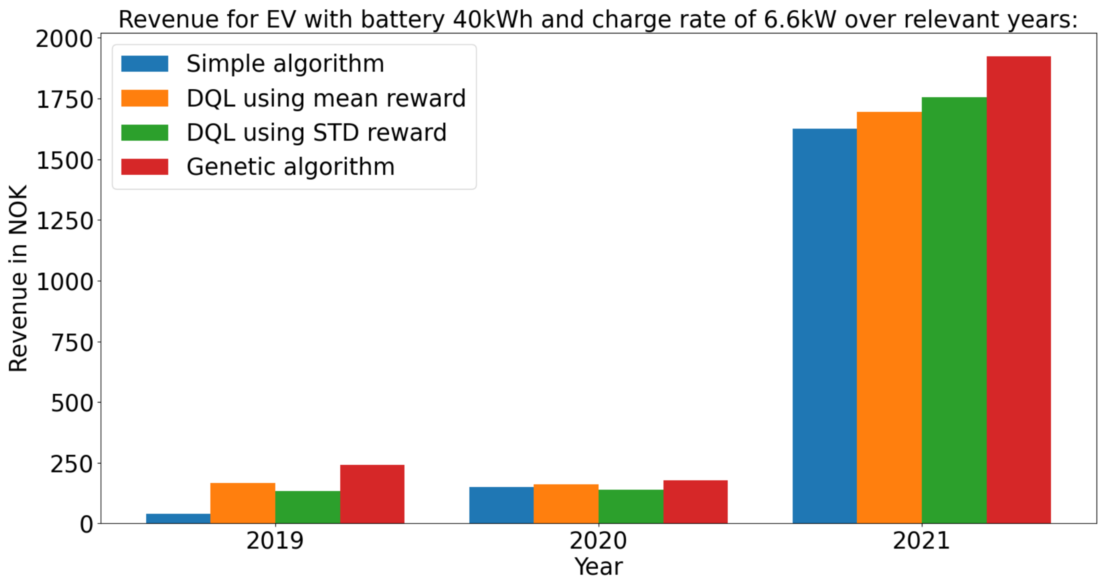


Figure 39: Shows the profit for each algorithm over the three relevant years for the 40kWh 6.6 kW EV.

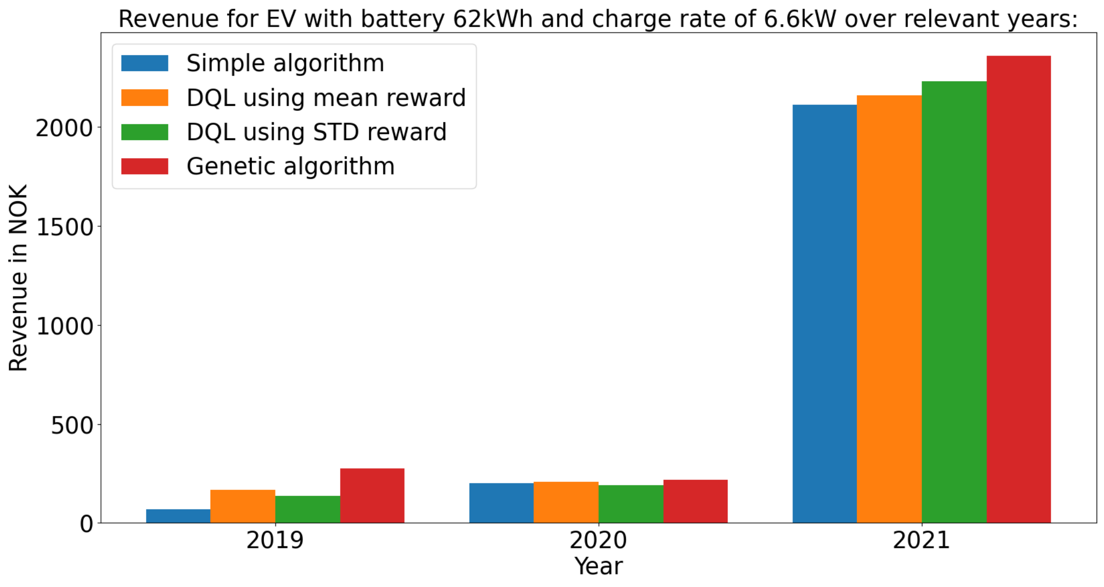


Figure 40: Shows the profit for each algorithm over the three relevant years for the 62kWh 6.6 kW EV.

Effect of no-profit regions

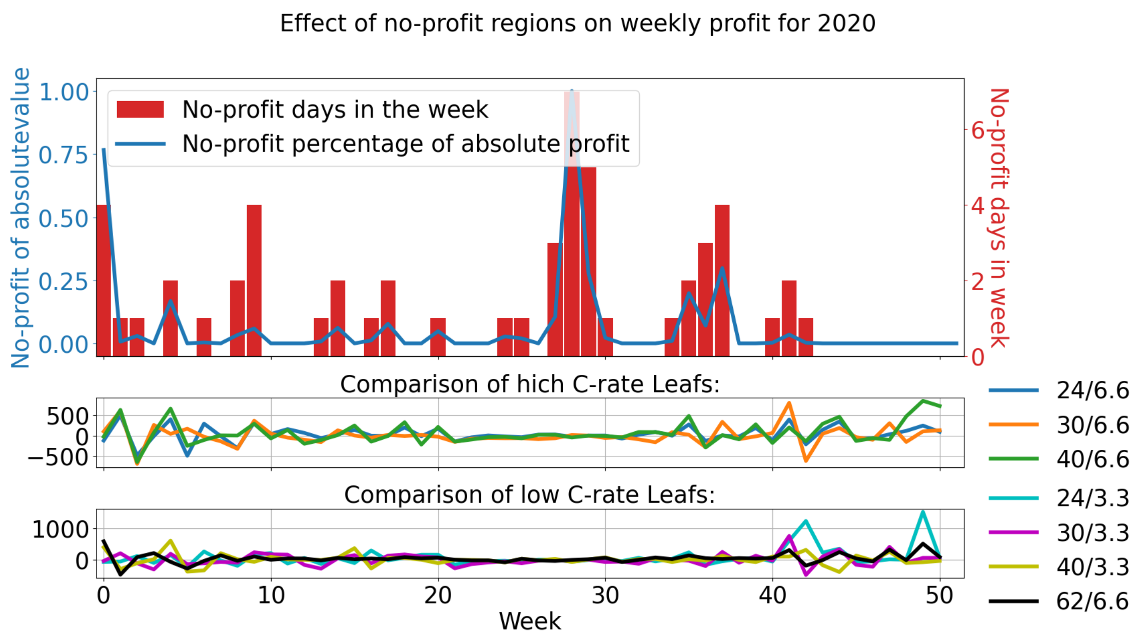


Figure 41: This figure shows the actions taken by the STD-DDQL in the period 2019-02-08 to 2019-02-13 for the EV having a battery of 24 kWh and a charging rate of 6.6 kW.

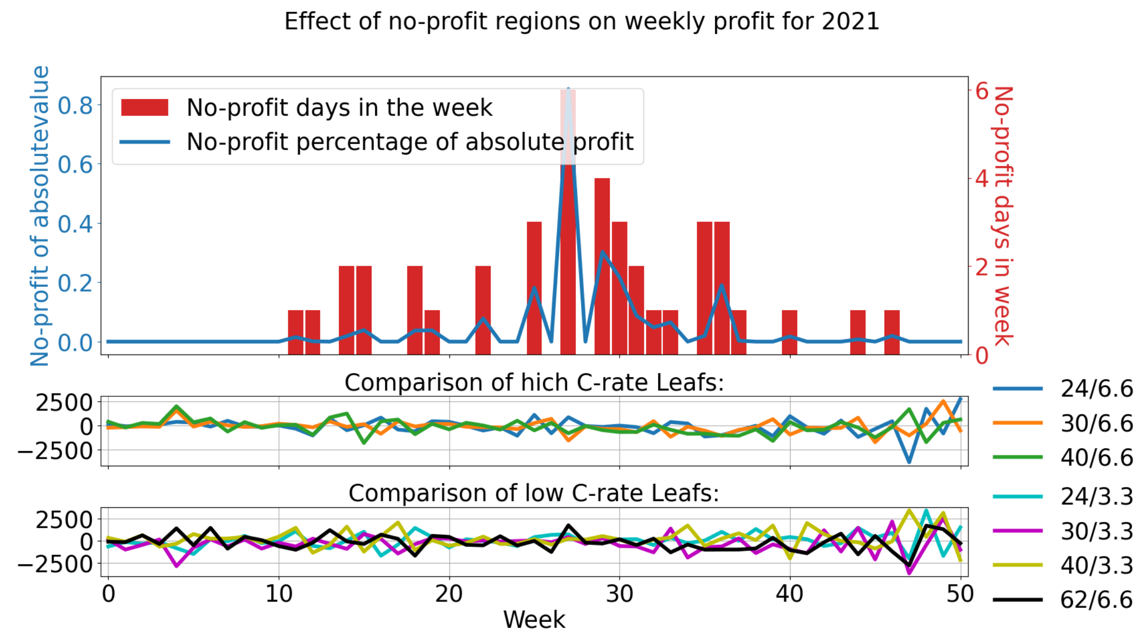


Figure 42: This figure shows the actions taken by the STD-DDQL in the period 2019-02-08 to 2019-02-13 for the EV having a battery of 24 kWh and a charging rate of 6.6 kW.

Decisions plots

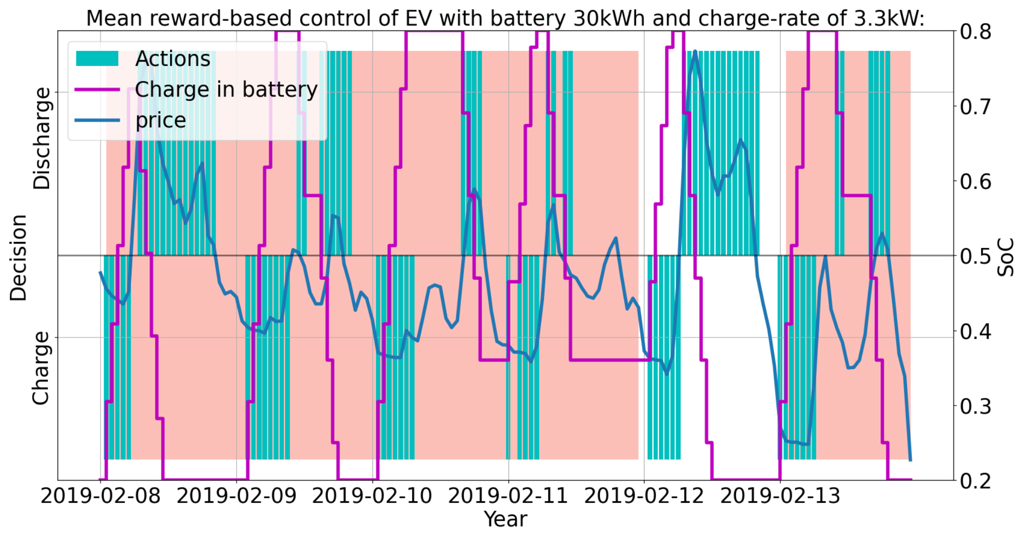


Figure 43: This figure shows the actions taken by the mean-DDQL in the period 2019-02-08 to 2019-02-13 for the EV having a battery of 30 kWh and a charging rate of 3.3 kW.

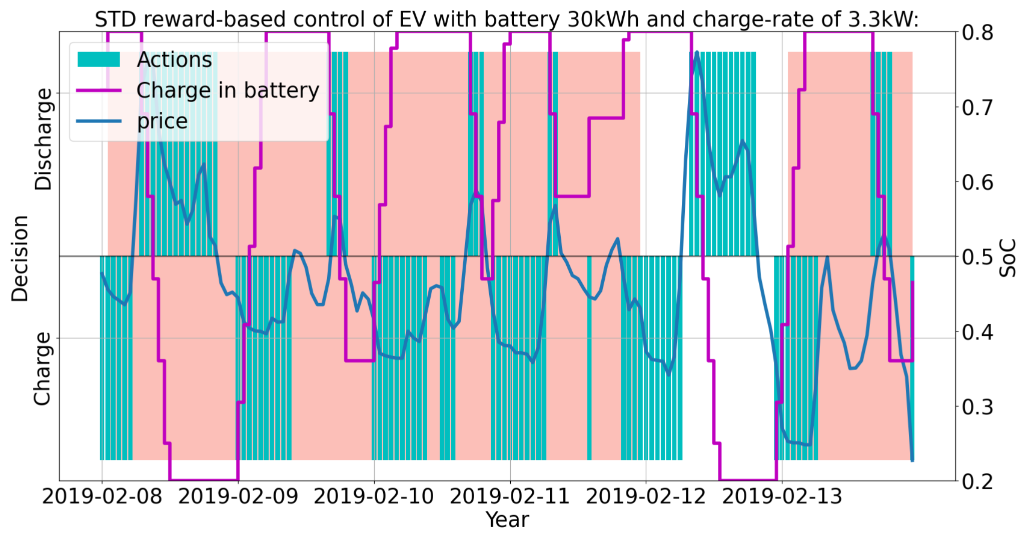


Figure 44: This figure shows the actions taken by the STD-DDQL in the period 2019-02-08 to 2019-02-13 for the EV having a battery of 30 kWh and a charging rate of 3.3 kW.

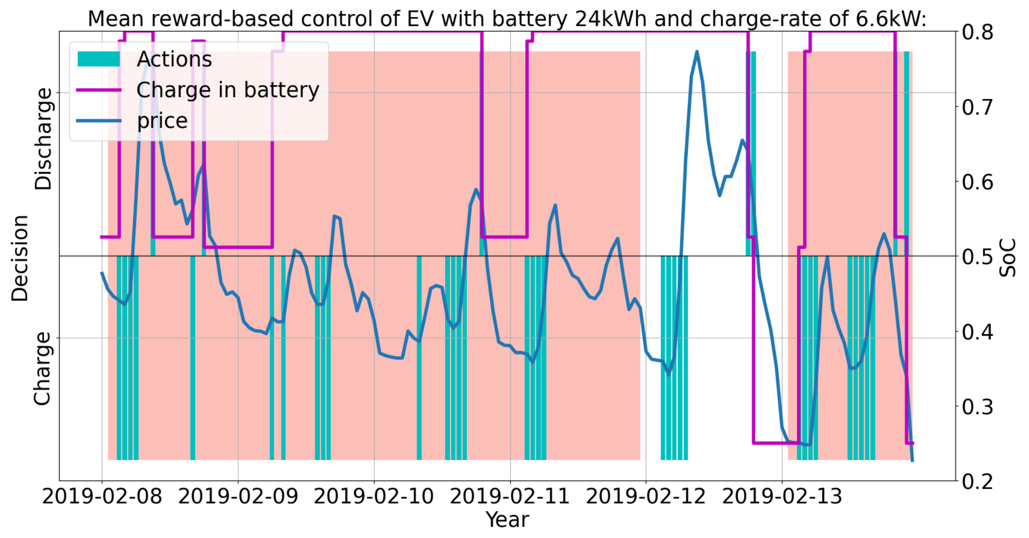


Figure 45: This figure shows the actions taken by the mean-DDQL in the period 2019-02-08 to 2019-02-13 for the EV having a battery of 24 kWh and a charging rate of 6.6 kW.

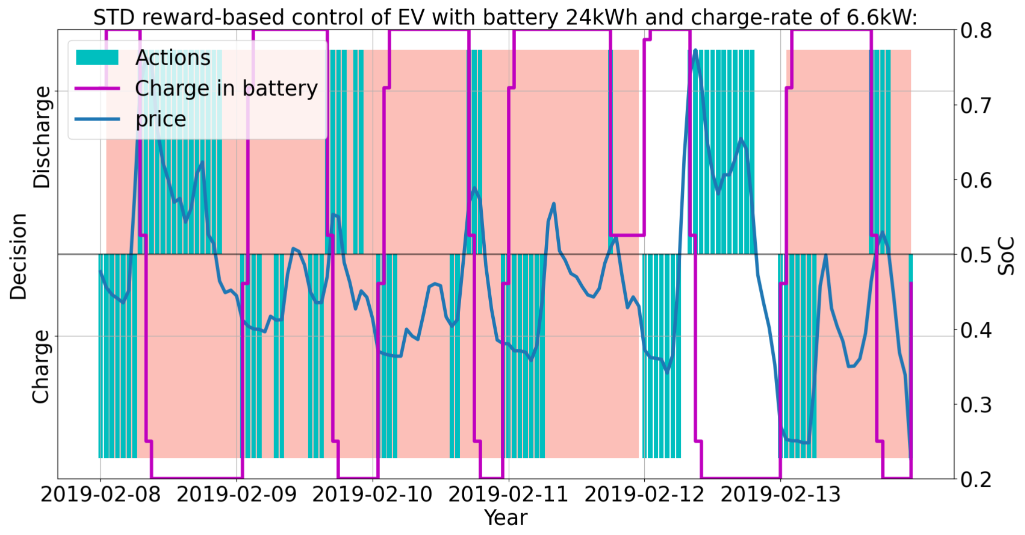


Figure 46: This figure shows the actions taken by the STD-DDQL in the period 2019-02-08 to 2019-02-13 for the EV having a battery of 24 kWh and a charging rate of 6.6 kW.

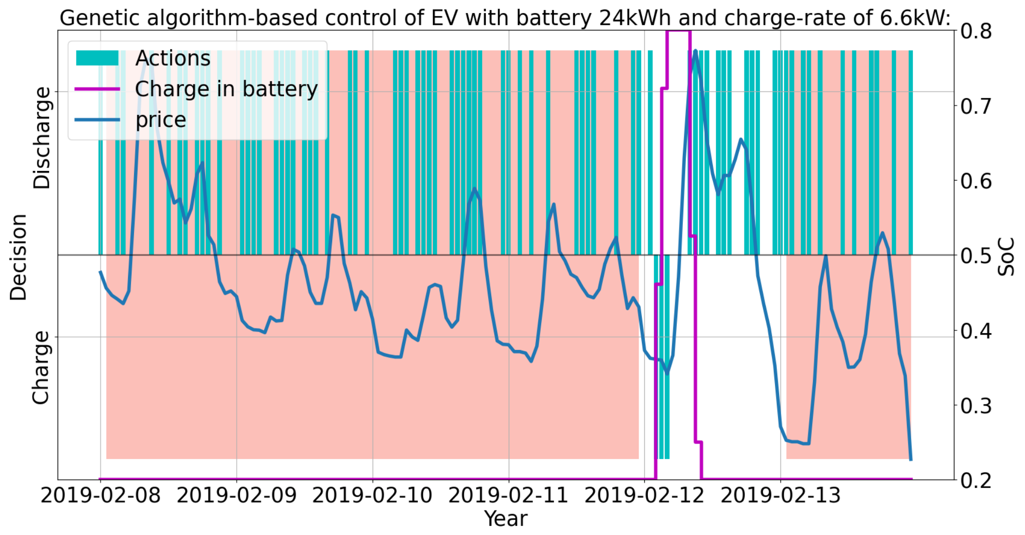


Figure 47: This figure shows the actions taken by the genetic algorithm in the period 2019-02-08 to 2019-02-13 for the EV having a battery of 24 kWh and a charging rate of 6.6 kW.

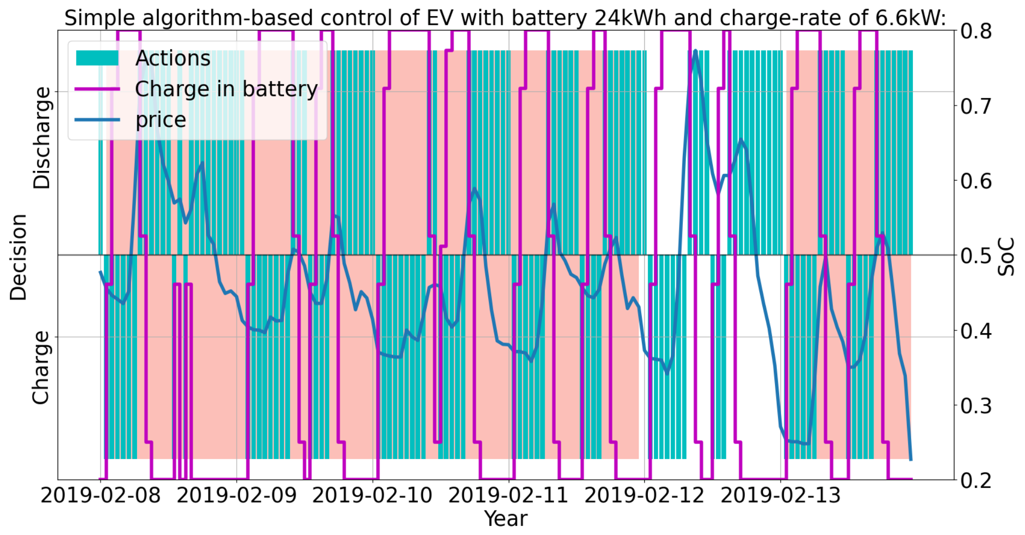


Figure 48: This figure shows the actions taken by the simple algorithm in the period 2019-02-08 to 2019-02-13 for the EV having a battery of 24 kWh and a charging rate of 6.6 kW.

Useful formulas

The **mean** formula is given in equation 23, where μ is the mean, n is the amount of points being taken the mean over and x_i is one data point. The mean can be thought of as the middle point of a set of data points such that 50% of the value will be above this mean value, while 50% will be bellow it.

$$\mu = \sum_{i=1}^{n} \frac{x_i}{n} \tag{23}$$

Standard deviation is given in equation 24, where σ is the standard deviation, and all the other variables are as defined for equation 23. Standard deviation is a measurement of the variation or dispersion of the given data points. This means that if the data points lays close to the mean then one will see a standard deviation value close to zero, while if they are far from the mean then one will see that the value is far from zero.

$$\sigma = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \mu)^2}{n}} \tag{24}$$

Standardization can be seen in equation 25. Where z_i is the standardized value, while all other values are the same as in equation 23 and 24. The point of standardizing the data is to try to get the data with different values on the same form so that it is easier to interpret from one dataset to the next and to compare different values[76].

$$z_i = (\frac{x_i - \mu}{\sigma}) \tag{25}$$

The developed computer program

As a part of this thesis then the development of a system to manage V2G was created. The main goals of this system was to register data from V2G operations, to allow for optimization algorithms to control the V2G operations and for there to be a ”Default” system that could handle the charging in the absence of any optimization algorithm controlling the charger. The basic concept behind the system can be seen in figure 49 and a full overview of the code can be found at this [2] gitlab.

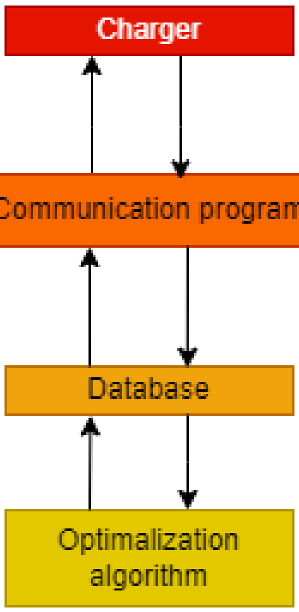


Figure 49: Shows how the program system works in very simple terms. The core of the system is the communication program. It will at a regular interval poll the charger and ask for data about the state of charge of the EV, the charging power and if it is charging or discharging. This is then stored to the database. Then the database will either have stored some entries about V2G operations, or it will not. The communication program will at regular interval request from the database what V2G actions to perform. If there are no data in the database then it will perform ”default” actions and store those actions to the database. When the communication program then has determined what actions to take then it will send those actions to the charger. Afterwards the cycle will repeat.

The communication program

The core of the system is the communication program, and its basic workings can be seen in figure 50. Its responsibility is to fetch readings from the charger, store those readings to the database and to decide which V2G actions that the charger is to perform. The way that the fetching of the readings from the charger is done is by the communication program. The communication program does this by polling the charger at a regular interval, to request the SoC, the power being transferred, and if the charger is charging or discharging. For this polling to work then the pc running the communication program has to be on the same network as the charger, or they have to be connected by some sort of VPN-solution. At the time that the communication program requests information from the charger it will also force the charger to stop its current action if the SoC is about to reach the lower limit or the upper limit set for the battery. Suppose that the battery was set to only operate in between 20-80% SoC, then this would mean that the charger will stop charging if the SoC of the battery reaches 81%.

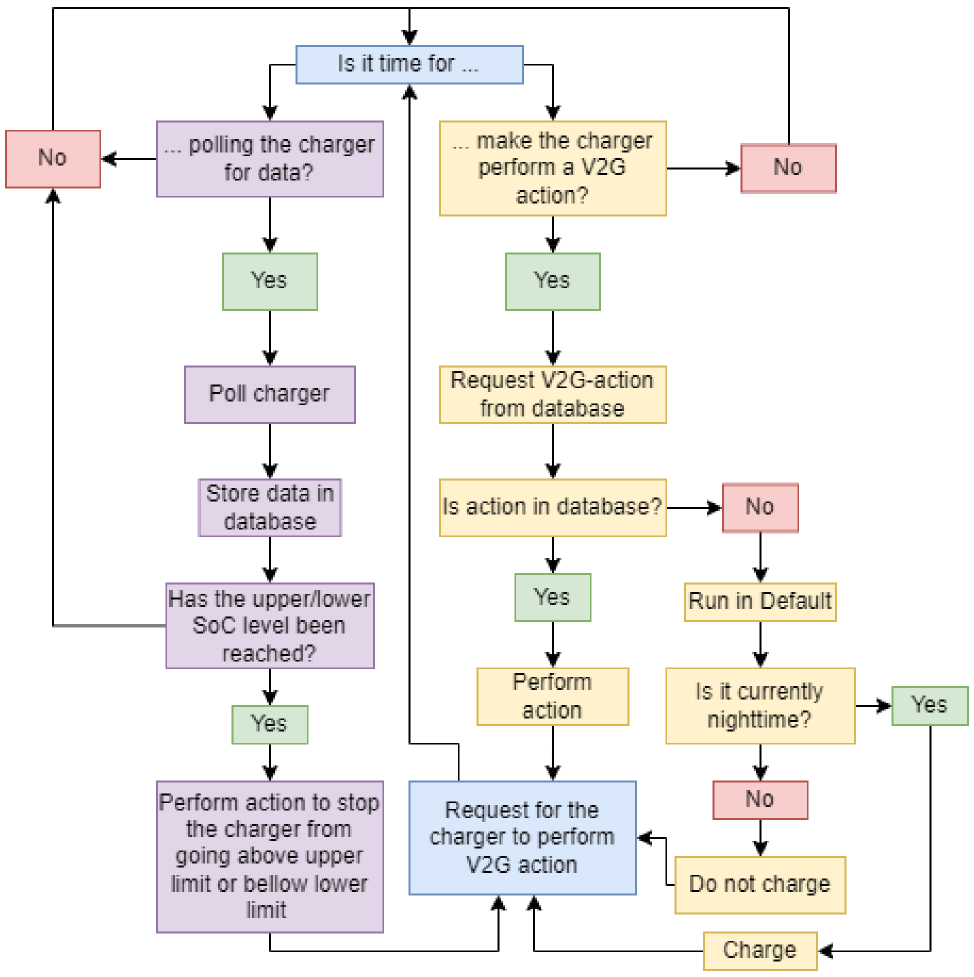

Is it time for ...
... polling the charger for data?
No
Yes
Poll charger
Store data in database
Has the upper/lower SoC level been reached?
Yes
Perform action to stop the charger from going above upper limit or bellow lower limit
... make the charger perform a V2G action?
No
Yes
Request V2G-action from database
Is action in database?
No
Yes
Run in Default
Perform action
Is it currently nighttime?
Yes
No
Request for the charger to perform V2G action
Do not charge
Charge

Figure 50: Shows the basic working of the communication program. It should be noted that in the step where the communication program determines that it wants to run in the "Default"-mode then it will store the action it takes to the database.

When the communication program is to ask the charger to perform V2G actions, then it will first try to fetch the actions from the database and if no actions are in the database then it will run on "default" mode. The action it fetches from the database or the actions chosen by the default mode will then be sent to the charger to be performed.

The default mode is a plan for how the communication program is supposed to operate if it is unable to fetch any actions from the database. The idea is that it should, if it has not been given any other signals, only charge during the night as this is the period with the lowest energy cost in addition to the lowest power consumption for Gardermoen. The hourly limits for the day and night cycle has been set so that it will charge at a maximum charge-rate between 22 in the evening and 6 in the morning and not charge from 6 till 22. Additionally then if it runs in default mode then it will store the actions it took in the database so that it can later be retrieved which actions that were taken by the communication program.

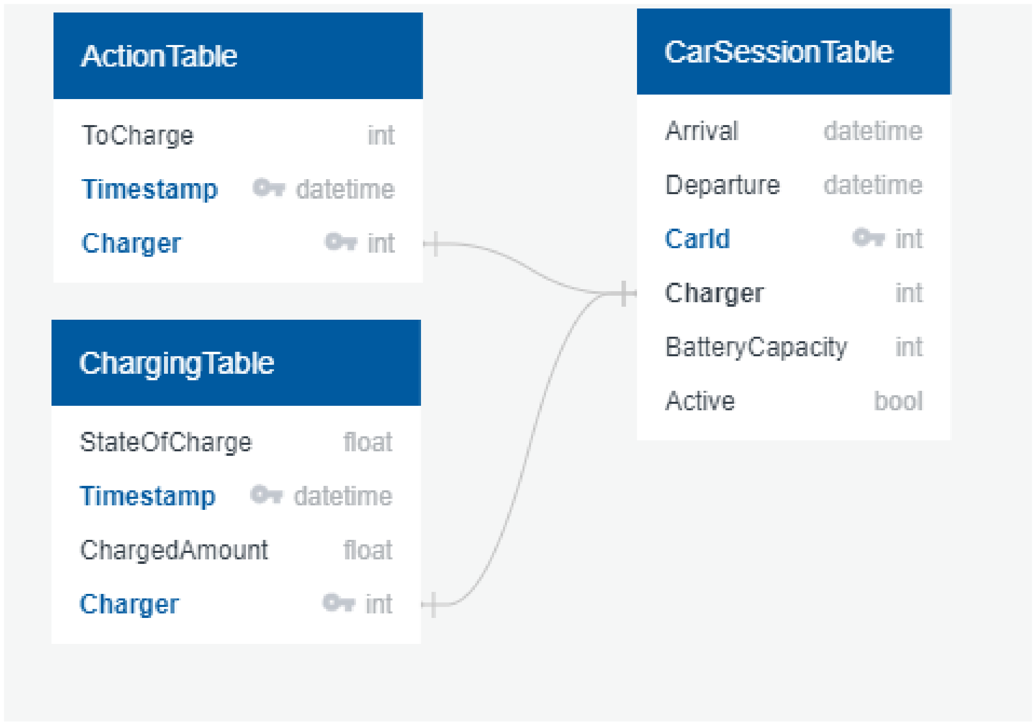


Figure 51: Shows the basic structure of the database. It consists of three tables, ActionTable, ChargingTable and CarSessionTable. The ActionTable stores all the actions that is to be taken or has been taken, the ChargingTable stores reading from the charging of the EV, while the CarSessionTable stores information about the cars that has and is currently charging.

The database

The database is structured as shown in figure 51. The database will store all of the readings that the communication program fetched in the "ChargingTable", this table will contain the

SoC and the amount of power that was charged during the given timestamp for a specific charger. All the actions provided by the optimization program, or done by the communication program in its default setting, will be stored in the ”ActionTable”. The final table, the ”CarSesionTable” that is not yet in use, but that will be used in the future to store when a car arrives and when it departs. It will contain key information about each car that can later be used to make more accurate models, or to get a better overview over the vehicle that was charged.

The optimization algorithm

The optimization algorithm is an optional part of this system, as the communication program will work in default mode in its absence. The way the optimization algorithm can make the system perform its decided V2G actions is by sending in its preferred actions to the ”ActionTable” table in the database. This will allow the communication program to pick up the decided actions, if they are available. The reason why the system is constructed like this, is so that the system can be set into operation without an optimization algorithm, and still work. Additionally this allows for replacement of the optimization algorithm without it affecting the system overall.

For the optimization algorithm to be able to optimize for the price data and the EV then it has to request from the database. This it can freely do, without disturbing the communication algorithm. The most likely table that it will request readings from will be the ”ChargingTable” as this table contains information about the latest state of the EV, such as SoC and latest charge rate.

The charger

On to the charger then this system only works with the Wallbox Quasar bidirectional charger [93]. This is due to the communication program using a modbus TCP protocol to communicate with the charger and the instructions that can be sent through this modbus protocol is specific to this charger. It could be that another charger will share the same modbus instructions as the Quasar charger, but this is not guaranteed. A more in depth explanation of how the modbus protocol functions can be found in this gitlab repository [2].