# Optimisation of battery usage in Smart Grids

## Comparing mathematical optimisation methods for making charging decisions for a private battery in a smart grid

E.A. de Swart BSc

# Optimisation of battery usage in Smart Grids

## Comparing mathematical optimisation methods for making charging decisions for a private battery in a smart grid

by

E.A. de Swart BSc

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday June 26, 2019 at 13:30.

| | | |
|---|---|---|
| Student number: | 4222717 | |
| Project duration: | March 5, 2018 – June 26, 2019 | |
| Thesis committee: | dr. ir. J. H. Weber, | TU Delft, supervisor |
| | ing. J. Kornet, | Witteveen+Bos |
| | dr. ir. F. A. Kuipers, | TU Delft |

# Preface

This thesis is written as a completion of the master Applied Mathematics at the Delft University of Technology. It was completed at the company Witteveen+Bos.

I would like to thank Jos Weber from the university and Johan Kornet from Witteveen+Bos for supervising my thesis. From Witteveen+Bos I would also like to thank Isabelle Vlasman for her help and feedback, Arie de Niet for the mathematical insight, and the other employees and interns that made the day-to-day experience of this internship very enjoyable.

Lastly, I would like to thank Arjen de Swart and Bregje Crum for proofreading my thesis and giving constructive criticism.

*E.A. de Swart BSc*
*Delft, June 2019*

# Abstract

It is predicted that in about 100 years most of the earth's fossil fuels will have been depleted. Currently, fossil fuels still make up 80% of the Dutch energy production. Thus, to handle this depletion, the research on renewable energy production and its usage is being stimulated by governments.

With the use of fossil fuel energy production, it is easy to increase production to meet the unexpected peaks in energy demand by burning more fuels. However, with renewable sources this is not possible. Thus, the way that the produced energy is being used needs to be altered. Furthermore, the amount of renewable energy that is privately generated has increased over the last couple of years. The energy that has been generated for private use and is not needed at that time, can either be sent back into the grid for other users or charged to a battery for later personal use. The electricity network that will regulate the buying and selling of energy is called a smart grid.

When using a battery to store privately generated energy, the decisions that are made for the (dis)charging of the battery are of great influence on the total energy cost at the end of the month. When implementing a battery in a household or company that privately generates energy, these decisions need to be made within a fixed time limit of 15 minutes. In this thesis, four mathematical optimisation methods are compared to each other on result and run time. These methods are dynamic programming, local search, tabu search, and simulated annealing.

Dynamic programming gives the solution with the lowest possible cost, but does not always have the lowest average run time. The total cost of the solution resulting from local search does not come close enough to the lowest possible cost generated by dynamic programming to be a viable alternative to dynamic programming. Tabu search is an extension of local search, it could result in a solution with a total cost close enough to the lowest possible cost if it runs more iterations than local search. However, due to this the average run time will exceed the run time of dynamic programming. Therefore, it is also not a viable alternative for dynamic programming. Simulated annealing has a shorter run time than dynamic programming when using a forecast time of 1 day or less. The total cost of the solutions come very close to those of dynamic programming.

Therefore, while the run time of dynamic programming still fits within the available time limit, it is advisable to use this method to determine the charging decisions for a private battery in a smart grid. However, the simulations that were run in for this thesis do not encompass the entire real-life case. If after the expansion of the problem to be implemented in real-life the run time of dynamic programming were to exceed the available time period, then simulated annealing would be a good alternative for implementation.

# Contents

**1 Introduction** 1
1.1 Research specifications . . . 3
1.2 Witteveen+Bos . . . 4
1.3 Structure . . . 4

**2 Literature study** 5
2.1 Electricity in the Netherlands . . . 5
2.2 Energy sources . . . 6
2.3 Smart Grids . . . 7
2.4 Initiatives in the Netherlands . . . 7
2.5 Battery charging strategies . . . 10

**3 Problem definition** 11
3.1 Problem variables . . . 11
3.2 Example situations . . . 17
3.3 Mathematical problem definition . . . 19

**4 Optimisation methods** 21
4.1 Dynamic programming . . . 22
4.2 Local search . . . 23
4.3 Tabu search . . . 23
4.4 Simulated annealing . . . 24
4.5 Comparison . . . 25

**5 Simulations** 27
5.1 Data . . . 27
5.2 Results . . . 30
5.3 Comparison . . . 37

**6 Conclusion and Recommendations** 45
6.1 Conclusion . . . 45
6.2 Recommendations . . . 46

**A Examples** 47
A.1 Dynamic programming . . . 47
A.2 Local search . . . 52
A.3 Tabu search . . . 56
A.4 Simulated annealing . . . 63

**Bibliography** 77

# 1 Introduction

The current Dutch electricity network consists of consumers, electrical substations, suppliers, producers, and network operators. A consumer is the end user of the energy, this could be for example a household or company. An electrical substation either connects multiple power grids with each other or connects the power grid to the low voltage grid. A supplier provides energy, this energy is either bought from a producer or the supplier is also the producer. A producer produces energy to be sold to consumers. [28]

The network operation is divided in one national network operator, TenneT, and several regional network operators. TenneT manages the transportation of electricity from producers via the power grid to electrical substations. The regional network operators manage the lower voltage grids for the transportation of electricity from electrical substations to consumers. [28] In figure 1.1 a simplified visual representation of the Dutch electricity grid can be found, the arrows indicate the direction of energy transportation.

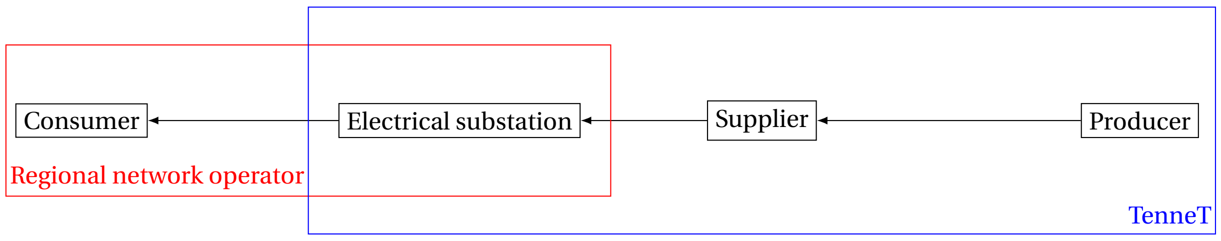


Figure 1.1: Schematic overview of the current Dutch electricity grid and the responsible parties for maintenance

Currently, over 80% of the Dutch energy is produced by burning fossil fuels. However, the fossil fuel sources are running out and it is predicted that in less than 100 years the world will have run out of most of its fossil fuels. Unfortunately, this potential decrease in energy production is not matched with the energy demand. [36] To stimulate the production of renewable energy, the Dutch government started several initiatives in 2013. The goal of these initiatives is for the energy production to be almost entirely renewable in 2050. [33] The results of these initiatives are clearly visible in the increase of the amount of solar panels that are in private use, as can be seen in figure 1.2. At the end of 2017, 9% of Dutch households had solar panels installed for private energy generation. [20].

With the increase of renewable energy sources, the electrical grid will also have to be altered. Currently, the grid is designed to deliver publicly generated energy to the customers. However, with the rise of private generation and consequently the rise of excess energy from private generation that is being sent back into the grid, the distribution of energy is becoming a much more complex problem. As a solution for this problem, the smart grid is introduced, which will be able to handle this change in energy distribution sources and varying amounts, a definition can be found in definition 1.1.

**Definition 1.1** *A* SMART GRID *is an electricity network that can intelligently integrate the actions of all users connected to it - producers, consumers, and those that do both - in order to efficiently deliver sustainable, economic, and secure electricity supplies. [42]*

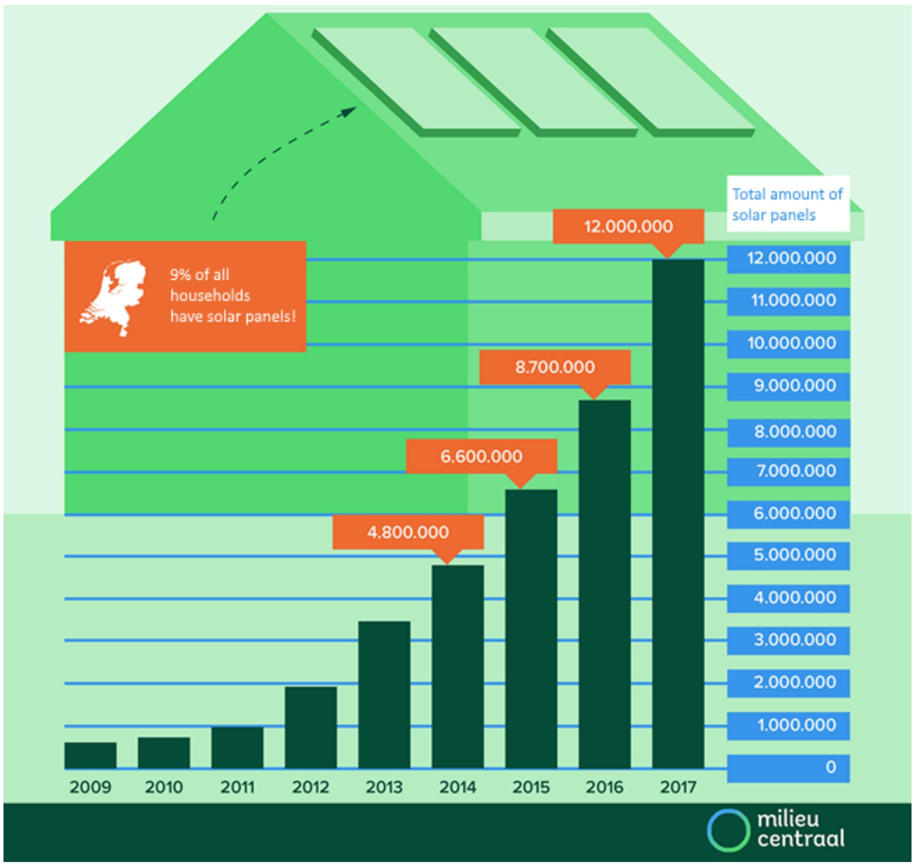


Figure 1.2: Number of private solar panels in the Netherlands over the years [20]

With the introduction of renewable energy sources and batteries in private setting, the flow of energy within the household will change. Without these elements, the only flow of energy is the amount that is bought from the grid directly to the appliances to meet the demand. With the introduction of a private energy source, the generated energy could be used to meet the demand, be stored in a battery for later usage, or be sold back to the grid. The energy that is stored in the battery could be discharged to help meet the demand or be sold back to the grid. A schematic representation of these possible energy flows can be found in figure 1.3. The amount of energy that is traded with the grid, will determine the size of the energy bill at the end of the month. The decisions that are made for the (dis)charging of the battery, are of great influence on the amount of energy that is traded with the grid and therefore on the size of the energy bill. To keep the size of the energy bill as low as possible, the (dis)charging of the battery can be optimised to minimise the cost.

The charging decision that needs to be made for a battery for each time interval is whether or not energy will be charged or discharged to the battery. And if this is the case, the amount of (dis)charged energy needs to be determined. This amount must be feasible with the amount of energy that is available and the specifications of the battery. These specifications include the amount of energy that is currently charged to the battery and the maximum capacity, since the battery can not have a negative charge or the amount charged to it can not exceed the maximum capacity.

The amount of energy that is being traded with the grid is based on the amount of energy that has been generated, the energy demand, and the charging decision that has been made for the battery. If the generation and the discharging of the battery combined does not supply enough energy to meet the demand, then the necessary amount of energy is bought from the grid to meet this shortage. Energy bought from the grid could also be stored in the battery for later usage. If the demand has been met, then any excess energy from the energy source or that has been discharged from the battery is sold back to the grid. Therefore, the charging decisions that are being made for the battery have a big influence on the expenses of the user.

For making decisions for the charging of the battery, the month is divided into time intervals of 15 minutes. For each of these intervals a decision for the (dis)charging of the battery has to be made. The size of the intervals is set to 15 minutes because of the fact that energy trading is done per 15 minutes. [27] This charging

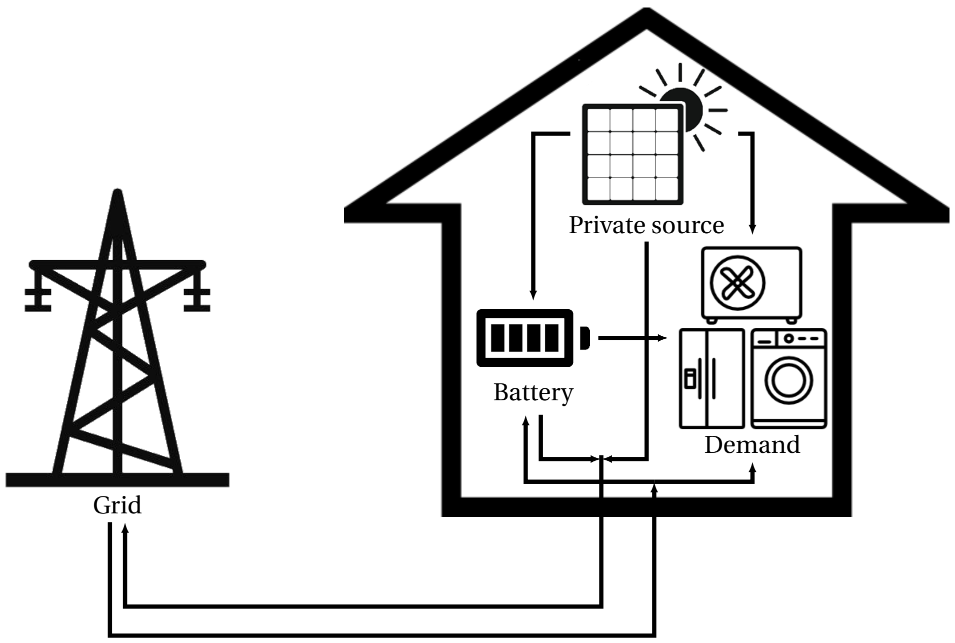


Figure 1.3: Schematic representation of the energy flow between the elements within an end user

decision is made in the 15 minute time interval prior and is based on the specific interval and a set of subsequent intervals. If only the interval itself were to be considered, the lowest cost decision would always be to discharge completely and sell energy back to the grid, if possible.

## 1.1. Research specifications

As was mentioned earlier in this chapter, the flow of energy between an end user and the grid is no longer a one-way street. Before energy was bought from the grid whenever there was a demand. With the introduction of smart grids, conscious decisions need to be made regarding the trading of energy with the grid. Because it would be very time consuming for the users if they had to make this decision themselves for each time interval, this process needs to be automated to make this decision for them. This software would need to make any decision regarding the use of energy. It will need to decide whether energy is charged to or discharged from a battery, and the amount of energy that will be traded with the grid. A mathematical optimisation method should be implemented in this software to help make these decisions, such that the end user benefits from it. In this thesis, several different optimisation methods will be compared to each other to determine which should be implemented in the software.

The thesis written by S. van der Kooij [40] discussed the determination of an optimal charging strategy with the use of the mathematical optimisation method dynamic programming. Dynamic programming uses (predicted) data for the energy demand, amount of generated energy, and the buying and selling price of energy for multiple consecutive upcoming intervals. With this data, a decisions for the next time interval is made which will result in the lowest cost at the end of this set of time intervals. Once this decisions is made for the next interval, it is executed. While this decision is being executed, the entire decisions process is repeated for the next time interval. The number of future time intervals that are used for this determination is the same for the decision of each time interval.

The research conducted by Van der Kooij created a good foundation for the use of dynamic programming, but still had a lot of opportunities for future work such as the integration of a private renewable energy source. The duration of a dynamic programming approach can be very large, which is not desirable when the charging decision must be made within a short time period. If the method were to be implemented for every day use, there is a time limit of 15 minutes for making the decision for the next interval. Therefore, it was decided to focus this research on the comparison of mathematical optimisation methods for the determination of charging decisions for a private battery in a smart grid. The considered methods besides dynamic programming are local search, tabu search, and simulated annealing. The different mathematical methods will be compared to each other on run time and result. Formulating this results in the following research question:

**What mathematical optimisation method should be used for making charging decisions for a private battery in a smart grid?**

To support this research question, several subquestions have been constructed. Each of these subquestions will be the main focus of the upcoming chapters of this thesis, these being:

(1) What research has currently been conducted on the implementation of smart grids?

(2) How can the problem be mathematically defined?

(3) Can other existing mathematical optimisation techniques, besides dynamic programming, be implemented to determine the optimal charging strategy for a battery?

(4) How do the considered implementations hold up in a simulation of a real-life case regarding run time and result?

## 1.2. Witteveen+Bos

This research has been performed in collaboration with the company Witteveen+Bos. This is a consultancy and engineering firm established in 1946 in the Netherlands. Their 19 offices are located in 11 countries in Europe, the former Soviet Union, and Southeast Asia. Their clients consist mainly of governments and other companies and they specialise in projects regarding water, infrastructure, environment, and construction. The internship was carried out at the section Smart Infra Systems. [44].

## 1.3. Structure

This thesis will be structured as follows. First, in chapter 2, a literature study for this thesis will be presented. This will include the current developments in the area of smart grids, research that has been conducted, the challenges that need to be overcome before smart grids can be implemented, and a short overview of research specific to charging strategies of batteries in a smart grid. This overview is followed by an introduction to the variables necessary to describe the process in a mathematical problem definition in chapter 3. In chapter 4, a description is given of the four mathematical optimisation methods that will be compared on run time and results. These methods are dynamic programming, local search, tabu search, and simulated annealing. These methods will be tested and compared to each other with the use of simulations and real-life data, this will be described in chapter 5. Lastly, chapter 6 will contain the conclusion and recommendations for future work regarding this thesis.

# 2 Literature study

The amount of research regarding smart grids has been steadily increasing over the years. However, there is still a long way to go before it can be fully implemented. In this chapter an overview of research relevant for this thesis is presented. First, a short history of energy generation in the Netherlands is given, including the goals set for the future. Next, several energy sources will be described and their renewability will be discussed. This is followed by a short description of the changes that will need to be made with the implementation of smart grids and a synopsis of smart grid initiatives that have been implemented in the Netherlands. Each of these initiatives highlighted the challenges that need to be overcome before smart grids can be fully integrated, including the current laws and regulations that limit the implementation. The chapter is concluded by a short outline of conducted research regarding the use of dynamic programming for the decision-making for private batteries in a smart grid.

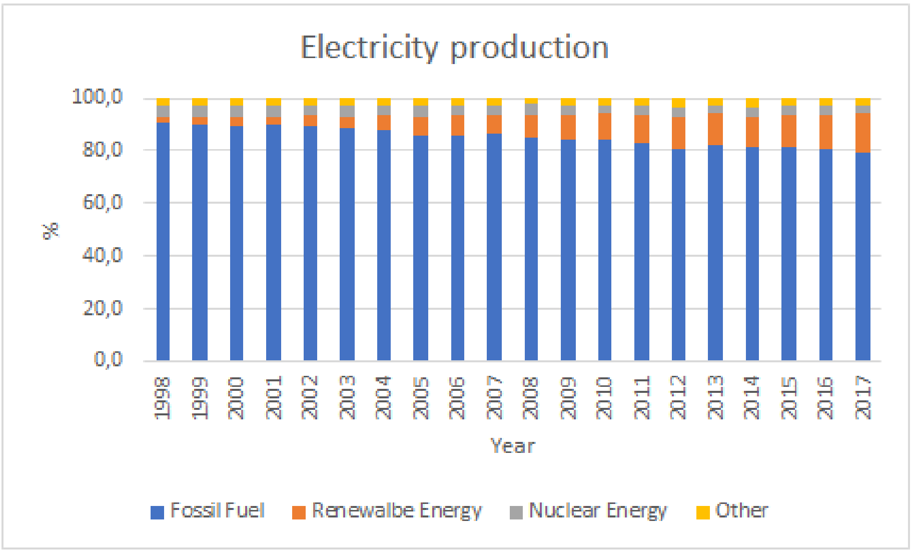


Figure 2.1: Electricity production percentages in the Netherlands [4]

## 2.1. Electricity in the Netherlands

The first public power supply in the Netherlands was put into use in 1884 in Rotterdam; a set of batteries was used to light a few buildings. Two years later, in 1886, the first official power plant was opened, a coal powered steam engine connected to two dynamos located in Kinderdijk. A year later, the regional energy companies united first under 'Samenwerkende Electriciteits Productiebedrijven' (Cooperating Electricity Production Companies) and in 1998 under its successor TenneT. The latter was a consequence of the introduction of the Electricity Law that secured the liberalization of the electricity market. This law stated that everyone was free to choose their own energy supplier and that there was one company, TenneT, that took care of the network management. [41]

Over the years, the amount of renewable energy that is being generated has increased. However, fossil fuels still covered over 80% of the total generation in 2016, see figure 2.1. [4] It is predicted that in less than 100 years the world will have run out of most of its fossil fuels, but the demand for energy will not have decreased. Thus, there is still a long way to go to satisfy the ever-increasing demand for energy. [36]

Currently, the government, companies and citizens are investing in renewable energy sources. It is the goal of the Dutch government to increase its national renewable energy percentage to 14% by 2020 and 16% by 2023. [37]

### 2.1.1. Selling energy to the grid

With the rise of the private generation of energy, the users had to buy less energy from the grid to cover their usage. However, when the owners of the source are at work or on a holiday, not all the energy that is being generated may be used. This excess energy can be sold back to the grid, such that other users can use it and it will not go to waste. The price for this sold back energy had to be determined.

Currently, for small and medium enterprises, and households, the energy that is sent back into the grid is registered and subtracted from the total amount of energy taken from the grid to determine the eventual bill. This is called net metering. [22] However, for larger companies, net metering is not possible. They already receive a feed-in tariff per kWh they send back into the grid. When at the end of a billing period, the client has generated more energy than they have used, they will receive a feed-in tariff per kWh they have sent back into the grid. This tariff varies per energy provider. [21]

The original plan was for net metering to be possible up until 2023. However, due to the large increase in acquisition of private renewable energy sources the cost of this has highly increased. So, this arrangement will be in existence until 2020, after which the feed-in tariff will hold for all the (produced) energy that is being sent back into the grid. [22] When this tariff holds, the purchase of a battery for private usage becomes profitable when an algorithm is implemented that has as a goal to maximize the total reward.

To anticipate this change to a feed-in tariff for all customers, it is implemented in this thesis. The reward for selling energy back to the grid is lower than buying energy from the grid, the reward depends on the amount that is sent back into the grid and the energy supplier. Each supplier is free to set its own tariff, most set this tariff to be equal to the bare energy price. [21]

## 2.2. Energy sources

Currently, fossil fuels is the main source for energy generation. However, due to the fact that the resources can not be reused, other renewable energy sources are considered to take its place, see definition 2.1.

**Definition 2.1** *A* RENEWABLE ENERGY SOURCE *is an energy source that will not run out or will be replenished in our lifetime. [24]*

To compare the different energy sources to each other, a capacity factor is determined, see definition 2.2. An overview for the capacity factors for renewable energy sources can be found in table 2.1. Nuclear and coal fired power plants have the highest capacity factor, thus are the most efficient sources. However, these methods are not entirely renewable, thus alternatives with a lower capacity factor need to be considered.

**Definition 2.2** *A* CAPACITY FACTOR *is the percentage of the theoretical maximal output that is generated by an energy source. [8]*

Solar panels are a renewable energy source, however they have the lowest capacity factor of all these energy sources. This is because its generation is completely dependent on both the amount of sun that shines on them throughout the day, and the orientation and location of the panels. [39]

Wind turbines are also a renewable energy source and they can generate energy throughout the entire day, as long as there is wind. However, due to zoning regulations they can be forbidden to install. [39]

Biomass that is left over from production processes in for example the agriculture, can be used to generate biogas, which can then be employed to generate heat. However, the biogas must be converted to green gas before it can be sent back into the natural gas network, which makes the process more complex. Biogas is considered to be a renewable energy source. [16]

With cogeneration, both electricity and heat are produced at the same time from one single fuel. This fuel could be natural gas, fuel oil, or pellets. By generating multiple energy types, the energy source is used

| Generation type | Capacity factor |
|---|---|
| Solar panels | 10-25% |
| Wind turbines | 25 % |
| Combined cycle gas plant | 38% |
| Cogeneration | 40% |
| Hydroelectric power plant | 40 % |
| Coal fired power plant | 80% |
| Nuclear power plant | 89% |

Table 2.1: Capacity factors of different type of energy sources [25], [29]

more efficient than without cogeneration. If the gas used with the cogeneration is green, then the generation is completely free of $CO_2$. [16] The renewability of cogeneration depends on the type of fuel that is used. Gas fired plants are not renewable, however biomass fueled plants are renewable. [43]

In order to be able to generate renewable energy with hydro power, a source of flowing water is necessary. The amount of energy that is generated depends on the amount of water and the vertical distance that it travels. This source can generate a steady amount of energy through the entire day. [39]

Energy is generated in coal fired power plant by burning coal, this is a non-renewable energy source since the coal can not be used again after the burning. It impacts the environment due to air and water pollution. [15]

A nuclear power plant harvests powerful energy in the core of an atom, nucleus, by splitting it. The energy itself is renewable, however the materials that are used in the plants are not. [24]

For businesses, the Dutch government has in 2018 a budget of €100 million to subsidies the installation of renewable energy sources. The amount that is received is dependent on the type of device that is installed. This amount varies between €500 and €2500. [32]

## 2.3. Smart Grids

As stated above, over 80% of the current electricity production is still generated by fossil fuels. Unlike most other energy sources, this production is independent of environmental influences. Because of this, the production of electricity is demand based, which means that end users can turn on any appliance requiring electricity at any time and the production will increase to meet this demand. Most renewable energy sources, on the other hand, are dependent of environmental circumstances such as wind power and cloud cover for their generation. They are therefore supply based. This means that the electricity production cannot easily be increased to meet any increase in demand there might be. The generation of energy will also no longer be located at a few select factories, but spread out over multiple locations in the country. [41]

To handle this change in the system, the energy grid will have to undergo a transformation to keep up. This envisioned new energy grid is called a smart grid. Currently, a lot of research is being conducted to determine how such a smart grid would function in the best possible way.

## 2.4. Initiatives in the Netherlands

The most recent practical experiments with smart grids in the Netherlands were conducted between 2011 and 2015. [30]. This set of twelve projects was overseen by Innovative Program Intelligent Nets (IPIN), the two main contributors involved with these projects were AgentschapNL and 'Rijksoverheid voor Ondernemend Nederland' (government for entrepreneurial Netherlands). These so called 'experimental gardens' covered different areas of society where possibilities for the implementation of smart grids are present. [1] Because these were some of the first experiments that took place in the Netherlands, their scope was very broad. However, the results did paint a clear picture of where the bottlenecks for general implementation are located. An overview of these projects and their respective market areas can be found in table 2.2.

The category 'Energy control systems and services' contains the projects that focus on implementing new systems for the use of smart grids, while the category 'Flexible energy infrastructure' contains projects that experiment with the infrastructure to implement the use of smart grids.

Below follows a short description of each of the projects and their results are presented.

| Markets / Products and services | Domestic and consumers | Domestic and business | Business and commercial including grid management | Industrial |
|---|---|---|---|---|
| Energy control systems and services | Couperus<br>Heijplaat<br>Lochem<br>PMC2<br>YESCON<br>Texel | INZET<br>ProSECco | EVANDER | |
| Flexible energy infrastructure | | | DeCent<br>TU Delft | MODIENET |

Table 2.2: IPIN experimental gardens market combinations [31]

### 2.4.1. Energy control systems and services

*Domestic and consumer*
In the domestic and consumer market area, six experimental gardens were constructed, Couperus, Heijplaat, Lochem, PMC2, YESCON and Texel. [1]

- Couperus is a newly constructed residential complex that does not contain gas pipes, the houses are heated with the use of a collective geothermal storage and individual heat pumps.
- Heijplaat is a residential area in Rotterdam that is transformed into a sustainable and energy-neutral district. A key element of this project is the usage of a feedback system, price incentive and active steering to optimise the energy usage.
- The experimental garden LochemEnergie actively involves the participants to balance the supply and demand of (sustainable) energy. With this cooperation the process of optimising energy usage is a lot smoother than without.
- PowerMatching City II (PMC2) is a continuation of the earlier project PowerMatching City, it validates the empirical value of smart grids and provides insight in the demands and wishes of end users.
- Your Energy moment: Smart Grid with the Consumer (YESCON) is situated in the newly build residential area Muziekwijk in Zwolle. Each of the houses is equipped with solar panels, facilities for electrical vehicles and smart appliances that can remotely be turned on and off based on the current energy usage.
- Cloud Power Texel is an initiative that has as goal to make its users and their community completely independent of the main electricity grid by only using the energy generated by their own renewable sources and sharing this with each other.

One of the main conclusions of these projects is the importance of user participation. Keeping them engaged and interested in the innovations results in a smoother transition and implementation of smart grids within the communities. [31]

*Domestic and business*
Two experimental gardens were implemented in the domestic and business market area, INZET and ProSECco. [1]

- Intelligent Network Zeewolde and Energy Transition (INZET) is an initiative that has as goal to locally use the locally generated energy from the several sustainable energy providers. The lack of generation due to insufficient weather conditions are compensated with the use of bio gas.
- In Experimental garden Smart Energy Collective & Co (ProSECco) two main services were delivered; flexibility in energy and the necessary information, which are integrated with the technologies present at the locations to optimise the overall usage. This project was implemented at multiple locations in different environments. These were Schiphol theGROUNDS (industry), Siemens and ABB (offices), Gorinchem (all-electric residential), Heerhugowaard (gas and electricity residential) and Goes (district heating residential).

The above initiatives have a lot of potential, however, they encounter difficulties due to the current laws and regulations. [31] In subsection 2.4.3 this is explained in more detail.

*Business and commercial including grid management*
The project EVANDER is situated in the business and commercial market area. [1]

- Electric Vehicles And Distributed Energy Resources (EVANDER) focuses on the integration of the (dis)charging possibilities of electric vehicles and other storage systems. The excessive energy is to be traded with other companies.

Just as for several other experimental gardens, the current laws en regulations restrict a proper implementation, see subsection 2.4.3. [31]

### 2.4.2. Flexible energy infrastructure

*Business and commercial including grid management*
In the business and commercial market area, two other projects were executed, DeCent and TU Delft. [1]

- The project DeCentralized electricity grid agriculture on direct current (DeCent) is located in the sustainable agriculture area PrimA4a close to Schiphol, with the use of wind and solar power a direct current grid is constructed and used to power the area.
- Intelligent heat net campus TU Delft uses the existing facilities to construct a geothermal grid for the heating the campus.

With both of the projects good results have been achieved and could be extended to cover a lager area. [31]

*Industrial*
The last project, MODIENET, is located in the industrial market area. [1]

- Modular Intelligent Energy network for business parks (MODIENET) focusses on the demand control of companies using their predicted flexibility.

Just as for several other experimental gardens, the current laws en regulations restrict a proper implementation, see subsection 2.4.3. [31]

### 2.4.3. Laws and regulations in the Netherlands

Currently, there are not many possibilities for the implementation of smart grids due to restrictions by the law. [31] First off, there is a need for more flexible network management. The European law has exceptions that allow for these implementations. However, the Dutch law is more restricting and therefore limiting possibilities for more flexibility. Secondly, there are the storage regulations. Many possibilities exist for the private storage of energy. In contrast, no clear regulations for storing energy in a public setting exist. Third is the fact that mediation by a larger energy supplier is necessary for small users to trade energy among themselves. Also, the energy suppliers have a big influence on the taxes that must be paid over the generation of energy. Furthermore, it has been indicated by users that they want one set of rules for each location of self-generating energy, whether this is at home, communally with other users, or at a third party. Lastly, different energy meters have occasionally different functionalities. However, it is currently impossible for users to decide which energy meter they want installed in their house, as these meters are provided by the energy supplier. They can install an additional one of their own, which leads to higher cost and potentially not always to the complete set of desired functionalities. [34] Thus, there is still a long way to go before smart grids can be fully implemented while following the current (Dutch) law.

## 2.5. Battery charging strategies

A significant part in the implementation of the private generation of renewable energy is determining what to do with the generated energy that is not being used by the producer. This energy can be sold back into the grid or it could be stored in a battery. In this thesis, several methods will be compared to each other on run time and results for the decision-making considering charging or discharging a private battery in a smart grid. The first method considered is the application of (stochastic) dynamic programming. When optimising with dynamic program, the final solution will be optimal in regard of the objective function. Van der Kooij started researching the implementation of this method in his thesis. Therefore, this was chosen as the first method. Below follows a brief description of the method and the situation cases that were considered in the aforementioned thesis. In chapter 4 the method is discussed in more detail and in section A.1 a numerical example is presented.

### 2.5.1. Dynamic Programming

In the thesis written by Van der Kooij, the application of (stochastic) dynamic programming for the determination of charging decisions for a battery in a smart grid is researched. For this he considers three cases, each an elaboration of the previous one.

Dynamic programming breaks a problem down in easier to solve, smaller (decision) problems. By solving each of these smaller problems only once, and storing the solutions so that they can be called upon when encountered a next time, the solution with the optimal reward is determined. In this case the decision problem is solved for each time period to either charge or discharge the battery and by which amount, given the current state.

Three cases were considered in the aforementioned thesis:

Case 1. Stand-alone battery model
: In this case, a household has a battery that can be (dis)charged in discrete steps in both time and charge. It is assumed that the energy is bought or sold on the market in such insignificant quantities that it does not influence the price. For each hour the decision is made for the amount of energy that will be (dis) charged from the battery, the energy prices are fixed.

Case 2. Home battery model
: This model is an elaboration of the previous model. The household now also has a (fixed) demand and a smart electric vehicle that also needs to be charged. The energy from the battery can be used to satisfy this demand at a later point in time.

Case 3. Price maker model
: This model is again an elaboration of the previous models. Now the battery is considered a price maker: its charging decisions influence the energy price. This can either be because it is large enough to have an influence on the system, or it is a collective group of batteries following the same decision pattern; they are a price maker together. In each of these cases, the decision also has to be made to either optimise for the users own benefit or the collective benefit of all the users combined.

This research leaves a lot of room for further research, by adding uncertainty in the demands, adding taxes for price makers, constructing a price function, or adding private generation of renewable energy. This method will be the first to be considered in this thesis. However, due to its potential to be time-consuming, other (faster) methods will be considered as well.

For the rest of this thesis, it is assumed that the consumer is not a price maker. Therefore, Case 2 will be used as a starting point for the comparison of the different optimisation methods. This case will be extended with a private energy source. The aspects of this case will be elaborated in chapter 3. In chapter 5 all the methods described in chapter 4 will be tested with this case and compared to each other on result and run time.

# 3 Problem definition

Before any method can be implemented, the problem needs to be defined mathematically. The goal of solving the problem is to determine when to charge or discharge a private battery to minimise the energy bill at the end of the month. This time period of one month is discretised in intervals of 15 minutes: $T = \{t_1, \dots, t_n\}$. For each of these intervals, a decision needs to be made for the usage of the battery. Whether to charge or discharge energy to it, or to not use it at all. If energy is (dis)charged to the battery, this amount of energy also needs to be determined.

In this thesis, the problem setting will be an office building with a private energy source and facilities to charge an electrical vehicle. However, the problem definition would still hold if all the variables were to be scaled up or down. Therefore, it is also applicable to any building with a private energy source, a private battery, a demand an a connection to the grid.

This chapter will start by introducing the variables and their restrictions to define this problem. This is followed by some example situations. The chapter will be concluded with the mathematical definition of the problem.

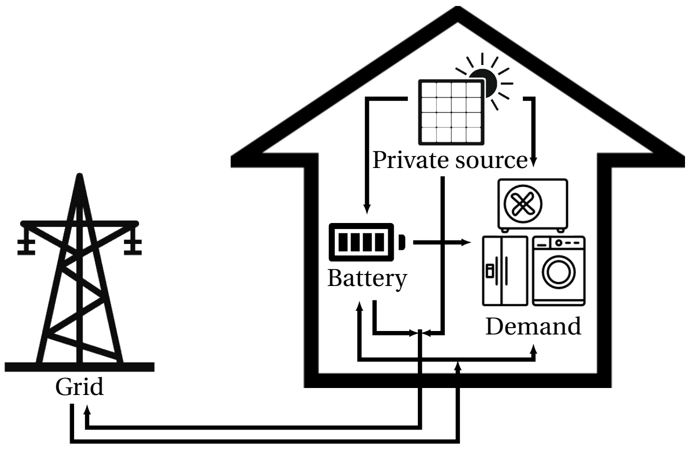


Figure 3.1: Schematic representation of the energy flow between the elements within an end user

## 3.1. Problem variables

The variables will be divided into two categories, independent and dependent variables. Independent variables are given (or estimated) at the start of the method, while independent variables have to be calculated with the use of other variables. When an independent variable has the same value for each of the time intervals, it is declared to be fixed. If a variable would need to be estimated, an uncertainty to this estimation would be in place. In this chapter it will be indicated which variables would have this uncertainty, however, for the rest of this thesis, the uncertainty will be ignored.

In figure 3.1 a schematic flow chart of all the aspects of the problem can be found. In the rest of this chapter all of these aspect will be highlighted and explained in more detail.

### 3.1.1. Independent variables

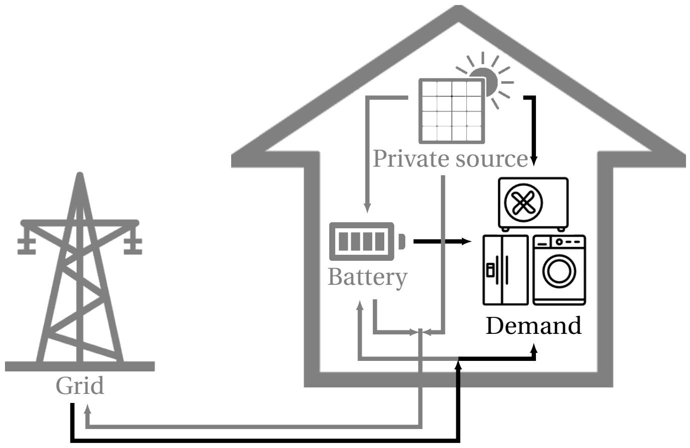


Figure 3.2: The energy demand of the end user

The first aspect of the problem to be discussed is the demand of the customer for each time interval $t$, this can be separated into the demand for the charging of electrical vehicles (EVs), $v_t$ and the rest of the users energy demand, $d_t$. This separation is made because of the potential of the electrical vehicles to be used as an additional battery from which energy could be discharged. In this thesis this possibility will not be considered, but for future research this variable is separated. Both of these demands should be real and larger than or equal to 0. The demand can not be known in advance with complete certainty, but can be estimated on prior usage or an indication of the user.

$$d_t, v_t \in \mathbb{R}_{\geq 0} \; \forall t \in T \tag{3.1}$$

In figure 3.2 the demand aspect is indicated in black. The demand can be satisfied with energy from the private source, with energy discharged from the battery or energy bought from the grid. Therefore, the sum of the energy flow of the arrows entering the demand node should be equal to the total demand. The values of the energy flow of each of the arrows is not known in advance, these will be discussed in the next section.

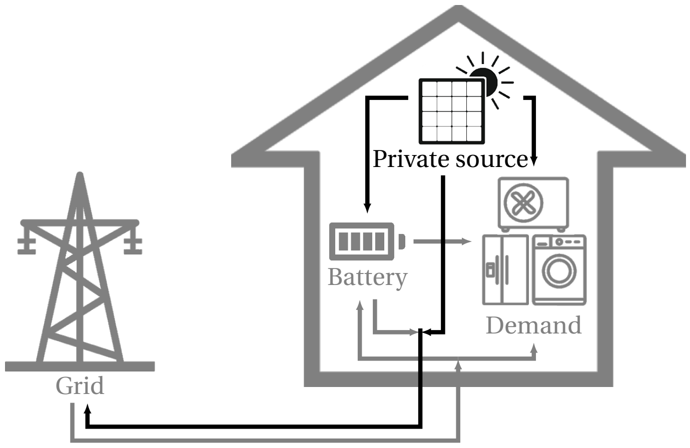


Figure 3.3: The renewable energy generation by the end user

The next variable is the amount of energy that is generated for each time interval. This variable is indicated by $g_t$ and should be real and larger than or equal to 0. The type of energy source does not need to be specified, only the generation per time interval. The generated amount of energy can be predicted, for example for solar panels this can be based on the weather forecast.

$$g_t \in \mathbb{R}_{\geq 0} \; \forall t \in T \tag{3.2}$$

In figure 3.3 the generation aspect is indicated in black. The generated energy can be charged to the battery, sold back to the grid, or used to meet the demand. Therefore, the sum of the energy flow of the arrows leaving the private source should be equal to the generation. The values of the energy flow of each of the arrows is not known in advance, these will be discussed in the next section.

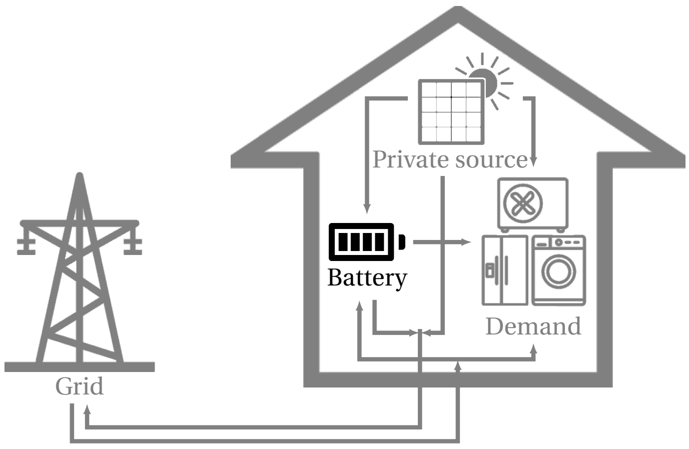


Figure 3.4: The specifications of the private battery

A battery is introduced in the problem to store the leftover generated energy for later usage. Currently, there exist multiple options when choosing a battery and due to the unknown possibilities of the future. The properties of the battery are implemented as variables to make sure it can be easily adapted for each type of battery. Five of the battery variables will introduced in this section.

In figure 3.4 the battery aspect is indicated in black. Because these variables are all properties of the battery, no energy flow arrows are indicated. The energy flow of the arrows connected to the battery will be discussed in the next section.

First up is the maximum capacity of the battery. This is the maximum amount of energy that can be stored within, this is indicated by $x_{max}$. This value is fixed for each time interval and should be real and larger than or equal to 0. The battery charge at the start of a time interval $t$ is indicated by $x_t$ and should be a real number between 0 and $x_{max}$. The value of $x_t$ is known with certainty at the start of interval $t$. However, the charging decisions are made prior to the start of the interval, thus the value is not known with complete certainty. The values at the start of previous intervals is known, as is the charging decision that is made for the previous interval. Combining these will give a close estimate to the value of $x_t$.

$$\begin{gathered} x_t, x_{max} \in \mathbb{R}_{\geq 0} \; \forall t \in T \\ x_t \leq x_{max} \; \forall t \in T \end{gathered} \tag{3.3}$$

**Definition 3.1** *The* MAXIMUM (DIS)CHARGING SPEED *is in this thesis defined as the maximum amount of energy that can be (dis)charged to a battery during one time interval.*

Besides the maximum capacity, each battery also has a limit on the (dis)charging speed, definition 3.1, defined as $u_{max}$. This value is the same for the charging and discharging of the battery. This is a fixed variable and should be real and larger than or equal to 0. The value of $u_{max}$ is known in advance. In real life, the (dis)charging of the battery is continuous, however for the simulations in this thesis it is discretised with a small step size, $u_{step}$. This indicates the minimum amount of energy that is (dis)charged to the battery, if it is (dis)charged at all. This step size is fixed and should be real and between 0 and $u_{max}$, equation 3.4.

$$\begin{gathered} u_{step} \in \mathbb{R} \\ u_{max} \in \mathbb{R}_{\geq 0} \\ 0 \leq u_{step} \leq u_{max} \end{gathered} \tag{3.4}$$

The last independent variable of the battery is the round-trip efficiency $\eta$. It indicates the ratio between the amount of energy stored in the battery and the amount of energy retrieved from the battery. This difference exists due to the conversion to DC-current for a battery and back to AC-current for the grid, and the loss of energy in the chemical process of storing and withdrawing energy in the battery. This variable is fixed and should be real and between 0 and 1.

The round-trip efficiency of the battery is taken into account by taking $\sqrt{\eta}$ with both charging and discharging of the battery. This will result in a total round-trip efficiency of $\eta$. For example, if $\eta = 0.16$ and $10kWh$ is charged and then discharged to the battery, then the resulting amount of energy should be equal to $0.16 \times 10 = 1.6$. If this is divided over both the charging and discharging by multiplying by $\sqrt{\eta} = \sqrt{1.6} = 0.4$, then this would result in $0.4 \times 10 = 4$ after charging and $0.4 \times 4 = 1.6$ after discharging.

$$\begin{gathered} \eta \in \mathbb{R} \\ 0 \leq \eta \leq 1 \end{gathered} \tag{3.5}$$

The last independent variables to be introduced are those relating to the energy price. The bare price of energy for a time interval $t$ will be indicated by $p_t$. The VAT is indicated by $\gamma$, if for example the VAT is equal to 10%, then $\gamma = 1.10$. The fixed tax cost per $kWh$ is indicated by $\tau$, this consists of both the energy tax and the tax for the storage of renewable energy. All of these variables should be real. The value of $p_t$ is known with certainty shortly before the relevant time interval and $\gamma$ and $\tau$ are fixed should be larger than or equal to 0.

$$\begin{gathered} p_t \in \mathbb{R} \; \forall t \in T \\ \gamma, \tau \in \mathbb{R}_{\geq 0} \end{gathered} \tag{3.6}$$

In table 3.1 an overview of all the independent variables and their unit is given. The next section will introduce and discuss all the dependent variables that need to be calculated with the use of other variables. When the independent variable contained some uncertainty, then this uncertainty is passed on to the variables that depend on it.

| Symbol | Description | Unit |
|---|---|---|
| $d_t$ | Demand of devices at time interval $t \in T$ | $kWh$ |
| $v_t$ | EV demand at time interval $t \in T$ | $kWh$ |
| $g_t$ | Amount of energy generated at time interval $t \in T$ | $kWh$ |
| $x_t$ | Battery charge at the start of time interval $t \in T$ | $kWh$ |
| $x_{max}$ | Total capacity of battery | $kWh$ |
| $u_{step}$ | Step size of the (dis)charging of energy | $kWh$ |
| $u_{max}$ | Maximum amount of energy that can be charged to the battery in one time interval | $kWh$ |
| $\eta$ | Round-trip efficiency | |
| $p_t$ | Energy price at time interval $t \in T$ | €/kWh |
| $\gamma$ | VAT | |
| $\tau$ | Fixed tax cost | €/kWh |

Table 3.1: Independent variables

### 3.1.2. Dependent variables

The first dependent variable is the amount of energy that is (dis)charged to the battery during time interval $t$, it is indicated by $u(t)$ and is a real number. This variable has a range of possible values. This range is defined by a lower and upper bound for each time interval $t$, these are indicated by $LB_t$ and $UB_t$ respectively. For the lower bound of this range, the battery can not be discharged more than its current charge or more than $u_{max}$. For the upper bound of this range, the battery can not be charged more than its capacity or more than $u_{max}$. As stated above, the charging of the battery would be continuous in real life. However, in this thesis it is discretised in steps of $u_{step}$, this results in equation 3.7.

$$
\begin{aligned}
LB_t &= \max\{-u_{max}, -x_t\}\ \forall t \in T \\
UB_t &= \min\{u_{max}, x_{max} - x_t\}\ \forall t \in T \\
u(t) &\in [LB_t : u_{step} : UB_t]\ \forall t \in T
\end{aligned}
\tag{3.7}
$$

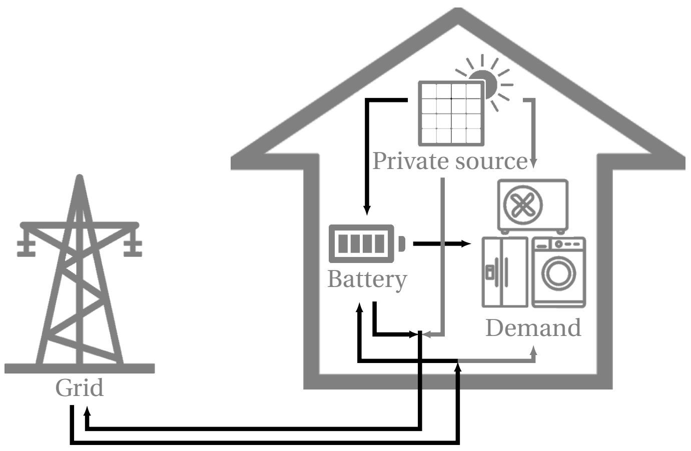


Figure 3.5: (Dis)charging of the private battery

In figure 3.5 the arrows that influence the value of $u(t)$ are highlighted. The arrows exiting the battery indicate the discharging of the battery and have a negative value. The arrows entering the battery indicate the charging of the battery and have a positive value. The arrows exiting the battery indicate the discharging of the battery and have a negative value. The variable $u(t)$ is equal to the sum of the values of these arrows.

As stated above, $x_t$ is the battery charge at the start of interval $t$ and $u(t)$ is the amount of energy that is charged to the battery during interval $t$. Thus, the battery charge at the start of interval $t+1$ is equal to the sum of these two variables.

$$x_{t+1} = x_t + u(t)\ \forall t \in T \tag{3.8}$$

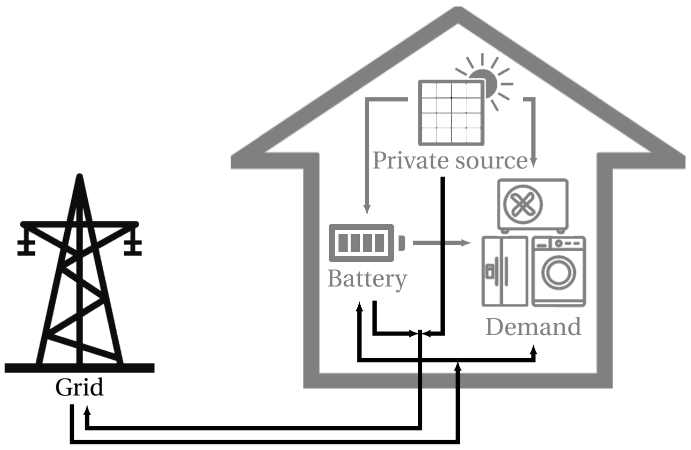


Figure 3.6: Trading energy with the grid

The next dependent variable is the amount of energy that is traded with the grid at each time interval $t$, this is indicated by $a(t)$. This value is what is leftover when the demand is subtracted from the generation and settled with the (dis)charging of the battery. If the remaining amount is positive, then more energy is necessary to meet the demand or charge to the battery, thus this is bought from the grid. If the remaining amount is negative, then this is excess energy that is sold back to the grid. As stated in the previous section, the round-trip efficiency of the battery is taken into account by multiplying with $\sqrt{\eta}$. This results in equation 3.9.

$$\mathbb{R} \ni a(t) = \sqrt{\eta}u(t) + v_t + d_t - g_t \; \forall t \in T \tag{3.9}$$

In figure 3.6 the arrows that influence the value of $a(t)$ are shown. The arrows exiting the grid indicate the buying of the battery and have a positive value. The arrows entering the battery indicate the selling of energy and have a negative value. The variable $a(t)$ is equal to the sum of the values of these arrows.

Next up is the buying and selling price of the energy that is being traded with the grid. The buying price of energy, $p_B(t)$ depends on the bare energy price and the taxes and is represented in equation 3.10.

$$\mathbb{R} \ni p_B(t) = \begin{cases} \gamma p_t + \tau & \text{if } p_t > 0 \\ p_t + \tau & \text{if } p_t \leq 0 \end{cases} \; \forall t \in T \tag{3.10}$$

In chapter 2 the difference between net metering and a feed-in tariff was explained. In this thesis, a feed-in tariff will be used to determine the selling price of energy. Most energy suppliers use the bare energy price as a feed-in tariff, thus this will also be used in this thesis. This results in equation 3.11.

$$\mathbb{R} \ni p_S(t) = p_t \; \forall t \in T \tag{3.11}$$

The last aspect of the problem definition, is the objective function. The eventual result of making decisions for charging the battery is located in the energy bill at the end of the month, this is determined by calculating the cost for each interval $t$. Thus, the objective function is defined as minimizing the sum of the cost of the trading with the grid of each time interval $t$. Thus, for each interval either the amount of energy bought from the grid is multiplied with the buying price or the amount of energy that has been sold back to the grid is multiplied with the selling price, this results in equation 3.12.

$$\mathbb{R} \ni C(t) = \begin{cases} a(t)p_B(t) & \text{if } a(t) > 0 \\ a(t)p_S(t) & \text{if } a(t) \leq 0 \end{cases} \; \forall t \in T \tag{3.12}$$

In table 3.2 an overview of all the dependent variables and their unit is given.

| Symbol | Description | Unit |
|---|---|---|
| $u(t)$ | Amount of energy that is charged to the battery at time interval $t \in T$ | $kWh$ |
| $a(t)$ | Amount of energy traded with the grid at time interval $t$ | $kWh$ |
| $p_B(t)$ | Price of buying energy from the grid at time interval $t$ | €$/kWh$ |
| $p_S(t)$ | Price of selling energy back to the grid at time interval $t$ | €$/kWh$ |
| $C(t)$ | Cost of trading with the grid at time interval $t$ | € |

Table 3.2: Dependent variables

## 3.2. Example situations

To illustrate the possible energy flows, this section includes four example situations. First up is a situation where the private energy source generates enough energy to meet the demand and the excess energy that has been generated is stored in the battery for later usage, see figure 3.7.

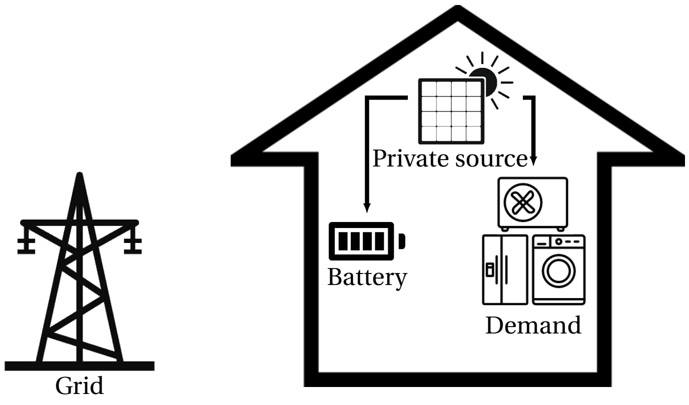


Figure 3.7: Example: meeting demand with energy from the private source and the excess is charged to the battery

Next up is the case when the private source does not generate enough energy to meet the demand. Now energy is being discharged from the battery to cover this lack of energy, see figure 3.8.

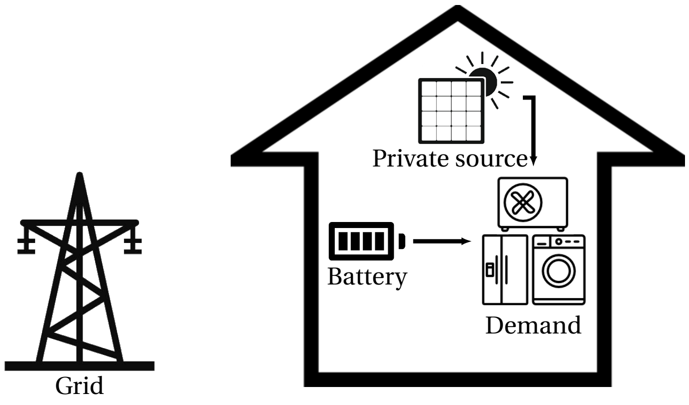


Figure 3.8: Example: meeting demand with energy from the private source and by discharging the battery

When no energy is being generated by the private source, for example with solar panels at night time, then energy can be bought from the grid to meet the demand. At night, the energy cost is low, thus extra energy can be bought and stored to the battery for later usage when the energy prices go up again, see figure 3.9.

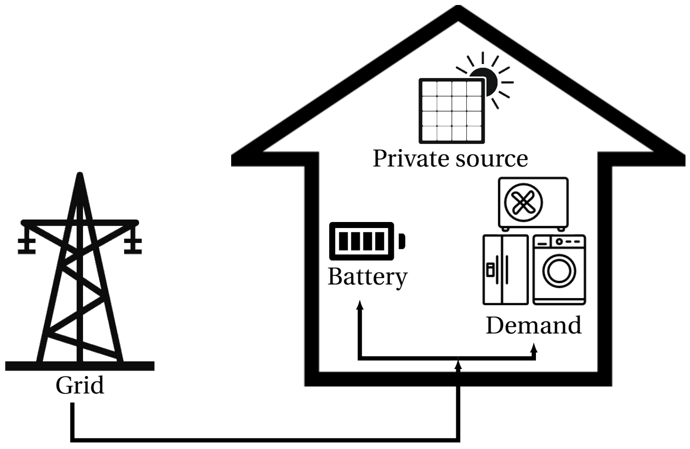


Figure 3.9: Example: meeting demand with energy from the private source and by buying from the grid, extra energy is bought to charge to the battery

However, if the energy source is not producing energy when the prices are very high, then it would be a more profitable decision to use energy that had been stored in the battery to meet the demand. Energy from the battery can then also be sold back to the grid for a higher profit, see figure 3.10.

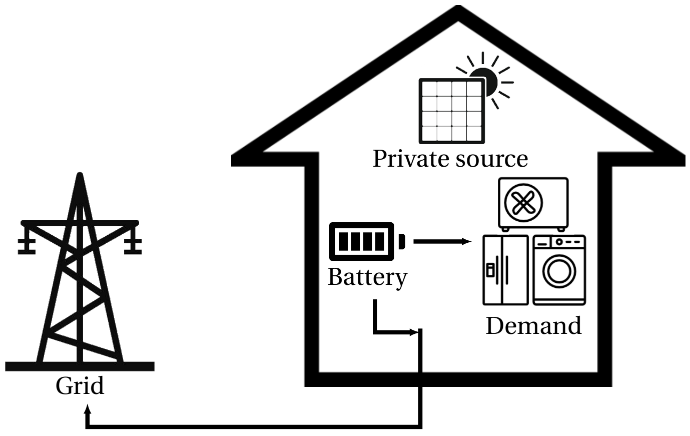


Figure 3.10: Example: meeting demand with energy from the private source and selling the excess back to the grid

## 3.3. Mathematical problem definition

Combining the variables and formulas given in the previous section results in the minimisation problem described in equation 3.13. For this definition holds $u = (u(t_1), \ldots, u(t_n))$.

To recap, the goal of the problem was to minimize the total cost at the end of the month. Therefore, the optimisation problem is a minimisation and the objective function is $\sum_{t \in T} C(t)$ with $C(t) = \begin{cases} a(t) p_B(t) & \text{if } a(t) > 0 \\ a(t) p_S(t) & \text{if } a(t) \leq 0 \end{cases}$.

One of the main requirements, is for the energy demand to be met. This energy can be generated by the private energy source, discharged from the battery, or bought from the grid. When there is an excess of energy, this can be stored in the battery or sold back to the grid. In the end, no energy can be lost, which results in the equation: $a(t) - \sqrt{\eta} u(t) = v_t + d_t - g_t$.

The charge of the battery at the start of an interval is equal to the charge of the battery at the start of the previous interval combined with the (dis)charged amount: $x_{t+1} = x_t + u(t)$.

The charge of the battery should be larger than or equal to 0 and smaller than or equal to the maximum capacity of the battery: $x_t \geq 0$ and $x_t \leq x_{max}$.

For the (dis)charging of the battery, a lower and an upper bound were set. The lower bound was $LB_t = \max\{-u_{max}, -x_t\}$ and the upper bound was $UB_t = \min\{u_{max}, x_{max} - x_t\}$. Therefore, four different inequalities can be derived: $u(t) \geq -u_{max}$, $u(t) \geq -x_t$, $u(t) \leq u_{max}$, and $u(t) \leq x_{max} - x_t$.

The three variable that need to be determined for each time interval $t \in T$ are $a(t)$, $x_t$, and $u(t)$. Each of these variables must be a real number. The charging of the battery is discretised for the simulations. However, this is not incorporated in the mathematical problem definition, because this definition represents the real-life problem.

$$\begin{aligned}
\underset{u}{\text{minimize}} \quad & \sum_{t \in T} C(t) \\
\text{subject to} \quad & a(t) - \sqrt{\eta} u(t) = d_t + v_t - g_t, && \forall\, t \in T, \\
& -x_{t+1} + x_t + u(t) = 0, && \forall\, t \in T, \\
& x_t + u(t) \leq x_{max}, && \forall\, t \in T, \\
& -x_t - u(t) \leq 0, && \forall\, t \in T, \\
& u(t) \leq u_{max}, && \forall\, t \in T, \\
& -u(t) \leq u_{max}, && \forall\, t \in T, \\
& x_t \leq x_{max}, && \forall\, t \in T, \\
& -x_t \leq 0, && \forall\, t \in T, \\
& a(t), x_t, u(t) \in \mathbb{R}, && \forall\, t \in T
\end{aligned} \tag{3.13}$$

# 4 Optimisation methods

In this chapter, four different mathematical optimisation methods will be introduced and adapted such that they could be used to solve the problem presented in equation 3.13. These methods are dynamic programming, local search, tabu search, and simulated annealing. The methods will be presented as minimisation methods, because the problem discussed in this thesis is also a minimisation.

The goal for the use of these methods is to find a feasible solution, see definition 4.1, that minimises the objective function (so the most profitable charging strategy) as fast as possible. Dynamic programming is known to always give the optimal reward solution. However due to the fact that each possible battery charge for each time interval is considered, it can take a very long time to compute. Heuristics methods are known to have a shorter run time, but do not always give the optimal reward solution. Local search was chosen as the first heuristic method due to the fact that it is simple to implement and if there are no local optima it will always give the optimal value solution. Because tabu search is an extension of local search this was chosen as the next method. Simulated annealing was chosen because of its reputation of being able to quickly solve for large, discrete spaces. [45]

**Definition 4.1** *A* SOLUTION *is an ordered set of numbers that indicate the battery charge at the end of each time interval. A solution is* FEASIBLE *when there is no negative charge, no charge larger than the maximum capacity of the battery, and when the absolute difference between the charges of two consecutive time intervals is less than the maximum charging speed.*

The solution space of the problem, is represented as the set of possible battery charge values for each time $t$. For the heuristic methods, the neighbours of a solution are considered, see definition 4.2. The neighbourhood of a solution $S$, $\mathcal{N}(S)$, is the set of all the neighbours of $S$. In figure 4.1 an example of a solution with all its neighbours is represented, its neighbourhood.

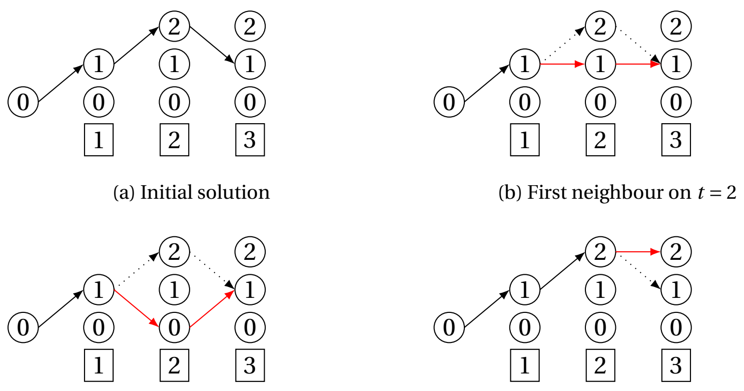
**Definition 4.2** *A* NEIGHBOUR *of solution $S$ is defined as a feasible solution that has a different battery charge at one time $t$ than $S$.*

To keep the example simple, the starting charge, $u_0$, is set to 0, $x_{max} = 2$, $u_{step} = 1$ and $u_{max} = 1$. Therefore, the possible decisions for each time instance are either charging by 1, discharging by 1 or doing nothing. The red arrows in the neighbour images indicate the difference from the initial solution.

Now for each time interval the possible charging en discharging decisions will be discussed to determine if they result in a feasible solution and therefore a neighbour.

- For the time interval 0, the charge $u_0$ was fixed. Therefore, there are no neighbours with a different charge for interval 0.

- Considering time interval 1. If the charge $u_1$ were to be increased to 2, then it is an infeasible solution because $|u_0 - u_1| = 2 > 1 = u_{max}$. If the charge $u_1$ were to be decreased to 0, then $|u_1 - u_2| = 2 > 1 = u_{max}$. Thus there are no neighbours with a different charge for interval 1.


(a) Initial solution (b) First neighbour on $t = 2$

(c) Second neighbour on $t = 2$ (d) Neighbour on $t = 3$

Figure 4.1: Example of a neighbourhood

- Next is time interval 2. If the charge $u_2$ were to be increased to 3, then it is an infeasible solution because $u_2 = 3 > 2 = x_{max}$, $|u_1 - u_2| = 2 > 1 = u_{max}$, and $|u_2 - u_3| = 2 > 1 = u_{max}$. When the charge $u_2$ is decreased to 1 or 0, then these are feasible solutions, thus neighbours of the initial solution, see Figure 4.1b and Figure 4.1c. If the charge of $u_2$ were to be decreased to $-1$, then it is an infeasible solution because $u_2 = -1 < 0$, $|u_1 - u_2| = 2 > 1 = u_{max}$, and $|u_2 - u_3| = 2 > 1 = u_{max}$. Thus there are two neighbours with a different charge for interval 2.
- Finally, time interval 3. When the charge $u_3$ is increased to 2, a feasible solution is found, see Figure 4.1d. If the charge of $u_3$ were to be increased to 3, then $u_3 = 3 > 2 = x_{max}$ and $|u_2 - u_3| = 2 > 1 = u_{max}$. If the charge of $u_3$ were to be decreased to 0, then it is an infeasible solution because $|u_2 - u_3| = 2 > 1 = u_{max}$. Thus there is one neighbour with a different charge for interval 3.

## 4.1. Dynamic programming

Dynamic programming starts at the last interval of the considered time period. For this interval, there exists a set of possible charge values that the battery could have at the start of the interval. For each of these values, a value at the end of the interval has to be determined. This is done by choosing the value that results in the lowest cost. This determined change in charge and the cost are saved.

Then working backwards in time, each of the other intervals is considered. Again for each of the possible charge values that the battery could have at the start of the interval, the value at the end of the interval is determined. This time not only the cost for the current interval, but also the cost of all the subsequent intervals have to be considered. The cost of the subsequent intervals for each value is already determined in previous steps and can be taken from where it is stored. This new chain of battery charge values for each interval and the total cost is saved again.

When all the time periods have been considered, the solution originating from the starting charge with the lowest total cost is selected as the final solution. [45]

An algorithmic representation can be found in algorithm 4.1. In this algorithm, the function $C(x)$ determines the cost value for the new charge $x$ and $Cost(i, j)$ stores this value for interval $i$ with starting charge $j$. A numerical example of this method can be found in appendix A.1.

**Algorithm 4.1:** DYNAMIC PROGRAMMING

```
$P$=number of time instances
$i = T$
for $j = 0 : stepsize : x_{max}$ do
    $m^- = \max\{-u_{max}, -j\}$
    $m^+ = \min\{u_{max}, x_{max} - j\}$
    $x_i = j + \arg\min_{k\in[m^-,m^+]}\{C(j+k)\}$
    $Cost(i,j) = C(x_i)$
end
for $i = T-1 : -1 : 0$ do
    for $j = 0 : stepsize : x_{max}$ do
        $m^- = \max\{-u_{max}, -j\}$
        $m^+ = \min\{u_{max}, x_{max} - j\}$
        $x_i = j + \arg\min_{k\in[m^-,m^+]}\{C(j+k) + Cost(i+1, j+k)\}$
        $Cost(i,j) = C(x_i)$
    end
end
return solution with the lowest cost C
```

## 4.2. Local search

The heuristic method local search starts with a (random) feasible solution to the problem and looks at its neighbours for an improvement. The neighbour with the lowest total cost will then be chosen, but only if it is lower than the total cost of the current solution. Then the neighbours of the new solution will be considered. This process will be repeated until no improving neighbour can be found.[45]

The downside to this algorithm is the potential to end up in a local optimum. Where all of its neighbours have a higher total cost, but there might exist a completely different solution with a lower total cost, that can not be reached from this point.

An algorithmic representation can be found in algorithm 4.2. In this algorithm $C(S)$ represents the total cost of solution $S$. A numerical example of this method can be found in appendix A.2.

**Algorithm 4.2:** LOCAL SEARCH

```
Choose an initial solution $S$
while $\exists P \in \mathcal{N}(S)$ st $C(P) < C(S)$ do
    Set $S = \arg\min_{P\in\mathcal{N}(S)}\{C(P)\}$
end
return $S$
```

## 4.3. Tabu search

Tabu Search is an extension of local search. The difference between the methods can be found in the selection of the new solution to consider. For local search, the neighbour that improves the most on the current solution is chosen. For tabu search, the neighbour with the lowest total cost is selected, even if it does not improve on the current solution. During the entire process, the lowest cost solution that has been encountered is saved and selected as the final solution. A stopping criterion is introduced to determine when to end the method.

To prevent the possibility of entering a loop of solutions, recently visited solutions are declared ‘tabu’ for a set amount of iterations. This is done to prevent getting stuck in a local optimum, such as the problem can be with local search.[45]

The largest problem for this method is located in the way solutions are stored in the tabu list. Storing an entire solution in the list and checking each of these can take up a lot of time. In this case, the altered time interval is declared tabu. Therefore, any neighbour where this interval is changed has become tabu. This will exclude the previous solution, but potentially some other neighbours as well. This larger set of tabu solutions is tolerated in order to possibly prevent ending up in a local optimum.

The number of problems that are tabu, the size of the tabu list, can cause problems as well. If there is no limit on the size, or if it is just too big, checking the tabu list takes up a lot of time and it may occur that the global optimum can't be reached due to the tabu solutions blocking it. If it is too small, the tabu might not prevent getting stuck in a local optimum.

If no stopping criterion is placed upon this method, it will run indefinitely. The stopping criterion can either be a set number of iterations or a set number of iterations without finding an improvement for the best solution. Two algorithmic representations can be found in algorithm 4.3, on the left it runs for a set number of iterations, on the right until a set number of iterations without improvement on the best solution. In this algorithm $TL$ represents the list of tabu items, $TL_{max}$ the maximum size of this list, $\mathcal{N}(S) \setminus TL$ the neighbourhood without the tabu solutions, and $I$ the number of iterations for the stopping criterion. Both of these stopping criterion will be considered when simulating tabu search as a comparison. A numerical example of this method using the latter stopping criterion can be found in appendix A.3.

**Algorithm 4.3:** TABU SEARCH

```
Choose an initial solution $S$
Set $S_{best} = S$, $TL = \{\}$, $TL_{max}$, $I$
$counter = 0$
while $counter < I$ do
    $S' = \arg\min_{P \in \mathcal{N}(S) \setminus TL}\{C(P)\}$
    $i$ = altered time state
    Set $S = S'$
    if $C(S) < C(S_{best})$ then
        $S_{best} = S$
    end
    $counter{+}{+}$
    if $size(TL) < TL_{max}$ then
        $TL = TL \bigcup i$
    else
        $TL = TL(2:end) \bigcup i$
    end
end
return $S_{best}$
```

```
Choose an initial solution $S$
Set $S_{best} = S$, $TL = \{\}$, $TL_{max}$, $I$
$counter = 0$
while $counter < I$ do
    $S' = \arg\min_{P \in \mathcal{N}(S) \setminus TL}\{C(P)\}$
    $i$ = altered time state
    Set $S = S'$
    if $C(S) < C(S_{best})$ then
        $S_{best} = S$
        $counter = 0$
    else
        $counter{+}{+}$
    end
    if $size(TL) < TL_{max}$ then
        $TL = TL \bigcup i$
    else
        $TL = TL(2:end) \bigcup i$
    end
end
return $S_{best}$
```

## 4.4. Simulated annealing

Simulated Annealing is a heuristic method that is derived from nature. Heating a solid metal until it becomes a liquid and then very slowly decreasing the temperature again results in a solid metal with neatly arranged molecules. This method can be interpreted for this problem as over multiple iterations slowly approaching the global optimum by at the beginning choosing the next solution almost at random and in the end only choosing those whose total cost is better or not a lot worse than the total cost of the current solutions. Thus it gravitates to solutions with a low total cost over time.

From a starting solution, a random neighbour is selected. If this neighbour has a lower total cost than the current solution, it is chosen as a new solution. If it has a higher cost, it is chosen as the new solution with a probability depending on the difference in total cost of both the solutions and how far along in the process the check is being made. The further in the process, the lower the probability of choosing a worse solution.

The stopping criterion of the problem is dependent on four variables: a control parameter $P$, a stopping criterion $P_{min}$, a set number of iterations $M$, and decrease variable $\alpha$. For each value of $P$, $M$ iterations are executed, afterwards the value of $P$ is decreased by multiplying it by $\alpha$. This is repeated until $P$ becomes smaller than $P_{min}$. As the value of $P$ decreases, the probability of choosing a worse solution decreases as well. The smaller the difference between the total cost of the two solutions, the larger the probability of choosing the neighbour as the new solution. [45] The values for $P$, $P_{min}$, $M$, and $\alpha$ are of large influence on the running time and efficiency of the method. An algorithmic representation can be found in algorithm 4.4 and a numerical example of this method can be found in appendix A.4.

**Algorithm 4.4:** SIMULATED ANNEALING

```
Choose an initial solution $S$
Set $P$, $M$, $P_{min}$, $\alpha$
while $P > P_{min}$ do
    repeat
        Randomly select $S' \in \mathcal{N}(S)$
        if $C(S') < C(S)$ then
            $S = S'$
        else
            With probability $\exp\left(\frac{-|C(S')-C(S)|}{P}\right)$, set $S = S'$
        end
    until $M$ times;
    $P = \alpha \times P$
end
return $S$
```

## 4.5. Comparison

In table 4.1, an overview of the methods is given with their specifications regarding result and run time.

Dynamic programming is a method that checks each of the possible solutions, and will therefore find the global optimum. However, this might take a long time to compute. Local search looks from a starting solution to similar solutions with a lower objective function, it stops when none can be found. This method is not guaranteed to find the global optimum, it may end up in a local optimum. Tabu search is an extension of local search. It will also consider solutions with a worse objective function, this is done to prevent ending up in a local optimum. However, it still is not guaranteed to find the global optimum. A stopping criterion is necessary for this problem and is, together with the specifications of the tabu list, of great influence on the run time. Simulated annealing was the final considered method. It is based on annealing in nature and converges to the global optimum, but is also not guaranteed to achieve it. The stopping criterion for simulated annealing considers multiple variables, each of these variables influences the run time and the eventual solution that is found.

| Method | Reward | Run time |
|---|---|---|
| Dynamic programming | Global optimum | Depends on the size of the solution space |
| Local search | Local optimum<br>Potentially global optimum | Depends on the size of the neighbourhood |
| Tabu search | Local optimum<br>Potentially global optimum | Depends on:<br>- size of the neighbourhood<br>- stopping criterion<br>- size of the tabu list<br>- how tabu solutions are stored |
| Simulated annealing | Converges to global optimum<br>No guarantee | Depends on the stopping critertion<br>and therefore on the choice of variables |

Table 4.1: An overview of the specifications of the methods presented in this chapter

# 5 Simulations

In the previous chapter, four methods have been described to determine the charging decisions when using a battery in combination with a renewable energy source in a private setting. The goal is to minimize the total cost of the solution. To compare the methods, they were programmed in MATLAB, several simulations were run to determine the result and run time of the methods

This chapter will first describe the data that was used for these simulations and their sources. This is followed by the specifications for the simulations that were run. Next are the variations in variables for each of the methods that were simulated and their results. Based on these results, the methods are compared to each other on run time and result and a conclusion is constructed for the applicability of the methods.

## 5.1. Data

For each of the methods, a program was written in MATLAB to run the simulations. They were all run on a Lenovo Yoga 700-14ISK laptop. [17] The simulations were run with data for the month August of 2018, with time intervals of 15 minutes. These simulations were run after the month had ended, thus all the data was known with certainty. However, if the methods were to be implemented in real-life, then not all data would be known and uncertainties would be in place. These uncertainties are not taken into account for these simulations.

If one of these methods were to be implemented in real-life, a simulation would have to be run for at the start of each of the time intervals to make sure the charging decisions are still concise with potential newly added data. For a month with 31 days and time intervals of 15 minutes, this would be 2976 simulations for that month. Simulations were run for different forecast time. For each forecast time, the time intervals of the month are divided into sets of the size of this forecast time. A simulation is run for each of these sets, thus each time interval is only considered in one simulation. For example, if the forecast time of 1 day is taken, then a simulation is run for each day of the month. Thus 31 simulations in total. The methods will be compared to each other on average run time per iteration and the sum of the cost of all the iterations.

All the independent variables need to be assigned before the simulations can be run. The first independent variable is the demand of office building and the demand of charging electrical vehicles, $d_t$ and $v_t$. For the energy demand, the total usage of one of the buildings of Witteveen+Bos for August was used: 16,860 kWh. The usage for each interval of 15 minutes was not known. Therefore, this had to be approximated. The 'Nederlandse Energie Data Uitwisseling' (Dutch Energy Data Exchange) has defined a set of user profiles to estimate the energy usage of customers for each 15 minute interval of every day. The profile of categories E3B/E3C was chosen, because this encompassed the energy usage of the building. The data indicates for each 15 minute interval of the year, the percentage of total annual usage. Because only August is considered, the percentages are scaled to a month. [26] The data that is used for the simulations is visualized by figure 5.1.

For the demand of the charging of electric vehicles at the office, it is assumed that the company has the use of one Engie BusinessLine. This is a charging hub that regulates and distributes energy over a set of electric vehicles. It has a power of 11kW per hour, thus 2.75kW per 15 minutes, this is divided over all the vehicles that are present. It is assumed that the first vehicle arrives at 07:00 and the last vehicles leaves at 18:00, at night and in weekends, no vehicle is charging. [9] The data that is used for the simulations is visualized by figure 5.1, the summed data of both of the demands is presented as well.

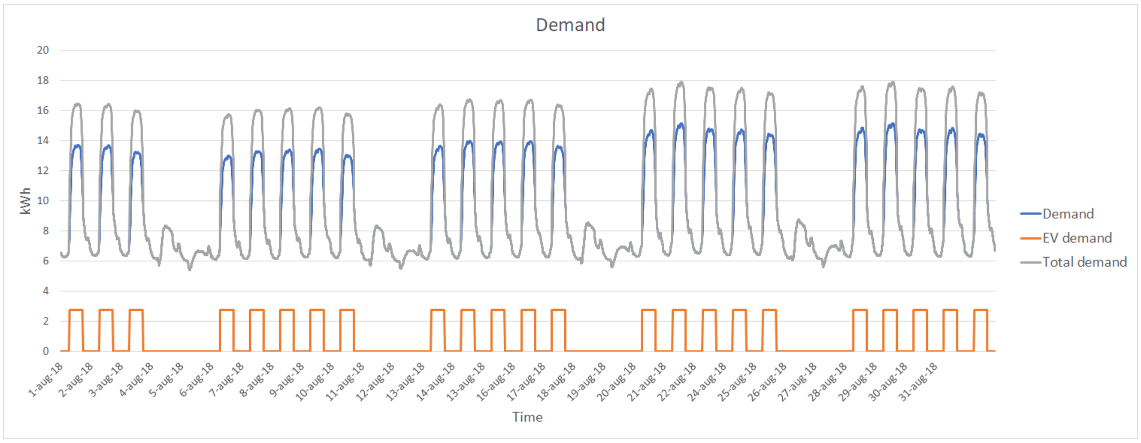


Figure 5.1: Total energy demand of August 2018

The next independent variable is the energy generation, $g_t$. The available generation data of renewable energy sources for the Netherlands was known for intervals of 10 minutes, while the generation data of wind power for Belgium was given for 15 minutes intervals. Since Belgium is located directly next to the Netherlands, it was assumed that the generated amounts would not be very different, thus the data from Belgium was used. However, this data is the total amount of generated wind power of Belgium, thus it was first scaled down from MW to kW and then divided by 20 to be a similar magnitude as the rest of the data. [7] The data that is used for the simulations is visualized by figure 5.2.

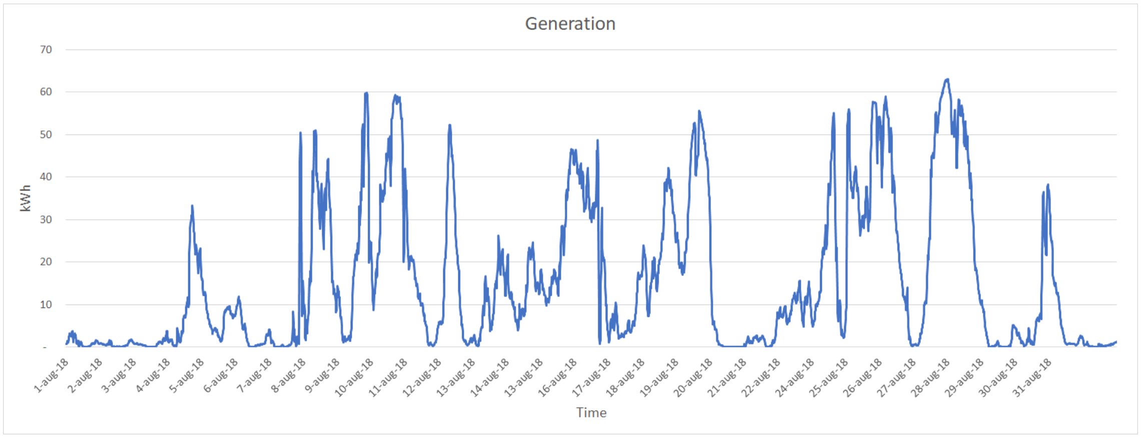


Figure 5.2: Energy generation of August 2018

As a battery, the Tesla Powerwall 2.0 was selected. This is a battery specifically designed to store leftover self-generated energy. This battery has a capacity of 13.5kWh, it has an efficiency of 90% and its power is equal to 5kW. Thus the maximum (dis)charging speed is set to be 1.25kWh for each interval of 15 minutes. [38] The battery charge at the beginning of the first day of the month is set to 0, simulating the case of introducing a battery to an existing system. The starting charge at the start of each of the following days will be set to the last charge of the battery of the previous day for continuity. An overview of the fixed variables for the battery can be found in table 5.1.

| Variable | Value |
|---|---|
| $\eta$ | 0.90 |
| $x_{max}$ | $13.5kWh$ |
| $u_{max}$ | $1.25kWh$ |
| $x_{start}$ | $0kWh$ |

Table 5.1: Values used for variables of the battery in the simulations

The last variables that need to be assigned are the energy prices. The buying and selling price, $p_B(t)$ and $p_S(t)$, are dependent variables, however they depend only on a set of independent variables that will be assigned in advance. The actual energy prices, $p_t$, of this day in the Netherlands are taken. These prices are per hour, thus for each of the 4 intervals of the hour, the same price is used. The price is given including 21% VAT, but excluding energy tax, which is set at 12.654 ct/kWh, and the tax for storage of renewable energy, 1.597 ct/kWh. The price without the VAT, $p_t$ , is calculated and the taxes are included as variables such that the buying and selling price can be determined. An overview of the these fixed variables for the energy price can be found in table 5.2. [6] The data that is used for the simulations is visualized by figure 5.3.

| Variable | Value |
|---|---|
| $\gamma$ | 1.21 |
| $\tau$ | 0.14251 |

Table 5.2: Values used for variables of the energy price in the simulations

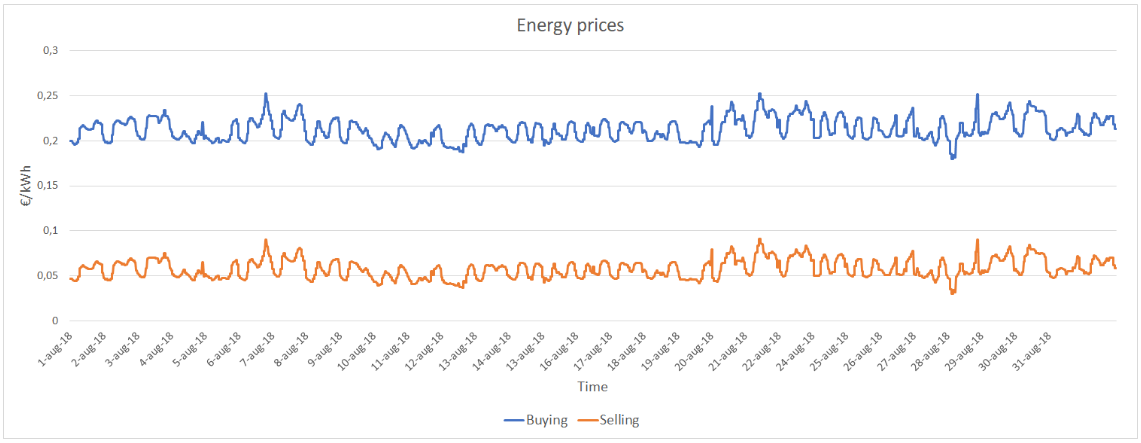


Figure 5.3: Buying and selling price of energy of August 2018

For local search, tabu search and simulated annealing, a valid initial solution has to be determined. This is set to be the solution where the battery is not used; nothing is charged or discharged from it. This solution is always valid.

For several of the data variables, such as the energy price and the generation data, an uncertainty is in place. This uncertainty becomes larger when the prediction is made for a moment further in the future. In these simulations, it is assumed that all the data is known. Therefore is this uncertainty excluded.

### 5.1.1. Data variations

To compare these methods with each other, multiple simulations will be run. The comparison criteria will be the run time and result. The simulations will vary from each other in forecast time and data ratios. If one of these methods were to be used in real-life, the maximal duration of the method would have to be less than 15 minutes to be done in time for the upcoming interval. However, the run time is not only dependent on the methods, but also on the computer it runs on. Thus, solutions found with any of these methods that exceed this time limit will not immediately be discarded.

The first aspect of the methods that will be varied, is the forecast time. These simulations will be run with the original data set. Due to the uncertainty of several of the data variables, the reliability of the results becomes smaller when the forecast time increases. This is not taken into account when running the simulations, but will be considered at the conclusion. The size of the neighbourhood depends on the forecast time, thus the differences in run time should be clearly visible. The chosen forecast times are:

- 1 hour
- 12 hours
- 1 day

- 1 week
- 1/2 month
- 1 month

Next is the ratio between the data for the demand, the energy price, and the generation. Because these aspects have a big influence as well, these are varied for each simulation. The methods are compared for seven simulations, with different variations for the generation and demand data, which will result in a different ratios between the values of demand, generation and price. They are all run with the same forecast time of 1 day. The different data variations and their abbreviation used in the tables are:

- The original data
  (G D)
- The generation data is multiplied by 10
  (G×10 D)
- The demand data is multiplied by 10
  (G D×10)
- The generation and demand data are multiplied by 10
  (G×10 D×10)
- The generation data is divided by 10
  (G/10 D)
- The demand data is divided by 10
  (G D/10)
- The generation and demand data are divided by 10
  (G/10 D/10)

To illustrate these data ratio variations, consider this example for one time instance. Let the generation be $g = 8$, the demand be $d = 5$, and the price be $p = 3$. In table 5.3 an overview of the values of each of these variables can be found for the different data ratio variations. By varying only the generation and the demand, the ratio between all three of the variables differs for each of the simulations.

| Data ratio / Example variables | G D | G×10 D | G D×10 | G×10 D×10 | G/10 D | G D/10 | G/10 D/10 |
|---|---|---|---|---|---|---|---|
| Generation ($g$) | 8 | 80 | 8 | 80 | 0.8 | 8 | 0.8 |
| Demand ($d$) | 5 | 5 | 50 | 50 | 5 | 0.5 | 0.5 |
| Price ($p$) | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

Table 5.3: Example of the data ratio variations

## 5.2. Results

The variations in forecast time and data ratio are run for each of the methods and the results and run time are rounded to integers. For each of these simulations, both the case of no battery usage and the worst case are determined as a context. In the case of no battery usage, no energy is charged or discharged to the battery. Any excess energy is directly sold back to grid and any energy shortage is covered by buying energy from the grid. For the worst case, the battery is in use, but the charging decisions are determined that result in the highest possible cost. Dynamic programming was used to determine this worst case, but with a maximisation instead of a minimisation. These results can be found for the forecast time and data ratio variations in table 5.4 and table 5.5 respectively.

In table 5.4, the total cost of the no battery case is for each of the simulations about the same value. This makes sense because it uses the same data set for each of the simulations. The differences in cost are caused by rounding errors. As stated earlier, the time intervals of the month are divided into set of the size of the forecast time for the simulations. The methods are run for each of these sets, and therefore optimise for this set. Therefore, the total cost for the worst case is maximized for each of these sets. Because the different variations have different run times, this results in different total cost for each of the simulations. For a forecast

| Case \ Forecast time | 1 hour | 12 hours | 1 day | 1 week | 1/2 month | 1 month |
|---|---|---|---|---|---|---|
| No battery | €1163 | €1163 | €1163 | €1161 | €1160 | €1163 |
| Worst case | €1174 | €1213 | €1215 | €1216 | €1215 | €1218 |

Table 5.4: Total cost in euros of the forecast time simulations with no battery and the worst case

| Case \ Data ratio | G<br>D | G×10<br>D | G<br>D×10 | G×10<br>D×10 | G/10<br>D | G<br>D/10 | G/10<br>D/10 |
|---|---|---|---|---|---|---|---|
| No battery | €1163 | €-22808 | €48146 | €7581 | €5351 | €-1953 | €566 |
| Worst case | €1215 | €-22756 | €48164 | €7629 | €5372 | €-1878 | €696 |

Table 5.5: Total cost in euros of the data ratio simulations with no battery and the worst case

time of 1 hour, only 4 time intervals are considered per simulation. Thus there are not a lot of possibilities to differ this solution from the no battery case. However, with the other forecast times, the number of time intervals do allow a greater difference from the total cost of the no battery case.

In table 5.5 a different data set is used for each of the variations, thus the results can not be directly compared to each other. However, it is clearly visible that the total cost increases for the variations where the ratio between the demand and generation results in a higher demand, and the total cost decreases for the variations where the ratio between the demand and generation results in a higher generation.

The simulations described in the previous section are run for each of the methods. For dynamic programming and local search, only one set of iterations has been run for each of the simulations because there is no variation in variables possible. However, for tabu search and simulated annealing several variables have to be chosen, thus multiple combinations of these variables are run to find the combination with the lowest total cost and the lowest average run time.

For tabu search and simulated annealing, different variables are considered until either the total cost of the solutions is ‘close enough’, see definition 5.1, or the run time exceeds the run time of dynamic programming for the same simulation. The lowest possible cost will be the solution found with dynamic programming, the highest possible cost is the worst case solution. For example, if the lowest possible cost is 0 and the highest possible cost is 100, then a solution would be close enough if its total cost would be smaller than or equal to 5. These close enough values will be determined in the next section when the total cost from dynamic programming is known.

**Definition 5.1** *The result of a solution is considered to be* CLOSE ENOUGH *to the lowest possible cost if, considering the difference between the highest and lowest possible cost, the cost of the solution has a difference of less than* 5% *from the lowest possible cost.*

### 5.2.1. Dynamic programming

The first method to be considered is dynamic programming, this will always result in the solutions with the lowest possible cost. The run times and cost for the forecast time variations can be found in table 5.6

| Result \ Forecast time | 1 hour | 12 hours | 1 day | 1 week | 1/2 month | 1 month |
|---|---|---|---|---|---|---|
| Average run time | 2s | 49s | 101s | 836s | 1806s | 3239s |
| Total cost | €1162 | €1122 | €1120 | €1115 | €1114 | €1117 |

Table 5.6: Average run time in seconds and total cost in euros of the forecast time simulations for dynamic programming

Similar with the worst case, the total cost for the simulations for a forecast time of 1 hour does not differ a lot from the no battery case. While for larger forecast times, there are more possibilities to differ from total cost of the no battery case. The average run time of the simulations increases as the forecast time increases. This is again to the fact that for larger forecast times, more time intervals are considered in a simulations and therefore there are more solutions to be considered.

Now that the lowest possible cost for the forecast time variations are determined, the minimum cost values for a solution to be close enough can be determined. In table 5.7 these minimum values are given for each of the forecast time variation.

| Result \ Forecast time | 1 hour | 12 hours | 1 day | 1 week | 1/2 month | 1 month |
|---|---|---|---|---|---|---|
| Minimum value | €1163 | €1127 | €1125 | €1120 | €1119 | €1122 |

Table 5.7: Benchmark values for the total cost to be close enough to the lowest possible cost for the forecast time simulations

In figure 5.4, the ranges of the cost for each of the forecast time variations are given. The dots represent from left to right the lowest possible total cost as determined with dynamic programming, the close enough benchmark, the total cost when no battery is used, and the highest possible total cost (worst case). When the total cost of a solution is located between the two leftmost dots, then it is considered to be close enough.

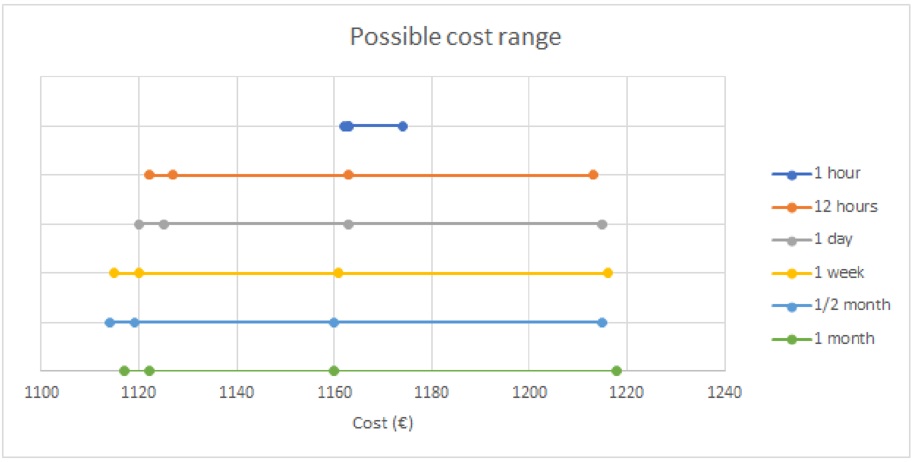


Figure 5.4: The cost ranges for each of the forecast time simulations.
The dots represent from left to right: lowest possible cost, close enough benchmark, cost without a battery, and highest possible cost

The run times and cost for the data ratio variations can be found in table 5.8.

| Result \ Data ratio | G D | G×10 D | G D×10 | G×10 D×10 | G/10 D | G D/10 | G/10 D/10 |
|---|---|---|---|---|---|---|---|
| Average run time | 101s | 90s | 117s | 98s | 117s | 89s | 102s |
| Total cost | €1120 | €-22853 | €48131 | €7537 | €5335 | €-1988 | €528 |

Table 5.8: Average run time in seconds and total cost in euros of the data ratio simulations for dynamic programming

The average run times of the simulations do not differ a lot. Because all of them have the same forecast time, the number of different solutions is the same. The small differences are caused by the differences in values which result in different solutions.

Now that the lowest possible cost for the data ratio variations are determined, the minimum cost values for a solution to be close enough can be determined. In table 5.9 these minimum values are given for each of the data ratio variation.

| Result \ Data ratio | G D | G×10 D | G D×10 | G×10 D×10 | G/10 D | G D/10 | G/10 D/10 |
|---|---|---|---|---|---|---|---|
| Minimum value | €1125 | €-22848 | €48133 | €7542 | €5337 | €-1983 | €536 |

Table 5.9: Benchmark values for the total cost to be close enough to the lowest possible cost for the forecast time variations

In figure 5.5, the ranges of the cost for each of the data ratio variations are given. The dots represent from left to right the lowest possible total cost as determined with dynamic programming, the close enough benchmark, the total cost when no battery is used, and the highest possible total cost (worst case). When the total cost of a solution is located between the two leftmost dots, then it is considered to be close enough.

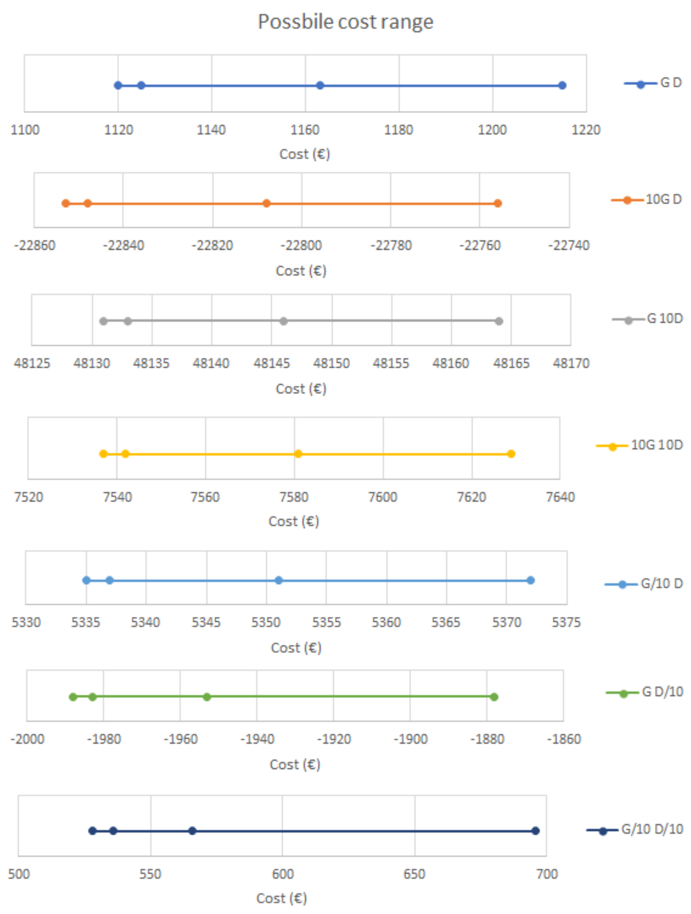


Figure 5.5: The cost ranges for each of the data ratio simulations.
The dots represent from left to right: lowest possible cost, close enough benchmark, cost without a battery, and highest possible cost

### 5.2.2. Local search

For local search, one simulation was run for each forecast time and data ratio variation. The run times and total cost for the forecast time variations can be found in table 5.10. The average run times that exceed those of dynamic programming as presented in table 5.6 and the total cost that subceed the minimum values determined in table 5.7 are made bold.

| Method \ Forecast time | 1 hour | 12 hours | 1 day | 1 week | 1/2 month | 1 month |
|---|---|---|---|---|---|---|
| Average run time | 0.03s | 18s | **133s** | **6583s** | - | - |
| Total cost | **€1162** | €1158 | €1158 | €1115 | - | - |

Table 5.10: Average run time in seconds and total cost in euros of the forecast time simulations for local search

For the forecast time of 1 week, the average run time exceeds the run time of dynamic programming for the simulations for 1 week, 1/2 month, and 1 month. Due to the fact that the run time of local search is longer if the size of the neighbourhood becomes larger, it follows that the simulations for 1/2 month and 1 month will also exceed the run time of dynamic programming. Thus these simulations are not run. For the forecast times of 1 hour the total cost is close enough to the lowest possible total cost. However, for the other forecast time variations, it is not.

In figure 5.6, scatter plots are presented for each of the simulation that were run. In these plot the total cost is set out against the average run time. The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming. Thus the goal is for the solution to end up in the lower left corner.

The run times and total cost for the forecast time variations can be found in table 5.11. The average run

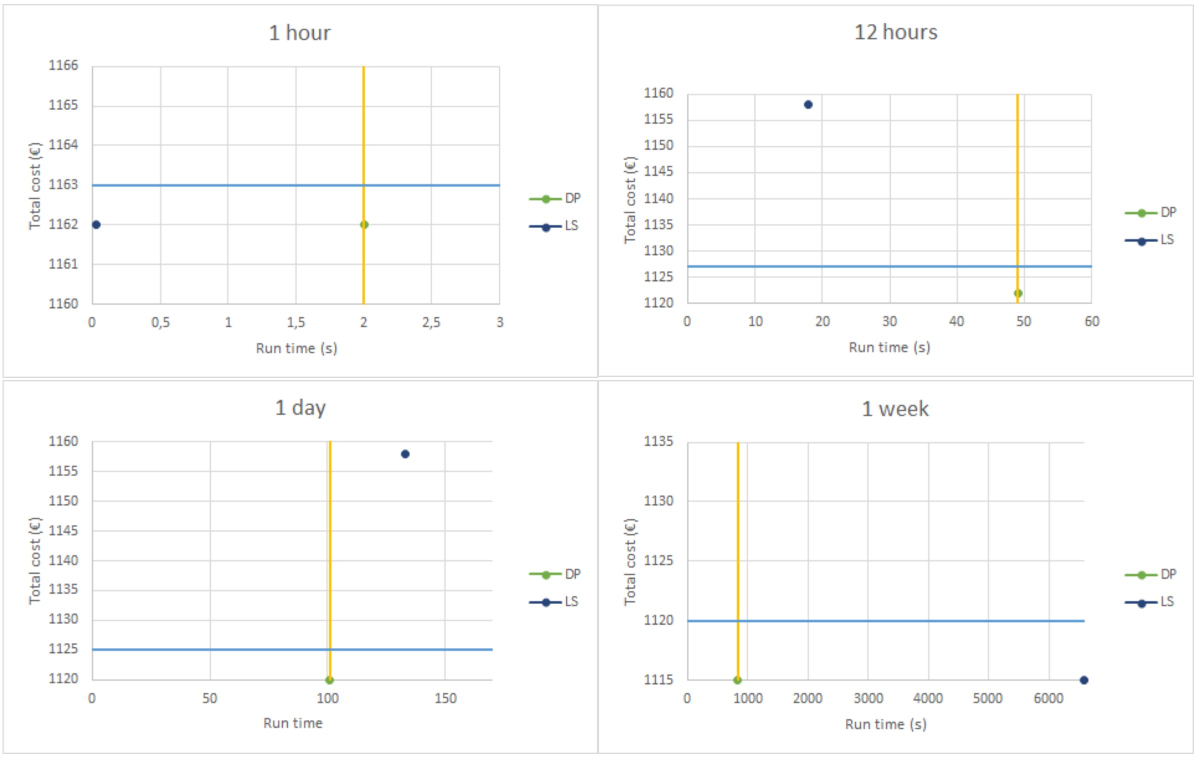


Figure 5.6: Scatter plots of the results of forecast time simulations of local search.
The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming.

times that exceed those of dynamic programming as presented in table 5.8 and the total cost that subceed the minimum values determined in table 5.9 are made bold.

| Data ratio / Method | G<br>D | G×10<br>D | G<br>D×10 | G×10<br>D×10 | G/10<br>D | G<br>D/10 | G/10<br>D/10 |
|---|---|---|---|---|---|---|---|
| Average run time | **133s** | **97s** | 92s | 79s | 113s | **98s** | **182s** |
| Total cost | €1158 | €-22814 | €48145 | €7575 | €5349 | €-1956 | €564 |

Table 5.11: Average run time in seconds and total cost in euros of the data ratio simulations for local search

For about half of the data ratio simulations, the run time exceeds the run time of dynamic programming. The total cost of the simulations was in none of the variations close enough to the lowest possible cost.

In figure 5.10 at the end of this chapter, scatter plots are presented for each of the simulation that were run. In these plot the total cost is set out against the average run time. The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming. Thus the goal is for the solution to end up in the lower left corner.

Thus for most of the iterations with local search, the run time exceeded the run time of dynamic programming and the results are not the global optimum. A change in initial solution could improve on the result, but finding this other solution would take extra time. Besides, there is no guarantee that a solution that quickly finds the optimum for one simulation will also yield the same result in another simulation.

### 5.2.3. Tabu search

For tabu search there were two possible options for a stopping criterion: a total number of iterations ('total criterion') or a number of consecutive iterations without finding an improvement on the lowest cost solution found so far ('unchanged criterion'). For both of these options the forecast time, the size of the tabu list and the number of iterations can be varied.

For the size of the tabu list, three different options were considered: $\{0.25T, 0.5T, 0.75T\}$. Where $T$ was the number of time intervals. For each of these options for list size, the number of iterations were first set to $0.5T$ and simulations were run. Then the simulations were run with each time an increase of $0.5T$ for the number of iterations until either the run time exceeded the run time of dynamic programming from the respective simulations found in table 5.6 and table 5.8, or until the total subceeded the values from table 5.7 and table 5.9.

As stated before, tabu search is an extension of local search, it continues when local search stops in a local optimum. One of two situations can occur. First, if the selected number of iterations is less than the number of iterations necessary for local search to reach the local optimum, then the cost of the solution will be worse than the cost of the solution found with local search. Due to the fact that the method is stopped before the local optimum has been reached. With the exception of the forecast time of 1 hour, the cost of the solutions found with local search are not close enough to the lowest possible cost. Thus the solutions found with tabu search would also not be close enough. Second, if the selected number of iterations is more than or equal to the number of iterations necessary for local search to reach the local optimum, then the run time of tabu search will exceed the run time of local search. Due to the fact that more iterations are run, which requires more time. Because for several of the simulations the run time of local search already exceeded the run time of dynamic programming, it follows that the run time for tabu search will also exceed the run time of dynamic programming. Therefore, there exists a set of simulations where for local search the run time exceeds the run time of dynamic programming and the total cost is not close enough. For this set of simulations, no simulations were run for tabu search.

It is clear that the simulations with the unchanged criterion will have at least as many iterations as the simulations with the total criterion. Therefore, the run time for the unchanged criterion simulations will exceed the run time for the total criterion simulations. Thus, if the run time for the total criterion simulations already exceeds the run time of dynamic programming, then the simulations with the unchanged criterion will not be run as they will also exceed this run time.

The average run time and the total cost for the forecast time variations and the data variations can be found at the end of this chapter in table 5.12 and table 5.13 respectively. The average run time that exceeded the run time of dynamic programming as presented in table 5.6 and table 5.8 and the total cost that subceed the minimum values determined in table 5.7 and table 5.9 are made bold.

In figure 5.7, figure 5.8, and figure 5.9 scatter plots are presented for each of the simulation that were run. In these plot the total cost is set out against the average run time. Each of the dots represent a simulation. For each chosen value for $TL_{max}$, they represent the different values for the number of iterations. Because the run time increases with the number of iterations, they are in order from left to right in accordance with the values from the aforementioned result tables. The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming. Thus the goal is for the solution to end up in the lower left corner.

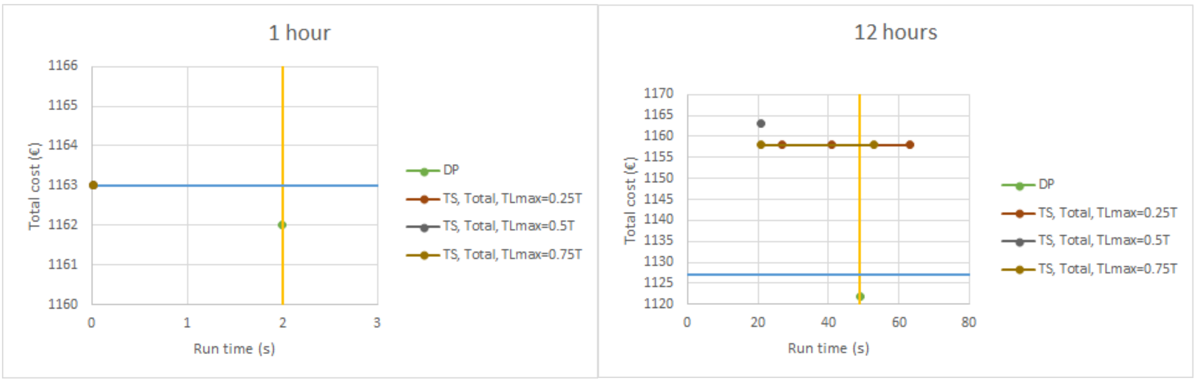


Figure 5.7: Scatter plots of the results of forecast time variations of tabu search with the total criterion.
The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming.

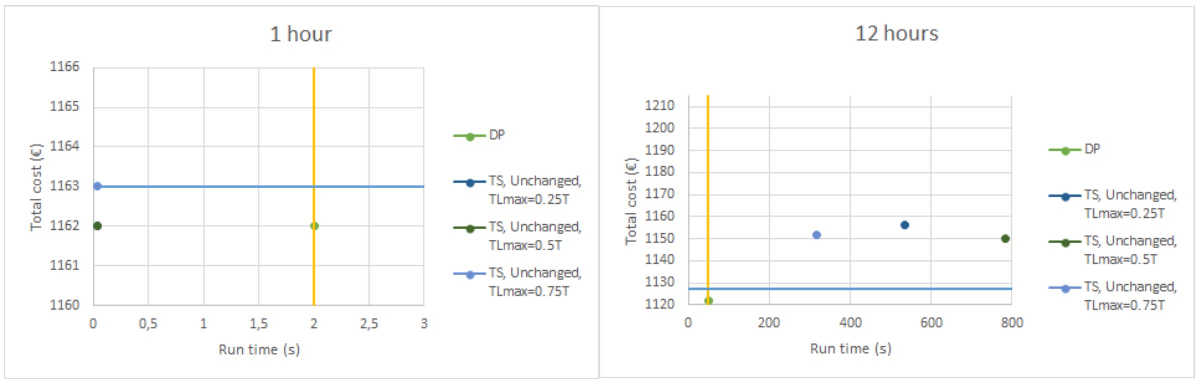


Figure 5.8: Scatter plots of the results of forecast time variations of tabu search with the unchanged criterion. The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming.

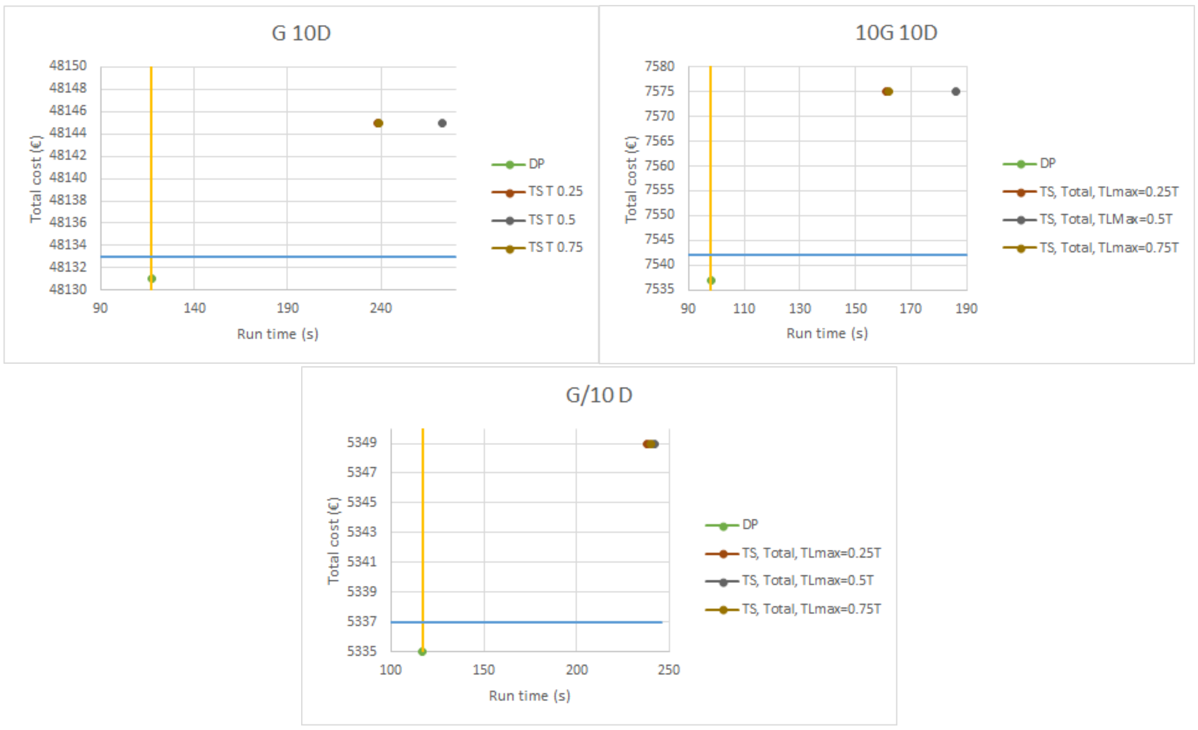


Figure 5.9: Scatter plots of the results of data ratio variations of tabu search with the total criterion. The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming.

For the forecast time of 1 hour, the total cost of the solution is close enough but not equal to the lowest possible cost solution. For every other forecast time and for each of the data variations, the average run time exceeds the run time of dynamic programming before the total cost became close enough.

Thus, for most of the simulations the average run time for tabu search exceeded the run time of dynamic programming. Just as for local search, changing the initial solution would not guarantee better results.

### 5.2.4. Simulated Annealing

For simulated annealing, four variables need to be defined. The control parameter $P$, stopping criterion $P_{min}$, set number of iterations $M$, and decrease variable $\alpha$. The control parameter $P$ is set to the largest absolute difference between the total cost of the initial solution and the total cost of its neighbours, such that each of these could be chosen as the nexzt solution. The stopping criterion $P_{min}$ is set to be equal to $\frac{P}{M}$.

The number of iterations $M$ and the decrease variable $\alpha$ are varied during the simulations. The value of $\alpha$ is usually located between 0.8 and 0.99. For the simulations, three different values for $\alpha$ were chosen: $\{0.8, 0.9, 0.99\}$. For each of these values of $\alpha$, the number of iterations $M$ is first set to $T$. For the next combination, $M$ is set to $2T$ and for each of the following simulations it is increased again by $2T$. These simulations are run until either the run time exceeds the run time of dynamic programming from table 5.6 and table 5.8, or until a solution has been found whose total cost is smaller than the values determined in table 5.7 and table 5.9. The average run time and the total cost for the forecast time and data ratio variations can be found at the end of this chapter in table 5.14 and table 5.15 respectively. The run times that exceeded and the total cost that subceeded the aforementioned values, are made bold.

In figure 5.11 and figure 5.12 at the end of this chapter, scatter plots are presented for each of the simulation that were run. In these plots the total cost is set out against the average run time. Each of the dots represent a simulation. For each chosen value for $\alpha$, they represent the different values for the number of iterations $M$. Because the run time increases with the number of iterations, they are in order from left to right in accordance with the values from the aforementioned result tables. The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming. Thus the goal is for the solution to end up in the lower left corner.

For the lower forecast times variations, the cost of the solution comes close enough to the lowest possible cost before the run time exceeds that of dynamic programming. However, for forecast times of 1 week or longer, the run time exceeds that of dynamic programming before the cost of the solution becomes close enough.

Since for these simulation the assumption is made that all the data for the entire month is known, large forecast times could be used for the simulations. However, in real-life there will always be an uncertainty in the data. This uncertainty increases, the further away the considered time interval is. This would result in an unreliable prediction of the eventual cost when large forecast times are used.

For each value of $\alpha$, increasing the value of $M$ always results in solutions with a total cost close enough to the lowest possible cost. The average run time of all the simulations differs for the different values of $\alpha$ and $M$. The larger the value of $\alpha$ or $M$, the longer the average run time. The value of $\alpha$ is of more influence, thus this should first be chosen to be small while $M$ is increased.

## 5.3. Comparison

Now the different methods will be compared to each other. Dynamic programming always results in the solutions with the lowest possible cost. For local search the solutions that were found were usually local optima that were not close enough to the lowest possible solution. Also, the average run times exceeded those of dynamic programming most of the times. Tabu search is an extension of local search. When its run time is less than that of local search, its final solution will have a higher cost. Local search often exceeded the run time of dynamic programming, consequently the same holds for tabu search. Therefore, these two heuristic methods would not be advisable alternatives to dynamic programming.

The last considered method was simulated annealing, several variables can be varied to compare their results to each other. Combinations are found for which the total cost of the solution comes very close to the lowest possible cost. When considering the forecast time variations, it shows that when the forecast time is 1 day or less, the run time of simulated annealing is always less than the run time of dynamic programming. However, when the forecast time becomes 1 week or more, the run time with the selected variables exceeds the run time of dynamic programming. Due to the uncertainty of several of the data variables, a forecast time of more than a week will probably yield unreliable results.

In conclusion, when taking a forecast time of a day or less, a fixed, small value of $\alpha$ and the variable of $M$ sufficiently high, then simulated annealing will result in a solution that has a total cost that is close enough to the lowest possible cost and its run time will be lower than that of dynamic programming.

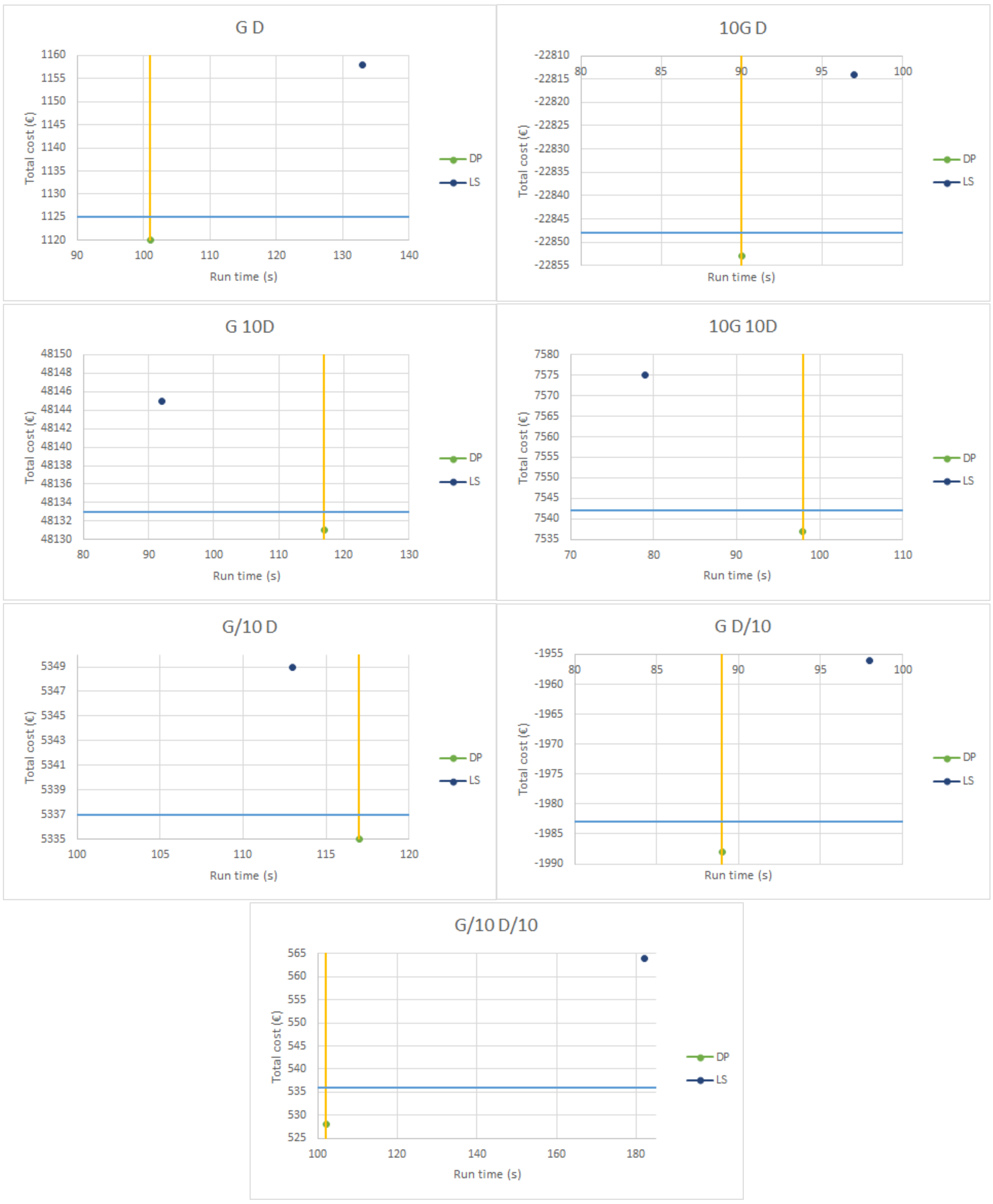


Figure 5.10: Scatter plots of the results of data ratio variations of local search.
The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming.

| Forecast time / Method | 1 hour | | 12 hours | | 1 day | | 1 week | | 1/2 month | | 1 month | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Total, list size=$0.25T$, iterations=$0.5T$ | 0.02s | **€1163** | 27s | €1158 | **168s** | €1158 | - | - | - | - | - | - |
| Total, list size=$0.25T$, iterations=$T$ | - | - | 41s | €1158 | - | - | - | - | - | - | - | - |
| Total, list size=$0.25T$, iterations=$1.5T$ | - | - | **53s** | €1158 | - | - | - | - | - | - | - | - |
| Total, list size=$0.5T$, iterations=$0.5T$ | 0.02s | **€1163** | 21s | €1158 | **156s** | €1158 | - | - | - | - | - | - |
| Total, list size=$0.5T$, iterations=$T$ | - | - | **53s** | €1158 | - | - | - | - | - | - | - | - |
| Total, list size=$0.75T$, iterations=$0.5T$ | 0.02s | **€1163** | 21s | €1158 | **157s** | €1158 | - | - | - | - | - | - |
| Total, list size=$0.75T$, iterations=$T$ | - | - | **53s** | €1158 | - | - | - | - | - | - | - | - |
| Unchanged, list size=$0.25T$, iterations=$0.5T$ | 0.03s | **€1163** | **535s** | €1156 | - | - | - | - | - | - | - | - |
| Unchanged, list size=$0.5T$, iterations=$0.5T$ | 0.03s | **€1163** | **783s** | €1150 | - | - | - | - | - | - | - | - |
| Unchanged, list size=$0.75T$, iterations=$0.5T$ | 0.03s | **€1163** | **317s** | €1152 | - | - | - | - | - | - | - | - |

Table 5.12: Average run time in seconds and total cost in euros of the forecast time simulations for tabu search

| Data ratio / Method | G D | | G×10 D | | G D×10 | | G×10 D×10 | | G/10 D | | G D/10 | | G/10 D/10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Total, list size=0.25$T$, iterations=0.5$T$ | - | - | - | - | **238s** | €48145 | **161s** | €7575 | **238s** | €5349 | - | - | - | - |
| Total, list size=0.5$T$, iterations=0.5$T$ | - | - | - | - | **273s** | €48145 | **186s** | €7575 | **242s** | €5349 | - | - | - | - |
| Total, list size=0.75$T$, iterations=0.5$T$ | - | - | - | - | **239s** | €48145 | **162s** | €7575 | **240s** | €5349 | - | - | - | - |
| Unchanged, list size=0.25$T$, iterations=0.5$T$ | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Unchanged, list size=0.5$T$, iterations=0.5$T$ | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Unchanged, list size=0.75$T$, iterations=0.5$T$ | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

Table 5.13: Average run time in seconds and total cost of euros of each of the data ratio simulations for tabu search

| Method \ Forecast time | 1 hour | | 12 hours | | 1 day | | 1 week | | 1/2 month | | 1 month | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\alpha = 0.8$, $M = T$ | 0.009s | **€1163** | 0.5s | €1155 | 2s | €1154 | 152s | €1148 | 671s | €1146 | 2675s | €1148 |
| $\alpha = 0.8$, $M = 2T$ | - | - | 1s | €1150 | 4s | €1148 | 300s | €1142 | 1301s | €1138 | **5547** | €1141 |
| $\alpha = 0.8$, $M = 4T$ | - | - | 2s | €1143 | 9s | €1140 | **1058s** | €1132 | **2654s** | €1129 | - | - |
| $\alpha = 0.8$, $M = 6T$ | - | - | 3s | €1137 | 12s | €1133 | - | - | - | - | - | - |
| $\alpha = 0.8$, $M = 8T$ | - | - | 4s | €1132 | 17s | €1130 | - | - | - | - | - | - |
| $\alpha = 0.8$, $M = 10T$ | - | - | 6 | €1129 | 22s | €1125 | - | - | - | - | - | - |
| $\alpha = 0.8$, $M = 12T$ | - | - | 7s | **€1127** | 27s | **€1125** | - | - | - | - | - | - |
| $\alpha = 0.9$, $M = T$ | 0.01s | **€1163** | 0.9s | €1151 | 4s | €1150 | 281s | €1142 | 1255s | €1140 | **5203s** | €1140 |
| $\alpha = 0.9$, $M = 2T$ | - | - | 2.3s | €1145 | 8s | €1142 | 637s | €1133 | **2692s** | €1129 | - | - |
| $\alpha = 0.9$, $M = 4T$ | - | - | 4s | €1135 | 19s | €1131 | **1274s** | €1123 | - | - | - | - |
| $\alpha = 0.9$, $M = 6T$ | - | - | 7s | €1129 | 26s | **€1125** | - | - | - | - | - | - |
| $\alpha = 0.9$, $M = 8T$ | - | - | 9s | **€1126** | - | - | - | - | - | - | - | - |
| $\alpha = 0.99$, $M = T$ | 0.06s | **€1163** | 8s | €1132 | 33s | €1127 | **2576s** | **€1118** | **11740s** | **€1116** | **49891s** | **€1119** |
| $\alpha = 0.99$, $M = 2T$ | - | - | 19s | **€1127** | 76s | **€1123** | - | - | - | - | - | - |

Table 5.14: Average run time in seconds and total cost in euros of the forecast time simulations for simulated annealing

| Data ratio / Method | G D | | G×10 D | | G D×10 | | G×10 D×10 | | G/10 D | | G D/10 | | G/10 D/10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\alpha = 0.8$, $M = T$ | 2s | €1154 | 2s | €-22818 | 3s | €48145 | 2s | €7572 | 3s | €5349 | 3s | €-1958 | 2s | €563 |
| $\alpha = 0.8$, $M = 2T$ | 4s | €1149 | 3s | €-22825 | 6s | €48143 | 4s | €7566 | 6s | €5348 | 4s | €-1962 | 4s | €558 |
| $\alpha = 0.8$, $M = 4T$ | 9s | €1140 | 6s | €-22833 | 12s | €48141 | 8s | €7558 | 12s | €5345 | 6s | €-1970 | 8s | €553 |
| $\alpha = 0.8$, $M = 6T$ | 12s | €1133 | 9s | €-22838 | 18s | €48138 | 12s | €7552 | 18s | €5342 | 9s | €-1976 | 17s | €548 |
| $\alpha = 0.8$, $M = 8T$ | 17s | €1130 | 14s | €-22841 | 25s | €48136 | 17s | €7547 | 25s | €5340 | 14s | €-1980 | 19s | €545 |
| $\alpha = 0.8$, $M = 10T$ | 22s | €1126 | 17s | €-22844 | 33s | €48134 | 23s | €7544 | 33s | €5339 | 18s | €-1982 | 25s | €538 |
| $\alpha = 0.8$, $M = 12T$ | 27s | **€1125** | 22s | €-22847 | 42s | **€48133** | 29s | **€7542** | 42s | €5338 | 22s | **€-1985** | 31s | €538 |
| $\alpha = 0.8$, $M = 14T$ | - | - | 25s | **€-22848** | - | - | - | - | 48s | **€5336** | - | - | 36s | **€536** |
| $\alpha = 0.9$, $M = T$ | 4s | €1150 | 3s | €-22822 | 5s | €48144 | 4s | €7577 | 6s | €5348 | 3s | €-1961 | 4s | €559 |
| $\alpha = 0.9$, $M = 2T$ | 9s | €1141 | 6s | €-22830 | 11s | €48141 | 7s | €7559 | 11s | €5346 | 6s | €-1969 | 8s | €553 |
| $\alpha = 0.9$, $M = 4T$ | 19s | €1131 | 12s | €-22841 | 24s | €48137 | 16s | €7549 | 23s | €5342 | 12s | €-1978 | 17s | €544 |
| $\alpha = 0.9$, $M = 6T$ | 26s | **€1125** | 20s | €-22846 | 38s | €48134 | 27s | €7543 | 38s | €5338 | 21s | **€-1984** | 31s | €539 |
| $\alpha = 0.9$, $M = 8T$ | - | - | 29s | **€-22848** | 54s | **€48133** | 39s | **€7541** | 53s | **€5337** | - | - | 39s | **€535** |
| $\alpha = 0.99$, $M = T$ | 33s | €1127 | 24s | €-22845 | 47s | €48138 | 33s | €7545 | 47s | €5343 | 25s | €-1982 | 34s | €540 |
| $\alpha = 0.99$, $M = 2T$ | 78s | **€1123** | 53s | **€-22850** | 102s | **€48133** | 71s | **€7540** | 105s | €5338 | 56s | **€1986** | 77s | **€532** |
| $\alpha = 0.99$, $M = 4T$ | - | - | - | - | - | - | - | - | 226s | **€5336** | - | - | - | - |

Table 5.15: Average run time in seconds and total cost in euros of each of the data ratio simulations for simulated annealing

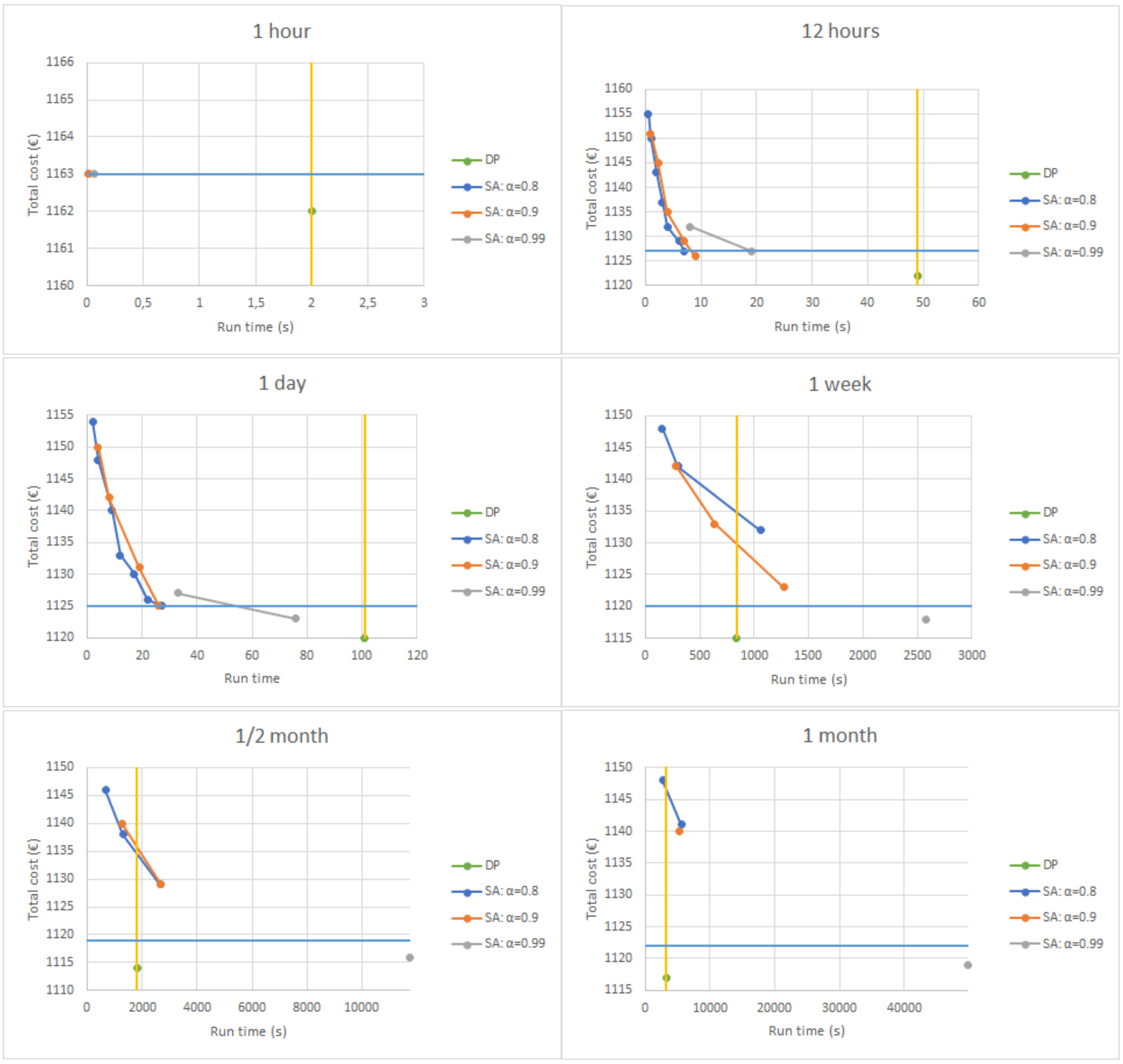


Figure 5.11: Scatter plots of the results of forecast time variations of simulated annealing.
The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming.

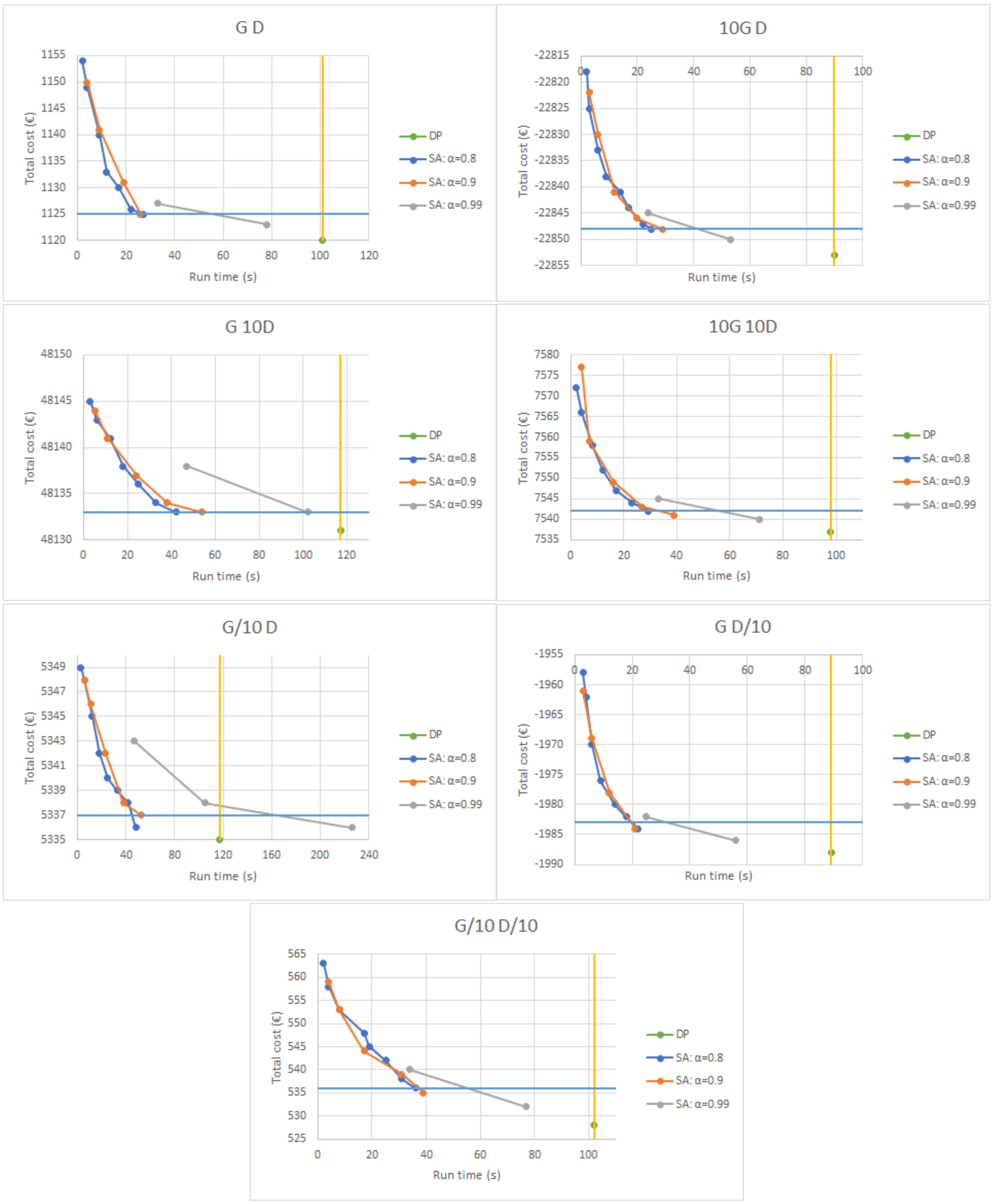


Figure 5.12: Scatter plots of the results of data ratio variations of simulated annealing.
The horizontal line indicates the close enough benchmark and the vertical line indicates the run time of the simulation with dynamic programming.

# 6 Conclusion and Recommendations

## 6.1. Conclusion

In this thesis, the question 'What mathematical optimisation method should be used for making charging decisions for a private battery in a smart grid?' has been researched. The focus was put on the use of a private battery in a smart grid in combination with a private renewable energy source. Four methods have been compared to each other on run time and result. These methods were dynamic programming, local search, tabu search, and simulated annealing.

The goal of this optimisation of the use of a battery is to minimise the monthly energy bill. Therefore, the total cost of the determined solution of each of the methods is the first comparison criterion. A benchmark was set for the total cost, for which the solution was determined to be 'close enough' to the lowest possible total cost. It was considered close enough if considering the difference between the lowest and highest possible cost, it would be within a 5% difference from the lowest possible cost.

As the decision for the (dis)charging is made for each time interval, the second comparison criterion is the run time of the method. This time interval would be 15 minutes. Because the run time highly depends on the specifications of the computer on which the calculations are run, it is not considered as a definite restriction.

For dynamic programming it is guaranteed that the solution that will be given, has the lowest possible cost. For the simulations with a forecast time of 1 week or lower, the average run time is still less than 15 minutes. The average run time for the forecast time of 1 week does come very close to 15 minutes.

The total cost of the solutions given by local search, do not come close enough to the lowest possible cost to be considered a viable alternative. The average run time of local search can exceed the run time of dynamic programming for a forecast time of 1 day and more.

Tabu search is an extension of local search. If it has a lower average run time than local search, then the cost of the solution will be higher. The cost of the solutions of local search were not close enough to the solution with the lowest possible cost, thus those of tabu search would also be too high. If tabu search has a higher average run time than local search, then it will exceed the average run time of dynamic programming at the same instances as local search.

For simulated annealing, the cost from the solutions come very close to those of dynamic programming. The average run time depends on the forecast time and the choice of variables. When the forecast time is kept to 1 day or less, then the average run time is shorter than the average run time of dynamic programming. When the forecast time is over 1 week, then it may exceed the average run time of dynamic programming. However, due to the uncertainty of several of the data variables, a forecast time of more than a week will probably yield unreliable results. Thus when taking a forecast time of a day or less, simulated annealing will result in a solution that has a total cost that is close enough to the lowest possible cost and its average run time will be lower than that of dynamic programming.

Thus, it is advisable to let the forecast time not exceed one day for both reliability and run time. The simulations that were executed are not a complete representation of a real-life case. Making the adjustments to fit a real-life situation will increase the run time of the methods due to the added complexity. As long as the run time of dynamic programming does not exceed the available time limit after these adjustments, it should be used to determine the charging decisions that result in the solution with the lowest possible cost. However, if the run time does exceed the 15 minute limit, then the switch to simulated annealing should be made which has a lower average run time. Simulated annealing does not guarantee to give the solution with the lowest possible cost, but it comes close enough.

## 6.2. Recommendations

In this thesis, a start has been made to construct an algorithm to determine the optimal charging decision for each time of the day. However, there is still a long way to go before this can be fully implemented. Several suggestions to continue on this work follow below.

### 6.2.1. Genetic algorithm

The heuristic method genetic algorithm could also be considered for implementation. This method is based on Darwin's theory of evolution. It starts with a set of solutions and constructs new solutions with mutations, crossovers, and selections. It could be researched if this different approach results in solutions close enough to those found with dynamic programming, but with a shorter average run time than dynamic programming.

### 6.2.2. Electric vehicles as batteries

An electric vehicle could be used as an extra battery by discharging the stored energy to satisfy the customer's demand. Adding this extra battery to the methods could improve the use of private generated energy and would bring the methods closer to reality.

### 6.2.3. Uncertainties

In real life, most of the data used is unknown. Weather and demand is constantly changing. Adding this uncertainty to the methods would increase the difficulty and run time of these method but would also bring them closer to reality.

# A
# Examples

In this appendix, an example of each of the considered methods is given. The data used for these examples is given in equation A.1, a timespan of 4 intervals is considered. Here $x_{start}$ is the starting charge of the battery, $x_{max}$ is the capacity of the battery and $x_{step}$ is the set amount of the battery can be charged or discharged with for each of the 4 time instances. The demand is indicated by the vector $d$, the amount of generated energy by vector $g$ and the energy price by vector $p$. To keep the examples simple, it is assumed that energy is bought and sold for the same price.

The goal is to minimise the total cost, which is determined by the formulas in equation A.2.

For each instance, there are 3 possible decisions: charging by 1, discharging by 1, or neither. If the state is 0, discharging is impossible because the battery charge can not be negative and if the state is 2, charging is impossible due to the maximum capacity of the battery. If a solution does not adhere to these requirements, it is deemed infeasible.

$$
\begin{gathered}
x_{start} = 0,\ x_{max} = 2,\ x_{step} = 1 \\
d = \begin{pmatrix} 3 \\ 8 \\ 4 \\ 5 \end{pmatrix},\ g = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 2 \end{pmatrix},\ p = \begin{pmatrix} 1.8 \\ 1.2 \\ 2 \\ 0.8 \end{pmatrix}
\end{gathered}
\tag{A.1}
$$

$$
\begin{gathered}
C_i(u_i, d_i, g_i, p_i) = \left(u_i + d_i - g_i\right) p_i \\
u_i = x_i - x_{i-1} \\
C_{total} = \sum_{i=1}^{4} C_i(u_i, d_i, g_i, p_i)
\end{gathered}
\tag{A.2}
$$

## A.1. Dynamic programming

With dynamic programming, the optimal solution is found by backtracking an optimal path for each charging state from the last instance. The first step is determining each charging decision that can be made for each state for each instance, figure A.1.

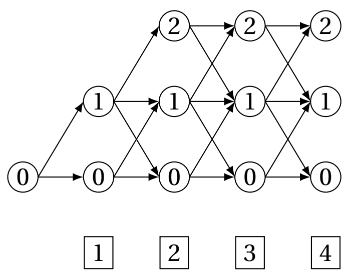
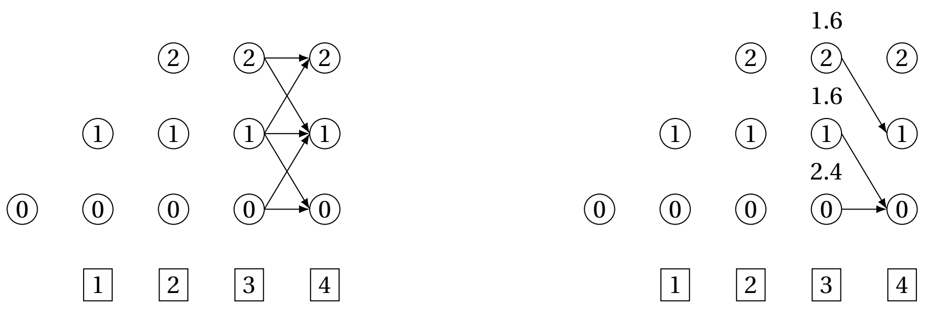


Figure A.1: All possible charging and discharging options

### A.1.1. Third to fourth time instance

Starting with the third to fourth instance, which has 3 possible states. All the options for the states are represented in Figure A.2a.



(a) All options for the fourth instance

(b) Optimal charging decisions for the fourth instance

Figure A.2: Fourth instance

Only the difference between the charging state of instance 4 and the charging state of instance 3 is of influence to the cost function.

If $u_4 = 1$ (charging), then $C_4 = \left(u_4 + d_4 - g_4\right) p_4 = (1+5-2) \times 0.8 = 3.2$

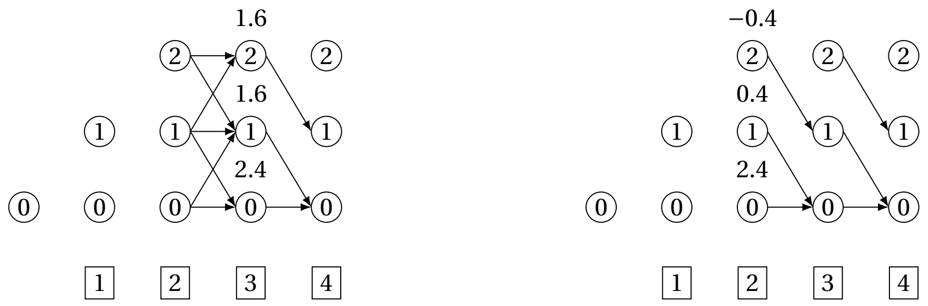
If $u_4 = 0$ (neither), then $C_4 = \left(u_4 + d_4 - g_4\right) p_4 = (0+5-2) \times 0.8 = 2.4$

If $u_4 = -1$ (discharging), then $C_4 = \left(u_4 + d_4 - g_4\right) p_4 = (-1+5-2) \times 0.8 = 1.6$

Thus the best choice is to discharge, if this is not possible nothing should be done. This solution is represented in Figure A.2b and the total cost for each decision path from the third instance is placed above each of the states.

### A.1.2. Second to third time instance

Next is considering all the options for the second to third instance, Figure A.3a.



(a) All options for the third instance

(b) Optimal charging decisions for the third instance

Figure A.3: Third instance

Again determining the cost for each of the charging decisions.
If $u_3 = 1$ (charging), then $C_3 = \left(u_3 + d_3 - g_3\right) p_3 = (1 + 4 - 4) \times 2 = 2$
If $u_3 = 0$ (neither), then $C_3 = \left(u_3 + d_3 - g_3\right) p_3 = (0 + 4 - 4) \times 2 = 0$
If $u_3 = -1$ (discharging), then $C_3 = \left(u_3 + d_3 - g_3\right) p_3 = (-1 + 4 - 4) \times 2 = -2$

The eventual goal is to minimise the total cost, thus for now minimising $C_3 + C_4$.

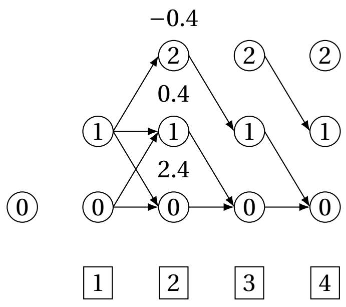
For state 2:
If $u_3 = 0$, then $C_3 + C_4 = 0 + 1.6 = 1.6$
If $u_3 = -1$, then $C_3 + C_4 = -2 + 1.6 = -0.4$
The best choice is to discharge.

For state 1:
If $u_3 = 1$, then $C_3 + C_4 = 2 + 1.6 = 3.6$
If $u_3 = 0$, then $C_3 + C_4 = 0 + 1.6 = 1.6$
If $u_3 = -1$, then $C_3 + C_4 = -2 + 2.4 = 0.4$
The best choice is to discharge.
For state 0:
If $u_3 = 1$, then $C_3 + C_4 = 2 + 1.6 = 3.6$
If $u_3 = 0$, then $C_3 + C_4 = 0 + 2.4 = 2.4$
The best choice is to do nothing.

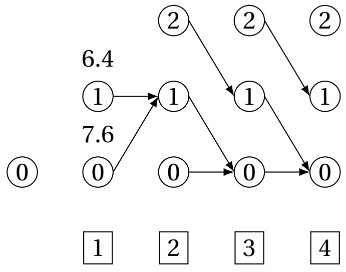
This solution is represented in Figure A.3b and the total cost for each decision path from the second instance is placed above each of the states.

### A.1.3. First to second time instance

Next is considering all the options for the first to second instance, Figure A.4a.



(a) All options for the second instance



(b) Optimal charging decisions for the second instance

Figure A.4: Second instance

Again determining the cost for each of the charging decisions.
If $u_2 = 1$ (charging), then $C_2 = \left(u_2 + d_2 - g_2\right) p_2 = (1+8-3) \times 1.2 = 7.2$
If $u_2 = 0$ (neither), then $C_2 = \left(u_2 + d_2 - g_2\right) p_2 = (0+8-3) \times 1.2 = 6$
If $u_2 = -1$ (discharging), then $C_2 = \left(u_2 + d_2 - g_2\right) p_2 == (-1+8-3) \times 1.2 = 4.8$

The eventual goal is to minimising the total cost, thus for now minimising $C_2 + C_3 + C_4$.

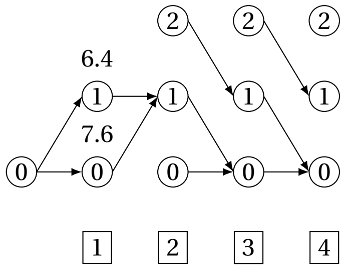
For state 1:
If $u_2 = 1$, then $C_2 + C_3 + C_4 = 7.2 - 0.4 = 6.8$
If $u_2 = 0$, then $C_2 + C_3 + C_4 = 6 + 0.4 = 6.4$
If $u_2 = -1$, then $C_2 + C_3 + C_4 = 4.8 + 2.4 = 7.2$
The best choice is to do nothing.

For state 0:
If $u_2 = 1$, then $C_2 + C_3 + C_4 = 7.2 + 0.4 = 7.6$
If $u_2 = 0$, then $C_2 + C_3 + C_4 = 6 + 2.4 = 8.4$
The best choice is to charge.

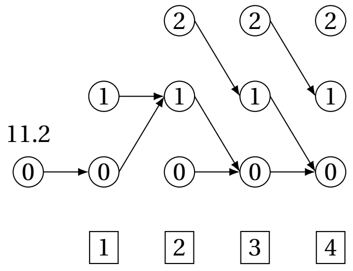
This solution is represented in Figure A.4b and the total cost for each decision path from the second instance is placed above each of the states.

### A.1.4. Initial to first time instance

Last is considering all the options for initial to first instance, Figure A.5a.



(a) All options for the first instance



(b) Optimal charging decisions for the first instance

Figure A.5: First instance

Again determining the cost for each of the charging decisions.
If $u_1 = 1$ (charging), then $C_1 = \left(u_1 + d_1 - g_1\right) p_1 = (1+3-1) \times 1.8 = 5.4$
If $u_1 = 0$ (neither), then $C_1 = \left(u_1 + d_1 - g_1\right) p_1 = (0+3-1) \times 1.8 = 3.6$

The goal is to minimising the total cost, thus for now minimising $C_{total} = C_1 + C_2 + C_3 + C_4$.

For state 0:
If $u_1 = 1$, then $C_{total} = 5.4 + 6.4 = 11.8$
If $u_1 = 0$, then $C_{total} = 3.6 + 7.6 = 11.2$
The best choice is to do nothing.

This solution is represented in Figure A.5b and the total cost for each decision path from the second instance is placed above each of the states.

Now the solution with the lowest total cost has been found, figure A.6, with a total cost of 11.2.

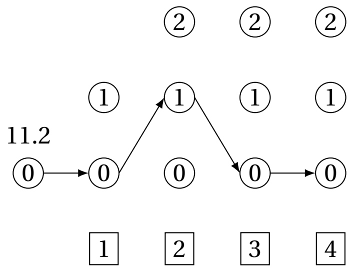


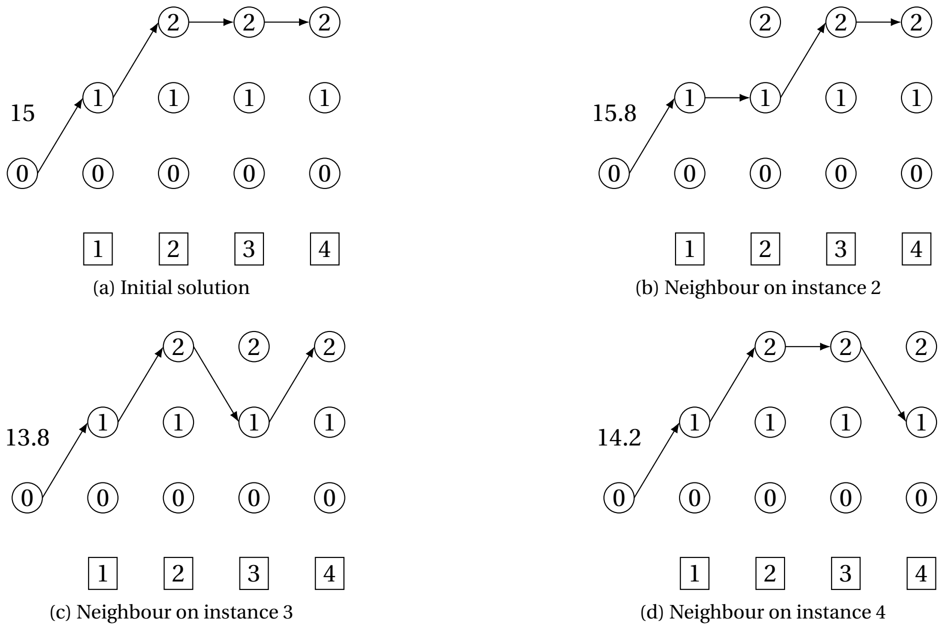
Figure A.6: Optimal solution of dynamic programming

## A.2. Local search

For local search, an initial solution is defined and from this solution all similar 'neighbour' solutions are considered to find a better one. This is continued until no better solution is present in the 'neighbourhood'. In this case the neighbourhood is defined as changing the charge of one of the instances by 1 while keeping a feasible solution.

A random feasible initial solution is represented in Figure A.7a, the total cost is written above the starting charge of 0.

### A.2.1. First iteration



(a) Initial solution (b) Neighbour on instance 2 (c) Neighbour on instance 3 (d) Neighbour on instance 4

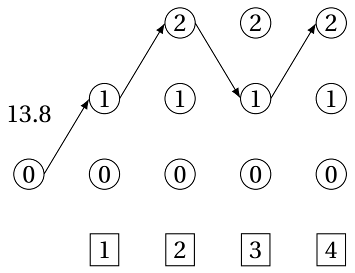
Figure A.7: First iteration

Considering the neighbours of the initial solution:

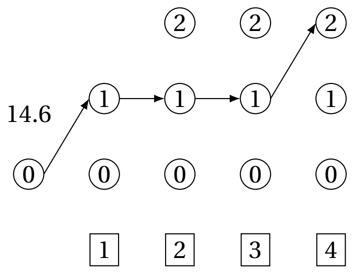
- For the first instance, changing the charge will only lead to infeasible solutions.
- For the second instance, the charge can be changed to 1, Figure A.7b. It has a total cost of 15.8, this is worse than the total cost of the initial solution.
- For the third instance, the charge can be changed to 1, Figure A.7c. It has a total cost of 13.8, this is better than the total cost of the initial solution and for now the best neighbour solution.
- For the fourth instance, the charge can be changed to 1, Figure A.7d. It has a total cost of 14.2, this is better than the total cost of the initial solution, but it is not the best neighbour solution.

All the neighbours are considered and a better solution has been found, Figure A.7c, with a total cost of 13.8.

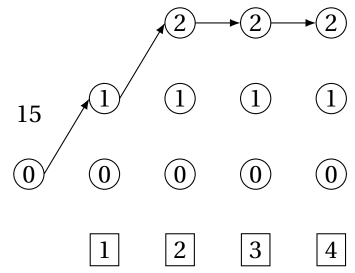
### A.2.2. Second iteration



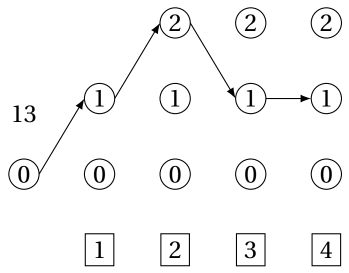
(a) Current solution



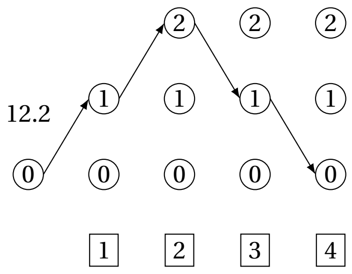
(b) Neighbour on instance 2



(c) Neighbour on instance 3



(d) First neighbour on instance 4



(e) Second neighbour on instance 4

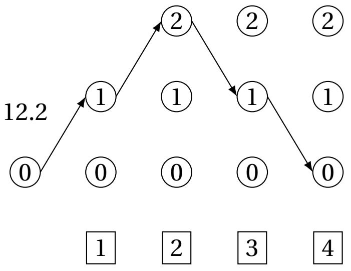
Figure A.8: Second iteration

Considering the neighbours of the current solution:

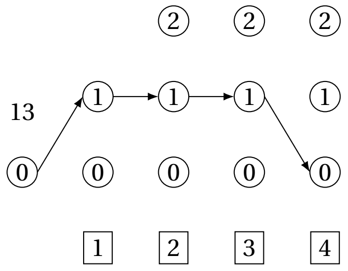
- For the first instance, changing the charge will only lead to infeasible solutions.
- For the second instance, the charge can be changed to 1, Figure A.8b. It has a total cost of 14.6, this is worse than the total cost of the current solution.
- For the third instance, the charge can be changed to 2, Figure A.8c. It has a total cost of 15, this is worse than the total cost of the current solution.
- For the fourth instance, the charge can either be changed to 1 or 0, Figure A.8d and Figure A.8e respectively. The first neighbour has a total cost of 13, the second neighbour has a total cost of 12.2. Both improve on the total cost of the current solution, the total cost of the latter is lower and is for now the best neighbour solution.

All the neighbours are considered and a better solution has been found, Figure A.8e, with a total cost of 12.2.

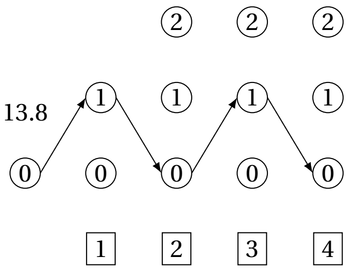
### A.2.3. Third iteration



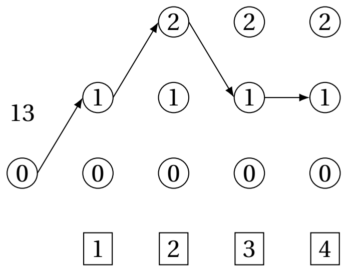
(a) Current solution



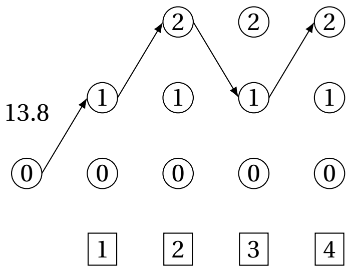
(b) First neighbour on instance 2



(c) Second neighbour on instance 2



(d) First neighbour on instance 4



(e) Second neighbour on instance 4

Figure A.9: Third iteration

Considering the neighbours of the current solution:

- For the first instance, changing the charge will only lead to infeasible solutions.
- For the second instance, the charge can either be changed to 1 or 0, Figure A.9b and Figure A.9c respectively. Both of these solutions have a total cost that exceeds the total cost of the current solution.
- For the third instance, changing the charge will only lead to infeasible solutions.
- For the fourth instance, the charge can either be changed to 1 or 2, Figure A.9d and Figure A.9e respectively. Both of these solutions have a total cost that exceeds the total cost of the current solution.

No better neighbouring solution has been found, thus the method is done. The best solution that has been found with a total cost of 12.2, figure A.10. This is worse than the solution found by dynamic programming, this must be a local optimum.

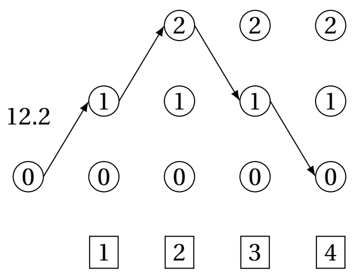


Figure A.10: Optimal solution of local search

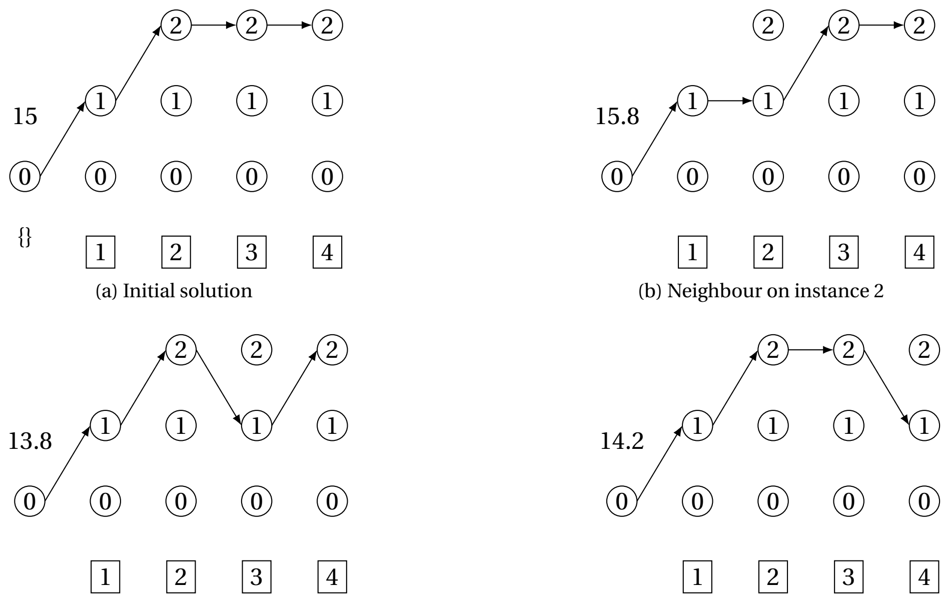
## A.3. Tabu search

Tabu search is similar to local search, the difference is that with tabu search the neighbouring solution with the lowest total cost is chosen to continue with, even if this is not better than the current solution. A list of tabu solutions is being kept to prevent cycling and a stopping criterion is introduced to end the method. The method keeps track of the solution with the lowest total cost found so far. If a neighbouring solution with a lower total cost than the current best is found, but is tabu. Then the tabu is ignored and it is still chosen as a new solution.

In this case, the tabu solutions are indicated by which instances have been altered in the previous iterations. The size of the tabu list is set to 2, the oldest tabu is replaced by a new tabu if the list is full. The stopping criterion is set to two consecutive iterations of no improvement on the best solution.

The same feasible initial solution as for local search is being used, Figure A.11a. The list of tabu instances is noted beneath the starting charge 0.

### A.3.1. First iteration



(a) Initial solution (b) Neighbour on instance 2 (c) Neighbour on instance 3 (d) Neighbour on instance 4

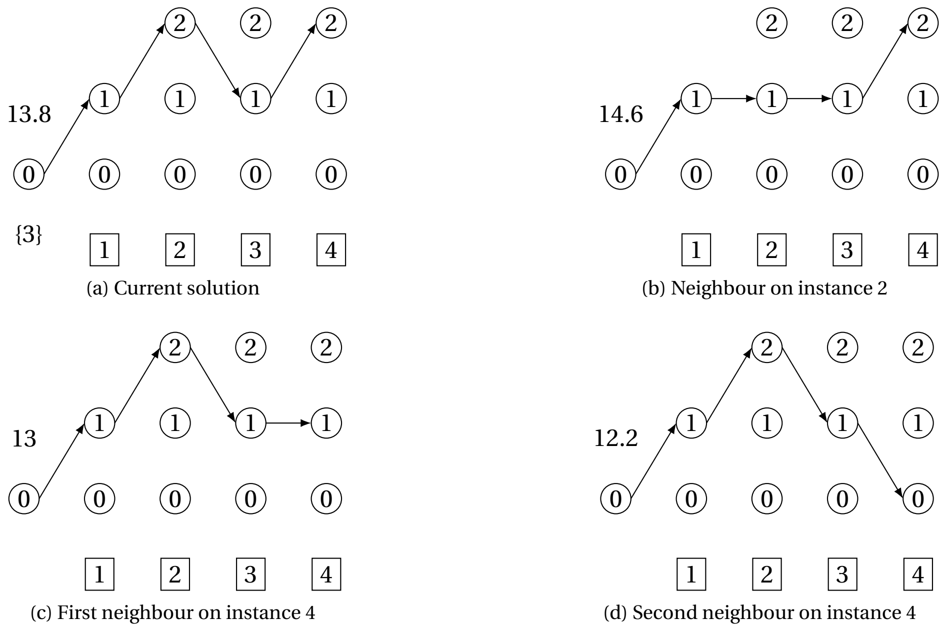
Figure A.11: First iteration

Considering the neighbours of the initial solution:

- For the first instance, changing the charge will only lead to infeasible solutions.
- For the second instance, the charge can be changed to 1, Figure A.11b. It has a total cost of 15.8, this is worse than the total cost of the initial solution.
- For the third instance, the charge can be changed to 1, Figure A.11c. It has a total cost of 13.8, this is better than the total cost of the initial solution and for now the best neighbour solution.
- For the fourth instance, the charge can be changed to 1, Figure A.11d. It has a total cost of 14.2, this is better than the total cost of the initial solution, but it is not the best neighbour solution.

All the neighbours are considered and a better solution has been found, Figure A.7c, with a total cost of 13.8. Tabu list: {3}.

### A.3.2. Second iteration



(a) Current solution
(b) Neighbour on instance 2
(c) First neighbour on instance 4
(d) Second neighbour on instance 4

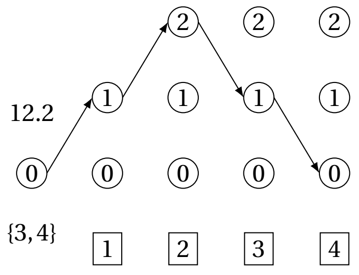
Figure A.12: Second iteration

Considering the neighbours of the current solution:

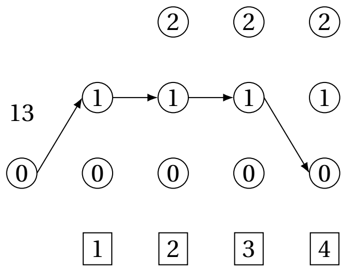
- For the first instance, changing the charge will only lead to infeasible solutions.
- For the second instance, the charge can be changed to 1, Figure A.12b. It has a total cost of 14.6, this is worse than the total cost of the current solution.
- The third instance is tabu.
- For the fourth instance, the charge can either be changed to 1 or 0, Figure A.12c and Figure A.12d respectively. Both of these solutions have a lower total cost than the current solution. The latter has the lowest cost and is for now the best neighbour solution.

All the neighbours are considered and a better solution has been found, Figure A.12d, with a total cost of 12.2. Tabu list: $\{3,4\}$.

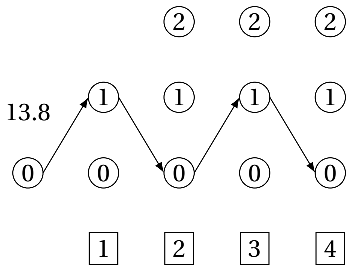
### A.3.3. Third iteration



(a) Current solution



(b) First neighbour on instance 2



(c) Second neighbour on instance 2

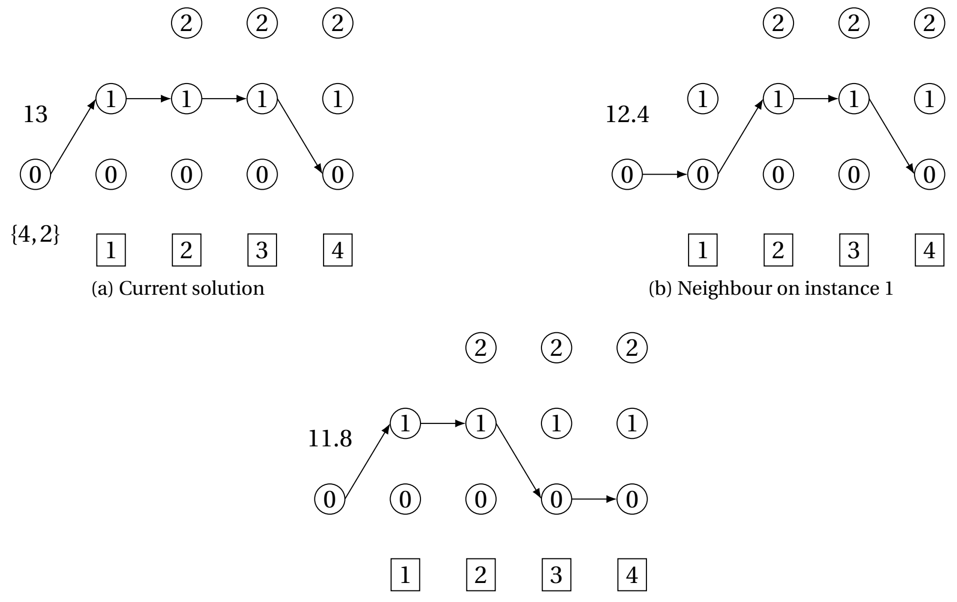
Figure A.13: Third iteration

Considering the neighbours of the current solution:

- For the first instance, changing the charge will only lead to infeasible solutions.
- For the second instance, the charge can either be changed to 1 or 0, Figure A.13b and **??** respectively. The total cost of both of these solutions are worse than the total cost of the current solution. The former has the lowest cost of the two.
- The third instance is tabu.
- The fourth instance is tabu.

No better neighbouring solution has been found, thus the best neighbouring solution is chosen. This is Figure A.13b with a total cost of 13, the best total cost is 12.2 of the solution in Figure A.12d. Tabu list: {4,2}.

### A.3.4. Fourth iteration



(a) Current solution

(b) Neighbour on instance 1

(c) Neighbour on instance 3

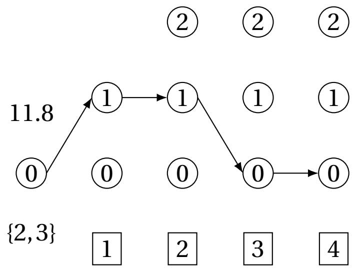
Figure A.14: Fourth iteration

Considering the neighbours of the current solution:

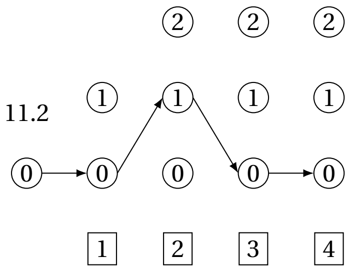
- For the first instance, the charge can be changed to 0, Figure A.14b. It has a total cost of 12.4, this is better than the total cost of the current solution but worse than the total cost of the best solution.
- The second instance is tabu.
- For the third instance, the charge can be changed to 0, Figure A.14c. It has a total cost of 11.8, this is better than the total cost of the current solution and of the best solution.
- The fourth instance is tabu.

All the neighbours are considered and a better solution has been found, Figure A.14c, with a total cost of 11.8. Tabu list: $\{2,3\}$.

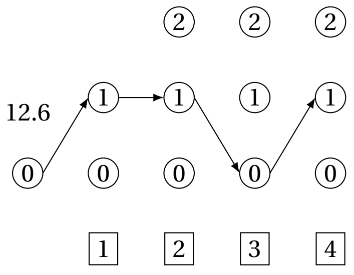
### A.3.5. Fifth iteration



(a) Current solution



(b) Neighbour on instance 1



(c) Neighbour on instance 4

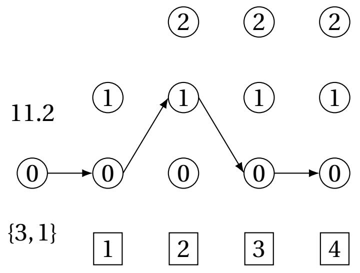
Figure A.15: Fifth iteration

Considering the neighbours of the current solution:

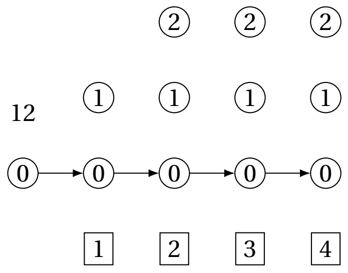
- For the first instance, the charge can be changed to 0, Figure A.15b. It has a total cost of 11.2, this is better than the total cost of the current solution and of the best solution.
- The second instance is tabu.
- The third instance is tabu.
- For the fourth instance, the charge can be changed to 1, Figure A.15c. It has a total cost of 12.6, this is worse than the total cost of the current solution.

All the neighbours are considered and a better solution has been found, Figure A.15b, with a total cost of 11.2. Tabu list: {3, 1}.

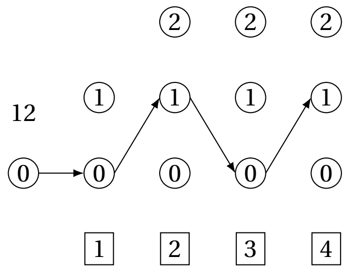
## A.3.6. Sixth iteration



(a) Current solution



(b) Neighbour on instance 2



(c) Neighbour on instance 4

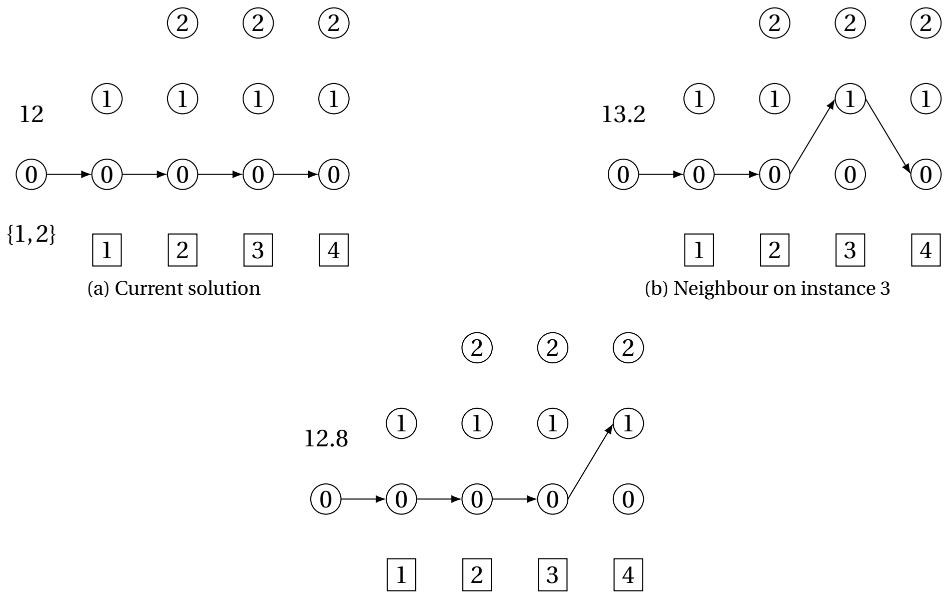
Figure A.16: Sixth iteration

Considering the neighbours of the current solution:

- The first instance is tabu.
- For the second instance, the charge can be changed to 0, Figure A.16b. It has a total cost of 12, this is worse than the total cost of the current solution.
- The third instance is tabu.
- For the fourth instance, the charge can be changed to 1, Figure A.16c. It has a total cost of 12, this is worse than the total cost of the current solution.

No better neighbouring solution has been found, thus the best neighbouring solution is chosen. Both of the neighbours have the same value, the solution from Figure A.16b is chosen at random, with a total cost of 12, the best total cost is 11.2 of the solution in Figure A.15b. Tabu list: $\{1, 2\}$.

### A.3.7. Seventh iteration



(a) Current solution (b) Neighbour on instance 3

(c) Neighbour on instance 4

Figure A.17: Seventh iteration

Considering the neighbours of the current solution:

- The first instance is tabu.
- The second instance is tabu.
- For the third instance, the charge can be changed to 1, Figure A.17b. It has a total cost of 13.2, this is worse than the total cost of the current solution.
- For the fourth instance, the charge can be changed to 1, Figure A.17c. It has a total cost of 12.8, this is worse than the total cost of the current solution.

No better neighbouring solution has been found, this is twice in a row thus the method is done. The best solution that has been found has a total cost of 11.2, figure A.18. This is better than the solution found by local search and the same as found by dynamic programming, this is the optimal value.

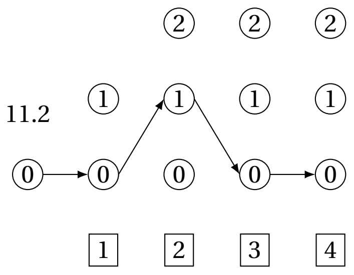


Figure A.18: Best solution found with tabu search

## A.4. Simulated annealing

Simulated annealing is based on the process of annealing in metallurgy. From a starting solution, a random neighbour is chosen. If this neighbour is a better solution, it is accepted as a new solution. If it has a worse solution, it is accepted with a probability depended on the difference in solution value and how far along in the process the check is being made. The later in the process, the lower the probability of accepting a worse solution.

In advance of the start of the method, an initial solution and four variables have to be determined. The initial solution is chosen equal to the initial solutions in local search and tabu search, Figure A.19a. Next is choosing a starting 'temperature' $T_{start}$, a minimum temperature $T_{min}$, a reduction value $\alpha$, and a set number of iterations for each temperature $M$. In this section, $C$ represents the current solution and $N$ the randomly selected neighbour.

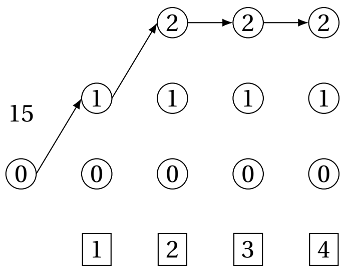
$M = 4$

$T_{start} = \max\{|C_{total}(\text{N}) - C_{total}(\text{initial solution})|\} = \max\{0.8, 1.2, 0.8\} = 1.2$

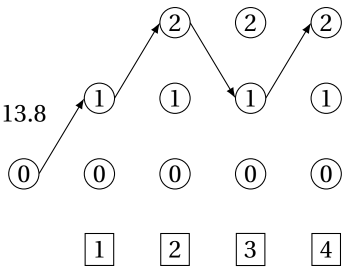
$T_{min} = \frac{T_{start}}{M} = \frac{1.2}{4} = 0.3$

$\alpha = 0.8$.

### A.4.1. First iteration for $T = 1.2$



(a) Initial solution

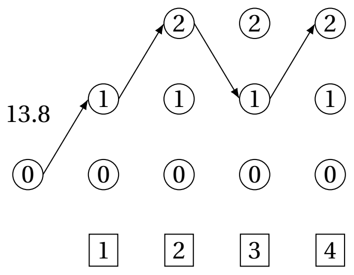


(b) Randomly selected neighbour

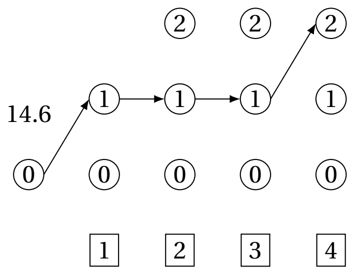
Figure A.19: First iteration for $T = 1.2$

A random neighbour is selected, Figure A.19b, this neighbour has a lower total cost, thus is accepted as the new solution.

### A.4.2. Second iteration for $T = 1.2$



(a) Current solution



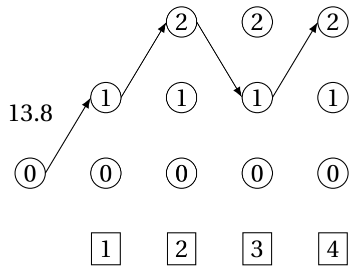
(b) Randomly selected neighbour

Figure A.20: Second iteration for $T = 1.2$

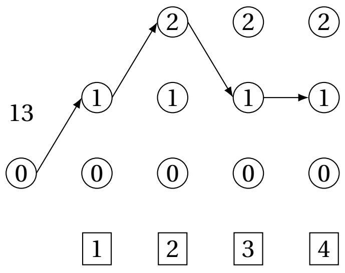
A random neighbour is selected, Figure A.20b, the neighbour has a lower total cost than the current solution.

For $p = \exp\left(\frac{-|C_{total}(N) - C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{1.2}\right) = 0.5134$, and $r$ a random number between 0 and 1 ($r = 0.6763$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.3. Third iteration for $T = 1.2$



(a) Current solution

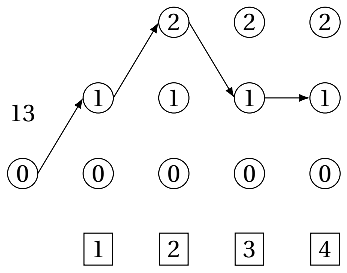


(b) Randomly selected neighbour

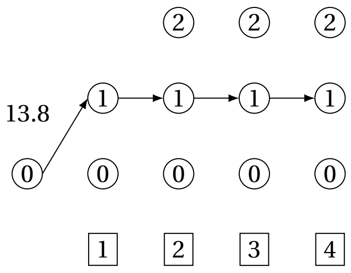
Figure A.21: Third iteration for $T = 1.2$

A random neighbour is selected, Figure A.21b, this neighbour has a lower total cost, thus is accepted as the new solution.

### A.4.4. Fourth iteration for $T = 1.2$



(a) Current solution



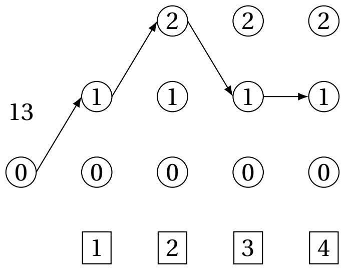
(b) Randomly selected neighbour

Figure A.22: Fourth iteration for $T = 1.2$

A random neighbour is selected, Figure A.22b, the neighbour has a lower total cost than the current solution.
For $p = \exp\left(\frac{-|C_{total}(N) - C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{1.2}\right) = 0.5134$, and $r$ a random number between 0 and 1 ($r = 0.6170$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

Four iterations have been executed for $T = 1.2$. Now calculating the new value for $T$:
$T = \alpha \times T = 0.8 \times 1.2 = 0.96 > 0.3 = T_{min}$, thus the method is continued.

### A.4.5. First iteration for $T = 0.96$


(a) Current solution

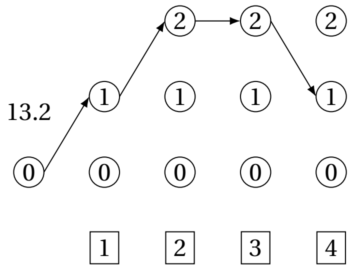
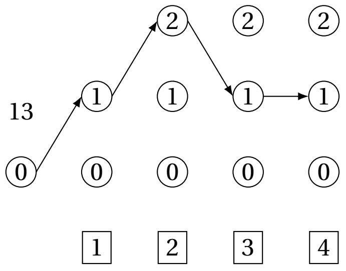

(b) Randomly selected neighbour

Figure A.23: First iteration for $T = 0.96$

A random neighbour is selected, Figure A.23b, the neighbour has a lower total cost than the current solution. For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.2}{0.96}\right) = 0.8119$, and $r$ a random number between 0 and 1 ($r = 0.9910$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.6. Second iteration for $T = 0.96$


(a) Current solution

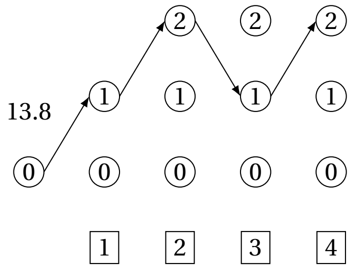
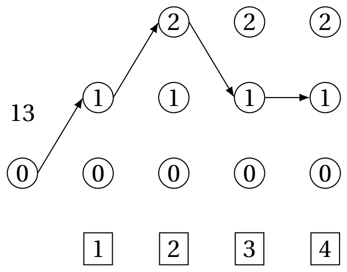

(b) Randomly selected neighbour

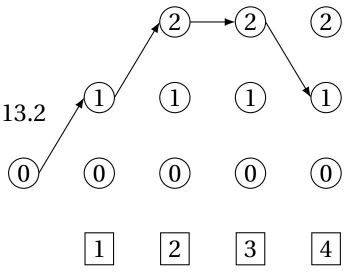
Figure A.24: Second iteration for $T = 0.96$

A random neighbour is selected Figure A.24b, the neighbour has a lower total cost than the current solution. For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.96}\right) = 0.4346$, and $r$ a random number between 0 and 1 ($r = 0.5303$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.7. Third iteration for $T = 0.96$



(a) Current solution

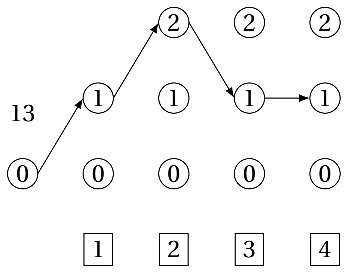


(b) Randomly selected neighbour

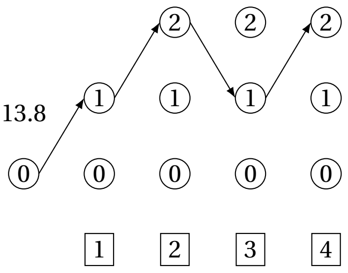
Figure A.25: Third iteration for $T = 0.96$

A random neighbour is selected, Figure A.25b, the neighbour has a lower total cost than the current solution.
For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.2}{0.96}\right) = 0.8119$, and $r$ a random number between 0 and 1 ($r = 0.8853$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.8. Fourth iteration for $T = 0.96$



(a) Current solution



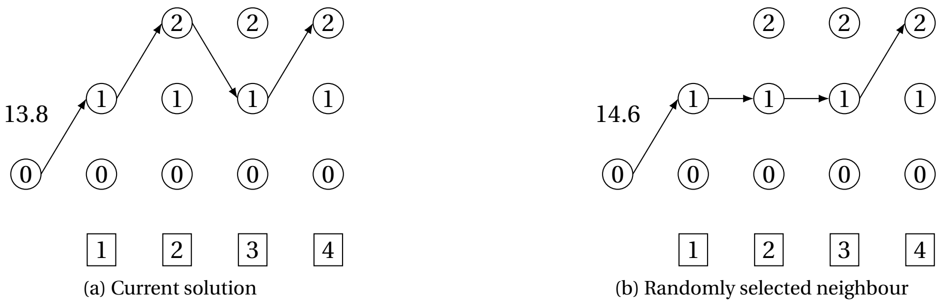
(b) Randomly selected neighbour

Figure A.26: Fourth iteration for $T = 0.96$

A random neighbour is selected, Figure A.26b, the neighbour has a lower total cost than the current solution.
For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.96}\right) = 0.4346$, and $r$ a random number between 0 and 1 ($r = 0.1117$), the neighbour is accepted if $r < p$. Thus, the neighbour is accepted.

Four iterations have been executed for $T = 0.96$. Now calculating the new value for $T$:
$T = \alpha \times T = 0.8 \times 0.96 = 0.768 > 0.3 = T_{min}$, thus the method is continued.

### A.4.9. First iteration for $T = 0.768$

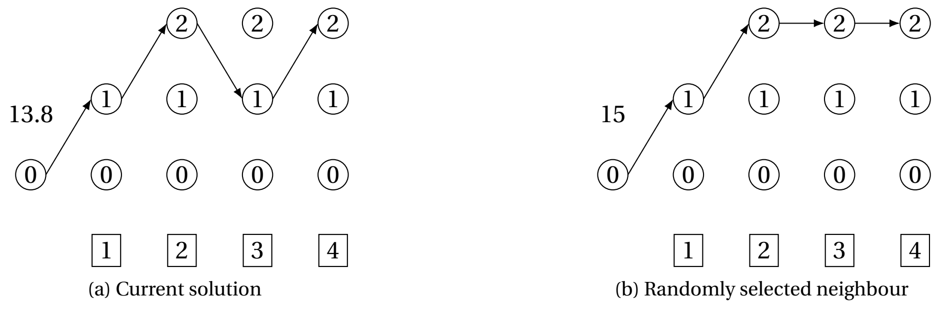


(a) Current solution (b) Randomly selected neighbour

Figure A.27: First iteration for $T = 0.768$

A random neighbour is selected, Figure A.27b, the neighbour has a lower total cost than the current solution. For $p = \exp\left(\frac{-|C_{total}(N) - C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.768}\right) = 0.5410$, and $r$ a random number between 0 and 1 ($r = 0.8062$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.10. Second iteration for $T = 0.768$

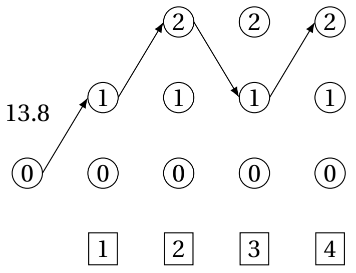


(a) Current solution (b) Randomly selected neighbour

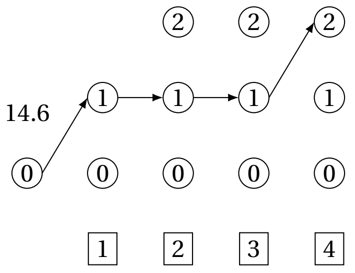
Figure A.28: Second iteration for $T = 0.768$

A random neighbour is selected, Figure A.28b, the neighbour has a lower total cost than the current solution. For $p = \exp\left(\frac{-|C_{total}(N) - C_{total}(C)|}{T}\right) = \exp\left(\frac{-1.2}{0.768}\right) = 0.2096$, and $r$ a random number between 0 and 1 ($r = 0.8062$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.11. Third iteration for $T = 0.768$



(a) Current solution



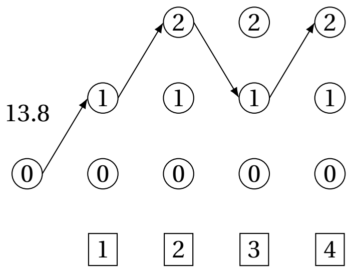
(b) Randomly selected neighbour

Figure A.29: Third iteration for $T = 0.768$

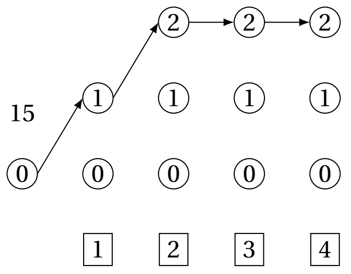
A random neighbour is selected, Figure A.29b, the neighbour has a lower total cost than the current solution.

For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.768}\right) = 0.5410$, and $r$ a random number between 0 and 1 ($r = 0.7442$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.12. Fourth iteration for $T = 0.768$



(a) Current solution



(b) Randomly selected neighbour

Figure A.30: Fourth iteration for $T = 0.768$

A random neighbour is selected, Figure A.30b, the neighbour has a lower total cost than the current solution.

For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-1.2}{0.768}\right) = 0.2096$, and $r$ a random number between 0 and 1 ($r = 0.5071$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

Four iterations have been executed for $T = 0.768$. Now calculating the new value for $T$: $T = \alpha \times T = 0.8 \times 0.768 = 0.6144 > 0.3 = T_{min}$, thus the method is continued.

## A.4.13. First iteration for $T = 0.6144$

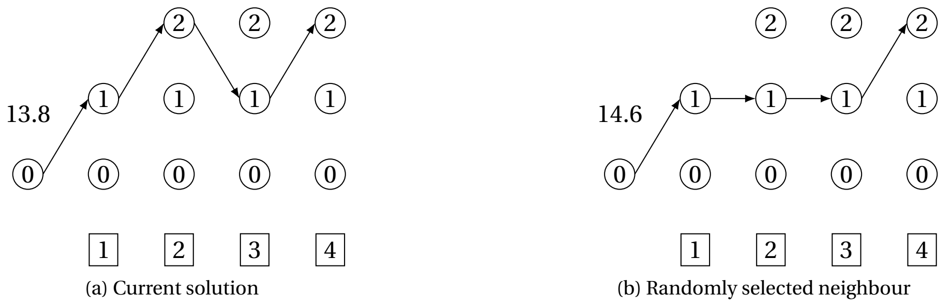


Figure A.31: First iteration for $T = 0.6144$

A random neighbour is selected, Figure A.31b, the neighbour has a lower total cost than the current solution.
For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.6144}\right) = 0.6117$, and $r$ a random number between 0 and 1 ($r = 0.8821$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

## A.4.14. Second iteration for $T = 0.6144$

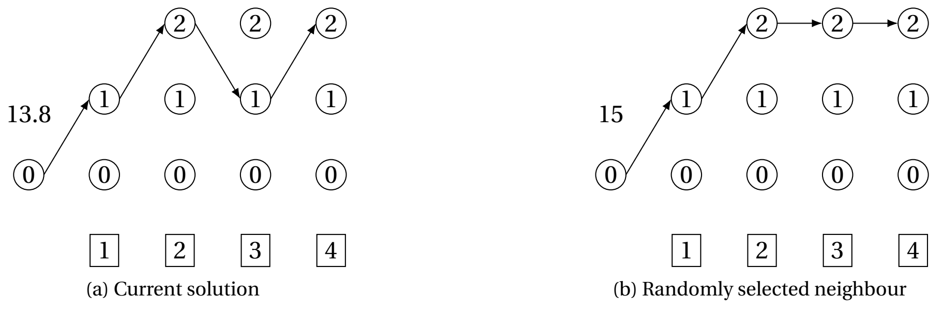
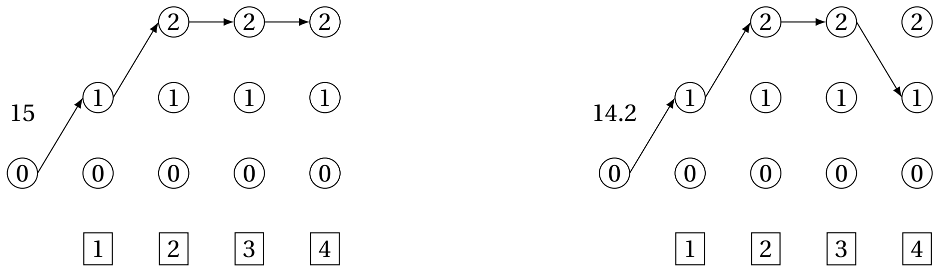


Figure A.32: Second iteration for $T = 0.6144$

A random neighbour is selected, Figure A.32b, the neighbour has a lower total cost than the current solution.
For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-1.2}{0.6144}\right) = 0.2096$, and $r$ a random number between 0 and 1 ($r = 0.1456$), the neighbour is accepted if $r < p$. Thus, the neighbour is accepted.

### A.4.15. Third iteration for $T = 0.6144$

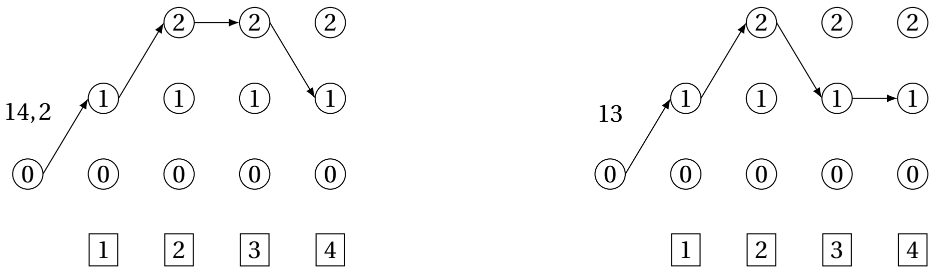


(a) Current solution (b) Randomly selected neighbour

Figure A.33: Third iteration for $T = 0.6144$

A random neighbour is selected, Figure A.33b, this neighbour has a lower total cost, thus is accepted as the new solution.

### A.4.16. Fourth iteration for $T = 0.6144$



(a) Current solution (b) Randomly selected neighbour

Figure A.34: Fourth iteration for $T = 0.6144$

A random neighbour is selected, Figure A.34b, this neighbour has a lower total cost, thus is accepted as the new solution.

Four iterations have been executed for $T = 0.6144$. Now calculating the new value for $T$:
$T = \alpha \times T = 0.8 \times 0.6144 = 0.4915 > 0.3 = T_{min}$, thus the method is continued.

### A.4.17. First iteration for $T = 0.4915$

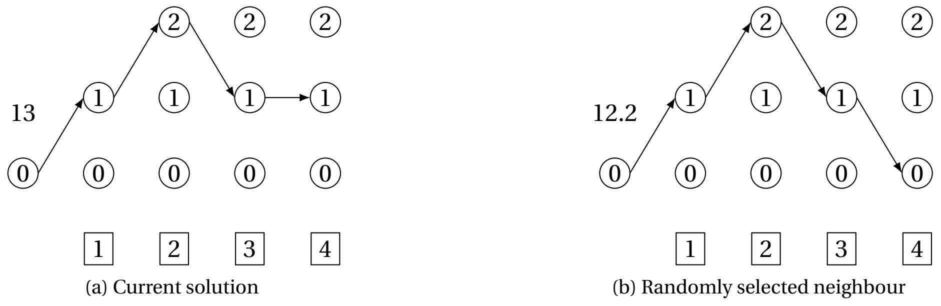


Figure A.35: First iteration for $T = 0.4915$

A random neighbour is selected, Figure A.35b, this neighbour has a lower total cost, thus is accepted as the new solution.

### A.4.18. Second iteration for $T = 0.4915$

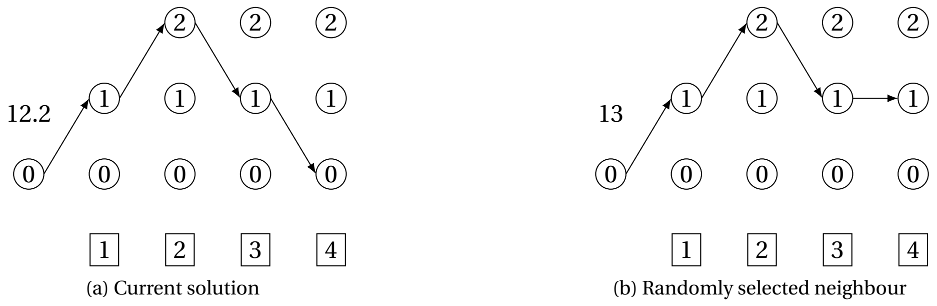
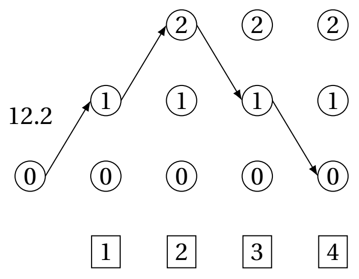


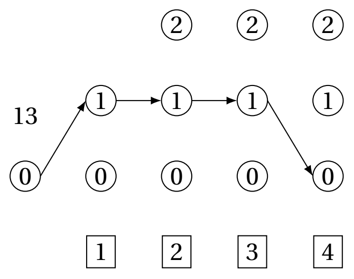
Figure A.36: Second iteration for $T = 0.4915$

A random neighbour is selected, Figure A.36b, the neighbour has a lower total cost than the current solution.
For $p = \exp\left(\frac{-|C_{total}(N) - C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.4915}\right) = 0.1964$, and $r$ a random number between 0 and 1 ($r = 0.4917$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.19. Third iteration for $T = 0.4915$



(a) Current solution

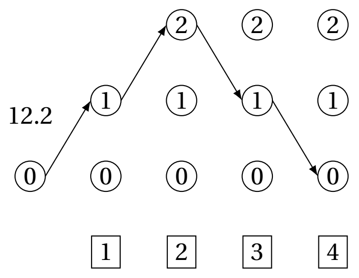


(b) Randomly selected neighbour

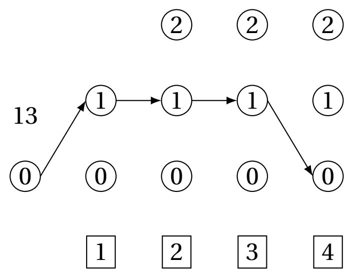
Figure A.37: Third iteration for $T = 0.4915$

A random neighbour is selected, Figure A.37b, the neighbour has a lower total cost than the current solution. For $p = \exp\left(\frac{-|C_{total}(N) - C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.4915}\right) = 0.1964$, and $r$ a random number between 0 and 1 ($r = 0.8104$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.20. Fourth iteration for $T = 0.4915$



(a) Current solution



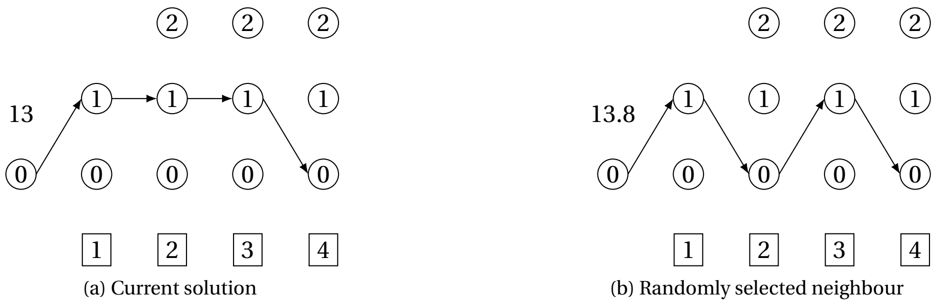
(b) Randomly selected neighbour

Figure A.38: Fourth iteration for $T = 0.4915$

A random neighbour is selected, Figure A.38b, the neighbour has a lower total cost than the current solution. For $p = \exp\left(\frac{-|C_{total}(N) - C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.4915}\right) = 0.1964$, and $r$ a random number between 0 and 1 ($r = 0.1765$), the neighbour is accepted if $r < p$. Thus, the neighbour is accepted.

Four iterations have been executed for $T = 0.4915$. Now calculating the new value for $T$: $T = \alpha \times T = 0.8 \times 0.4915 = 0.3932 > 0.3 = T_{min}$, thus the method is continued.

### A.4.21. First iteration for $T = 0.3932$



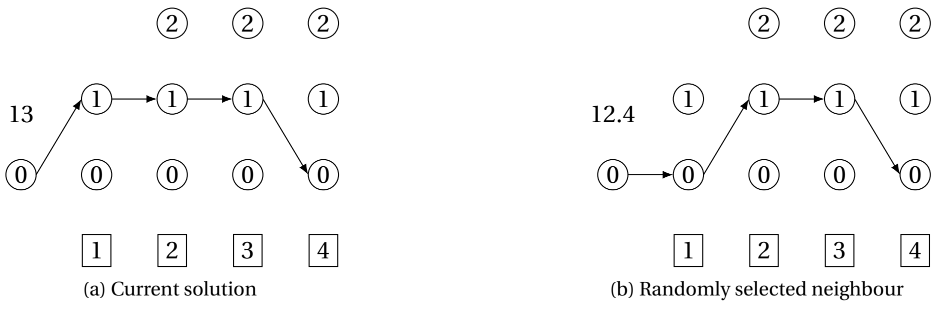
(a) Current solution (b) Randomly selected neighbour

Figure A.39: First iteration for $T = 0.3932$

A random neighbour is selected, Figure A.39b, the neighbour has a lower total cost than the current solution.

For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.3932}\right) = 0.1307$, and $r$ a random number between 0 and 1 ($r = 0.9338$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.22. Second iteration for $T = 0.3932$

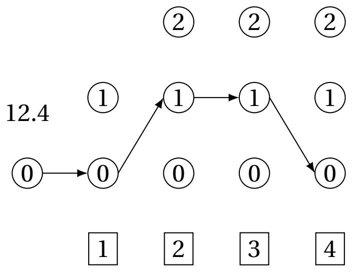


(a) Current solution (b) Randomly selected neighbour

Figure A.40: Second iteration for $T = 0.3932$

A random neighbour is selected, Figure A.40b, this neighbour has a lower total cost, thus is accepted as the new solution.

### A.4.23. Third iteration for $T = 0.3932$


(a) Current solution

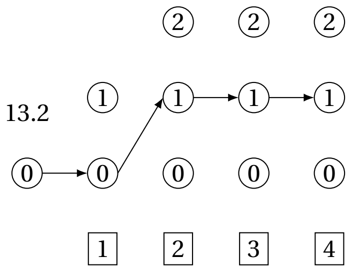
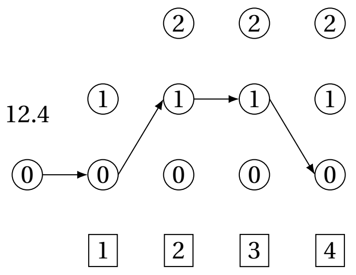

(b) Randomly selected neighbour

Figure A.41: Third iteration for $T = 0.3932$

A random neighbour is selected, Figure A.41b, the neighbour has a lower total cost than the current solution. For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.3932}\right) = 0.1307$, and $r$ a random number between 0 and 1 ($r = 0.3784$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.24. Fourth iteration for $T = 0.3932$


(a) Current solution

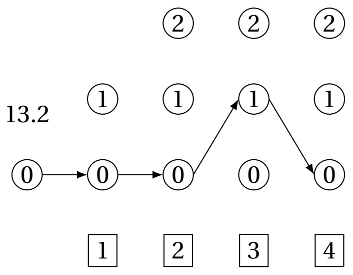
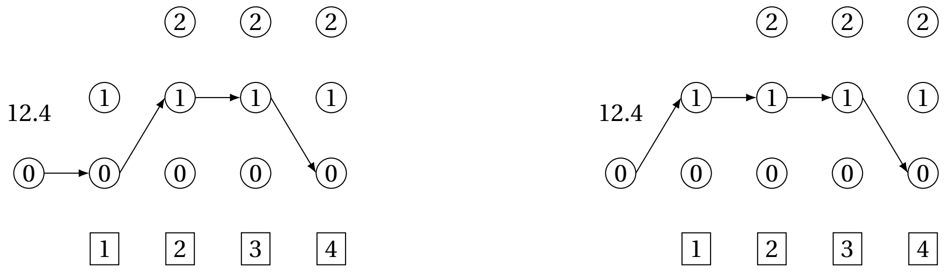

(b) Randomly selected neighbour

Figure A.42: Fourth iteration for $T = 0.3932$

A random neighbour is selected, Figure A.42b, the neighbour has a lower total cost than the current solution. For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.3932}\right) = 0.1307$, and $r$ a random number between 0 and 1 ($r = 0.2660$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

Four iterations have been executed for $T = 0.3932$. Now calculating the new value for $T$: $T = \alpha \times T = 0.8 \times 0.3932 = 0.3146 > 0.3 = T_{min}$, thus the method is continued.

### A.4.25. First iteration for $T = 0.3146$

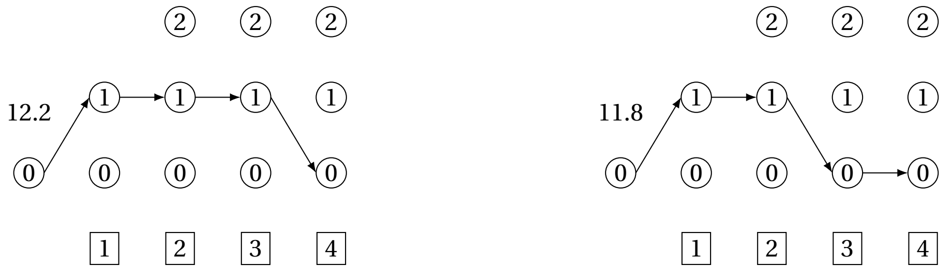


(a) Current solution (b) Randomly selected neighbour

Figure A.43: First iteration for $T = 0.3146$

A random neighbour is selected, Figure A.43b, the neighbour has a the same total cost, thus it is accepted.

### A.4.26. Second iteration for $T = 0.3146$

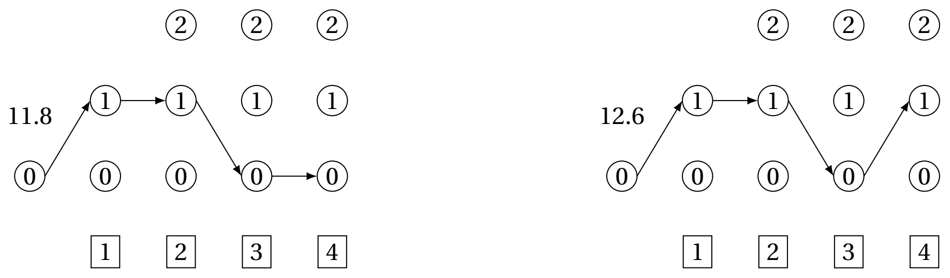


(a) Current solution (b) Randomly selected neighbour

Figure A.44: Second iteration for $T = 0.3146$

A random neighbour is selected, Figure A.44b, this neighbour has a lower total cost, thus is accepted as the new solution.

### A.4.27. Third iteration for $T = 0.3146$

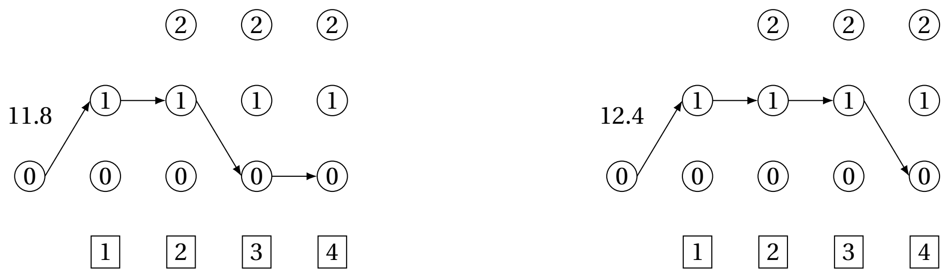


(a) Current solution (b) Randomly selected neighbour

Figure A.45: Third iteration for $T = 0.3146$

A random neighbour is selected, Figure A.45b, the neighbour has a lower total cost than the current solution. For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.8}{0.3146}\right) = 0.0786$, and $r$ a random number between 0 and 1 ($r = 0.3383$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

### A.4.28. Fourth iteration for $T = 0.3146$



(a) Current solution (b) Randomly selected neighbour

Figure A.46: Fourth iteration for $T = 0.3146$

A random neighbour is selected, Figure A.46b, the neighbour has a lower total cost than the current solution. For $p = \exp\left(\frac{-|C_{total}(N)-C_{total}(C)|}{T}\right) = \exp\left(\frac{-0.6}{0.3146}\right) = 0.1485$, and $r$ a random number between 0 and 1 ($r = 0.3973$), the neighbour is accepted if $r < p$. Thus, the neighbour is not accepted.

Four iterations have been executed for $T = 0.3146$. Now calculating the new value for $T$: $T = \alpha \times T = 0.8 \times 0.3146 = 0.2517 < 0.3 = T_{min}$, thus the method is ended.

The value of $T$ has become smaller than $T_{min}$, thus the method is ended. The final solution that has been found is figure A.47, with a total cost of 11.8. This is not equal to the optimal solution that had been found with dynamic programming, but is still quite good.

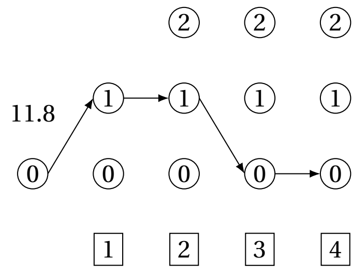


Figure A.47: Optimal solution of simulated annealing

# Bibliography

[1] AgentschapNL. Proeftuinen intelligente netten 2011-2015, April 2012.

[2] V. Bakker. Triana: a control strategy for smart grids: Forecasting, planning & real-time control. Master's thesis, University of Twente, January 2012.

[3] M. G. C. Bosman. Planning in smart grids. Master's thesis, University of Twente, 2012.

[4] Centraal Bureau voor de Statistiek. Elektriciteit en warmte; productie en inzet naar energiedrager. `https://opendata.cbs.nl/statline/#/CBS/nl/dataset/80030ned/table?ts=1524484063939`. Accessed: 22-05-2019.

[5] DNV GL. PowerMatching City: wonen en ondernemen in de energiewereld van morgen, April 2015.

[6] easyEnergy. Energietarieven. `https://www.easyenergy.com/nl/energietarieven`. Accessed:03-12-2018.

[7] Elia. Elia Data Download. `http://www.elia.be/en/grid-data/data-download`. Accessed:02-12-2018.

[8] Energy Numbers. What does the capacity factor of wind mean? `http://energynumbers.info/capacity-factor-of-wind`. Accessed:31-01-2019.

[9] Engie. BusinessLine - laadpaal voor uw bezoekers en werknemers. `https://www.engie.nl/laadpalen/onze-laadpalen/businessline`. Accessed:.

[10] Engie. BusinessLine: Laadpaal voor uw bezoekers en werknemers. `https://www.engie.nl/laadpalen/onze-laadpalen/businessline`. Accessed: 18-07-2016.

[11] European Wind Energy Association. Wind energy's frequently asked questions (FAQ). `http://www.ewea.org/wind-energy-basics/faq/`. Accessed: 21-07-2018.

[12] X. Fang, S. Misra, G. Xue, and D. Yang. Smart grid – the new and improved power grid. *IEEE Communications Surveys & Tutorials*, 14, 2012.

[13] H. Gharavi and R. Ghafurian. Smart grid: The electric energy system of the future. *Proceedings of the IEEE*, 99, June 2011.

[14] GreenWorld Partners. Energy storage. `https://www.greenworldpartners.com/energy-storage/`. Accessed:29-08-2018.

[15] J. Hanania, J. Jenden, K. Stenhouse, and J. Donev. Coal fired power plant. `https://energyeducation.ca/encyclopedia/Coal_fired_power_plant`. Accessed: 22-05-2019.

[16] Klimaatplein. Zelf energie opwekken. `https://www.klimaatplein.com/zelf-energie-opwekken`. Accessed: 21-07-2018.

[17] Lenovo. Lenovo Yoga 700-14ISK Tech specs. `https://www.lenovo.com/hk/en/laptops/lenovo-laptop/lenovo-yoga-series/Yoga-700-14/p/88YG7000615#tab-tech_specs`. Accessed:09-03-2019.

[18] P. A. Lombardi, K. Rudion, and M. Powalko. Optimal operation of a virtual power plant. *x*, April 2014.

[19] S. Lubach. Smart grids on industrial areas: business models to enhane the implementation of renewable energy. Master's thesis, Eindhoven University of Technology, March 2013.

[20] Milieu Centraal. Zonnepanelen kopen. `https://www.milieucentraal.nl/energie-besparen/zonnepanelen/zonnepanelen-kopen/`. Accessed:06-01-2019.

[21] minder.nl De zakelijke energievergelijker. Terugleververgoeding zakelijke energieleveranciers. `https://www.minder.nl/faq/groene-energie/terugleververgoeding-energieleveranciers`. Accessed: 20-06-2018.

[22] minder.nl De zakelijke energievergelijker. Zakelijk terugleveren. `https://www.minder.nl/zonnepanelen/zakelijk-terugleveren`. Accessed: 20-06-2018.

[23] Ministerie van Economische Zaken. Energieagenda: Naar een co$_2$-arme energievoorziening, December 2016.

[24] E. Morse. Non-renewable energy. `https://www.nationalgeographic.org/encyclopedia/non-renewable-energy/`. Accessed: 22-05-2019.

[25] J. P. J. Navarro, K. C. Kavvadias, S. Quoilin, and A. Zucker. The joint effect of centralised cogeneration plants and thermal storage on the efficiency and cost of the power system. *Science Direct Energy*, April 2018.

[26] Nederlandse Energie Data Uitwisseling. Verbruiksprofielen. `https://www.nedu.nl/documenten/verbruiksprofielen`. Accessed:22-11-2018.

[27] Netbeheer Nederand. Uit Net NL: Over prijzen, pieken & dalen. Accessed:29-03-2019.

[28] NUON. Hoe zit het Nederlandse elektriciteitsnetwerk in elkaar? `https://www.nuon.nl/grootzakelijk/energiekenners/business-bibliotheek/nederlandse-elektriciteitsnetwerk/`. Accessed: 14-05-2019.

[29] Ozgur. What is capacity factor and how do solar and wind energy compare? `https://sunmetrix.com/what-is-capacity-factor-and-how-does-solar-energy-compare/`, July 2013. Accessed: 21-07-2018.

[30] Rijksdienst voor Ondernemend Nederland. Smart grids en slimme energiesystemen. `https://www.rvo.nl/onderwerpen/duurzaam-ondernemen/energie-en-milieu-innovaties/smart-grids`. Accessed:18-04-2018.

[31] Rijksoverheid. Smart Grids proeftuinen IPIN. `https://www.energiekaart.net/wp-content/uploads/2015/11/2015-IPIN-projects-results-RVO-1.pdf`, September 2015. Accessed: 18-04-2018.

[32] Rijksoverheid voor Ondernemend Nederland. ISDE Zakelijke gebruikers. `https://www.rvo.nl/subsidies-regelingen/investeringssubsidie-duurzame-energie-isde/isde-aanvragen/isde-zakelijke-gebruikers`. Accessed: 21-07-2018.

[33] Rijksoverheid voor Ondernemend Nederland. Rijksoverheid stimuleert duurzame energie. `https://www.rijksoverheid.nl/onderwerpen/duurzame-energie/meer-duurzame-energie-in-de-toekomst`. Accessed: 14-05-2019.

[34] Rijksoverheid voor Ondernemend Nederland. Position paper kennis- en leertraject, Thema: wet- en regelgeving, September 2015.

[35] P. Samadi, A.-H. Mohsenian-Rad, V. W. S. Wong, and J. Jatskevich. Optimal real-time pricing algorithm based on utility maximization for smart grid. *IEEE International Conference on Smart Grid Communications*, 2010.

[36] S. Shafiee and E. Topal. When will fossiel fuel reserves be diminished? *ScienceDirect*, 37:181–189, January 2009.

[37] Sociaal Economische Raad. Energieakkoord voor duurzame groei, September 2013.

[38] Tesla. Powerwall. `https://www.tesla.com/nl_NL/powerwall`. Accessed:04-12-2018.

[39] A. Thompson. 7 ways to power your home with renewable energy. `https://www.popularmechanics.com/science/energy/g2825/7-ways-to-power-your-home-with-renewable-energy`. Accessed: 20-07-2018.

[40] S. van der Kooij. Optimal charging/discharging strategies in smart energy grids. Master's thesis, VU Amsterdam, November 2016.

[41] B. van Wezel. Elektriciteit in nederland, February 2015.

[42] T. Vijayapriya and D. P. Kothari. Smart Grid: An Overview. *Scientific Research*, June 2011.

[43] C. Wasilenkoff. The future of cogeneration is green. `https://breakingenergy.com/2013/11/01/the-future-of-cogeneration-is-green/`. Accessed: 23-05-2019.

[44] Witteveen+Bos. Witteveen+Bos - About us. `https://www.witteveenbos.com/about-us/`. Accessed:06-01-2019.

[45] L. A. Wolsey. *Integer programming*. Wiley-Interscience publication, 1998.

[46] Z. Zhu, S. Lambotharan, W. H. Chin, Z. Fan, and J. Tang. An integer linear programming based optimization for home demand-side management in smart grid. *IEEE PES Innovative Smart Grid Technologies*, 2012.

[47] B. Zientara. How much electricity does a solar panel produce? Accessed: 21-07-2018.