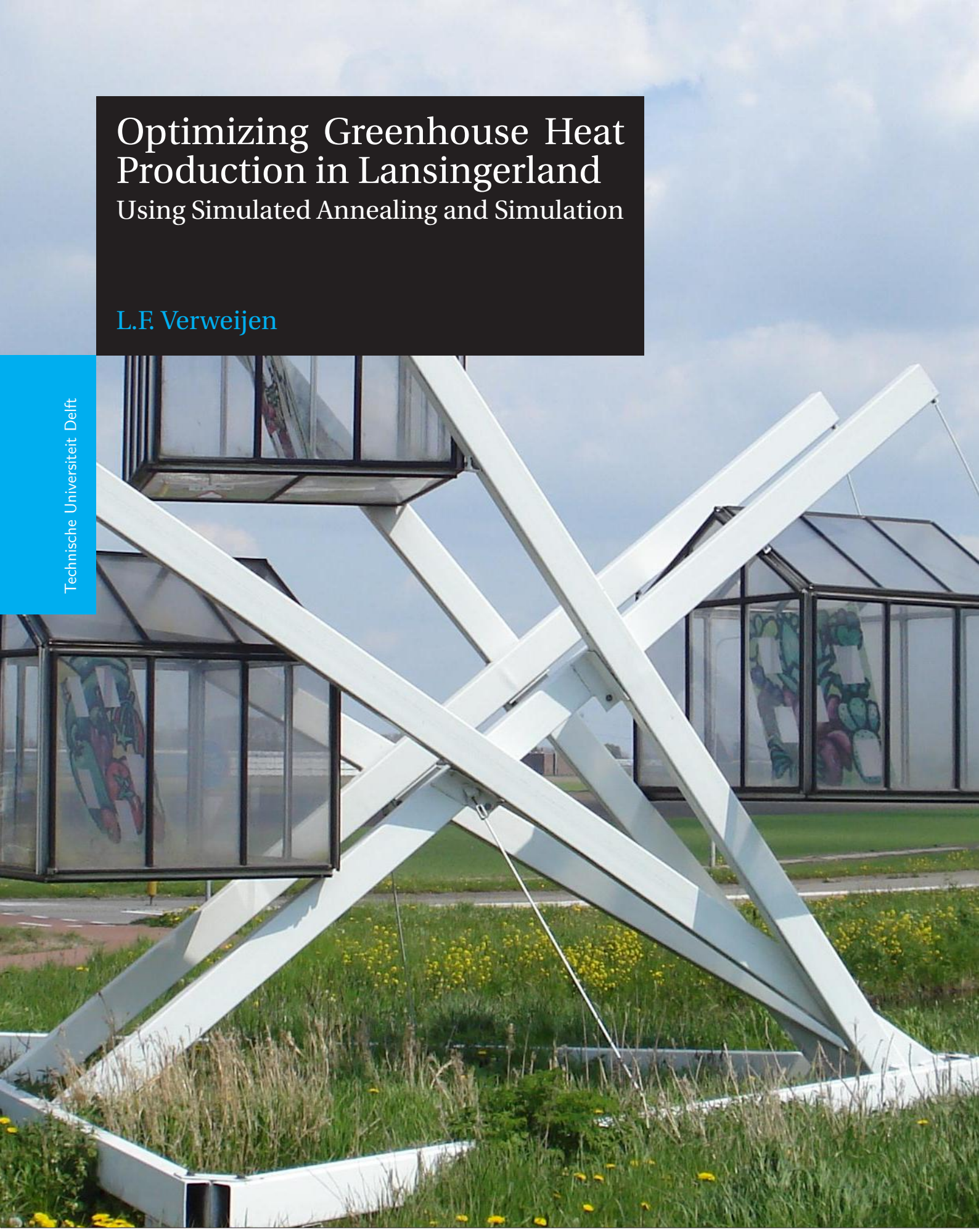
# Optimizing Greenhouse Heat Production in Lansingerland

## Using Simulated Annealing and Simulation

L.F. Verweijen

Technische Universiteit Delft



TU Delft Delft University of Technology

Challenge the future

# Optimizing Greenhouse Heat Production in Lansingerland

## Using Simulated Annealing and Simulation

by

**L.F. Verweijen**

in partial fulfilment of the requirements for the degree of

**Master of Science**
in Computer Science

at the Delft University of Technology,
to be defended publicly on Friday 16 December 2016 at 13:00.

| | |
|---|---|
| Supervisors: | Prof. dr. Sicco Verwer |
| | Prof. dr. Mathijs de Weerdt |
| Thesis committee: | Prof. dr. ir. Jan van den Berg |

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

TU Delft Delft University of Technology

# ABSTRACT

Many engineering problems require some objective to be optimized subject to certain constraints. For linear and convex optimization problems, techniques exist that can find the global minimum. For problems that have local minima or complex constraints, finding the global minimum is more difficult. An example of such a problem can be found in Lansingerland, where heat needs to be supplied by greenhouses.

In this work a heat generation problem that has a local objective but complex constraints is solved using simulated annealing. The problem has the form of a demand supply problem and is applied in the context of generating heat for greenhouses in the region of Lansingerland. The objective is the costs that we want to keep as low as possible. The constraints arise from the network used to transport the heat, because the pressure in the pipes and valves of this network needs to be kept under control.

Prior to coming up with a solution, the patterns of the heat demand are studied to decide what conditions the solutions should take into account. It was found that days follow a similar pattern and that the demand is higher in winter than in summer months. Furthermore the distribution of the heat demand over the different greenhouses is roughly equal for each day.

Then, a look is taken at how a heat control can be solved using a stochastic optimization technique without considering variations during the day. Instead the problem is solved as if the heat demand for a single day is constant. We improve on canonical simulated annealing by making our mutations more intelligent than random. Two better approaches are shown to accomplish this. One starts by an analysis of trying different solutions to see what works best under what circumstances. This will be referred to as a smart mutation. The other approach varies the type of mutation based on earlier iterations. This will be referred to as an adaptive mutation. Both approaches show similar results, but since the adaptive mutation doesn't require much domain knowledge, this one is easier to apply to other problems. The influence of the starting point is also explored. In our setup it mattered a lot whether the initial point was already located in the feasible domain or not, for how quickly a solution can be found. We found that a good initial scenario should be one that already starts in the feasible domain. The best result was obtained by creating an initial scenario that for each greenhouse randomly decides to use the CHP or the RoCa. The mutation that gave best results in our case was the adaptive mutation, but it is expected that if more domain knowledge is added the smart mutation would be the best one.

Finally, we will demonstrate how a heat control problem can be solved if the heat demand varies over time. In order to do so, a new way is introduced to segment time-series. This segmentation will be used to cluster the heat demand for a single day and to solve the problem for each segment individually. This improves the solution considerably, because it enables us to generate more heat at cold moments of the day like midnight, but to save on costs during the hot parts of the day.

# Contents

# PREFACE

*As the final project for my master studies, it has been my goal to come up with a method to find good production schemes to provide heat to greenhouses.*
*I would like to thank my supervisors, Mathijs de Weerdt and Sicco Verwer, for their feedback, counsel and for giving me the opportunity to work on this project. In addition, I want to thank Stephan Mes for helping me understand the physics that are of influence on this system. Second, I want to thank my friends and family (my parents, sister and dog) for their support. Third, I want to thank all people that I have shared the room in EWI/EEMCS with while working on this project for their company and occasionally joining me for a tea / coffee break.*

*L.F. Verweijen*
*Delft, December 2016*

# 1

# INTRODUCTION

This work is dedicated to finding an algorithm that produces a cost-efficient produce scheme to generate heat for greenhouses in Lansingerland given the energy demands over the day within the imposed constraints. Lansingerland is an area in Rotterdam consisting of the municipalities Bergschenhoek, Bleiswijk and Berkel and Rodenrijs. In this area greenhouses are located that are supplied heat by means of a thermal grid network. In the old situation, heat used to be generated centrally and provided through the network to the greenhouses. A picture of this network can be found in Figure 1.1. Because of economical reasons, the network is being transformed into a so called *smart thermal grid*. Unlike the old situation, this new grid will be decentralized and most of the heat will be produced at the individual greenhouses and be shared over the network.

Energy provider Eneco has started a project to investigate the situation and this project is divided in the following parts:

1. Development of a 'dynamic netmodel'. Unlike the old model, which was centralized, this new model can have multiple energy sources.
2. Greenhouse growers need to be able to supply and receive heat to the network.
3. Supply and demand of heat need to be matched in an optimal way, while satisfying the physical constraints of the netmodel. To evaluate the physical feasibility, a model called Drukval is used that comes in the form of a Fortran program written by Stephan Mes, who is an employee of Eneco.

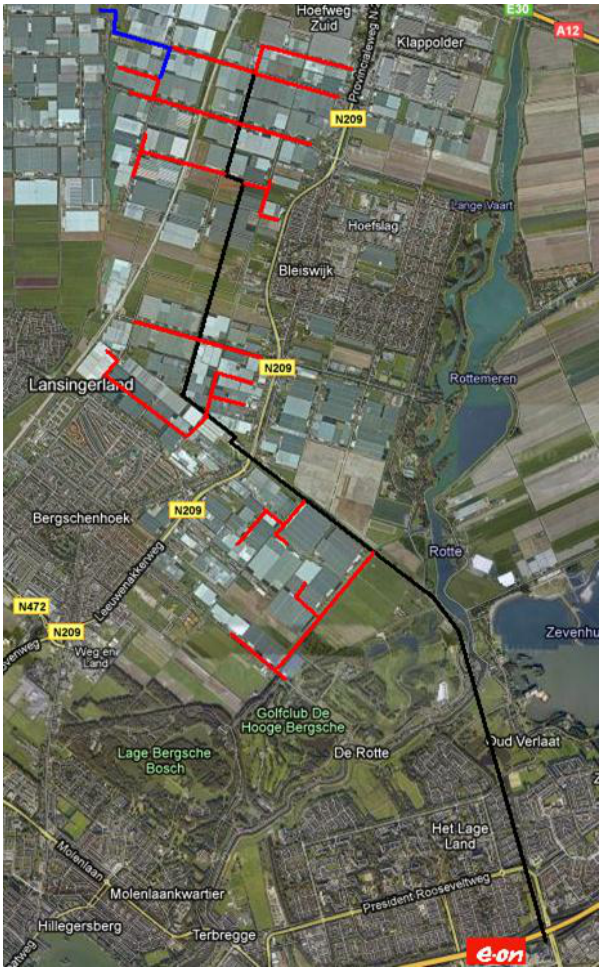



Figure 1.1: The old situation with the pipeline of RoCa in black and the pipelines to the greenhouses in red. (Arcadis, 2012).

In this thesis we explore how optimizing the energy production for production costs can be done as quickly and accurately as possible. The Drukval model can be used directly in the optimization process or indirectly by training an aggregate, a simplified model that can be optimized more efficiently. This will be explained further in Section 2.5.

### MAIN CONTRIBUTIONS

This work tackles a practical optimization problem with an objective and constraints using simulated annealing. It explores the choices that work well in the context of a constrained heat supply and demand problem. An initial heat generation scenario and a mutation need to be chosen for simulated annealing to work with. Although we apply our solution to the heat generation problem, some of the principles can also be used in other contexts like deciding the next best decision depending on what situation you are. Furthermore, a method is provided to cluster a linear time series that has some useful theoretical properties.

### THESIS OUTLINE

In Chapter 2, the problem will be explained in greater detail and the research question will be formulated. In Chapter 3, the data of our problem will be analysed in great detail. In Chapter 4, the problem will be tackled by simulated annealing. Ultimately, in Chapter 5 the contribution of this thesis and future work is discussed.

# 2

# PROBLEM DESCRIPTION

The goal of this project is to find a method such that given the heat demands over a given day, a cost efficient heat production scheme can be found in reasonable time that satisfies these demands subject to constraints.

## 2.1. PROBLEM SKETCH

In Lansingerland, 104 greenhouses are growing crops. Each of the greenhouses needs to operate on a fixed temperature. Due to warmth transmission, infiltration, ventilation and warmth used for the preparation of tap water, some of the heat gets lost. The amount of heat lost by these influences, linearly depends on the difference between the temperature in the greenhouses (which will from now on be referred to as the indoor temperature) and the outdoor temperature. This creates a heat demand for each greenhouse that varies during the time of day.

### 2.1.1. TRANSITION TO THE SMART GRID

In the old situation which is depicted in Figure 2.1a, heat was mainly produced by the RoCa from E.ON, which got transported through a unidirectional grid of heat pipes to the greenhouse owners, who need it for crop production. When the maximum production capacity of the RoCa was reached, the heat production would be supplemented by CHPs or boilers, but the heat produced by the CHPs was not shared through the network. In the network heat exchangers can be found. These are devices that can be used to transport heat from the grid to the individual greenhouses. There are two streams of water. One stream flowing forward from the RoCa to the greenhouses that has hot water and one flowing back from the greenhouses to the RoCa that has cold water.

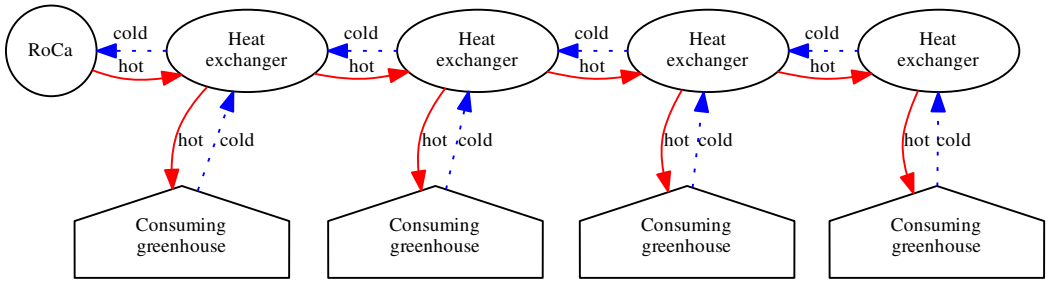
In the new situation, which is depicted in Figure 2.1b, some of the heat will still be produced centrally, but in addition heat produced by the RoCa of individual greenhouse owners can be put on the network as well and be used by other greenhouse owners. It can be said that in the new situation the greenhouses with a CHP have changed from consumers into prosumers, greenhouse that occasionally consumers and occasionally produce heat themselves. Software is needed to manage the supply and demand of heat in this new decentralized thermal grid and to prepare a heat plan for a single day. In this new situation the heat exchangers are also used to transport heat from the producing greenhouses to the grid.

## 2.2. HEAT SOURCES

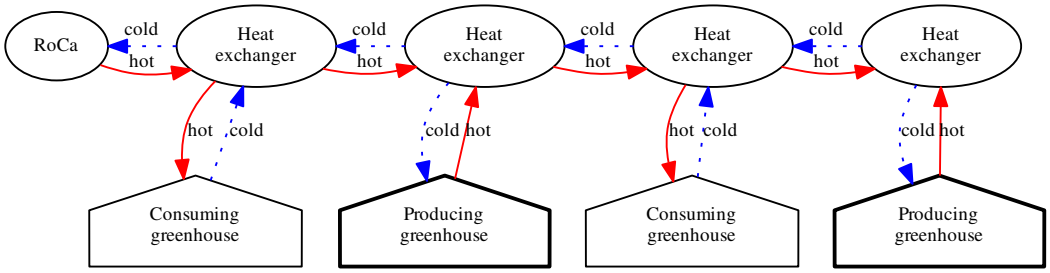
The available sources for generating heat are the RoCa, CHPs and the boilers. The buffers are available for temporary heat storage.

**RoCa** A huge powerplant owned by Eneco for centralized heat production. This heat will be put on the network and be transported to the individual greenhouses through the pipeline. This power plant produces gas and steam at the same time and has a maximum power of 200 MW. The RoCa has a start-up time of 4 hours, but is flexible in its heat production.

**CHP** Units that produce heat and electricity at the same time. Most greenhouse owners have one. They can choose to generate heat for their own usage or share it with the other greenhouses through the same pipeline. Hence these provide for decentralized heat production.




(a) The old situation. Heat was produced centrally by the RoCa and transported to the individual greenhouses.




(b) New situation. There will still be a RoCa, but the individual greenhouses can optionally temporarily become producers too.

Figure 2.1: The old and new situation of the heat grid.

Their production is linear, but when the CHPs deliver to the network, their outgoing debits are fixed to a contractual value. For each greenhouse, a fixed debit value is provided for the winter months (January, February and March) and the spring months (March, April and June).

**Boiler** Each greenhouse owner has a boiler. They are not used a lot since heat from the network is cheaper. It mainly serves as a backup heat source or when the heat demand is higher than the maximum heat generation of the other sources combined. They are only used for private usage so heat produced this way is not shared through the network. Because for this thesis infinite production capacity is assumed for the RoCa and the CHPs, the boilers are not considered. If production capacities were considered, it is not certain these would be reached, but in case they are, the result would be higher costs, but no extra pressure on the network.

**Buffer** Most of the greenhouses have a buffer unit in which heat can be stored for later usage. Each of these buffers has a different maximum capacity. When stored in the buffer the heat will have a slow decay that will be neglected. When the day ends, the amount of heat stored in the buffer should be the same as at the beginning, so all heat generated should be used within 24 hours. The buffers are not taken into account in this research. If they were taken into account, the buffer could be used to save up some of the heat during the afternoon hours, during which the demand is low and be used at night when it's colder. It would thus smoothen the demand curve, which will be discussed in more detail in Chapter 3.

The costs of each heat source will be discussed in detail in Section 2.6. A heat source that is not used yet but might become relevant in the future is geothermal energy. This heat source will be discussed in Paragraph 5.2.1.

## 2.3. DESCRIPTION OF THE DEMAND DATASETS

The heat demand for the months January to June (180 days) has been provided by Eneco which contains the heat demand for each of the individual greenhouse owners for every interval of 5 minutes of the day in MWh (that means 288 moments per day) for 104 greenhouses. There were some inconsistencies in the formatting of the date and for some months some of the intervals are missing. This has been solved by converting the date format to YYYY-MM-DD hh:mm and by linear interpolation of the missing data. The time intervals over which data was missing were never larger than 10 minutes, so it is expected that the resulting data after interpolation is still accurate. The months January to March will be considered to be winter months and the months April to June will be considered spring months. The system is going to be used for day-ahead planning, so each day, which starts and ends at midnight, will be considered a separate problem. These heat demands will be studied in detail in Chapter 3.

## 2.4. SCENARIO REPRESENTATION

The heat demand that is supplied to us is divided in intervals of 5 minutes for which the heat demand needs to be fulfilled. A solution is specified by the heat production of the RoCa and the 15 greenhouse CHPs for each interval. The simplest representation of a scenario is to indicate for each greenhouse for each point in time whether the greenhouse is producing or consuming and if they are producing, how much is being produced. The production and consumption per source depend on this in the following way:

- If a greenhouse is producing, it will deliver a fixed amount to the network. Technically, it takes 10 minutes for a CHP to run on its full potential. The greenhouse will produce slightly more than it delivers to the network, because it also needs to fulfil its own production. If a greenhouse is consuming, it will request heat from the network to fulfil its heat demand.
- If more heat is demanded by the greenhouses than the total production of all CHP's, the remaining heat is produced by the RoCa. This will create an ingoing flow at the greenhouse that are consuming.
- If more heat is demanded than the total production of all CHP's and the maximum RoCa production, the remaining heat is produced by the individual boilers. This will have no effect on the heat transportation network. Since no maximum production of the RoCa is known, this heat source is ignored.

The boilers are used to supplement the heat when the other heat sources are not fulfilling the demand and can be calculated by calculating how much of the demand $d(t)$ is unfulfilled by the heat delivery of the

CHPs and the RoCa:

$$H_{\text{boilers}} = \sum d(t) - H_{\text{CHP}} - H_{\text{RoCa}} \tag{2.1}$$

The boilers are only used for a greenhouse's own consumption and its heat is not put on the network. If the CHP produces, it produces both for the greenhouse itself and for the network. It is assumed there is no limit on the amount of heat CHP can produce, but its production is still limited, because the heat is distributed on the network on a fixed value, so the maximum it would produce is the debit of the greenhouse it belongs to and the maximum it can put on the network.

### 2.4.1. FEASIBILITY OF SCENARIOS

The problem is further complicated by the fact that in the new situation, the pressure on the pipes is much higher.

There are two kinds of constraints that would make a scenario infeasible:

- The heat demand consumption of a specific greenhouse at a specific time interval is lower than demanded. This problem is easily handled by employing private heaters, which are assumed to have unlimited capacity and high costs whenever the heat at a greenhouse is unfulfilled. This is the most expensive heat source, though.
- The pressure on the valves is too high. The physics are described by Van der Mes [1], but a summary of the relevant parts is given in Section 2.5.

## 2.5. TECHNICAL AND PHYSICAL CONSTRAINTS ON PIPES

When water flows through a pipe, there needs to be a pressure difference between both end points of that pipe. Water will flow from a point where the pressure is high to a point where the pressure is low. To calculate how much pressure is needed, first the pressure that gets lost through the pipes, the so called pressure drop, has to be calculated. The pressure loss $\delta P$ (measured in bar) in pipes depends linearly on the average streaming speed of the water through that pipe $v$ (measured in m/s) and the density of water $\rho$ (about 1000 kg/m$^3$, depending on how pure the water is). Furthermore, for straight pipes this pressure difference depends linearly on the length of that pipe $L$ (measured in m) and the resistance factor $\lambda$ and is inversely proportional to the diameter $d_i$ (measured in m) of that pipe. For a curved pipe $i$, the pressure loss depends linearly on a coefficient $\zeta_i$ which is the resistance coefficient for that pipe, a value which is supplied by the pipe manufacturer. So for a straight pipe without any curves, the dynamic pressure loss follows the following formula:

$$\delta P_i = \lambda \frac{L}{d_i} \cdot \frac{1}{2} \rho v^2 \tag{2.2}$$

For a curved pipe, the dynamic pressure loss follows the following formula:

$$\delta P_i = \zeta_i \cdot \frac{1}{2} \rho v^2 \tag{2.3}$$

The liquid speed $v$ can be calculated using:

$$v = \frac{Q_v}{\frac{\pi}{4} d_i^2} \tag{2.4}$$

Furthermore, valves might be of influence on the pressure loss. It is assumed that the effect of these valves is negligible.

The total dynamic pressure loss of the net can be calculated from those of the individual pipes and valves in the network:

$$\delta P_{\text{total}} = \sum_i \delta P_i \tag{2.5}$$

To realize a heat flow $\Phi_{th}[kW]$, water is used. Furthermore, heat can be transported faster when the temperature difference $\Delta T[K]$ is small. The volume stream or debit $Q_v[m^3/h]$ can be calculated by:

$$Q_v = \frac{3600 \cdot \Phi_{th}}{c \cdot \rho \cdot \Delta T} \tag{2.6}$$

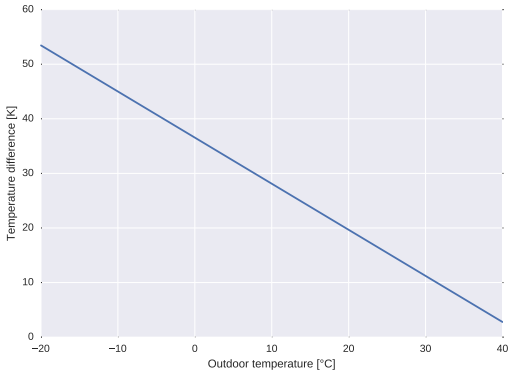



Figure 2.2: Temperature difference depending on the outdoor temperature

depending on the density of water $\rho = 1000\,\mathrm{kg\,m^{-3}}$, the specific heat $c = 4.19\,\mathrm{kJ\,kg^{-1}\,K^{-1}}$ and the temperature difference $\Delta T$[K] between the temperature at which heat is supplied and the return temperature.

The supply temperature is the temperature at which water goes into the greenhouse and the return temperature the temperature at which the water returns. The supply should be such that the required temperature in the greenhouse is achieved. The colder it is outside, the higher the supply temperature should be to compensate. This difference $\Delta T$[K] can be calculated by the following formula using the outdoor temperature $T_o[^\circ C]$, the supply temperature $T_s[^\circ C]$ and the return temperature $T_r[^\circ C]$:

$$T_s = \frac{35}{25}(15^\circ C - T_o) + 80^\circ C \tag{2.7}$$

$$T_r = \frac{10}{18}(8^\circ C - T_o) + 60^\circ C \tag{2.8}$$

$$\Delta T = \max(T_s - T_r, 0) \tag{2.9}$$

The result of these formulas is a linear dependency between the outdoor temperature and the temperature difference between the supply and return temperature as depicted in Figure 2.2. According to the KNMI[2] the average temperature in the Netherlands is $10^\circ C$, thus on average $\Delta T = 30^\circ C$.

The pressure difference that needs to be created should not just compensate for the pressure loss, but also needs to take the vertical distance $H$ between the liquid heights, the difference in height between where the liquid is put on the network and where it is removed, into account. This required pressure difference is called the discharge pressure and its formula is:

$$P = \rho \cdot g \cdot H + \delta P \tag{2.10}$$

where $g$ is the gravity acceleration ($9.81 ms^{-2}$) and $\rho$ is the density of water $1000\mathrm{kg/m^3}$.

However, the pressure can not be too high and there is a so called pump curve. A violation of this pump curve occurs when:

$$P > P_{\text{limit}} = P_{\max}\left(1 - \left(\frac{Qv}{Qv_{\max}}\right)^2\right) \tag{2.11}$$

Where $P$[bar] is the pressure of a pump, $P_{\max} = 8\mathrm{bar}$, $Q_v(m^3/h)$ is the debit of the CHP and $Qv_{\max}$ is the maximum debit which depends on the size of the pump which that CHP is using. This pump curve is plotted in Figure 2.3.

A program called *Drukval* exists that calculates the pressures of each pump from the debits at each greenhouse. As input to this program debits are specified for each greenhouse for every 5 minutes of a day. Those debits have positive values when a greenhouse is consuming heat from the network. When a greenhouse is supplying heat to the network, the debits have negative values.

There are two setups in which Drukval can be used. The straightforward way is to use the setup in Figure 2.4a, where we feed the program a scenario to figure out whether it is feasible or not. An alternative to speed up evaluation is shown in Figure 2.4b, where the choice is made to first learn a simplified linear model that behaves just like Drukval and to use the simplified model instead. Such an approach has been used before for auction problems[3]. Such an approach has the advantage that a linear model is easy to optimize. The disadvantage is that it is difficult to find a model that represents Drukval well. Our trained model might

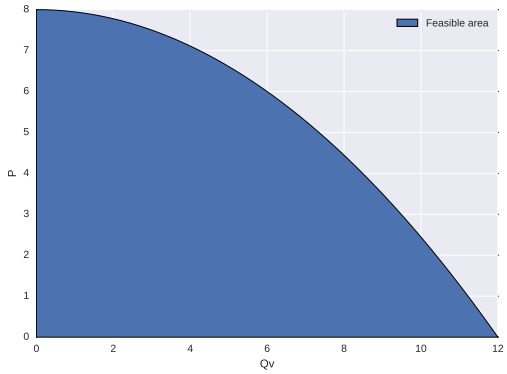



Figure 2.3: Pump curve

| Heat source | Marginal heat costs per GJ |
|---|---|
| CHP | € 6.37 |
| RoCa | € 8.30 |
| Boiler | € 9.89 |

Table 2.1: Costs by heat source according to Van den Ende [4]

erroneously report correct scenarios as incorrect (false negatives) or incorrect scenario as correct (false positives) and might therefore not be able to find the optimum scenario. The choice was made to use Drukval directly, but the alternative will be discussed again in Section 5.2.5.

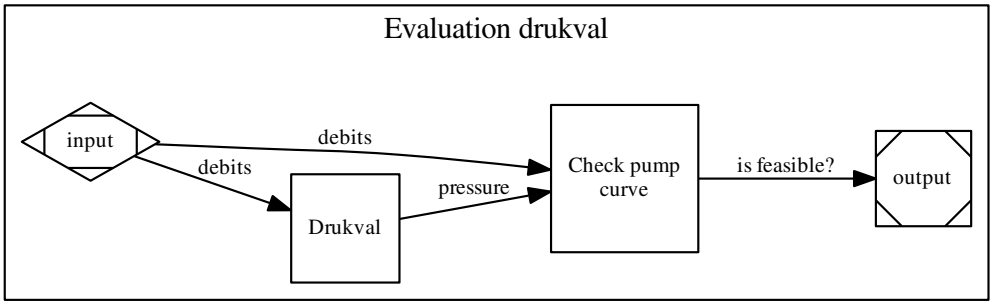
## 2.6. COSTS

When dealing with the costs for each greenhouse there are fixed costs, variable costs and earnings as can be seen in Equation 2.12. Fixed costs include deprecation, maintenance and heat losses. However, as these costs have no influence on the optimization routine, they will be ignored for the rest of this thesis. Technically heat losses are not fixed costs as they depend on the temperature difference between the heated water and its surroundings. As they are independent from the amount of heat transported through the network, they will be treated as fixed costs in this thesis.

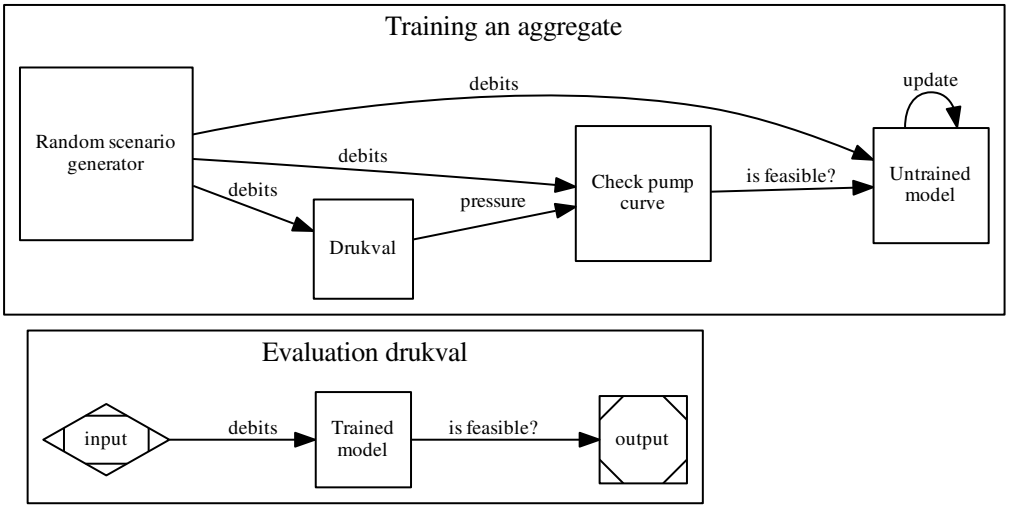
$$\text{Heat cost} = \text{Fixed costs} + \text{Variable costs} - \text{Earnings} \tag{2.12}$$

**Production costs** For the production costs, two prices play a role: the gas price and the electricity price. The gas price is stable and is treated as a fixed price per day. The electricity price is irregular and depends on the current rate on the APX, an energy exchange active in the Netherlands. Producing electricity saves the companies from buying it and a surplus can even be sold on the market. Electricity is produced by the CHPs and the RoCa. The consequence is that a low electricity price could make the boilers more profitable, whereas a high electricity price makes the CHPs and the Roca become the efficient option. This price will be treated as fixed per hour. The CHPs also produce $CO_2$, but according to the report of Samira [5] this $CO_2$ "is not directly applicable".

In Figure 2.5, graphs by Van der Ende [4] illustrate how the electricity and gas price influence the heat cost per MWh of heat produced. As can be seen both the production costs of the CHP and RoCa decline linearly when the electricity price increases. When the electricity price weakens, it becomes more profitable to use boilers. Because gas is needed for production, a higher gas price results in higher production costs. CHPs and the RoCa need more gas per unit of heat produced than boilers and hence their production costs are slightly more sensitive to fluctuations in gas price (which does not fluctuate as much as the electricity price). The figure also shows geothermal plants. Geothermal plants are not used yet, but if they were used, its production costs would be relatively cheap because of government subsidy.

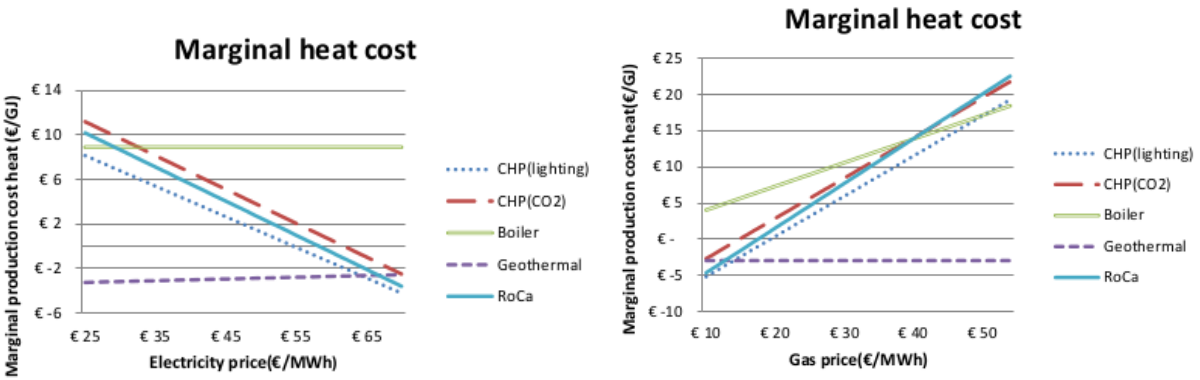



(a) Using drukval directly to determine if a scenario is feasible




(b) Using drukval indirectly by first training a machine learning model and using that model to determine if a scenario is feasible

Figure 2.4: Drukval can be used in two ways




(a) Heat cost depending on electricity price

(b) Heat cost depending on gas price

Figure 2.5: Heat cost graphs (created by Mark van den Ende [4])

**Transportation costs** There are also costs related to pumping the heat through the network. These costs depend linearly on the amount of heat pumped through the network and the pressure difference. In the thesis by Mark van den Ende[4], the following table of estimates for the pumping cost $T_{cost}$ are given. Values in-between can be estimated by interpolation, e.g. linear interpolation.

| Pressure loss(bar) | Pumping cost (per MWh) |
|---|---|
| 4 | €0.33 |
| 7 | €0.57 |
| 10 | €0.81 |
| 13 | €1.06 |

These costs only need to be paid when heat needs to be transported, which only happens when a greenhouse does not produce enough heat on its own, i.e. when the generated heat $H_G ij$ is lower than the demanded heat $H_D ij$ at time $i$ for greenhouse $j$:

$$C_{Tij} = \begin{cases} (H_{Dij} - H_{Gij}) \times T_{cost} & \text{if } H_{Gij} \leq H_{Dij} \\ 0 & \text{else} \end{cases} \tag{2.13}$$

### 2.6.1. SIMPLIFIED COST MODEL

In our optimization, only the production costs will be taken into account, because these costs are expected to be the dominant ones. For the different heat sources, the prices from Table 2.1 will be used. In reality, the prices are varied throughout the day, but these patterns are unknown to us. So for calculating the costs Equation (2.14) is used.

$$C = 6.37 \cdot P_{\text{CHP}} + 8.30 \cdot P_{\text{RoCa}} + 9.89 \cdot P_{\text{boiler}} \tag{2.14}$$

## 2.7. OPTIMIZATION

Figure 2.6 shows how a single scenario is ranked. For a demand, the CHP production $P_{\text{RoCa}_i}$ for each greenhouse is decided using a method that will be discussed in Chapter 4. The additional demand is satisfied using the RoCa, which will produce $P_{\text{RoCa}}$, and the boiler, which will produce $P_{\text{boiler}_i}$, as long as it's unfulfilled. Now the production costs can be calculated per heat source.

$$P_{\text{CHP}} = \sum_{i=1}^{15} P_{\text{CHP}_i} \tag{2.15}$$

The costs are given by:

$$C_{\text{RoCa}} = 6.37 \cdot P_{\text{RoCa}} \tag{2.16}$$

$$C_{\text{CHP}} = 8.30 \cdot P_{\text{CHP}} \tag{2.17}$$

$$C_{\text{boiler}} = 9.89 \cdot P_{\text{boiler}} \tag{2.18}$$

From the RoCa and CHP production the debits are calculated using Equation (2.6) and given to the Drukval program to calculate the pressure in the valves. The pump curve between these pressures and debits are checked for feasibility using Equation (2.11). Now the costs and pump curve are known for this single scenario.

Our goal is to minimize the total production costs throughout the day.

$$\min \sum_{t=1}^{24} (C_{\text{RoCa}}(t) + C_{\text{CHP}}(t) + C_{\text{boiler}}(t)) \tag{2.19}$$

### 2.7.1. MATCHING SUPPLY AND DEMAND

Given a heat demand throughout the day of $D(t)$ and the total production at time $t$ is given by:

$$P_{\text{total}}(t) = P_{\text{RoCa}}(t) + P_{\text{CHP}}(t) + P_{\text{boiler}}(t) \tag{2.20}$$

For there to be enough heat, we want supply and demand to match during each time of the day such that:

$$P_{\text{total}}(t) \geq D(t) \forall t \tag{2.21}$$

### 2.7.2. INCLUDING PHYSICAL AND TECHNICAL CONSTRAINTS

In addition to buffer constraints, we will also need to include physical and technical constraints from Section 2.5.

For this, the Drukval program needs to be run first to calculate the pressures for every greenhouse $g \in G$:

$$DP_g \forall g \in G \tag{2.22}$$

And make sure that:

$$f_g = \frac{dP_g}{P_{\text{limit}}} \tag{2.23}$$

$$= \frac{dP_g}{dP_{\max} \cdot \left(1 - \left(\frac{Qv_g}{Qv_{\max}}\right)^2\right)} < 1 \tag{2.24}$$

for all greenhouses $g$ as discussed in Section 2.5.

## 2.8. EXAMPLE CALCULATION

In Section 2.8 the heat demands $H_D$ and heat production $H_G$ are given of a small network with three greenhouses. For each of the greenhouses the heat demand $H_D$ and generated heat $H_G$ during a specific hour are given in MWh. Greenhouses WAS51 and WAS52 are both supplying heat to the network, whereas greenhouse WAS54 is only extracting heat.

In an 1-hour interval the demanded heat is $1257 + 21998 + 0 = 23225$ MWh and the generated heat is $0 + 21998 + 9428 = 31426$ MWh.

The role of buffers is ignored for now. When the demanded heat is larger than the generated heat for a greenhouse, heat is extracted from the network and when the demanded heat is smaller than the generated heat for a greenhouse, heat is put on the network. Furthermore, it is assumed that the outdoor temperature is $10°C$ anytime anywhere, which according to Figure 2.2 means that $\Delta T = 30K$. So in this example filling in Equation (2.6) the debit can be calculated as $Q = \frac{H_D - H_G}{1000 \cdot 4.19 \cdot 30}$.

Now for calculating the pressure difference $D_P$, the Drukval program by Stephan Mes is used, which only needs the debit as input.

To decide its feasibility, the following helper variable is created:

$$P_{\text{limit}} = dP_{\max} \cdot \left(1 - \left(\frac{Qv}{Qv_{\max}}\right)^2\right) \tag{2.25}$$

Another view of the feasibility is given by the variable $f$ defined as follows:

$$f = \frac{dP}{P_{\text{limit}}} \tag{2.26}$$

$$= \frac{dP}{dP_{\max} \cdot \left(1 - \left(\frac{Qv}{Qv_{\max}}\right)^2\right)} \tag{2.27}$$

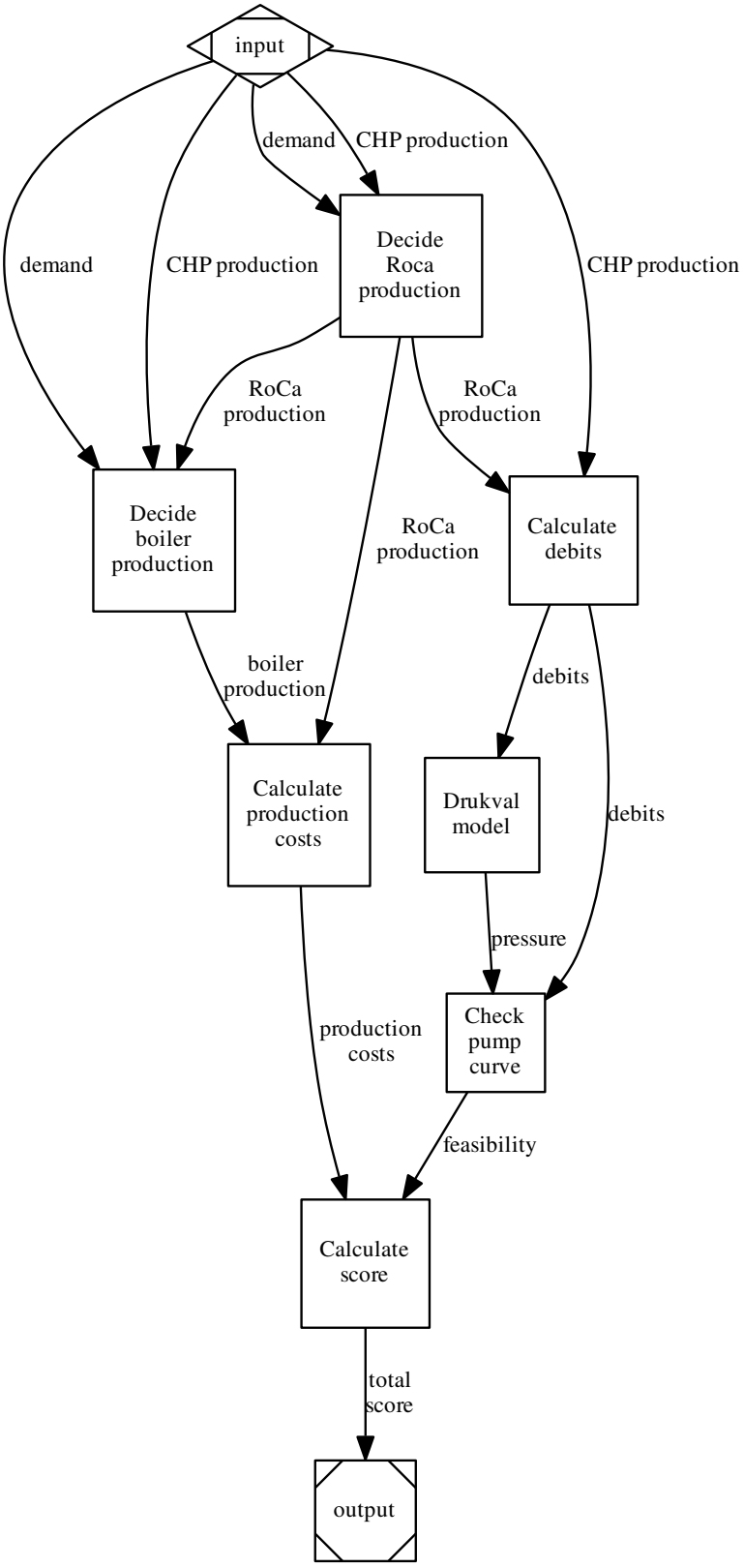



Figure 2.6: Evaluation of a single scenario

| | RoCa | | WAS052 | | WAS054 | | WAS051 | |
|---|---|---|---|---|---|---|---|---|
| | $H_D$ | $H_G$ | $H_D$ | $H_G$ | $H_D$ | $H_G$ | $H_D$ | $H_G$ |
| 12:00:00 | 0 | 1.06e+08 | 1201.4 | 25.1 | 781.5 | 8.8 | 2.8 | 19.9 |
| 13:00:00 | 0 | 1.06e+08 | 0.3 | 25.1 | 85.0 | 8.8 | 2.9 | 19.9 |
| 14:00:00 | 0 | 1.06e+08 | 0.2 | 25.1 | 3160.0 | 8.8 | 2.9 | 19.9 |
| 15:00:00 | 0 | 1.06e+08 | 0.2 | 25.1 | 69.9 | 8.8 | 3.0 | 19.9 |
| 16:00:00 | 0 | 1.06e+08 | 891.5 | 25.1 | 3157.3 | 8.8 | 3.0 | 19.9 |

Table 2.2: Example problem heat demands $H_D$ and heat production $H_G$

| | Production | Costs |
|---|---|---|
| RoCa | 1.06e+08 | 876.8 |
| CHP | 1.06e+06 | 6.8 |
| Boiler | 0 | 0.0 |

Table 2.3: Example problem costs per heat source

| | WAS52 | WAS51 |
|---|---|---|
| $Qv$ | 0.0 | 0.0 |
| $dP$ | 0.0 | 0.0 |
| $P_{\text{limit}}$ | 8.0 | 8.0 |
| $f$ | 0.0 | 0.0 |

Table 2.4: Example calculation of debit $Q_v$, pressure $dP$, pressure limit $P_{\text{limit}}$ and feasibility metric $f$

A solution is feasible if and only if $f \leq 1$ for all greenhouses at all times. The formula for $f$ will henceforth be referred to as the feasibility metric.

## 2.9. DEALING WITH VARIATIONS OF THE HEAT DEMAND DURING THE DAY

As can be seen Figure 2.7 and as will be discussed in detail in Chapter 3, the heat demand varies over the day. There are different solutions to this problem, but there are two requirements that should be taken into account:

- Scenarios should be continuous. For this research the choice has been made to not vary the state of a CHP more than once an hour.
- A CHP can only be turned on twice a day.

**Taking the maximum demand** The easiest solution is to ignore the daily variations and optimize for a single point. Such a single point can be found by taking the peak demand for each individual greenhouse and finding a static solution that produces enough heat for those peaks and not violate any physical constraints. If a stable solution can be found for this extreme case, this solution will work for the complete day. This approach is taken initially in Chapter 4 and it would be a waste or energy to produce the same amount of heat during when at night little heat is needed.

**Optimizing for each time moment separately subject to constraints** The most flexible solution would be to solve the problem separately for each individual moment. There are 104 greenhouses and 288 moments during a day. However the found solution needs to be continuous and satisfy the requirements mentioned above. This can be solved by introducing penalties for any discontinuity, but this makes optimization difficult. If this solution is implemented, varying the value of a heat source requires changing the value of the surrounding time moments too to prevent discontinuities from being introduced. The resulting scenarios will also needs more cooperation between the different actors and each of the actors needs to operate at a different time. It is also needed for all the greenhouses to stick to this plan.

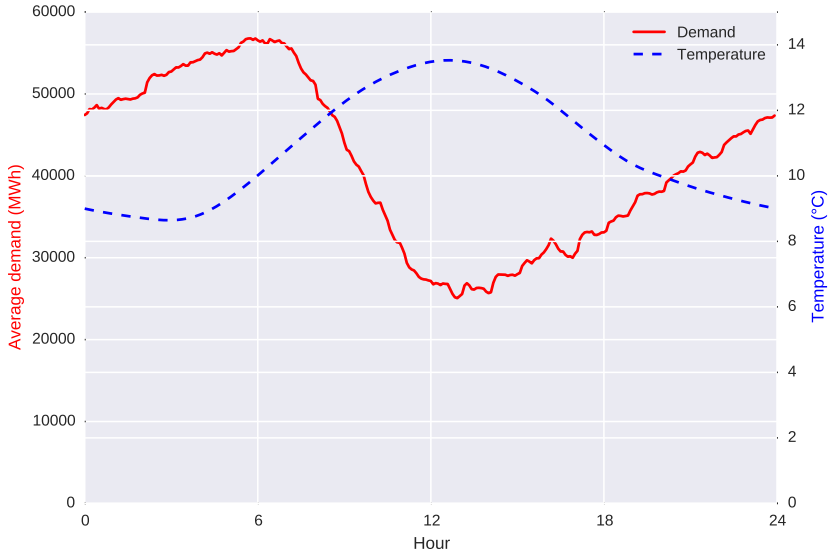



Figure 2.7: The heat demand varies over the day

| Time interval | CHP 1 | CHP 2 | CHP 3 |
|---|---|---|---|
| 1 | on | on | on |
| 2 | on | off | on |
| 3 | on | on | on |
| 4 | on | off | off |

Table 2.5: Example of turning CHP's on and off over four time intervals

**Segmenting the heat demand** Another way to deal with time is by dividing the day up in segments and solving for each segment separately. This solution is followed in Section 4.5. The same segmentation will be used for everyone, so all of the actors will have to operate at roughly the same time. Then for each of these segments, the peak demand is taken for each individual greenhouse and a static solution is found that produces enough heat for those peaks and not violate any physical constraints. Limiting the number of segments to 4 makes sure no CHP is turned on and off more than once a day, because for turning it on thrice at least 5 points of switching are needed. Handling continuity will have to be done, by having short transitions periods between the segments. In the experiments continuity is ignored, but dealing with these transitions is expected to be a simple problem and some of the ways these transitions can be done will be discussed in Section 5.2.3. An example of how a CHP can be turned on and off over multiple time intervals is given in Table 2.5.

## 2.10. OBJECTIVE AND RESEARCH QUESTIONS

The main research question of this thesis is:

> How can stochastic optimization algorithms be used to solve a heat control optimization problem that has a linear objective and black-box constraints for which a simulator exists?

We have a linear objective, which is to minimize the costs which depend linearly on the heat sources that are used. The RoCa is a linear heat source, which means that the costs depend on the state of this source. The CHPs are heat sources that can be on or off, so their production can be considered to be linear as well. The constraints are not linear and require an optimization method that can deal with non-linear constraints.

We have a general understanding of what the simulator is doing, but some of the information is unknown to us, like the positions and topology of the individual greenhouses and the thermic influences on the pressure that is currently not considered. A program has been supplied to us that can be used to evaluate the

physical constraints by calculating the pressure and comparing this to the maximum pressure that is tolerable. Since we have little knowledge about the network itself, we are going to use a stochastic method.

This problem will mainly be applied in the context of the Eneco heat grid, where heat needs to be provided to the greenhouses as cheaply as possible, but without causing physical constraints, while making sure every greenhouse has its demand satisfied.

**Research question 1** *What are the patterns that should be taken into account?*

Since the heat demands of the different days are used as the input for our problem, there are a few patterns that we want to investigate. The heat demand is specified as the need for heat per greenhouse per 5 minutes. This makes us wonder if the heat demand is actually different. If the heat demand remains more or less constant over the day, the heat demand can be averaged and we can solve the problem once for the average. Otherwise, it is needed to somehow adapt the amount of heat that is generated. If the demand remains constant over the year, the problem only needs to be solved once, but if the demand varies greatly from day to day, a single solution does not suffice. Therefore, we want to know if the demand differs over the days. Even if the demand differs over the day, it might be possible to divide the days into temperature groups, therefore the correlation between the heat demand and outdoor temperature is studied. If either the heat demand remains constant or the days can be divided into groups that have a constant temperature, we do need to know if the distribution remains equal. Two days might have a similar total demand, but have a different structure, for example on one day greenhouse A might have a high demand and greenhouse B a low one, whereas on another day greenhouse A has a low demand and greenhouse B a high one. The total demands of these days can be similar, but both problems need different solutions.

This results in the following subquestions:

1. What does the demand over the average day look like? Is there a fixed pattern?
2. What does the demand over the look like over the year?
3. Is there a correlation between temperature and demand?
4. How is the demand spread over the greenhouses?

Subquestion 1 will help us to answer Research Question 3.

**Research question 2** *How can a stochastic optimization algorithm solve a heat control problem if the heat demand does not vary over time?*

Therefore we need to start by choosing an optimization method. This is done in Chapter 4.

We will first see how this program can be solved if the assumption is made that the heat demand does not vary over the day. An easy way to do this is to take the maximum demand for each greenhouse for each time over the day. If we can solve this problem we have a scenario that always works but is probably overproducing for the times of the day when the demand is low.

**Research question 3** *How can a heat control problem be solved if the heat demand varies over time?*

The heat demand is not fixed and would vary over time. Once Research Question 2 has been answered, a way needs to be found to vary the pattern over the day to deal with these variations. This will be done by making use of segmentation to split the day in segments and solving the problems for each of these segments separately. This is done in Section 4.5.

# 3

# ANALYSIS OF DEMAND PATTERNS

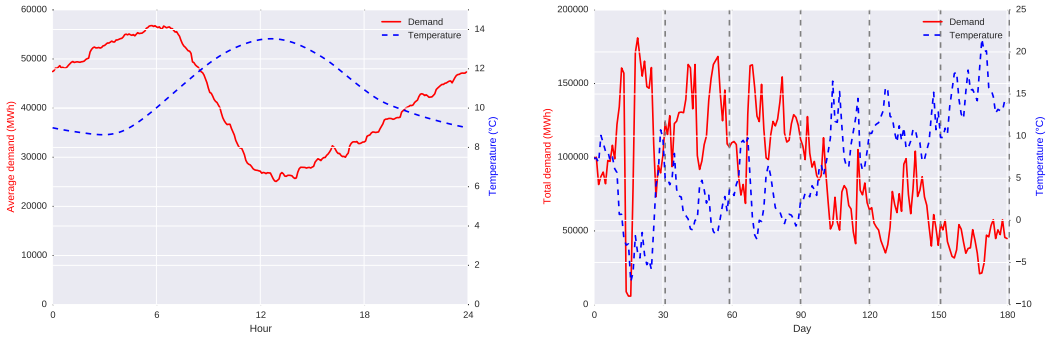
In this chapter, a look is taken at the data to see which patterns need to be taken into account. In Section 3.2 the change of the heat over the day is investigated. This result gives us insight into how best to split the problem in segments, which will be done in Section 4.5. In Section 3.3, the demand between the different days is compared. The result of this investigation has resulted in the decision to run the final solution on at least one summer month and one winter month. In Section 3.4 the correspondence between temperature and demand is investigated and in Section 3.5 the homogeneity of the demand patterns is investigated. This results of these investigations are not used in the Chapter 4, but could be used in future work. For example, the decision can be made to split the demand data sets into groups that are similar in structure and to find optimized solutions for these cases that can be used as a basis for the individual days, for example an optimized solution for winter months.

## 3.1. EXPECTED PATTERNS

As discussed in Chapter 2, the indoor temperature of the greenhouses needs to be kept constant and this indoor temperature is influenced by the outdoor temperature by transmission, infiltration and ventilation. So the more cold gets in, the more heat is needed for compensation. It is expected that the relation between the outdoor temperature and demand is a linear one: We expect that a linear increase of the outdoor temperature, results in a linear increase of the demand.

**Hypothesis 1** *There is a linear correlation between the outdoor temperature and the heat demand.*

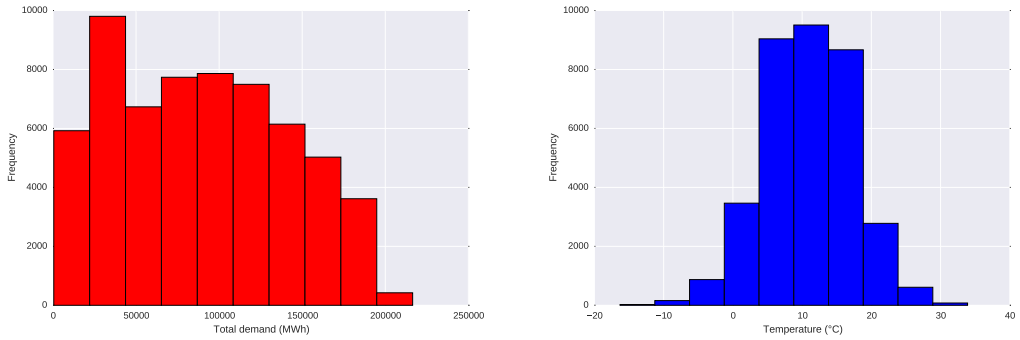
Because of day and night and season patterns, the outdoor temperature in turn depends on the time of day and the time of year. This understanding is depicted in Figure 3.3. During the average day, the lowest outdoor temperature is reached at 3:00 and the highest at 12:00. Therefore it is expected that the demand is




(a) An average day.

(b) January to June

Figure 3.1: Heat demand and outdoor temperature over the day and over the year




(a) Histogram of total demand

(b) Histogram of temperature

Figure 3.2: Histograms of data

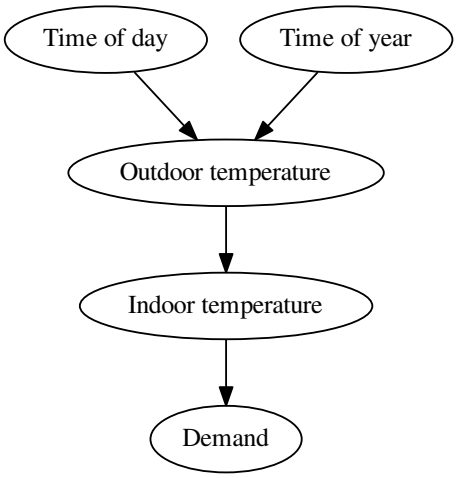



Figure 3.3: Expected dependencies

highest at 3:00 and lowest at 12:00. Our expectations for differences between summer and winter are similar. During the winter it's colder outside, so more heat needs to be produced to keep the temperature in the greenhouse constant and so the expected demand is higher for these months.

**Hypothesis 2** *The time of the day and time of the year influence the heat demand.*

Although we expect the heat demand to vary depending on time, it is expected that some greenhouses will always need more heat than others.

**Hypothesis 3** *Irrespective of time, the heat demand is distributed equally over the greenhouses.*

The implication of these hypotheses combined would be that if two days have similar temperatures and similar spread, that they might be able to share a similar scenario. For the rest of this chapter, we will refer to the outdoor temperature simply as temperature.

To find the temperatures in 2013 near Lansingerland, temperature data has been obtained from the nearest KNMI weather station (number 344) [2], which is located in Zestienhoven in Rotterdam. This contains the average temperature measured during each hour with a resolution of 0.1 °C. These data will be used in our analysis. The temperature and heat demand over the day can be found in Figure 3.1a. The temperature and heat demand over the year can be found in Figure 3.1b. The distribution of the heat demand and temperatures are plotted in Figure 3.2.

## 3.2. DEMAND OVER THE AVERAGE DAY

In this section, a look will be taken to the average day and describe what the demand looks like during a day. Finding such a pattern is useful for deciding how the data should be split in segments which is done in Section 4.5.

The heat demand in $J/h$ over the average day (based on the months January to June) is depicted in Figure 3.1a. As expected, the demand is high during the night but decreases during the afternoon. However, the times at which the peaks occur are different from our expectations; the demand is highest around 6:00 and lowest around 13:00, but these extremes were expected at 3:00 and 12:00 respectively similar to the temperature peaks. The temperature is about twice as high during 6:00 as it is during 12:00. A likely explanation is that a change in outside temperature slowly affects the indoor temperature and thus does not immediately change the demand. In other words, there is a reaction time between the action, a change of outside temperature, and the reaction, a change of inside temperature and thus a change of heat demand. This reaction time is not by a constant offset, which can be explained by the observation that the outdoor temperature increase in the daytime is faster around noon than the outdoor temperature decrease during the night. If buffers were included in our model, they could be used to equalize the heat production. For example, heat can be stored in the buffers when the heat demand is low and released again when the heat demand is high. This would make the problem easier to optimize because it reduces the variation over the day,

## 3.3. DEMAND OVER THE YEAR AVERAGED PER DAY

We have averaged over the day to ignore daily variations and see the variation over the year in Figure 3.1b. As can be seen, there is a valley from 15 to 18 January. It is expected that during this period something special might have happened like maintenance or that something went wrong with the measurement instruments. This period will be ignored for the experiments and for the rest of this analysis. As expected, the colder months like January have a higher heat demand than a summer month like June, although we have only looked at the year 2013. The graphs in Figure 3.1b show that an increase in temperature corresponds to a decrease in demand. Generally, temperature peaks correspond to heat demand valleys and vice versa. The correlation between temperature and demand is investigated in Section 3.4.

Per month, the energy demand over each day is depicted in Figure 4.10. The intuition that a winter month requires less energy seems to fit this data. Another thing that can be noticed is that during January and June the heat demand during the day is less varied than during March or April.

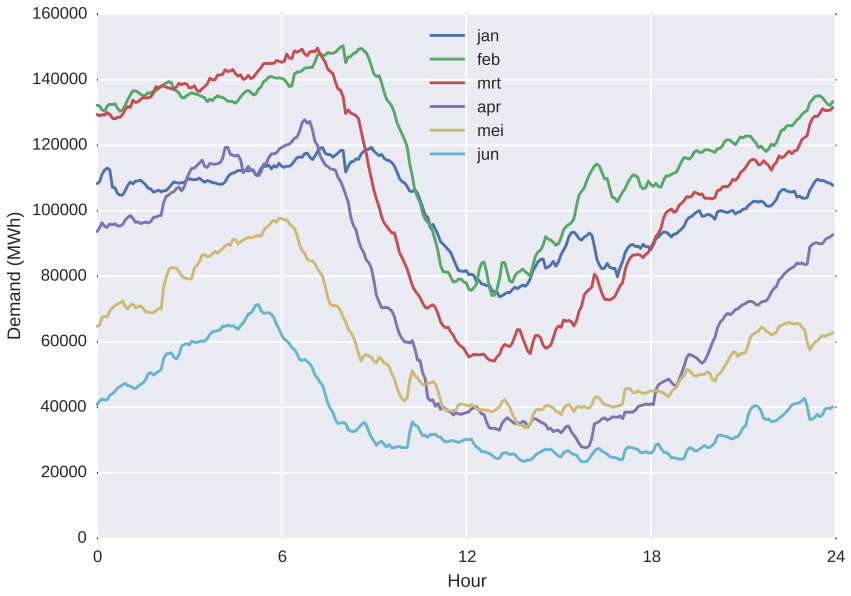



Figure 3.4: Average energy per day in each month

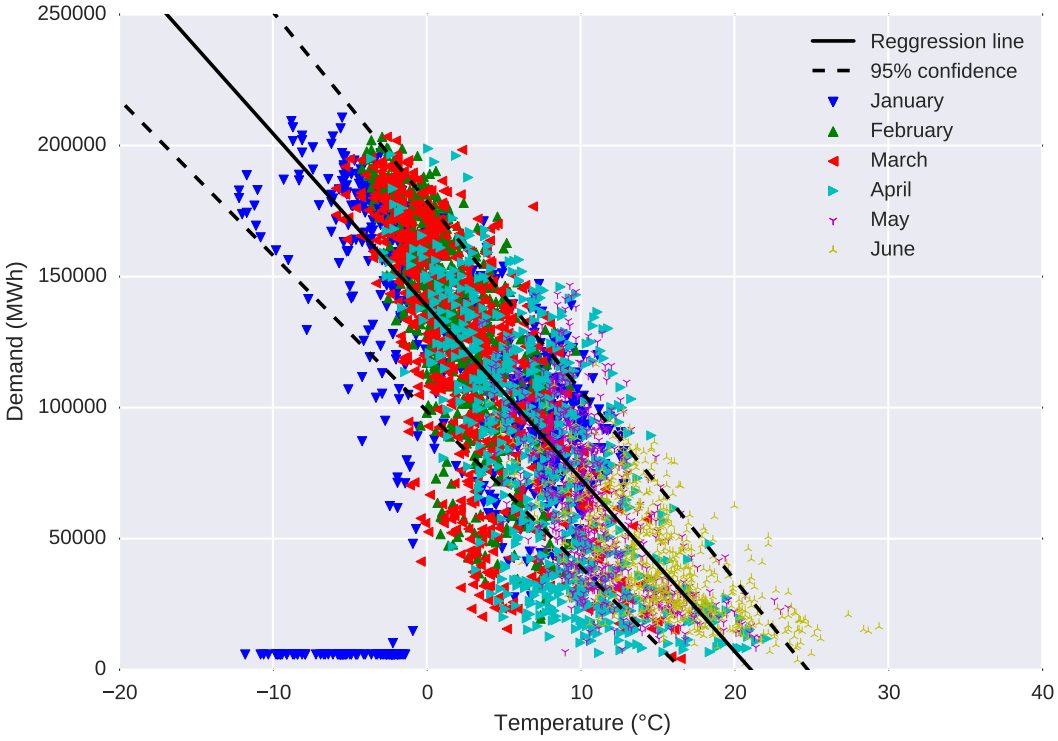



Figure 3.5: Correlation between demand and temperature with 95 % confidence interval.

## 3.4. CORRELATION BETWEEN THE TEMPERATURE AND HEAT DEMAND

The temperatures of each time of each day of each month have been plotted in Figure 3.5. From the figure it appears that there is a strong linear correlation between temperature and heat demand. The graph also shows the anomaly in January that was also observed in Figure 3.1b.

Testing the linear correlation is a bit tricky. Because we are dealing with a time series, the data are not independent. From Figure 3.1a and Figure 3.1b, two dependencies are known to us. Consecutive days tend to be similar and the same time moment on different days tends to be similar. If we take the temperature and the demand of two consecutive moments on a specific day, they would be similar because of continuity, but that doesn't mean they are correlated. It's not possible to achieve complete independence because all the data are from the year 2013. Because of this any test for correlation is likely to overestimate the strength of the correlation. If we test the statistical significance of the data as is, the Pearson coefficient turns out to be $-0.72$ with a tiny p-value for non-linear correlation. This correlation means that days of similar temperature, have a similar total heat demand. It does not mean the data sets are similar because the distribution of the demand might be different. This will be explored in Section 3.5.

## 3.5. SPREAD OF HEAT DEMAND OVER THE GREENHOUSES

In Figure 3.7 the heat demand per greenhouse is shown during each month. Visually some recurring patterns can be seen. For instance, greenhouses 116, 125, 139 and 147 are always peaking, whereas there is usually little demand from greenhouse 64. It is expected that the distributions are more or less the same if they are scaled. If this is so, the normal distribution of heat demand is shown in Figure 3.6.

To test that these distributions are actually similar, Pearson's chi-squared test for homogeneity [6] will be used. A contingency table $d_t(g)$ is created that has for each 5 minutes from January to June the demand for greenhouse $g$. Then we want to test if for different values of $t$, the distributions of $d_t(g)$ are similar.

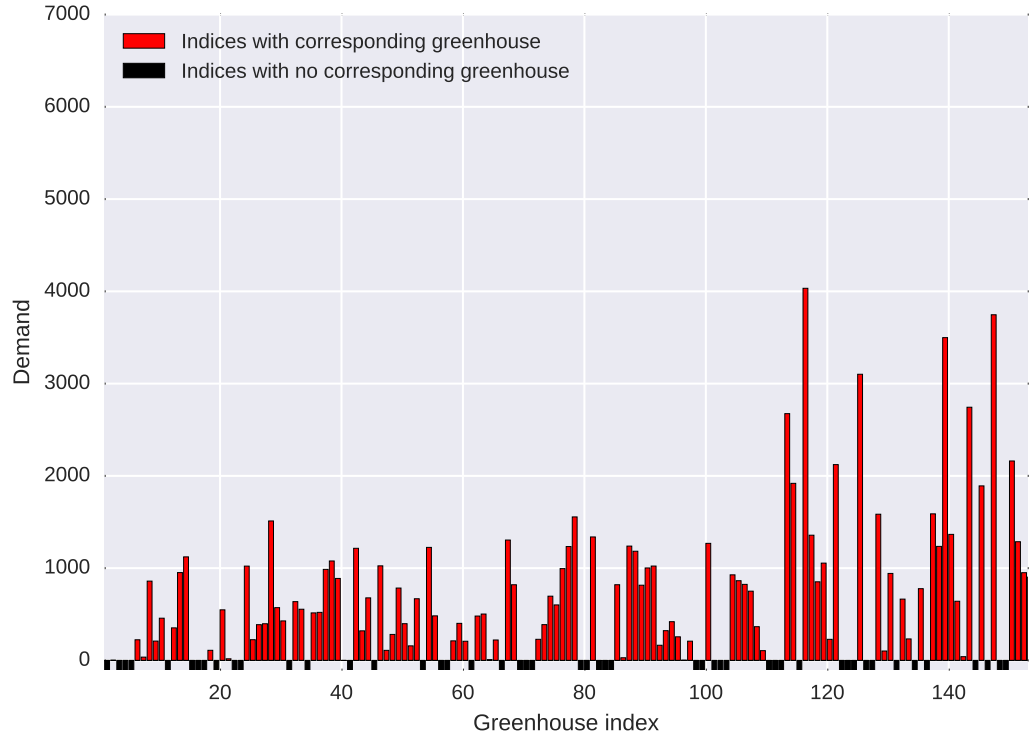



Figure 3.6: Total average distribution

First, the distribution of heat demand is scaled such that its values sum to 1.

$$D_t(g) = \frac{d_t(g)}{\sum_{h=1}^{T} d_t(h)} \tag{3.1}$$

Under the assumption that the distribution is the same for every moment the expected distribution of heat demand at a time $t$ can empirically be estimated to be:

$$E[D_t(g)] = \frac{\sum_{u=1}^{52122} D_u(g)}{52122} \text{for different } t$$

Then the chi-squared coefficient is calculated the following way:

$$\chi^2 = \sum_{t=1}^{52122} \sum_{g=1}^{104} \frac{\left(D_t(g) - E[D_t(g)]\right)^2}{E[D_t(g)]} = 97389 \tag{3.2}$$

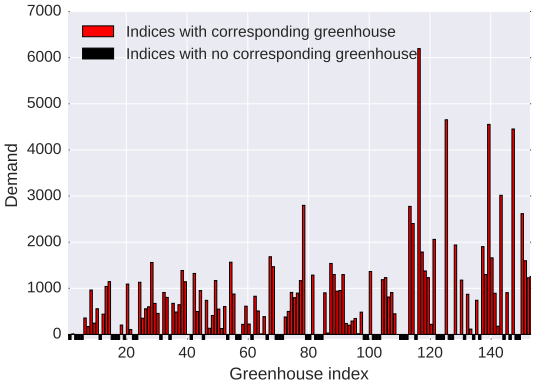
For the chi-squared test of homogeneity, the degrees of freedom are the number of greenhouses minus 1 times the number of moments minus 1, so

$$k = (104 - 1) \cdot (52122 - 1) = 5368463 \tag{3.3}$$

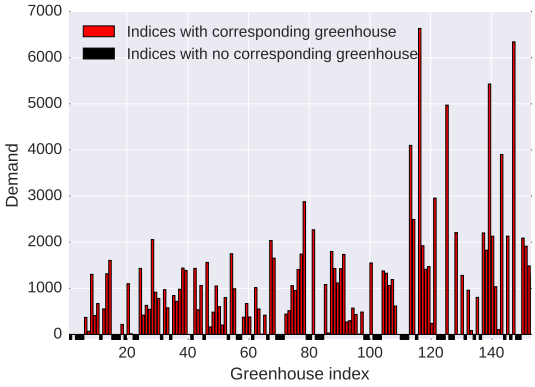
The p-value of the test turns out to be close to one, which implies that with high probability the distributions are correlated.

## 3.6. CONCLUSION ON DEMAND PATTERNS

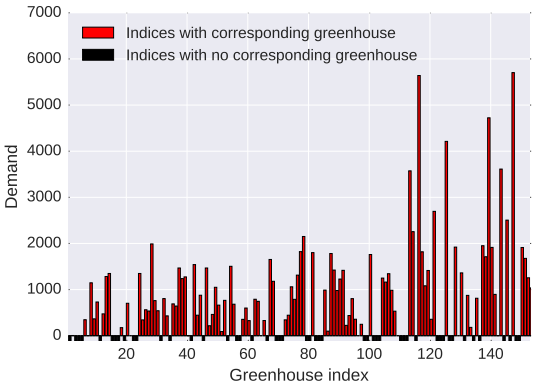
In conclusion, it can be said that a linear correlation between the temperature and the demand is probable, although we can't be sure if this correlation still holds if the dependency on time is removed. This manifests itself both in a regular demand pattern over the day as a regular demand pattern over the year. During the



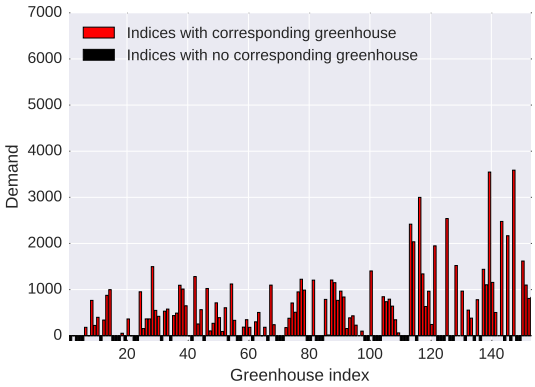

(a) January



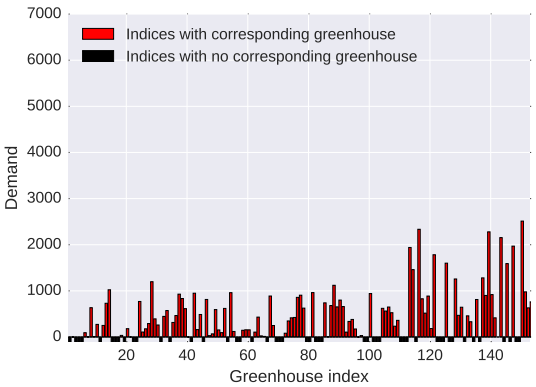

(b) February



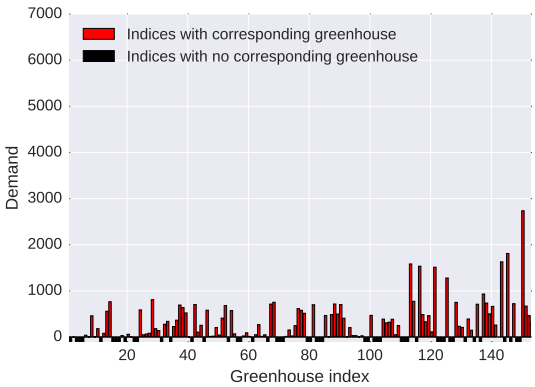

(c) March




(d) April




(e) May




(f) June

Figure 3.7: Average heat demand per greenhouse per month

day the heat demand is lower than during the cold night and during the summer the demand is lower than during the winter. During a day a change in outside temperature does not immediately affect the demand, which is most likely the result of heat transmission taking place slowly. And it can be observed that during spring (from June 21 to September 22) the heat demand varies more over the day than during the summer or the winter. Furthermore, we observed that the spread is more or less the same for every day, so we think that the demand will mainly depend on the outside temperature.

# 4

# SOLUTION WITH SIMULATED ANNEALING

In this chapter an optimization technique is described to solve the heat supply problem. The problem will first be solved without looking at time (thus we will solve the problem under the assumption that the demand doesn't vary over time), which will answer Research Question 2. In Section 4.5 we will consider how to incorporate time and have the demand vary over the day. This will answer Research Question 3.

We start by comparing different optimization techniques from literature in Section 4.1. The choice has been made to use simulated annealing and we will describe some of the high-level design decisions in Section 4.2. This optimization technique requires an initial scenario to be chosen, which is done in Section 4.3 and a mutation which is done in Section 4.4.

## 4.1. CHOICE OF OPTIMIZATION TECHNIQUE

As will be shown in Subsection 4.4.1, the problem of this thesis has multiple local optima, due to the physical constraints that can occur in multiple places and the many degrees of freedom to control the scenario (One RoCa, several CHPs and several boilers can be varied). When there is a shortage of heat at a particular greenhouse, the algorithm could increase the usage of a particular heat source to resolve it, but perhaps it can find a more economical solution by using a different heat source or a combination of heat sources instead. Therefore, using a method that can avoid getting stuck in local optima is desirable.

There are several methods that can deal with global minima. The most primitive one is random search, that tries many points to find the lowest, but this method requires a lot of sampling and doesn't guarantee that the solution found is a local minimum. Some other methods are multi-start local optimization, simulated annealing and genetic algorithms [7].

### 4.1.1. MULTI-START LOCAL OPTIMIZATION

Local start optimization is a widely used method, which follows the same process as local optimization, but repeats this process from different starting points. It has the advantage over random sampling that its solutions converge to a local minimum. The disadvantage is that it needs to be run multiple times and therefore requires many evaluations.

The number of evaluations could be reduced, if a good starting point or a region of good starting points can be found. Although multi-start local optimization will not be used for solving the heat generation problem, the problem of finding a good starting point, will be explored in Section 4.3.

If multi-start local optimization were used for our problem, two local optimization methods that could be used are Powell's perpendicular method [8] or the Nelder-Mead method [9]. The Levenberg-Marquardt and Newton algorithm for local optimization converge faster[7], but need the gradient and Hessian to be computed of our evaluation function, including the Drukval model. For the Drukval model, these functions are not known. They can be estimated by multiple evaluations of the Drukval model, but this is computationally expensive.

### 4.1.2. SIMULATED ANNEALING

Simulated annealing [10]is based on the process of physical annealing in metallurgy. Like downhill search the method iteratively tries to finds solutions that are closer to the optimum, but unlike downhill search the

solution will sometimes accept slightly worse solutions. During the early stages of simulated annealing the algorithm is free to accept slightly worse solutions, but during the later stages the algorithm becomes similar to a downhill search and will only accept better solutions. This transformation takes gradually place as the temperature drops.

Simulated annealing starts by choosing a random solution $s$. On each iteration, a mutation $s'$ is proposed produced by a mutation function. If $s'$ is better than the original solution $s$ according to evaluation function $v$, it replaces the original. Otherwise, the new solution is accepted with probability $p(s, s', i)$ which depends on the difference in quality between the current solution and the mutated one and on how for how many iterations the algorithm has been running: During the first few iterations of the algorithm, the algorithm is flexible and the probability of accepting a slightly worse solution is high. This makes it possible for the algorithm to escape from local minima and to explore the domain. Later on the probability of accepting a worse solution decreases and the algorithm starts converging. Eventually the solution that has had the highest score (minimum costs while satisfying constraints) throughout the algorithm is returned. Pseudocode is shown in Algorithm 1. In this code, $s$ is the solution that is being worked with, $s'$ is a proposed improvement and $s^*$ is the best solution encountered so far.

**Algorithm 1** Simulated annealing

$s$ is the current solution.
$s'$ is a proposed mutation.
$s^*$ is the best solution encountered so far.
$m(s)$ is the mutation function
$v(s)$ is the evaluation function
$p(s, s')$ is the acceptance function for accepting $s'$

```
s ← s* = initial scenario
for i ← 1...n do
    s' ← m(s)
    if v(s') < v(s) or p(s, s', i) > random(0...1) then    ▷ New state better than old state or small probability
        s ← s'
    end if
    if v(s') < v(s*) then                                  ▷ New state is best state seen so far
        s* ← s'
    end if
end for
return s*
```

Compared to downhill search, simulated annealing is able to escape from local minima so it can explore the domain before converging to a single solution. Another advantage of simulated annealing is that it is flexible, because it can work with any objective and constraints even when little domain knowledge is available (we don't know the topology of the network or the thermic influences on the pressure), whereas gradient descent requires the gradient of the function to be known or to be estimated. typography, we don't know

The disadvantage of simulated annealing is that the occasional step to accept a worse solution (an uphill move) makes the process converge slowlier, but in return for that it is less likely to get stuck in local minima. Another disadvantage is that although it is statistically likely that a good solution is found, there is no performance guarantee for the final solution other than that it is at least as good as the initial one. Finally, this method requires many choices to be made such as the decay rate of the temperature function, the number of iterations, and the mutation rate.

### 4.1.3. GENETIC ALGORITHMS

Another class of stochastic algorithms are genetic algorithms [11]. A genetic algorithm starts from a pool of random solutions. Each of the solutions in the pool is evaluated by an evaluation function $v(x)$, which is an indication of how good the solution is. Constraints are handled by including a penalty function (positive or negative) to punish bad solutions.

Every iteration, a new generation is formed by repetitively selecting 2 solutions from the old generation. Each solution is chosen with probability:

$$p(x_i) = \frac{v(x_i)}{\sum_{j=1}^{n} v(x_j)} \tag{4.1}$$

These solutions are combined by a crossover function $f(x_i, x_j)$, which combines two existing solutions to construct a new one, and a mutation function, $m(x_i)$, which makes a small change to a solution $x_i$ to create a solution $x_i'$.

**Algorithm 2** Genetic algorithm

$N$ is the size of the pool
$T$ is the numer of iterations
$P = \{P_1 \dots P_n\}$ is the pool of solutions
$v(x)$ is the evaluation function
$c(x, y)$ is the cross-over function
$m(x)$ is the mutation function

**for** $n \leftarrow 1 \dots N$ **do**
  $P_n$ = random initial solution
**end for**

**for** $t = 1 \dots T$ **do**
  **for** $n \leftarrow 1 \dots N$ **do** ▷ Create distribution $p$ from the scores of the solution pool.
    $p_n \leftarrow \frac{v(P_n)}{\sum_m v(P_m)}$
  **end for**

  **for** $n \leftarrow 1 \dots N$ **do** ▷ Create a new pool
    Let $P_i$ and $P_j$ be randomly selected from $P$ according to distribution $p$
    $c_n \leftarrow c(P_i, P_j)$ ▷ Apply crossover function on solutions to create a new one
    $m_n \leftarrow m(c_n)$ ▷ Apply mutation on new solution
    $P'_n \leftarrow m_n$ ▷ Store solution in new generation
  **end for**

  $P = P'$
**end for**
**return** best solution currently in $P$

In the context of our problem, a mutation function that operates on solutions $x_i$ and $x_j$ can iterate through all heat sources. With a probability of 50% the state of solution $x_i$ is copied and with a probability of 50% the state of the second solution is taken. The mutation function can take a random CHP and randomly turn it on or off or change how much it produces.

Like simulated annealing, genetic algorithms work well with local minima and large sets of parameters. The disadvantage of genetic algorithms is that it needs many iterations and requires many evaluations per round. Like simulated annealing, it also requires tuning a lot of parameters and has no performance guarantee other than that the final solution is statistically likely to be at least as good as the initial solution and that the solution generally improves during each step.

## 4.2. OVERVIEW OF DESIGN DECISIONS

A flow diagram of the whole program is shown in Figure 4.1. Solutions are generated by a mutator and have its costs and physical constraints evaluated on each iteration. The best solution encountered during the process is remembered and used as the final answer.

**Initial solution** Simulated annealing needs to start from an initial scenario before it starts mutating. It can be random, but it is expected to be better to start from a scenario that is already good, so that subsequent mutations can spend more time on improving the current solution than on finding a lead to a good solution. Three different initial solutions will be introduced and compared against each other in Section 4.3.

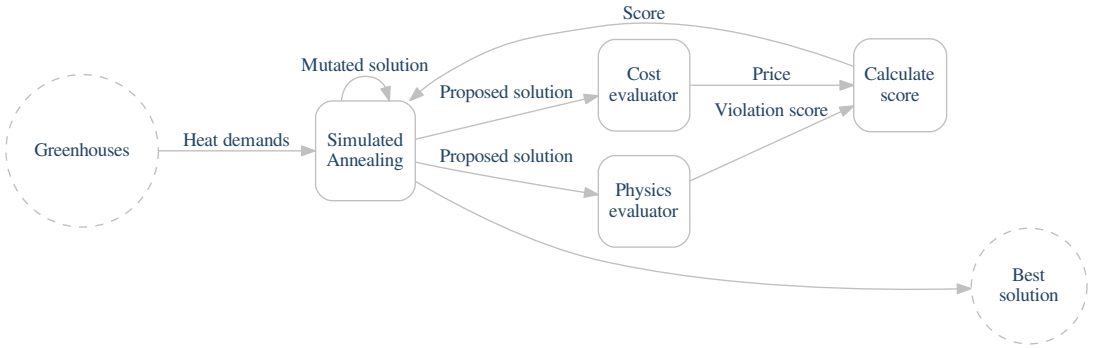



Figure 4.1: Program structure. The input are the heat demands provided by the greenhouses. The output is a best solution which takes the form of the heat production per heat source for every moment of the day (every 5 minutes).

As discussed in Chapter 2, the CHPs deliver a fixed amount $m^3/h$ to the network when they are in use. The maximum amount of heat that can be exported for a certain debit is calculated using formula 2.6. Furthermore, the assumption is made that the greenhouses with CHPs are self-sufficient when they produce (the production is their own demand plus the demand for the debit they are delivering to the network). Details about the generation of the initial solution are discussed in Section 4.3.

**Mutation function** The mutation function will be used to propose new scenarios. It can be random, but better results might be achieved if a mutation function is used that tries to turn infeasible scenarios into feasible ones and expensive feasible scenarios into ones scenarios that are cheaper but still feasible. Three different mutation methods will be introduced and compared to each other in Section 4.4.

**Evaluation function** In Chapter 2, we have described how the costs and constraints can be calculated. An overview of the procedure can be found in Figure 2.6. For the final result, a result is desired that has no constraints and low costs. As has been explained in Section 2.5, a constraint occurs when the pressure in a pipe is higher than allowed. This will be tested by calculating the debits and running the Drukval program to test the pressures that correspond to these debits. For the evaluation function, we want the costs to be as low as possible while not having any constraints. To do this, we make use of a penalty function[12] that depends on how much violations we have and how big these violations are.

Initially it was tried to make a penalty function that simply counts the number of violations, without regard to how big they are. So for each valve $i$, a variable $P_i$ was introduced such that $P_i = 1$ when a violation occurred in valve $i$ and $P_i = 0$ when no violation occurred. Because having a solution free of violations is more important than having an economical solution, the violations were weighted with a big constant $C$ to make the algorithm prioritize feasible solutions over ones that are economical The evaluation function was:

$$v = \text{total costs} + C \cdot \sum P_i \tag{4.2}$$

This has been tried in an experiment setup with a random day of every month (6 months) and been run for 1000 iterations. The mutation step was a random mutation, which will be introduced in Section 4.4. An initial solution was chosen that would enable 50% of all CHP's, but the demands of the individual greenhouses were inflated to 500% of their normal values to introduce a lot of overproduction. The result showed that although the costs improved, no progress was made in getting rid of the violations.

As an alternative, a soft penalty function was tried that depends on how much the violation is. A constant of 1 is still added to make sure constraints are solved completely instead of letting small violations pass. The shape of this formula depends on the extent to which dP exceeds $dP_{limit}$, which it should not do according to Equation (2.11). The new penalty function still has values in the range between 0 and 1.

$$P_i = \begin{cases} 0.5 \cdot \left(1 + \frac{dP - dP_{limit}}{dP}\right) & \text{if } dP > dP_{limit} \\ 0 & \text{else (no violation)} \end{cases} \tag{4.3}$$

Crashes in Drukval are resolved by setting the penalty to infinite, a so called 'death penalty'. This is needed because mutations of a scenario that makes Drukval crash tend to make Drukval crash too. Giving such scenarios, a penalty of infinity makes sure that it is never selected, because it is not an improvement and because putting infinity in Equation 4.6 (that will be discussed in paragraph Probability function) makes the transition probability 0.

For example, if the pressure at some point in the valves is 700 bar, but the maximum pressure allowed is 500 bar, the penalty becomes:

$$0.5 \cdot \left(1 + \frac{700 - 500}{500}\right) = 0.7 \tag{4.4}$$

Different values for C can be chosen. One option is to choose $C >>$ total costs; which guarantees that avoiding the constraints is prioritized over cost efficiency. On the other hand, choosing lower values for $C$ might make it easier for the algorithm to escape from local minima to find cost optimal solutions while temporarily tolerating some constraints violations, as long as the final solution is feasible. Values of $1 \times 10^5$, $1 \times 10^6$ and $1 \times 10^7$ have been tried on 3 different scenarios with a random mutation each running for 100 iterations and $1 \times 10^6$ gave the best results.

**Probability function** When the mutated solution is worse than the current one, it is accepted with a probability $P(s, s', i)$ This formula is described by Goffe e.a. [13]. The smaller the difference between the current solution and the mutation, the higher the probability that the mutation is accepted. Solutions are also more likely to be accepted at the beginning when the temperature is still high. Later on when the temperature drops, the probability of accepting a worse solution drops as well.

Goffe [13] used the following:

$$P(s, s', i) = e^{(v(s') - v(s))/T} \tag{4.5}$$

This formula has been tested on a summer and a winter day. My experience with this formula is that it worked well when trying to get rid of constraints, but less so in trying to reduce the costs. It was my intuition that the penalty function had a large impact compared to the costs, which makes it hard to balance the parameters of the temperature function. Therefore, it was decided to change the probability formula to measure the improvement relative to the old value.

$$P(s, s', i) = e^{(v(s') - v(s))/(T \cdot v(s))} \tag{4.6}$$

There is a trade-off between different temperature functions. Different temperature functions can be chosen, but they should always have a value between 0 and 1. If the temperature decreases quickly (fast cooling), the algorithm will converge faster and be less explorative. If the temperature decreases slowly, it will explore more solutions but converge at a slower rate.

One temperature function used by Goffe e.a. [13] that is expected to converge quickly is

$$T_a(t) = T_0 \cdot r_t^t \tag{4.7}$$

For $T_0$, no parameters other than $T_0 = 1$ have been tried. In hindsight, a lower value may have been better, because it will speed up the initial exploration phase. Because evaluations take long, I wanted the process to converge in at most 1000 iterations at most. Experiments have been done with $a = 0.5$, $a = 0.95$ and $a = 0.995$. Most processes converged in about 300 iterations when $a = 0.95$, so this value had been chosen.

Another temperature function has been proposed by Szu and Hartley [14]

$$T_a(t) = \frac{T_0}{1 + t} \tag{4.8}$$

It has not been tried, because the evaluation of a single scenario is slow and this formula is expected to converge more slowly than Goffe's requiring more evaluations.

## 4.3. INITIAL SCENARIO

Simulated annealing starts with an initial solution, explores the solution space to find a basin as deep as possible and eventually converges to the local minimum in that basin. It is assumed that finding an already good solution as a starting point will work better than starting from a random point. Of course this depends on what mutations are used, because if the mutations were directed in a certain direction, it would be wise

to position the initial position at some position such that the mutations are most likely to bring the solution towards the global minimum. For example, if the mutations are such that they create a more decentralized scenario, it might be better to start from a scenario that is centralized than one that is already decentralized. Ideally, the mutations should be able to find the right direction from any initial situation, in which case a scenario close to the optimum would mean that fewer mutations are necessary.

A minimum guarantee that an initial solution will bring is that the eventual solution will at least be as good as that initial solution, because there is always the possibility to fall back on it. Preferably, we even want to find the best basin to converge in. If we are able to start in the best basin, even a downhill search can be used instead. A secondary goal of this research is that it provides us insight into what kind of solutions work well in general.

### 4.3.1. PROPOSED METHODS

**Random approach** This will generate a random solution in the feasible domain. A solution is generated by considering each greenhouse. If the greenhouse has a boiler, then there is 50% probability that the greenhouse will produce heat using that boiler and 50% that the greenhouse won't. As mentioned in Paragraph 2.2, the CHPs produce a fixed contractual debit. The amount of heat produced will be within the minimum and maximum production values allowed. If the greenhouse doesn't have a boiler or doesn't use its boiler, it will subtract the amount it needs. To ensure there is enough heat, all heat consumed but not produced will be supplemented by a RoCa and all heat required but not consumed by a greenhouse is supplemented by boilers. If as much or more heat is produced than consumed,the RoCa will not be used. The guarantees of this solution are that at least the demand is fulfilled. It does not guarantee a cheap solution or a solution that is physically feasible.

- Greenhouses that don't have a CHP will always be consuming.
- Greenhouses that have a CHP have a 50% chance that they produce heat. If they produce, they will produce for their own demand and deliver a fixed amount to the network; otherwise they will consume from the network.
- The RoCa will be used to deliver all heat that is consumed from the network but not delivered to the network by any of the greenhouses.

In the example in Section 4.3.1, the random scenario solves the heat production problem partly by using the RoCa and partly by making greenhouses A and D produce. Greenhouse A produces for its own consumption and partly for the demand of B and C. Greenhouse D produces for its own consumption only and doesn't export to the network.

**Centralized production** This reflects the old situation in the network. The total heat demand is calculated and everything is produced by a RoCa. The greenhouses will each extract the exact amount of heat that is required from the network. No CHPs nor boilers will be used. Because this solution reflects the old situation, it has the advantage that the solutions it creates are feasible. Due to its heavy reliance on the RoCa, it might be an expensive solution.

- The CHPs are not used.
- The RoCa produces enough for everyone.

In the example in Section 4.3.1, the centralized scenario solves the heat production problem by making the Roca produce for all other greenhouses. The other greenhouses consume from the network and don't produce anything themselves.

**Greedy CHP production** This initializes the heat production scenario such that as few sources are used as possible. The greenhouses that have a CHP are sorted by the capacity of their CHPs. While the demand is unsatisfied, the greenhouse with the most productive CHP is selected to use its full production capacity to produce heat until everyone's demand is satisfied. For small greenhouses with a CHP, it might happen that another greenhouse with a bigger CHP fulfils their demand.

- The greenhouses with CHPs are ordered by their production capacity.

| Actor | Demand | Centralized | | Greedy | | Random | |
|---|---|---|---|---|---|---|---|
| | | Production | Ex/import | Production | Ex/import | Production | Ex/import |
| RoCa | – | 2000 | 2000 | 0 | 0 | 500 | 500 |
| A | 500 | 0 | -500 | 1000 | 500 | 1000 | 500 |
| B | 500 | 0 | -500 | 1000 | 500 | 0 | -500 |
| C | 500 | 0 | -500 | 0 | -500 | 0 | -500 |
| D | 500 | 0 | -500 | 0 | -500 | 500 | 0 |

Table 4.1: Example initial scenarios for the different initial methods. A positive value for export/import means heat is transported from the greenhouse to the network. A negative value means heat is transported from the network to the greenhouse. The value of export can be lower than the value for production if the greenhouse is producing for its own consumption.

- The greenhouse with the largest CHP is selected as a producer until the total demand is satisfied.
- The remaining greenhouses will consume heat.
- The RoCa is not used.

In the example in Section 4.3.1, the greedy scenario solves the heat production problem by by making greenhouses A and B produce enough for themselves and for greenhouses C and D. The RoCa isn't needed here and the production is completely decentralized.

### 4.3.2. HYPOTHESES ABOUT INITIAL SCENARIOS

The randomized program is deemed to be too simple. It doesn't take the heat demand into account and is therefore not expected to work well.

**Hypothesis 4** *The centralized and greedy scenario work better than the random one.*

It is expected that the resulting scenario produces much more than is needed. Because a centralized scenario reflects the original scenario, this one should always be feasible.

**Hypothesis 5** *A centralized scenario is always feasible.*

The greedy scenario seems like a cheap solution, because it heavily utilizes the CHPs which are cheaper than the RoCa and only produces what is needed. It might however end up with a scenario that isn't feasible.

**Hypothesis 6** *The greedy scenario creates cheap scenario regardless of their feasibility.*

Because in Figure 3.1a the maximum demand on the average day is about twice as high as the minimum demand and because our initial generation methods take make enough for the point of the day where the demand is highest, it is expected that the surplus for centralized is not more than a 100 %.

**Hypothesis 7** *The overproduction for centralized is at most 100%. And within the ratio between the maximum and minimum peaks of the demand.*

### 4.3.3. EXPERIMENTAL SET-UP FOR TESTING INITIAL SCENARIOS

For evaluation we will look at the quality of the scenario produced without doing simulated annealing. For each scenario the following metrics are stored:

**Stability**

$$\text{stability} = \frac{\text{number of scenarios on which Drukval didn't crash}}{\text{number of scenarios tested}} \tag{4.9}$$

How often Drukval doesn't hang on a scenario. In case Drukval does hang on a scenario, the scenario is unusable and can not be used for further optimization. This will be measured as the percentage of problems on which a method doesn't crash.

**Costs** The costs of feasible solutions. The lower the better. Only feasible solutions are considered for this metric.

| Method | Stability |
|---|---|
| Centralized | 100 % |
| Greedy | 100 % |
| Random | 99.37 % |

Table 4.2: Stability of initial scenarios

**Surplus percentage**

$$\text{surplus} = 100\% \cdot \frac{\text{heat generated} - \text{heat required}}{\text{heat required}} \tag{4.10}$$

The percentage of heat produced more than the amount of heat that gets actually used.

**Feasibility metric**

$$f = \frac{dP}{dP_{\max} \cdot \left(1 - \left(\frac{Qv}{Qv_{\max}}\right)^2\right)} \tag{4.11}$$

The feasibility metric has been defined in Equation (2.27). A solution is feasible as long as its value is between 0 and 1. A feasibility of 0.1 is not necessarily better than a value of 0.9 because both values are feasible as long as the value is below 1.

### 4.3.4. RESULTING INITIAL SCENARIOS

The initial scenario generators have been tested for every day between 1 January and 30 June (180 days).

For each initial scenario generation method, a summer day and a winter day have been plotted in Figure 4.2 using the metrics from Subsection 4.3.3.

An example of the generated initial scenarios for a summer and a winter can be found in Figure 4.2. The debit graph shows the debit $m^3/h$ that is transported to or from the greenhouse. A negative debit for a greenhouse means that it is delivering to the network; a positive value means that the greenhouse has no CHP or that is not active and that the greenhouse is consuming from the network. In the centralized scenario in Figure 4.2a all debits are positive, because the greenhouses are receiving from but not delivering to the network. In the randomized and greedy scenario, the greenhouses are delivering heat to and receiving heat from the network.

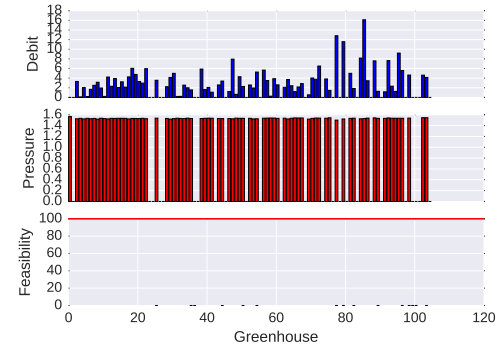
The pressure graph contains the pressure. As discussed in Chapter 2, these pressures are a result of the debits. High pressures are usually a result of high debits from greenhouse to network or from network to greenhouse. The centralized scenario manages to keep the pressure in the network low to a level of 1.6 bar. The greedy and centralized scenarios have higher pressures reaching 6 bar.

The feasibility graph contains the feasibility metric, which depends on the pressure and the debit at the node of an individual (< 100 means feasible). The graphs are normalized so that a value above 100 indicates a violation at some point in the network. For the winter day, a greedy scenario has been picked where a physical violation occurs at the 45th greenhouse. The other scenarios are free of violations.

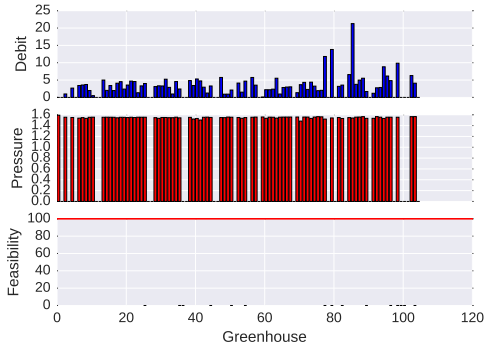
**Stability** Unfortunately, the Drukval program would hang on some of the generated scenarios, but most of them work. This is likely to be a Drukval bug. The scenarios on which this bug occurs are ignored. In my final experiment the centralized and greedy one always succeeded. The random one failed 1% of the time. Future work on the Drukval program might make the scenarios stable at all times.

**Costs** The costs are listed in Table 4.3. As can be observed, the random approach is cheaper than the centralized one. The random method has an average cost of €548.85, but the centralized method has an average cost of €614. The greedy approach is the most expensive one and has an average cost of €978.58.

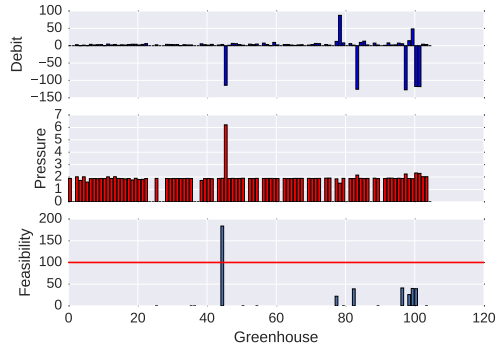
**Surplus percentage** The surplus in production is listed in Table 4.4. The greedy approach performs poorly (490% overload on average) and has much more overload than the random and centralized approaches. The random and centralized approaches (187% and 215% respectively) are close to each other, but the centralized approach has more overproduction than the random one on average. Even in their minimum cases,



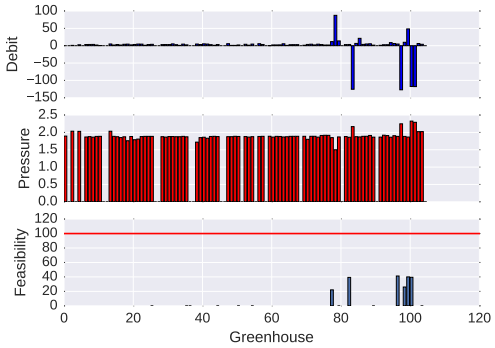

(a) Centralized scenario for winter day



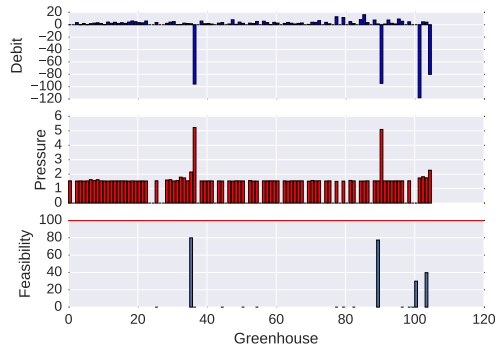

(b) Centralized scenario for summer day



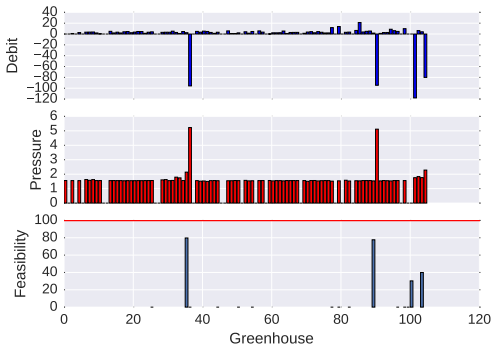

(c) Greedy scenario for winter day




(d) Greedy scenario for summer day




(e) Random scenario for winter day




(f) Random scenario for summer day

Figure 4.2: Initial scenarios with the debits, the resulting pressures and the resulting feasibility scores. The debit $m^3/h$ shows the quantity of heat transported to (positive value) or from (negative value) the greenhouse. The pressures are a result of the debits. The feasibility violation depends on the pressure and the debit. A value above 100 indicates a violation at some point in the network.

| Method | Min | Max | Mean | Std dev |
|---|---|---|---|---|
| Centralized | 332.85 | 729.73 | 589.96 | 94.90 |
| Greedy | 820.22 | 984.20 | 902.06 | 39.71 |
| Random | 287.82 | 660.45 | 524.60 | 89.27 |

Table 4.3: Costs of the heat production of a single day in Euros

| Method | Min | Max | Mean | Std dev |
|---|---|---|---|---|
| Centralized | 55.41 | 1637.28 | 215.92 | 155.47 |
| Greedy | 142.05 | 4591.93 | 490.56 | 443.43 |
| Random | 45.48 | 583.39 | 187.00 | 99.49 |

Table 4.4: Surplus (= 100% · (generated – required)/required) as a percentage

| Method | Min | Max | Mean | Std dev | Percentage feasibile |
|---|---|---|---|---|---|
| Centralized | 0.00 | 0.00 | 0.00 | 0.00 | 100 |
| Greedy | 0.41 | 1.86 | 1.15 | 0.72 | 48.73 |
| Random | 0.80 | 0.81 | 0.80 | 0.00 | 100 |

Table 4.5: Feasibility metric. The feasibility region is from $0 \leq f \leq 1$. Percentage feasible is how often the solution lies into this region.

all the approaches still show more than 45% of overproduction. It is expected this will reduce when adding mutations and turning the schedule in one for a complete day.

**Feasibility metric** The feasibility metric has been explained in Equation 2.27. A value below 1 means feasible. A value above 1, means infeasible. The centralized and random approaches were constantly feasible, because their feasibility scores are always lower than 1. The greedy one ended up in the feasible region only half the time, so more work will need to be done by the mutation to solve these constraints.

### 4.3.5. CONCLUSION ON INITIAL SCENARIOS

All the generated scenarios are still producing more than is required and it is expected that this is because these scenarios generate enough for the whole day don't take the fluctuations into account. The greedy scenario performs worse than was expected. The distribution of the surplus percentage isn't normally distributed, but has a positive skew and the effectiveness of this approach varies a lot. One reason the surplus is so high, is probably because the algorithm works for the whole day, but bases its scenario on the moment of the day when the demand is highest. It is expected that when scenarios are created that vary over the day, the surplus will decrease.

The random scenario performs much better than initially expected. It was expected that the random one would not performs so well, because of the simplicity of the method, but it outperformed the other ones with respect to costs, so Hypothesis 4 turned out to be false. The most promising one is the random approach. From Table 4.5, the maxima for centralized are below 1 and thus Hypothesis 5 is most likely to be true. From Table 4.4, the surpluses for centralized are below 100 and thus Hypothesis 5 appears to be true. As expected the centralized approach usually results in a valid outcome as this reflects the old situation when physical feasibility wasn't a problem yet. This makes Hypothesis 5 true. The greedy approach ended up being more expensive than expected. We expected the greedy one to produce cheap scenarios, because the CHP are cheaper than the RoCa. The weakness of the CHPs is that they have to operate on a fixed debit, whereas the RoCa is more flexible. This makes Hypothesis 6 false. We still think the RoCa's can be used, but they should not be running during the whole day. The RoCa did end up with feasible scenarios more often than expected, but not in all cases. At this point the random one looks most promising in every region except for its stability. The random one comes up with a reasonable solution, but due to the way it's set up it has a chance to produce solutions that are similar to both the centralized and the greedy one. The greedy one doesn't look promising at at all. Nevertheless it is interesting to keep this one because it brings the solution in another position the mutations will have to handle. The centralized initial solution has been shown to be a good starting point and so far all solutions it generated ended up being feasible, so Hypothesis 5 is likely to be true. The random initial method was usually able to yield lower costs but wasn't always stable. The advantage of the random method is that it runs quickly and can be run again with a different seed if it fails. It's also likely that the result is caused by an error in the Drukval model, that makes it hang. When running the non-clustered form, all initial generators had overproduction, so Hypothesis 7 is false. This is probably mainly the result of non-clustering, which forces the algorithm to make a scenario that works even though some moments of the day require a small amount of heat.

## 4.4. MOST EFFECTIVE MUTATION STEPS UNDER DIFFERENT CONDITIONS

In this section the combination of mutations that works best is determined. Initially, this is done by generating different continuations to see which one works best. The stochastic search is balancing between fulfilling the following two goals:

- Exploration: Explore as many regions as possible and find the region that is most likely to contain the global optimum.
- Exploitation: Hill-climbing towards the global minimum

The optimal value that we are trying to find is the one that has lowest costs under the constraint that there should be no physical constraints. Hence, when a given solution has been found a better solution is one that has fewer violations or one that has the same costs but fewer physical violations. There is some conflict between the steps towards the optimum, because a solution that would generate all energy purely centralized would avoid all physical constraints but be expensive due to the high costs of the central plant. On the other hand, a cheap solution might not be physically possible.

### 4.4.1. FINDING CONTINUATIONS

Initially, in order to gain a notion of what constitutes a good mutation, an experiment is run that evaluates all possible mutations in sequence. As initial scenarios we take the methods discussed in Section 4.3 and run them each for a random day. Only one day is tried because Figure 4.2 showed that there is little difference between the different months. Then an algorithm is run on each of these scenarios that tries the following 1-step mutations for each greenhouse with a CHP:

- Make the greenhouse deliver a quantity between 0 and the maximum amount that can be produced using their fixed debit according to Formula 2.6.
- Make the greenhouse produce for own usage either. If not enough is produced the remaining amount is complemented by the RoCa.

#### RESULTING CONTINUATIONS

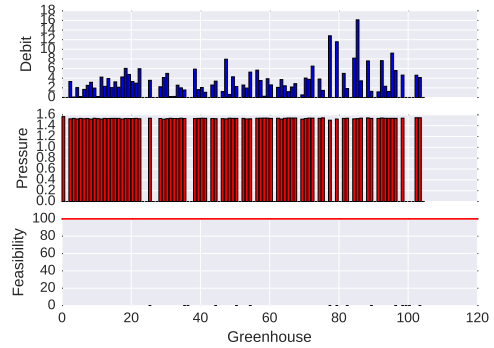
The results of the initial scenarios can be found in Figure 4.3. These graphs show the debits, the pressure and the feasibility constraint at the valve near each of the greenhouses. The graphs on the right are the improvement graphs. They show the influence on the costs and violations score if a single greenhouse changes its heat transfer to or from the network. The costs are linear on the production, but for the violations it matters only if the CHP is turned on or turned off at all with no regard to how much it is producing. A violation score above 0 means that the scenario is infeasible. The graphs contain two horizontal lines. The dash-dotted line gives the situation of the current solution and the dashed lines the situation after applying the best one-step improvement. The feasibility is prioritized over the cost, so the best solution might sometimes be more expensive than the current one if it solves a physical violation. Although it was shown in Chapter 3 that there is more demand in winter than in summer, the improvement graph for summer and winter were similar to each other.

In Figure 4.2a there were no violations and the best move is to make on of the greenhouses deliver much more to the network to reduce the costs without introducing new constraints. This results in a cost improvement from €550 to €500. The situation in Figure 4.2a is very similar. Although it does make use of the CHPs, it does so up to a very small degree and can be further improved by making more use of the CHP. This results in a cost improvement from €490 to €440. The situation in Figure 4.2c is different. There is exactly one violation around greenhouse 45 as can be seen in the feasibility graph. This violation can be eliminated completely, by stopping a single greenhouse from its consumption from the network with minimal effect on the costs which remain stable at €880.

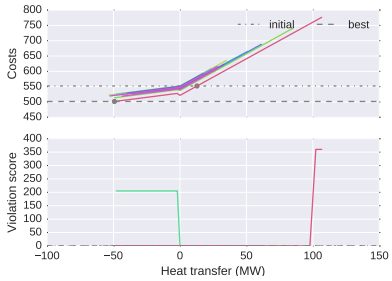
In general, the direction of the best transformation depends on the context. If there are physical constraints are if the costs are because of CHP usage, the production should decrease. If on the other hand, there are no constraints and the usage of the CHPs is low, the costs can be decreased by utilizing the CHPs.

### 4.4.2. IMPROVEMENT STRATEGY

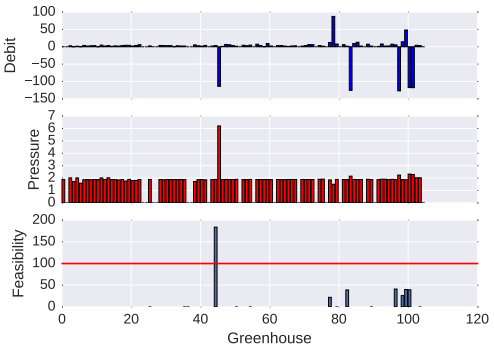
As was clear from the results above, it makes sense to increase the CHP production when the costs are high as happens in Figure 4.3b. On the other hand, when violations are occurring or when there is a lot of overproduction the costs should be reduced again.



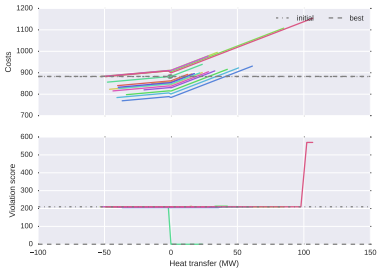

(a) Centralized scenario



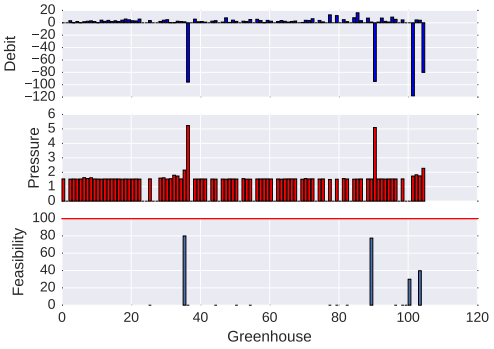

(b) Improvements



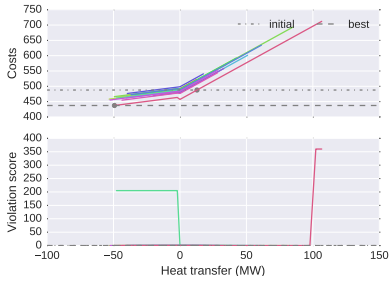

(c) Greedy scenario




(d) Improvements




(e) Random scenario




(f) Improvements

Figure 4.3: Resulting initial scenarios. The debit $m^3/h$ shows the quantity of heat transported to (positive value) or from (negative value) the greenhouse. The pressures are a result of the debits. The feasibility violation depends on the pressure and the debit. A value above 100 indicates a violation at some point in the network. Improvements show costs and violation score. A violation score above 0 means that the scenario is infeasible. The dashdotted line gives the optimality of the current solution and the dashed lines the optimality of the best one-step improvement. Feasibility is prioritized over costs.

If there are no constraints and the RoCa produces more than the CHPs, it will be tried to reduce the RoCa usage as CHPs are cheaper. This leads to the pseudo-code in Algorithm 3. It will consider each greenhouse and with a small probably $\alpha$ it will decrease the greenhouse production if there are constraints or overproduction and increase the greenhouse production if there aren't.

**Algorithm 3** Smart mutation

| | |
|---|---|
| $a$ | Mutation rate |
| $D$ | Total heat demand |
| $H$ | Total heat production $H_{\text{RoCa}} + \sum H_{\text{CHP,}} + H_{\text{boiler}}$ |
| $H_{\text{RoCa}}$ | Heat production by the RoCa before mutation |
| $\sum H_{\text{CHP}}$ | Heat production by all greenhouses before mutation |
| $H_{\text{CHP},g}$ | Heat production by greenhouse $g$ before mutation |
| $H'_{\text{RoCa}}$ | Heat production by the RoCa after mutation |
| $H'_{\text{CHP},g}$ | Heat production by greenhouse $g$ after mutation |

**for** every greenhouse $g$ with CHP **do**
  **if** random$(0\ldots1) < \alpha$ **then**
    **if** (¬feasible **or** H > D) **and** $\sum H_{\text{CHP}} > H_{\text{RoCa}}$ **then**
      $h \leftarrow \text{random}(0\ldots H_{\text{CHP,g}})$
      $H'_{\text{CHP,g}} \leftarrow H_{\text{CHP,g}} - \min(h, H_{\text{CHP,g}})$
      $H'_{\text{RoCa}} \leftarrow H_{\text{RoCa}} + \min(h, H_{\text{CHP,g}})$
    **else**
      $h \leftarrow \text{random}(0\ldots H_{\text{CHP,g}})$
      $H'_{\text{CHP},g} \leftarrow H_{\text{CHP},g} + \min(h, H_{\text{RoCa}})$
      $H'_{\text{RoCa}} \leftarrow H_{\text{RoCa}} - \min(h, H_{\text{CHP,g}})$
    **end if**
  **end if**
**end for**

This will be compared to the minimal random method in Algorithm 4, which works by considering every greenhouse and with low probability $\alpha$ it will flip a coin. On heads it will increase the CHP production in favour of the RoCa production by a random amount and on tails it will decrease the CHP production in favour of the RoCa production.

A less transparent but more autonomous approach is followed in Algorithm 5. It would let the algorithm itself decide in which direction it performs a step. Initially it would decrease the RoCa usage and increase CHP usage, but if that worsens or doesn't change the solution, it will change direction and increase the RoCa while decreasing CHP usage. Just as in Algorithm 3 it will be possible to perform both a mutation that balances the production towards more RoCa usage and a mutation that balances the production to using the CHPs. However the direction is decided on by the algorithm. If for example a CHP increasing step improves the result, it will be repeated on the next step. But if a CHP increasing step deteriorated the result or had no effect, a decreasing CHP motion will be tried on the next turn.

**Hypothesis 8** *The improvement strategy in Algorithm 3 works better than a random mutation. It will converge in fewer iterations and find a better solution.*

**Hypothesis 9** *The mutation in Algorithm 3 converges to a local optimum.*

### 4.4.3. SETUP OF SIMULATED ANNEALING WITH PROPOSED SOLUTIONS

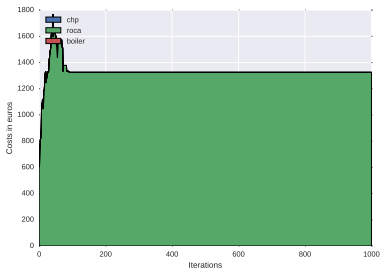
Both the random mutation in Algorithm 4 and our proposed solutions in Algorithm 3 and Algorithm 5 are compared. They will be tested on 6 days of January, a winter month; and 6 days of June, a spring month. This is a small amount of samples, but unfortunately the experiments have a long running time. Both the initial method and the mutation are varied.

**Algorithm 4** Random mutation

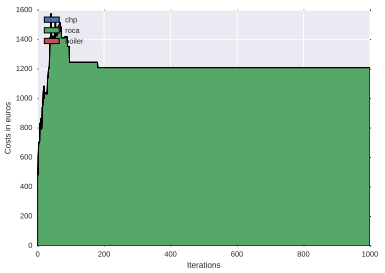
**for** every greenhouse $g$ with CHP **do**
  **if** random$(0 \ldots 1) < \alpha$ **then**
    **if** random$(0 \ldots 1) < 0.5$ **then**
      $h \leftarrow \text{random}(0, H_{\text{CHP,g}})$
      $H'_{\text{CHP,g}} \leftarrow H_{\text{CHP,g}} - \min(h, H_{\text{CHP,g}})$
      $H'_{\text{RoCa}} \leftarrow H_{\text{RoCa}} + \min(h, H_{\text{CHP,g}})$
    **else**
      $h \leftarrow \text{random}(0, H_{\text{CHP,g}})$
      $H'_{\text{CHP,g}} \leftarrow H_{\text{CHP,g}} \min(h, H_{\text{RoCa}})$
      $H'_{\text{RoCa}} \leftarrow H_{\text{RoCa}} \min(h, H_{\text{RoCa}})$
    **end if**
  **end if**
**end for**

**Algorithm 5** Adaptive mutation

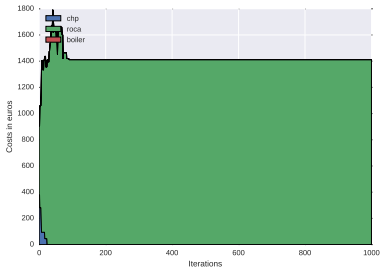
**if** previous mutation was not an improvement **then** ▷ Decide on strategy $s$
  **if** s=1 **then**
    $s \leftarrow 2$
  **else if** s=2 **then**
    $s \leftarrow 1$
  **end if**
**end if**
**for** every greenhouse $g$ with CHP **do** ▷ Execute selected strategy
  **if** $s = 1$ **then**
    $h \leftarrow \text{random}(0, H_{\text{CHP,g}})$
    $H'_{\text{CHP,g}} \leftarrow H_{\text{CHP,g}} - \min(h, H_{\text{CHP,g}})$
    $H'_{\text{RoCa}} \leftarrow H_{\text{RoCa}} + \min(h, H_{\text{CHP,g}})$
  **else if** $s = 2$ **then**
    $h = \text{random}(0, H_{\text{CHP,g}})$
    $H'_{\text{CHP,g}} \leftarrow H_{\text{CHP,g}} \min(h, H_{\text{RoCa}})$
    $H'_{\text{RoCa}} \leftarrow H_{\text{RoCa}} \min(h, H_{\text{RoCa}})$
  **end if**
**end for**



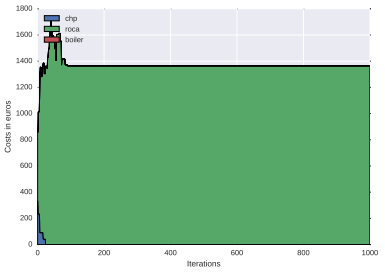

(a) Winter random mutation op centralized



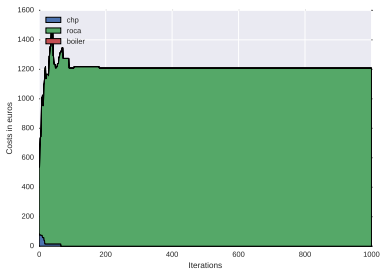

(b) Summer random mutation op centralized



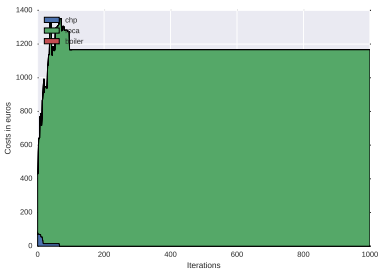

(c) Winter random mutation on greedy




(d) Summer random mutation on greedy




(e) Winter random mutation op random




(f) Summer random mutation op random

Figure 4.4: Random mutation

**Score (costs + penalty for violations)** For evaluation, the score of the optimal solution as in Equation 4.2 is taken for each month. When there are no violations (which is expected to be the typical case) this will be equal to the costs.

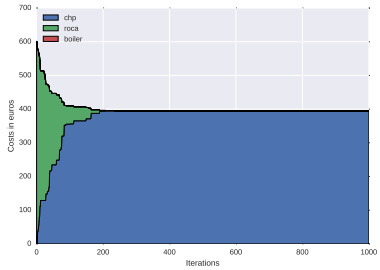
**Convergence rate** The number of mutations until convergence. This tells us something about the efficiency of the solution. It is calculated as the last score after which no changes have occurred. Our actual convergence criterion checks if nothing has changed during the last 100 runs. Because of time constraints, the algorithm will also terminate if it takes more than 1000 iterations.

**Percentage violations** The amount of scenarios that did not converge to a solution without constraints.

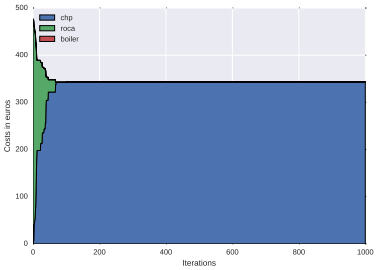
**Violation score** The violation score used by the SA algorithm as described before in Equation 4.3. A value of 0 means no physical violations. Any other (positive) value is bad.

### 4.4.4. RESULT OF SIMULATED ANNEALING

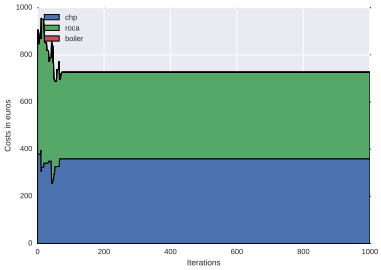
The result of running simulated annealing with different initial solutions and mutations is found in Table 4.6. The raw results are also included in Appendix A.1. Because the number of trials used for this experiment was low, an ANOVA test has been done for the equivalence of costs when starting with a random scenario and the equivalence of costs. These p-values can be found in Table 4.7 and 4.8 respectively. On average, the best result was achieved by combining a random initial solution with a smart mutation which resulted in an average cost of €351 per day. The second-best result was obtained using an adaptive mutation which resulted



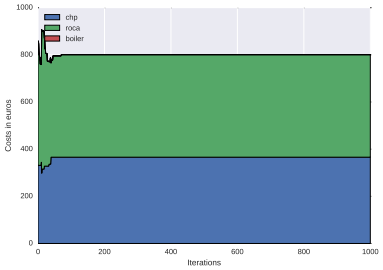

(a) Winter scenario from centralized initial solution



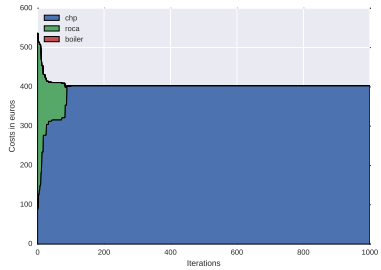

(b) Summer scenario from centralized initial solution



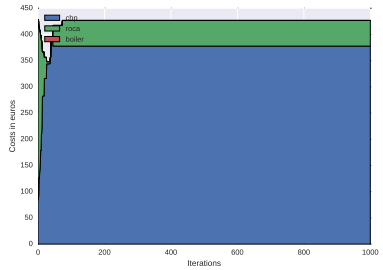

(c) Winter scenario from greedy initial solution




(d) Summer scenario from greedy initial solution

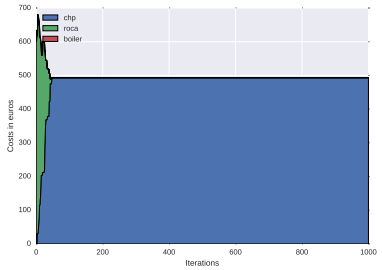



(e) Winter scenario from random initial solution

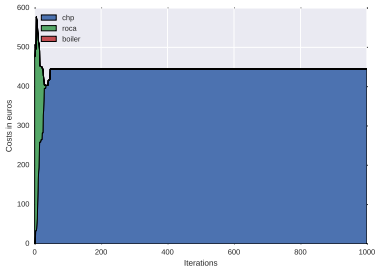



(f) Summer scenario from random initial solution

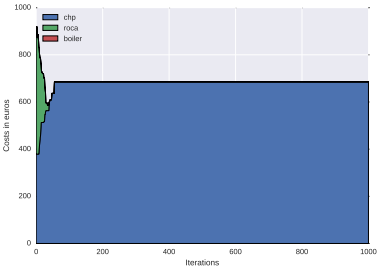
Figure 4.5: Smart mutation



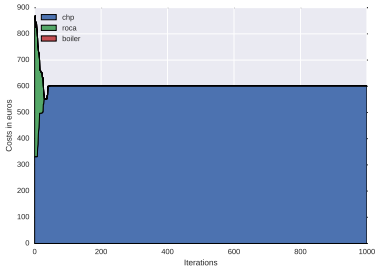

(a) Winter scenario from centralized initial solution



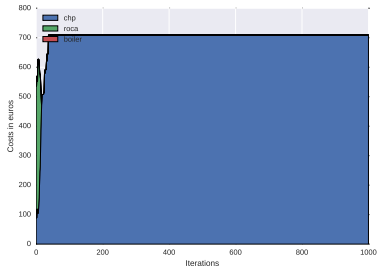

(b) Summer scenario from centralized initial solution



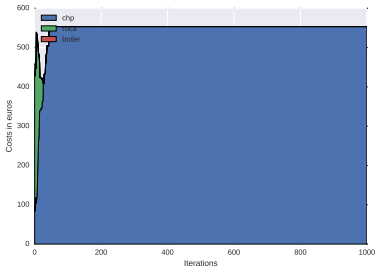

(c) Winter scenario from greedy initial solution




(d) Summer scenario from adaptive initial solution




(e) Winter scenario from random initial solution




(f) Summer scenario from random initial solution

Figure 4.6: Adaptive mutation

| | | Centralized | Greedy | Random |
|---|---|---|---|---|
| No mutation | Mean cost | 517.8 | 876.9 | 457.4 |
| | Std dev cost | 67.7 | 39.9 | 62.2 |
| | Mean convergence | 0.0 | 0.0 | 0.0 |
| | Std dev convergence | 0.0 | 0.0 | 0.0 |
| | Percentage violations | 0.0 | 8.3 | 0.0 |
| | Mean violation score | 0.0 | 209.7 | 0.0 |
| | Std dev violation score | 0.0 | 0.0 | 0.0 |
| Random mutation | Mean cost | 517.8 | 917.1 | 457.4 |
| | Std dev cost | 67.7 | 138.6 | 62.2 |
| | Mean convergence | 135.5 | 90.0 | 142.5 |
| | Std dev convergence | 45.5 | 0.0 | 38.5 |
| | Percentage violations | 0.0 | 0.0 | 0.0 |
| | Mean violation score | 0.0 | 0.0 | 0.0 |
| | Std dev violation score | 0.0 | 0.0 | 0.0 |
| Adaptive mutation | Mean cost | 430.9 | 547.0 | 426.9 |
| | Std dev cost | 46.9 | 30.2 | 41.1 |
| | Mean convergence | 46.8 | 52.9 | 41.8 |
| | Std dev convergence | 4.0 | 7.4 | 5.2 |
| | Percentage violations | 0.0 | 0.0 | 0.0 |
| | Mean violation score | 0.0 | 0.0 | 0.0 |
| | Std dev violation score | 0.0 | 0.0 | 0.0 |
| Smart mutation | Mean cost | 354.4 | 693.6 | 351.2 |
| | Std dev cost | 35.1 | 43.2 | 32.2 |
| | Mean convergence | 201.4 | 75.4 | 200.0 |
| | Std dev convergence | 155.0 | 5.7 | 215.4 |
| | Percentage violations | 0.0 | 8.3 | 0.0 |
| | Mean violation score | 0.0 | 205.1 | 0.0 |
| | Std dev violation score | 0.0 | 0.0 | 0.0 |

Table 4.6: Combination of initial methods and Simulated Annealing

in an average cost of €426. Table 4.7 shows that the difference between the adaptive and smart mutation is significant ($8.34 \times 10^{-5} \ll 0.05$).

There differences between the initial scenarios are very small. The smart and centralized initial scenario seem very close to each other and the results of the smart mutation are only slightly better. Unfortunately table 4.8 doesn't confirm the random scenario to be significantly better than the centralized one, because the p-value is not below 0.05. The greedy initial scenario seems like a bad one, because it's costs are in always higher than the others and because it shows violations, yet the ANOVA test doesn't confirm with high significance that the random and the centralized are better than the greedy one. However the greedy one is the only one that has violations.

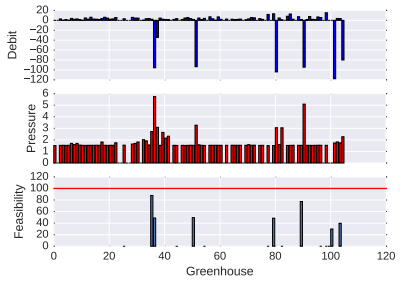
Table 4.6 includes the average costs of the solutions that ended up having no violations. In most cases no violations occurred, with the exception of one solution obtained by taking the greedy initial solution with no further mutations and the same scenario followed by a smart mutation. The solution not solved by smart mutation is depicted in Figure 4.8. The adaptive mutation performs much better on this scenario.

| | No mutation | Smart mutation | Adaptive mutation | Random mutation |
|---|---|---|---|---|
| No mutation | - | 4.933e-05 | 0.1884 | 1 |
| Smart mutation | 4.933e-05 | - | 8.342e-05 | 4.933e-05 |
| Adaptive mutation | 0.1884 | 8.342e-05 | - | 0.1884 |
| Random mutation | 1 | 4.933e-05 | 0.1884 | - |

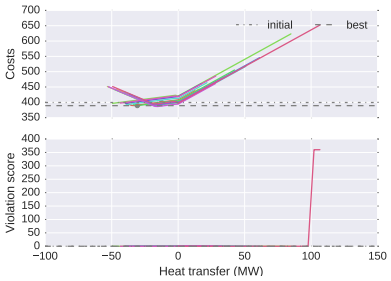
Table 4.7: P-values for ANOVA test for the hypothesis for equivalent costs after a random initial scenario. When the value is $< 0.05$ we conclude that the results are different.

| | Centralized | Greedy | Random |
|---|---|---|---|
| Centralized | - | 6.198e-07 | 0.8309 |
| Greedy | 6.198e-07 | - | 8.564e-08 |
| Random | 0.8309 | 8.564e-08 | - |

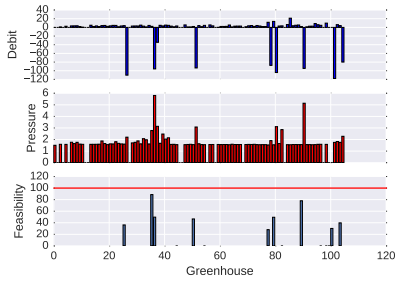
Table 4.8: P-values for ANOVA test for the hypothesis for equivalent costs when performing a smart mutation. When the value is $< 0.05$ we conclude that the results are different



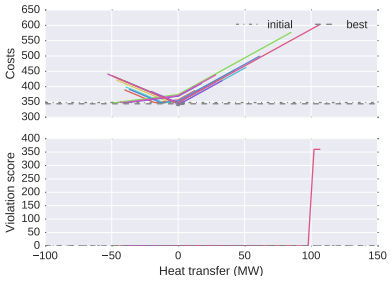

(a) Winter month after simulated annealing




(b) Winter improvements after simulated annealing




(c) Summer month after simulated annealing




(d) Summer improvements after simulated annealing

Figure 4.7: Result of simulated annealing on centralized initial scenario and improvements after optimizing

As Tables 4.6 shows, random mutations are hardly better than no mutation and it actually never improves the costs. One thing the random mutation is capable of is getting rid of the violations as it gets rid of the constraints generated by the greedy initial method. As a result, the average costs slightly go up, because one extra data point is added that was excluded from "no mutation", because it had violations.

In Figure 4.5, the mutations seem to converge when Algorithm 3 is used. To check if the solution has converged to a local optimum, the process in Subsection 4.4.1 is run on the best solution, that is some of the solutions generated by a combination of a centralized scenario and simulated annealing with a smart mutation. The results are in Figure 4.7b and Figure 4.7d and as can be seen the result hasn't fully converged, but the mutations are marginal improvements only.

### 4.4.5. CONCLUSION ON SIMULATED ANNEALING

The mutations have shown a lot of improvement. By using a smart mutation, the average costs have decreased from €457 to €351 which is an improvement of 23 %. Both the smart and the adaptive mutation performed better than a random mutation, which seems in favour of Hypothesis 8. In fact, a random mutation was only able to get rid of the constraints, but didn't improve the solution by any significant amount. The smart mutation performed better than an adaptive mutation

Although the results in Section 4.3.4 showed the random scenario to be better than the centralized one, the differences between these initial scenarios mostly disappeared after the mutations. Both these scenarios started in the feasible region in all cases. It is suspected that for the smart and adaptive mutations it doesn't matter what starting point is chosen as long as it's feasible. In theory the random one can produce the same scenarios as both the centralized and greedy one, so for most stability the centralized one is recommended.

For the smart mutation it does seem important what initial scenario is given as it's the only mutation that wasn't able to get rid of the constraints introduced by the greedy scenario. It could be that the code in

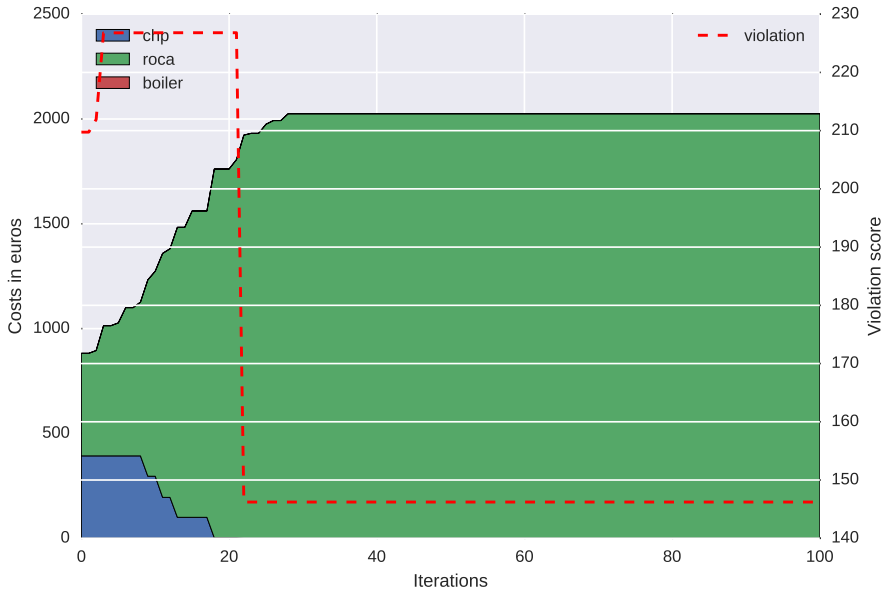



Figure 4.8: Scenario on which smart mutation fails to get rid of violations

Algorithm 3 is not very good at resolving infeasible scenarios. The algorithm could when given an infeasible scenario be adapted to follow the same rules as the adaptive or random scenario or the rules could be changed to better deal with infeasible scenarios.

The adaptive mutation is better at dealing with input given by the adaptive scenario. It did end up with worse results than the smart mutation when it had to deal with solutions that started in the feasible region. It is expected this is because the adaptive one is adaptive to different situations, it adapts more slowly than the smart one. The smart one knows what to do in each situation. The adaptive one on the other hand is pushed in a certain direction and the result needs to worsen before it realizes it should change its direction. It also may end up changing direction continuously when it gets stuck in a situation where it is hard to find a nearby region that improves the situation. If this happens, adaptive would would continuously switch strategies, whereas the smart one sticks to a single strategy.

The final result converged to a good solution, but when seeking improvements by brute force, it still managed to find an improvement, although a marginal one, so strictly speaking Hypothesis 9 turned out to be false. It is still expected that the result converges close to the local minimum. A more definite result could be obtained by running the scenario improver for each solution generated by SA, however this requires a lot of evaluations and will take a long time. Much harder to tell would be if the result is close to the global optimum.

## 4.5. SEGMENTATION

Section 4.4 discussed how to solve the problem for a demand that doesn't change during the day. In reality, the demand does vary over the day and so coming up with a single scenario for the whole day is probably not going to be optimal. On the other hand, the greenhouse owners are not willing to constantly adapt and there is some start-up time involved for putting CHPs on and off. This problem is solved by dividing the day into segments and solving the problem for each segment separately.

**Hypothesis 10** *Segmentation improves the result (lower total costs) over generating a single scenario for the whole day compared to creating a single scenario for the whole day.*

**Hypothesis 11** *Segmentation converges in fewer iterations compared to generating a single scenario for the whole day.*

### 4.5.1. OBJECTIVE OF SEGMENTATION

Clustering should be done for each individual day. The total demand of all greenhouses during a day is a continuous function of time, where $t$ is time in seconds:

$$d(t) \in \mathbb{R}^+ \tag{4.12}$$

This function has been sampled in intervals of 5 minutes. So during a day there are $\frac{24\times 60}{5} = 288$ intervals. These intervals will be denoted as $d(1) \dots d(288)$.

These intervals should be segmented into $N$ segments such that:

- Each interval is part of exactly one segment, but a segment can contain many intervals.
- If for an interval $j$ there exist a prior interval $i$ and a later interval $d$ and both intervals $i$ and $d$ are assigned to a segment $c$, then interval $j$ should be assigned to segment $c$ as well.

So if the original time series is $T$, then the segmentation into $M$ segments should be of the form:

$$C = \{[d(l_1) \dots d(r_1)] \dots [d(l_M) \dots d(r_M)]\} \tag{4.13}$$

where $l_1 = 0$, $r_M = n$ and $l_j = r_{j-1} + 1$ for all $2 \leq j \leq M$.

We want the elements within a segment to be as similar as possible and to minimize the variance of the group. Mathematically, we want the following term (the sum of variances within each cluster) to be minimized:

$$z(C) = \sum_{[d(l_j) \dots d(r_j)] \in C} \left( \sum_{t=l_j}^{r_j} \left| d(t) - \overline{d_{c_j}} \right|^2 \right) \tag{4.14}$$

$$\text{where } \overline{d_{c_j}} = \frac{\sum_{u=l_j}^{r_j} d(u)}{r_j - l_j + 1} \tag{4.15}$$

The runtime of performing a single evaluation of a segmentation takes order $O(n)$.

### 4.5.2. REQUIREMENTS

Practical constraints:

- A CHP can only be turned on twice a day
- A heat source should not change its state more than once an hour

Limiting the number of segments to 4 satisfies the first constraint. The second constraint can be satisfied by forcing a minimum segment duration of an hour.

### 4.5.3. LITERATURE ON SEGMENTATION AND CLUSTERING

There are two common approaches to clustering: [15]

**Hierarchical** This method works bottom-up. It starts with single-element clusters and repetitively joins the clusters that are most similar according to a (dis)similarity metric and a linkage criterion. This has a runtime complexity of $O(n^3)$.

**Divisive** This method works top-down. It starts with a cluster that contains all the intervals and is repetitively split into smaller clusters according to a splitting criterion like variance. This has a runtime complexity of $O(2^n)$.

The similarity criterion is the function that is used to decide how close two elements are. This makes sense if data points are multidimensional, but because the demand for our problem is in the interval $\mathbb{R}^+$ (1-dimensional), the value itself can be used. Because segments consist of multiple points a linkage criterion is needed which tells us which points of the cluster should be used for comparison. For our problem the average would be used as this makes most sense considering Equation 4.15.

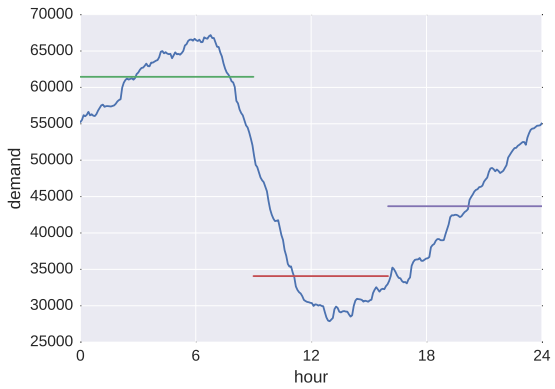



Figure 4.9: Expected segmentation. A morning peak, afternoon drop and evening rise are expected which will have to be divided over 4 segments.

**Adaptive Piecewise Constant Approximation Segmentation** ACPA segmentation [16] is a technique that produces unequal segments. In its original form it scans the data from left to right and creates a new segment whenever the total error so far has exceeded a certain threshold. Using this method in its original form however does not give us a way to specify the number of segments in advance.

### 4.5.4. SOLUTION

The costs of evaluation Equation 4.15 for a clustering takes $O(n)$ and the decision needs to be made at what index the right borders of the clusters, $r_1$, $r_2$ and $r_3$, need to be placed. If we have 4 segments at most, the calculation costs of a bruteforce solution that tries all solutions to find the minimum are bounded by:

$$T(n) = \prod_{r_1=1}^{n-3}\left(\prod_{r_2=r_1+1}^{n-2}\left(\prod_{r_3=r_2+1}^{n-1} n\right)\right) \in O\left(\binom{n}{3}\cdot n\right) \subseteq O(n^4) \tag{4.16}$$

From the equation above it's clear that more generally a brute-force solution takes $O(n^{\#c})$ where $\#c$ is the number of clusters. With dynamic programming, the runtime can be reduced to $O(n^3 \cdot \#c)$. This approach is possible, because the evaluation can be calculated for each cluster independently. The runtime can even be reduced to $O(n^2 \cdot \#c)$ if one realizes that the variance can be calculated as a rolling (geometric) average.

### 4.5.5. EXAMPLE

Looking at Figure 3.1a and Figure 4.10, segmenting the day in three parts seems like an option.

The expected segmentation (three parts) is as follows:

- Morning peak (from 0:00 to 8:59)
- Afternoon drop (from 9:00 to 15:59)
- Evening rise (from 16:00 to 23:59)

The resulting segmentation is illustrated in Figure 4.9 for the average day discussed earlier in Figure 3.1a. Instead of three segment, it is chosen to split it up in four segments for more flexibility. When a day behaves as expected, one of these segments might be split up into 2 segments. If the day has an atypical peak or hill, the fourth segment can be used for that anomaly. The constraint of not having to put CHPs on and off more than twice a day is still fulfilled.

### 4.5.6. RESULTING SEGMENTATION

The segmentation process is close to what was expected. The example of Figure 4.9 has been turned into Figure 4.10a, which is almost exactly as expected but since there are 4 clusters, the fourth cluster has been split into two parts. A typical example is shown in Figure 4.10b. In Figure 4.10c a segment is seen that according to our intuition in subsection 4.5.5 is regarded as a single segment. However this isn't a problem as the split improves the representation of the individual intervals. Some less trivial examples are shown in Figure 4.10d

| | | Centralized | Greedy | Random |
|---|---|---|---|---|
| No mutation | Mean cost | 382.6 | 929.4 | 345.3 |
| | Std dev cost | 74.3 | 25.6 | 63.4 |
| | Mean convergence | 0.0 | 0.0 | 0.0 |
| | Std dev convergence | 0.0 | 0.0 | 0.0 |
| | Percentage violations | 0.0 | 0.0 | 8.3 |
| | Mean violation score | 0.0 | 0.0 | inf |
| | Std dev violation score | 0.0 | 0.0 | 0.0 |
| Random mutation | Mean cost | 382.6 | 929.4 | 345.3 |
| | Std dev cost | 74.3 | 25.6 | 63.4 |
| | Mean convergence | 695.7 | 645.6 | 670.8 |
| | Std dev convergence | 389.4 | 362.7 | 397.5 |
| | Percentage violations | 0.0 | 0.0 | 8.3 |
| | Mean violation score | 0.0 | 0.0 | inf |
| | Std dev violation score | 0.0 | 0.0 | 0.0 |
| Adaptive mutation | Mean cost | 301.9 | 702.4 | 297.1 |
| | Std dev cost | 59.3 | 50.2 | 57.3 |
| | Mean convergence | 49.8 | 50.5 | 36.7 |
| | Std dev convergence | 19.1 | 16.6 | 11.9 |
| | Percentage violations | 0.0 | 0.0 | 8.3 |
| | Mean violation score | 0.0 | 0.0 | inf |
| | Std dev violation score | 0.0 | 0.0 | 0.0 |
| Smart mutation | Mean cost | 331.2 | 733.3 | 309.2 |
| | Std dev cost | 78.5 | 45.6 | 61.8 |
| | Mean convergence | 902.2 | 968.8 | 928.2 |
| | Std dev convergence | 148.4 | 23.6 | 97.7 |
| | Percentage violations | 0.0 | 0.0 | 8.3 |
| | Mean violation score | 0.0 | 0.0 | inf |
| | Std dev violation score | 0.0 | 0.0 | 0.0 |

Table 4.9: Combination of initial methods and Simulated Annealing with clustering

to Figure 4.10f where outliers make up their own segment. In Figure 4.10e the influence of a restriction of minimum of an hour is shown, where around 9 a clock an outlier is stretched until its length is exactly an hour.

### 4.5.7. INTEGRATION OF SEGMENTATION INTO MAIN ALGORITHM

The simulated annealing algorithm is run again, but this time the scenario is clustered in advance. It is expected that this will improve the performance because the generated scenarios are clustered to more specific problems. So the steps are the following:

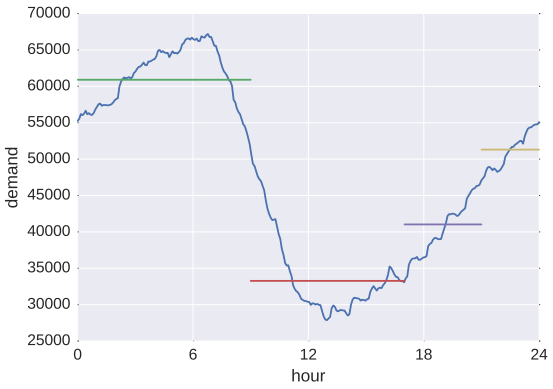
1. Cluster the demand into 4 partitions.
2. Generate initial solution for each cluster separately.
3. Perform the SA mutation for each cluster separately.

The transitions between the clusters are ignored, but will briefly be discussed in Contributions and Future work.

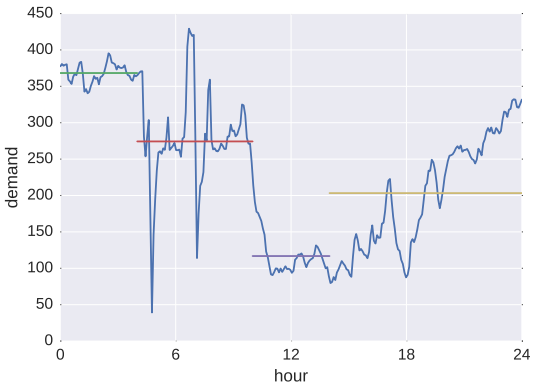
### 4.5.8. RESULT OF SEGMENTATION ON SIMULATED ANNEALING

The result of running simulated annealing on the clustered data is shown is shown in Table 4.9. The raw results are also listed in Appendix A.2. Some example results are in Figure 4.11.

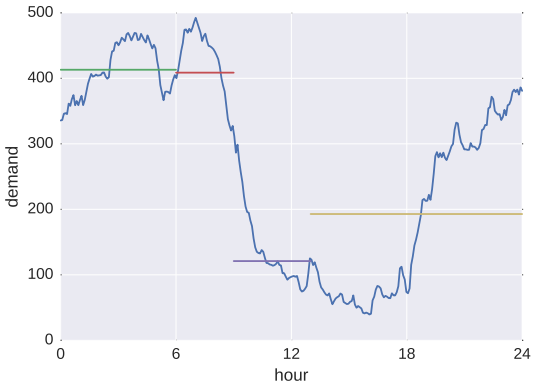
For all scenarios that started from a centralized or a random one, the costs massively improved by clustering. Whereas our optimal scenario without clustering had a cost of €351, the best solution with clustering has a cost of only €297.1.



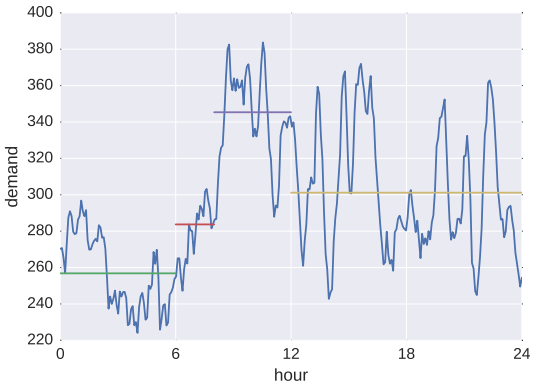

(a) Average day



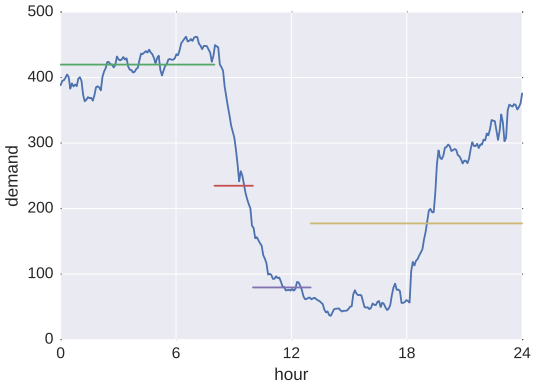

(b) Typical situation



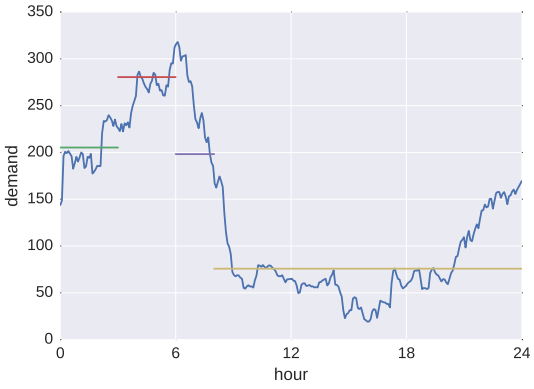

(c) Split of afternoon drop segment




(d) Much variance 1




(e) The second cluster is forced to the length of an hour




(f) One large segment

Figure 4.10: Some particular clusterings

| | No mutation | Smart mutation | Adaptive mutation | Random mutation |
|---|---|---|---|---|
| No mutation | - | 0.2117 | 0.08968 | 1 |
| Smart mutation | 0.2117 | - | 0.6561 | 0.2117 |
| Adaptive mutation | 0.08968 | 0.6561 | - | 0.08968 |
| Random mutation | 1 | 0.2117 | 0.08968 | - |

Table 4.10: P-values of ANOVA test for the null hypothesis that the mutations give different results when using a random initial scenario. If a significance level of 0.05 is used, none of these results is significant.

| | Centralized | Greedy | Random |
|---|---|---|---|
| Centralized | - | 3.388e-14 | 0.5817 |
| Greedy | 3.388e-14 | - | 3.459e-13 |
| Random | 0.5817 | 3.459e-13 | - |

Table 4.11: P-values of ANOVA test for the null hypothesis that the initializers give different results when using a smart mutation.

On average, the best result was obtained by combining a random initial scenario with an adaptive mutation. The number of trials was similar to the number of trials used in Section 4.4.3 The difference between the smart mutation and adaptive mutation is much smaller. When we didn't split the data into segments, the adaptive mutation scored 18 % better than the smart mutation when paired with a random initializer, but when the results are segmented, the results of the smart and adaptive mutation are close to each other and the adaptive one actually scores slightly better. It is also interesting to note that the adaptive mutation generally converged in about 50 mutations, whereas the scenarios by the smart mutation may not have converged at all, because they were capped at 1000 iterations. Unfortunately, the number of trials is small and the results are very close to each other, so it is not possible at this point to say whether the adaptive mutation is better than the smart mutation if more experiments would be done.

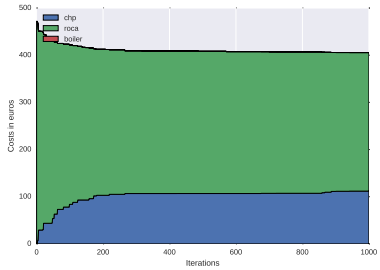
There was one random scenario though that made Drukval crash, which is expected to be a Drukval bug, which explains why the mean violation score is infinite. The greedy initial scenario is the only one that got more expensive when solving the clustered version, no matter what mutation was chosen. It is not known why this happens. An ANOVA test has been done in Table 4.10 to see if the results of the adaptive and smart mutation are different, but the p-value of 0.6561 is not enough to reject the null hypothesis that the results are the same. In fact, this test indicates that if we pay attention to the costs none of the mutations performs significantly better if we start from a random initial scenario. Some other ANOVA tests have been done in Table 4.11 to see if the initial scenarios matter. This test doesn't show the random and centralized scenarios to be significantly different, but both tests differ from the greedy scenario.

### 4.5.9. CONCLUSION ON SEGMENTATION

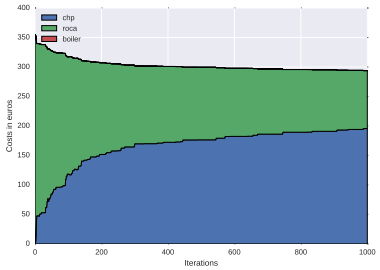
All results improved by clustering and the costs of the best solution decreased from €351 to €297, which is an improvement of 15 %, so Hypothesis 10 turned out to be true. We expect the improvement in results because we are optimizing for smaller segments in which the spread of the demand is low, so we can make better scenarios than before we clustered when we had to come up with a single scenario for the whole day, even though the demand varies a lot throughout the day as was shown in Figure 3.1a.

In general, the segmented problems took more iterations to converge than the unsegmented problems. It was expected that solutions would converge faster, because the segmented problem should be easier, but Hypothesis 11 turned out to be false. It is expected that the convergence takes longer, because the clustered form gives the algorithm more room for finetuning, whereas in the clustered form the algorithm needs to be more careful not to run into violations. The unclustered form might thus end up more often in a situation where it is hard to find a good mutation that doesn't introduce new constraints or requires an extra heat source to be enabled.

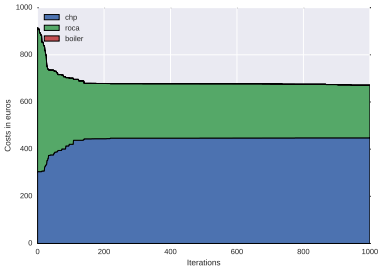
Although the results in subsection 4.3.4 showed the random initial scenario to be better than the centralized one, the differences between these initial scenarios mostly disappeared after the mutations. Both these scenarios started in the feasible region in all cases. Only the greedy initial scenario performed much worse. Because of the small amount of data and the small differences in result, it can not be said if the random initial solution is a better starting point than the centralized initial one. It is suspected that for the smart and



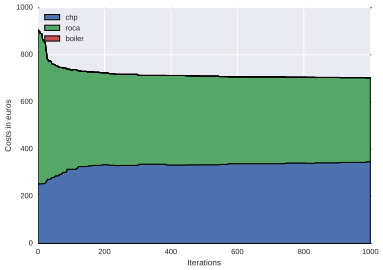

(a) Winter scenario from centralized initial solution



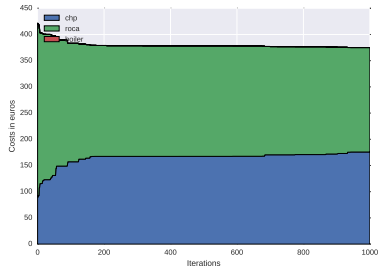

(b) Summer scenario from centralized initial solution



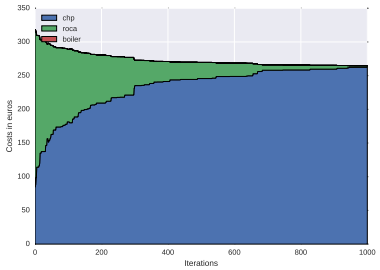

(c) Winter scenario from greedy initial solution




(d) Summer scenario from greedy initial solution




(e) Winter scenario from random initial solution




(f) Summer scenario from random initial solution

Figure 4.11: Smart clustered mutation

adaptive mutations it doesn't matter what starting point is chosen as long as it is feasible. If a choice has to be made between the centralized and random initial scenario, the centralized initial scenario always ends up in the feasible domain, whereas the random initial scenario has a small probability to produce an infeasible solution, so the centralized initial scenario is recommended for production as it produces the most stable solutions.

The adaptive mutation is better at dealing with input given by the adaptive scenario. It did end up with worse results than the smart mutation when it had to deal with solutions that started in the feasible region. It is expected to be because in case the situation changes, the adaptive one first needs to adapt to the new situation first, whereas the smart one knows what to do. The adaptive one on the other hand is pushed in a certain direction and the result needs to worsen before it realizes it should change its direction. It also may end up changing direction continuously when it gets stuck in a situation where it is hard to find a nearby region that improves the situation. If this happens, adaptive would continuously switch strategies, whereas the smart one sticks to a single strategy.

## 4.6. CONCLUSION ON SIMULATED ANNEALING

Different approaches have been tried to get to this result. An overview of choices can be found in Figure 4.12 Segmentation has been shown to work better than maximizing the heat and although other ways can be construed to deal with time, segmentation should at least be considered an option, because it's easy to implement and has shown good results.

The initial solution that worked best for our experiments is the random one. Proportional is not recommended, because it puts much pressure on the network and is inflexible, because the CHP's need to operate on a fixed debit. A better starting point is to use the RoCa and to enable a few CHP's. Our experiment showed slightly better results when using a random scenario than a centralized one that solely relies on the RoCa, but the centralized one is safer, because the random one still has a small probability to end up with an infeasible scenario from which it is more difficult to optimize.

The random mutation is not recommended, unless you have no other option. The other mutations gave better results. Both the adaptive and smart mutation gave good results, but neither of them was significantly better than the other. The general advantage of the adaptive is that its basic ideas can be applied in different contexts too, whereas the smart mutation relies on our domain knowledge. If more insight into the domain is added to the problem, it is recommended to try the smart mutation, otherwise the adaptive mutation is still a good choice.

For the simulated annealing parameters, the choice was made to use the temperature function of Goffe, but to change the acceptance probability to measure the change as a fraction of the total amount, which was done to make the acceptance probability work with small differences when the step is made from a a solution that is physically feasible to another solution that is still physically feasible but cheaper and to work with steps from an infeasible solution to a slightly more feasible solution, which are much bigger because of the penalty function.

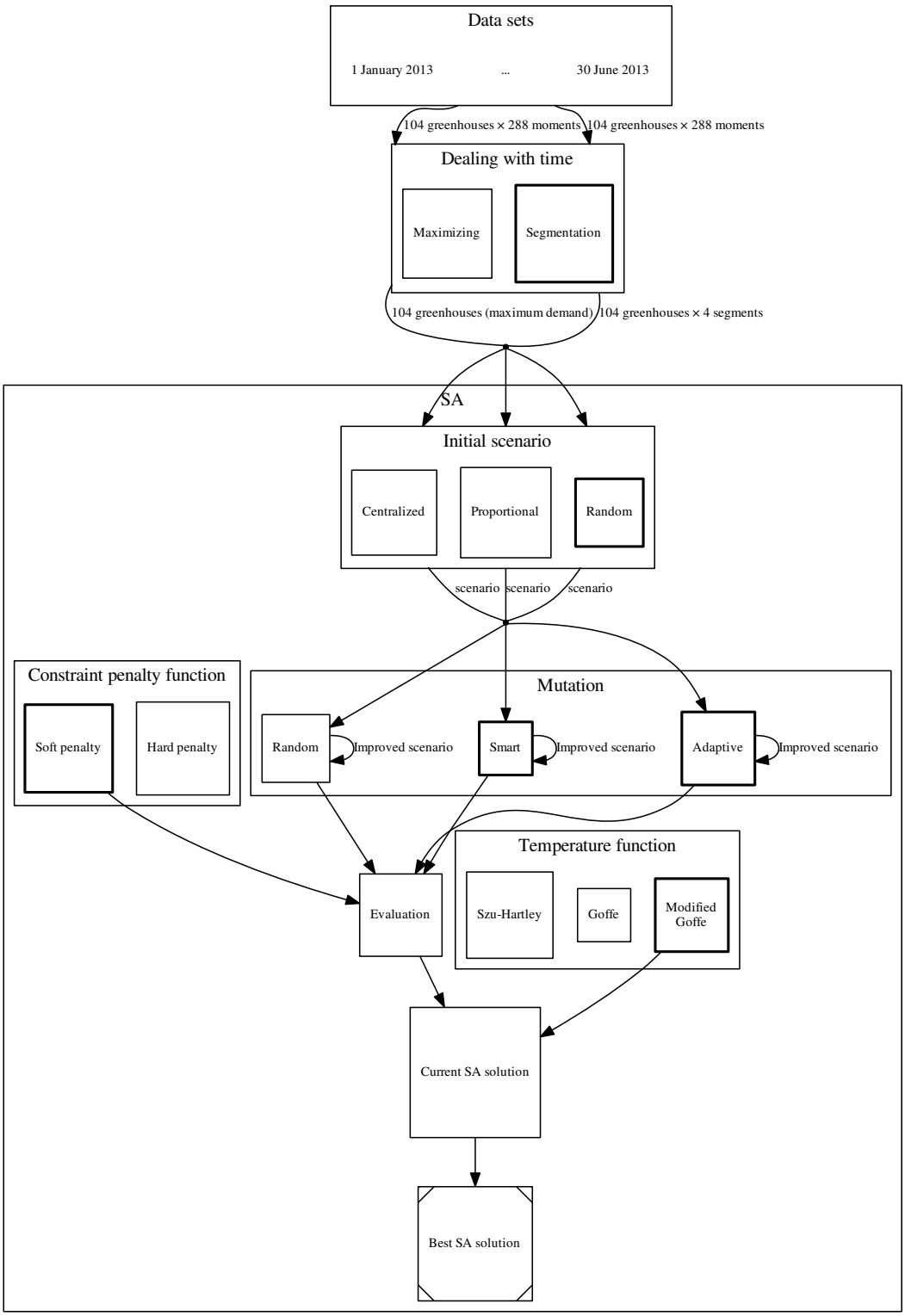



Figure 4.12: Overview of all decisions made. The recommended decisions have been made bold

# 5

# CONTRIBUTIONS AND FUTURE WORK

## 5.1. CONCLUSIONS

The focus of this thesis was the question "How can a heat control optimization problem with both linear and non-linear constraints be solved as optimally and efficiently as possible?". We came up with a way to solve heat demand scheduling problems under several constraints by making use of simulated annealing and load balancing algorithms.

We have shown that techniques such as simulated annealing and segmentation can end up with better scenarios than can be found by simple techniques such as using a centralized scenario for the whole day, although such simple techniques can be used as a initial scenario to optimize using simulated annealing. Instead of using the basic form of simulated annealing that works using a random initial solution and random mutations, we have decided to put some intelligence into the way the initial scenario and the mutations are done. It was found that a random solution creates a better starting point for finding other solutions than by starting from a centralized scenario or a greedy scenario that tries to use as few heat sources as possible. For simulated annealing, we found that solutions that are in the physically feasible domain already are easier to optimize than scenarios that need to be made feasible first. The problem with non-feasible solutions is that two criteria need to be optimized: the solution should be physically possible and the solution should be efficient. If given a feasible solution to start with, the algorithm only needs to optimize the costs. We found that an easy way to get a solution in the feasible domain is to start from the centralized scenario, that makes the RoCa responsible for all heat production. Finally, we have shown a way to solve the problem over time, by dividing the time series into segments and solving the problem per segment.

**Days follow similar patterns with a lot of variations and there is a difference between summer and winter months** In Chapter 3, we have looked at the data and found some patterns in the data. There is a strong correlation between the outdoor temperature and the collective heat demand. Related to this, we found that the day of the year indirectly influences the demand through temperature. Another factor of influence on the demand is the time of the day. The demand over individual days roughly follow the same pattern: there is a morning peak, a dip during the afternoon and a rise in the evening. However as we saw in Section 4.5, the individual days tend to vary a bit from this general pattern. This concludes research question 1.

**Simulated annealing has many advantages** At the beginning of Chapter 4, the choice was made to use simulated annealing. Advantages of this method are that it is flexible, that it still works when little domain knowledge is available, that it can escape from local minima and that when given enough time, it is highly probable (but not guaranteed) to find the global optimum. Disadvantages of this method are that it converges slowly, requires choosing a temperature function and tuning the associated parameters, and that it has no performance guarantee. The parameters that need to be chosen are the decay rate of temperature, the weight that is given to the penalty function and the amount of change that should be done on each mutation. We have partly avoided these choices by putting our focus on making the mutations itself as clever as possible. It is thus expected that our solution would work well in a simulated annealing setting that performs downhill-search only. Some work can still be performed on making better choices for these parameters. Because of the instability of the Drukval program, it is needed to check if new solutions fail or succeed, though.

Another method that was considered are genetic algorithms. The advantages of simulated annealing are that it is able to escape from local minima and that it works with a large set of parameters. It wasn't chosen because it needs a huge amount of evaluations per generation and needs to run for multiple generations, whereas for our problem evaluations are expensive. Also considered was using local optimization with or without multiple starting points. This method was not chosen, because the problem of finding a feasible solution was expected to be a difficult problem. Since our initial solutions end up with feasible solutions, this method might be considered a try as well in future work. The disadvantage of this method is that it is less flexible.

**The choice for the initial scenario and the mutation are important** In Section 4.3, three different methods of generating an initial scenario were compared to each other with the intention to get an idea of what kind of scenarios work well and to be used as an initial scenario for simulated annealing. Another reason for experimenting with initial scenarios is that simulated annealing has no performance guarantee, except that the final solution is at least as good as the initial one and that the solution generally improves over iterations. The centralized initial scenario reflects the old scenario, in which all heat would be produced by a central heat source. The greedy initial scenario tries to use as few heat sources as possible. The random scenario randomly enables or disables each of the CHPs. Contrary to what was expected the greedy one didn't perform well and resulted in both a lot of pressure and overproduction. The centralized one performed well, but the random initial scenario, which was originally considered to be too naive, performed best.

For the mutation, two approaches were observed to be successful. The smart mutation uses experimentally obtained domain knowledge to make a random decision within the set of mutations that seems reasonable. The adaptive mutation follows a more general idea that can also be applied in different problem domains. It tries a random mutation and if successful, it remembers the direction of the mutation. However if the solution doesn't improve or deteriorates, it reverses the direction of the mutation to seek the solution in the opposite direction. It is expected that such an approach would work in many other problem domain contexts as well. Both the smart and adaptive approaches gave better results than random mutations.

**Dealing with time can be done using a segmentation algorithm** To deal with time, the choice was made to use segmentation on the time series and to solve for the resulting segments individually. Three techniques from literature have been considered, but since our dataset consists of 288 elements only, a dynamic programming method was used that solves the problem in quadratic time and minimizes the variances within each segment.

## 5.2. FUTURE WORK

A lot of work has been done, but the work can still be extended in some ways.

### 5.2.1. PROBLEM ENHANCEMENTS

In this subsection, a look is taken how additional requirements and changes to the problem can be solved.

**Equal spread of heat production** In the scenarios the current solution comes up with, some greenhouses might be made to produce more heat than others. The greenhouse owners might want the solution to be such that it equalizes the heat production over the greenhouses. This can be done in a soft way by changing the evaluation function such that little used greenhouses yield a better score than overworked ones or in a hard way by putting quota on the minimum and maximum amount of heat a greenhouse is allowed to produce.

**Geothermal energy** Geothermal energy is another energy source that is more stable in price compared to CHP's. The only variable source is in the price of the electricity, which influences the pomp. The temperature that can be reached by such a geothermal heat source depends on how deep the geothermal sources are built. There is a trade-off between the depth of the geothermal plant and the costs. The deeper the plant is built, the higher the temperature, but the more expensive the costs will be. If these become the main source, it means that the costs are less volatile and that the same solution can be used for multiple days. The same program structure as in Figure 4.1 can still be used with multiple heat sources.

**Stability if a heat source stops working** A solution that can be applied if a heat source stops working is to calculate the loss(the amount of heat not produced by the heat source) and make this amount be generated by the private boilers of the greenhouses. This solution should be safe, because private boilers don't need the network.

### 5.2.2. MODEL ENHANCEMENTS

In this subsection, a look is taken at how the current model for assigning a score to a heat generation scenario can be improved. Improvements can be made to calculate the pressure more precisely or to include additional costs that were left out of scope for our model.

**More advanced Drukval model** The current Drukval model only looks at hydraulic factors, but ignores thermodynamic ones. It is not clear if the thermodynamic constraints can safely be ignored and the model might need to be changed to incorporate thermodynamic factors too.

**Variable costs** In the current cost model the costs per heat source are kept constant throughout the day. The costs that are used per heat source can be found in Table 2.1. Although the solution is made such that it can work with variable costs, it hasn't been tested, because these are difficult to model without more domain knowledge.

**Transportation costs** At the moment transportation costs aren't considered. In Chapter 2, it was shown that the transportation costs linearly depend on the quantity, which makes them comparable to the production. For an accurate estimation of the total costs these sources would have to be included, but they don't change the problem or solution method.

**Influence of buffers** At the moment buffers are not considered, but incorporating those would improve the solution. Research needs to be done on how best buffers can be added to the solution. The existing set-up can be used for emptying the buffer by treating them as yet another heat source with the restriction that they should always be filled beforehand or always be refilled afterwards. In the current clustered set-up, it is recommended to find the time segment that has the lowest cost and use that one for filling up the buffer. Since the prices are the same for everyone, it is expected that every greenhouse should fill their buffers at the same time.

**Delay in water streaming through pipes** At the moment, it is assumed that water flowing through the pipes will arrive at its destination instantly. In practice this is more complicated. Depending on the actual layout of the network it will take time for the current to get from one point to another. It is assumed the network is stable enough to deal with such fluctuations.

### 5.2.3. SOLUTION ENHANCEMENTS

In this subsection a look is taken at ways at improvements that can be made to the current solution as is.

**Don't always run Drukval** It seems that once the constraints have been solved, the simulated annealing algorithm is unlikely to reintroduce them. One might wonder if it's really necessary to run Drukval every iteration. Maybe it can be run once every 100 iterations to verify the solution is still in the feasible domain and speed up calculations.

**Transitions between time segments** At the moment each time segment is solved separately. The transitions between these clusters are very abrupt, but should be done more gradually. It is suggested to perform these transitions gradually. A simple way to do so is by linear interpolation For example, a transition can take place in the following way:

| Time | RoCa production | CHP1 production | CHP2 production | Total production |
|---|---|---|---|---|
| 0 | 100 | 150 | 0 | 250 |
| 5 | 75 | 75 | 100 | 250 |
| 10 | 50 | 0 | 200 | 250 |

It needs to be verified that such transitions don't cause overflows in the middle. If this turns out to be a problem, the safest way is to perform reductions in heat production, before the increase of other heat sources. In this case a transition would look like this:

| Time | RoCa production | CHP1 production | CHP2 production | Total production |
|---|---|---|---|---|
| 0 | 100 | 150 | 0 | 250 |
| 5 | 75 | 75 | 0 | 150 |
| 15 | 50 | 0 | 0 | 50 |
| 20 | 50 | 0 | 100 | 150 |
| 25 | 50 | 0 | 200 | 250 |

This latter scenario would have a brief temporary period of reduction in heat production and it is not clear if the greenhouses would mind.

**More flexible solution than segmentation** At the moment, time is handled by segmenting the demand and the scenario globally and every greenhouse will have to adjust its production at the same time. A solution that uses a separate segmentation per greenhouse might yield better results.

**Finding good model parameters** Because Drukval runs slowly, not a lot of time has been spend on optimizing the parameters of simulated annealing. Ideally these parameters should be optimized separately in combination of initial solution and mutation. There are also some additional parameters that could be varied like making the mutation step size smaller depending on the temperature function as defined in Section 4.2.

### 5.2.4. SOLVING THE PROBLEM BY ACTIVE LEARNING

As an alternative to simulated annealing this problem could be solved using Machine Learning. This machine learning method would model the result of the evaluations and based on that model it would build a new solution. To get a model that is easy to optimize, the model would ideally be linear or quadratic, but something convex would work as well. The advantage of linear and quadratic problems is that there exist state of the art solutions for solving these problems rapidly in theory [17] and in practice [18]. Optimization of linear models has been done before using Lasso and decision tree models [3]. Convex optimization problems can be solved using Subgradient projection and Interior point methods.

### 5.2.5. ACTIVE LEARNING OVERVIEW

A high-level description of how this problem can be solved using active learning is depicted in Figure 5.1. This description leaves several details open.

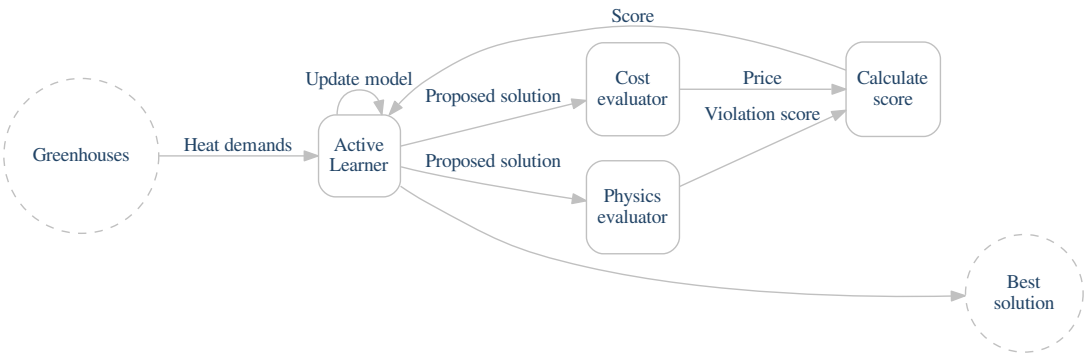



Figure 5.1: Machine learning program structure

As with simulated annealing, the input is the heat demands per greenhouse per time interval and the output is the best heat production scenario. The system has access to a cost and a physics evaluator to evaluate

proposed solutions. Internally, the system tries to build a model that can predict the outcome of the physical evaluator but that lends itself better for optimization. It does this by first generating instances to build its model and then using that model to build instances that it predicts will test positively on the physical evaluator.

The production costs are mostly linear, so the physical model is the thing that needs modelling. From the heat production scenario, the debits can be calculated, whose instances are in the domain $\mathbb{N}^{288}$, because there are 288 periods of 5 minutes in a day. From personal experience, I know machine learning works best when there is either a lot of data or when the number of features is small. 288 features is large, but perhaps feature reduction could be used, by taking groups of related greenhouses together at the cost of generalizing. Fortunately, the number of instances can be reduced.

**Number of models** We could either learn a single model that predicts if a complete scenario is feasible or we could train a model per greenhouse. If we train a single model that model will label a scenario (debits per greenhouse) either as feasible or infeasible. If only a single greenhouse fails to meet the constraints, that single model should mark the scenario as infeasible. Only if every single greenhouse meets the constraints, the single model should predict the scenario as feasible. As failing constrains can always be traced back to a single greenhouse, an alternative approach is to train a model per greenhouse. As input every model will get a full scenario with the debit values for every single greenhouse, however as output it will only predict if the constraints of a single greenhouse are met. Expected is that training will be faster in this case, as infeasible solutions can be traced back to a single greenhouse and only the model for that single greenhouse needs to be updated.

### 5.2.6. OPTIMIZING THE LEARNED MODEL

**Linear and quadratic programming problems** A linear programming problem (LP) is a problem of the following basic form.

$$\begin{aligned} &\text{Minimize } \vec{c}^{\top}\vec{x} \\ &\text{Subject to } A\vec{x} \leq b \text{ for } i = 1 \cdots m \\ &\qquad \vec{x} \geq 0 \text{ for } i = 1 \ldots p \\ &\text{Where } \vec{x} \text{ is a free variable} \\ &\text{Where } A, \vec{b} \text{ and } \vec{c} \text{ are fixed} \end{aligned}$$

It has a linear objective function and linear constraints. The vector $c = (c_1 \ldots c_n) \in R^n$ needs to be supplied in advance. The solution $\vec{x} = (x_1 \ldots x_n) \in \mathbb{R}^n$ needs to be found such that the objective is as small as possible while satisfying all constraints.

There are a couple of extensions to the linear programming problem. If some of the solutions of $x$ lie in the integer domain $\mathbb{Z}^+$ it's an integer linear problem. This problem is NP-hard, but solvers such as CPLEX are specialised in solving many such problems efficiently. In other extensions the objective can be quadratic, making the problem a Quadratic Programming Problem or the problem might have quadratic constraints in which case it becomes a Quadratically Constrained Programming Problem. An even more general problem class is the Mixed Integer Programming class, in which the objectives and constraints of a problem can be linear or quadratic, any of the variables can be constrained to the real of integral domain and constructs such as Special Ordered Sets and semi-continuous variables can be included. [18] Some tools that can can solve all these problem classes are CPLEX [18] and Gurobi [19].

**Linearizing the physical constraints** When optimizing over scenarios to find a cost efficient scenario, we would like to specify the physical limitations as linear or quadratic constraints such that the problem is formulated as a MIP and the efficiency of CPLEX can be utilized. Unfortunately these physical limitations aren't linear, but perhaps they can be approximated by a linear or quadratic model if that simplified model predicts the physical feasibility accurately enough. An attempt is made to find such a linear model by machine learning.

### 5.2.7. MACHINE LEARNING MODEL

**Linear classifiers** The easiest type of models that is solvable in polynomial time are linear models. Linear models are simple and easy to optimize, but aren't the most accurate solution because they ignore interactions. If the current physical model (or an improved one that is still based on formulas) is used, we expect not to find noise or faults in the data, so it is expected that simple methods like LDA and LASSO could work. Otherwise, if it is decided to train on real world data, an advanced method like SVM could be used that is more robust to noise while still trying to be as accurate as possible.

**Decision tree** Decision trees can be used for both classification in which case they are called classification trees and for regression in which case they are called regression trees. As regressors can be turned into classifiers by thresholding, we can consider these regression methods to be classifiers as well. In its simplest setting, at each branch of a decision tree a single feature and a threshold is used to advance to one of the subtrees. The leafs of a decision tree contain labels (classification) or values (regression). Unlike more advanced techniques such as neural networks models created by decision trees are relatively easy to understand. A popular method for training regression trees is M5 proposed by J.R. Quilan in 1992. [20] Instead of labels, the leafs consist of multivariate linear models and therefore this tree based learner is called a model tree. Also pruning is used for simplifying the tree and smoothing to resolve discontinuities. Wang and Witten made some improvements to this M5 model like improved dealing with missing values. Frank, Wang, Inglis, Holmes and Witten turn the M5 model into a classifier. [21] As a model tree in its simplest form is a combination of a decision tree and a linear classifier, it is likely that this model can be turned into a set of MIP constraints, but it's not clear a this point if that will improve the modelling. Landwehr, Hall and Eibe Frank use a model tree with a logistic classifier at the leafs and claim that this improves the modelling. [22] As shown by Sicco e.a., decision trees can be optimized by mapping it to an MIP problem and using a tool such as CPLEX. [3]

### 5.2.8. INSTANCE GENERATION

There is something to say for generating instances purely randomly. It provides the classifier with the most unbiased sample of the data, whereas all other models favour certain instances more than others. However having an unbiased distribution isn't what we are interested in. As we would rather have a biased model that works well for solutions near the optimal one, than an unbiased model that performs well for all instances. less optimal solutions.

**Active learning inspired methods** In a literature survey by Burr Settles [23], different query strategies for active learning are described. The methods discussed in his survey intend to make the classifier work well on the whole distribution. *Uncertainty sampling* chooses the instances the classifier is most uncertain about how to label. There is also a variation, query-by-committee that uses multiple classifiers and selects the label the classifiers agree on the least.

More expensive methods are *Expected model change* and *Expected error reduction. Expected model change* selects the instance that would change the model the most. *Expected error reduction* tries to reduce the generalization error, the error on unseen instances the most. This requires the classifier to be retrained at every step. Because *expected error reduction* is expensive, *Variance reduction* can be used which is about selecting instances that reduce the variance most and indirectly reducing the generalization error.

An advanced method is *Density weighted reduction*, where instances are selected that are both uncertain as well as those that are representative for the underlying distribution.

**Contextual bandit** Another class of instance generation methods is *Contextual bandit*, which tries to balance between exploitation (trying to find a best solution in the area explored so far) and exploration (hunting the instance space for more promising areas). This method comes from recommender systems, another field of research, where the goal is partly to exploit what is known about the user preferences and partly to come up with new suggestions which the user might like. Standard $\epsilon$-greedy chooses with probability $\epsilon$ the best solution according to the current model of the user preferences, and with probability $(1-\epsilon)$ a random policy. [24] In recommender systems it is sometimes needed to keep the variable $\epsilon$ high to not lose the interest of the customer, but for us this is not as relevant.

**Q-learning methods** Q-learning methods are used in reinforcement learning, which studies how software agents should behave in an environment where they can perform actions and obtain rewards. Q-learning methods are about choosing the best action in such situations.

Two common exploitation strategies often used in Q-learning are semi-uniform learning and Boltzman exploration. Semi-uniform learning is very similar to the contextual bandit method $\epsilon$-greedy. With probably $p$ the best solution is selected and with probability $1-p$ a random solution is selected. Initially $p$ is quite low and is gradually increased. So initially the algorithm is very explorative and slowly shifts into an exploitative one. Another approach is Boltzman exploration that select actions with probability:

$$Pr(a) = \frac{e^{Q(s,a)/T}}{\sum_{a'} e^{Q(s,a')/T}} \tag{5.1}$$

The variable $T$ stands for temperature and is initially assigned a high value, but is slowly decreased to transition from exploration to exploitation. This is conceptually similar to the temperature function for Simulated Annealing used in Chapter 4. Q-learning itself can not be directly applied to our problem, because the solution space of our problem is enormous, whereas Q-learning works with a small, discrete set of actions.

### 5.2.9. DIFFERENCES WITH SIMULATED ANNEALING

Both active learning (actually a combination of active learning and optimization) and simulated annealing require multiple evaluations of the solution. The advantage of simulated annealing over active learning is that it is easier to set up. Active learning requires training a general model first and then applying that model to an individual day.

Simulated annealing can do without such a model and can be run directly. The difference is efficiency, the active learning model only needs to be learned once which will take long if an accurate model is required, but after having learned it once, it can be applied over and over again to efficiently generate a solution. Simulated annealing takes longer to generate an individual solution, but doesn't require a long initial learning process. Since this initial learning only needs to be done once, active learning would be better. The initial learning phase only needs to be done again if the underlying model changes.

Whether or not active learning will find a better solution remains unclear. Solution methods like simplex are definitely better than simulated annealing at finding global minima in complex solution spaces; however, the simplex method assumes the linear model is completely accurate, which might not be the case because the active learning model might be oversimplifying the problem and might even come up with a solution that is not actually feasible. Simulated annealing on the other hand would, given enough time, effectively do a brute-force search, which is guaranteed to find the best solution.

It would also be possible to combine them. Active learning could be used to come up with an initial solution, which can then in turn be fine-tuned using simulated annealing.

## 5.3. RECOMMENDATIONS TO ENECO

It has been shown that by changing from a centralized to a decentralized approach the costs can be decreased by about 25 %. Potentially more costs can be saved if the costs are made variable and the solution is extended to include buffers for which some suggestion are made in Section 5.2.2 In either case, the solution can only improve, because even if the solution has worse performance, there is always the option to use the current one.

The Drukval program from Eneco has been used for the evaluation of physical constraints. However, this program is still a bit unstable, as it would freeze or even segfault on some scenarios. This makes it difficult to use the program as part of an optimization routine and considerably slows down the evaluation time. Furthermore, the program only pays attention to the hydraulic physical laws to calculate the pressure but neglects the thermic ones. It is not certain that the thermic pressure can safely be ignored. It is recommended to stabilize the program and to make it work with a modern compiler or to make the program available as a shared library that can be used in any environment.

If the recommendations in this thesis are followed, it is expected that solutions can be created that are better than could be found by hand. Eneco and the greenhouses need to consider what additional requirements they have to the system. Solutions might be biased to make use of certain heat sources more often than others, but the system can be extended with the requirement to equalize the production over the greenhouses or to split the costs by compensating greenhouses that need to produce more. Since the total costs decrease, an agreement can and should be made such that nobody has to lose and everybody wins.

# BIBLIOGRAPHY


[1] S. B. Mes, *Warmtedistributie deel 1. algemene warmtedistributietechniek,* (2001).

[2] KNMI, *Daggegevens van het weer in nederland,* http://www.knmi.nl/nederland-nu/klimatologie/daggegevens (2015).

[3] S. Verwer, Y. Zhang, and Q. C. Ye, *Auction optimization with models learned from data,* (2014), unempty.

[4] M. van den Ende, *The Decision Problem Of The Development Of A Smart Thermal Grid*, Master's thesis, Delft University of Technology, the Netherlands (2013).

[5] S. Farahani, *Final Report for the Warmteweb Project*, Tech. Rep. (TU Delft, 2014).

[6] H. Chernoff and E. L. Lehmann, *The use of maximum likelihood estimates in $\chi^2$ tests for goodness of fit,* Ann. Math. Statist. **25**, 579 (1954).

[7] T. van den Boom and B. de Schutter, *Lecture Notes SC 4091. Optimization in Systems and Control* (Delft Center for System and Control, Delft University of Technology, 2012) pp. 75–78.

[8] M. J. D. Powell, *An efficient method for finding the minimum of a function of several variables without calculating derivatives,* **7**, 155 (1964).

[9] J. A. Nelder and R. Mead, *A simplex method for function minimization,* **7**, 308 (1965).

[10] A. Khachaturyan, S. Semenovsovskaya, and B. Vainshtein, *The thermodynamic approach to the structure analysis of crystals,* Acta Crystallographica Section A **37**, 742 (1981).

[11] C. M. Fonseca, P. J. Fleming, *et al.*, *Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization.* in *ICGA*, Vol. 93 (Citeseer, 1993) pp. 416–423.

[12] Ö. Yeniay, *Penalty function methods for constrained optimization with genetic algorithms,* Mathematical and Computational Applications **10**, 45 (2005).

[13] W. L. Goffe, G. D. Ferrier, and J. Rogers, *Global optimization of statistical functions with simulated annealing,* Journal of Econometrics **60**, 65 (1994).

[14] H. Szu and R. Hartley, *Fast simulated annealing,* Physics letters A **122**, 157 (1987).

[15] C. Ding and X. He, *Cluster merging and splitting in hierarchical clustering algorithms,* in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on* (IEEE, 2002) pp. 139–146.

[16] L. Junkui and W. Yuanzhen, *Apcas: An approximate approach to adaptively segment time series stream,* in *Advances in Data and Web Management*, Lecture Notes in Computer Science, Vol. 4505, edited by G. Dong, X. Lin, W. Wang, Y. Yang, and J. Yu (Springer Berlin Heidelberg, 2007) pp. 554–565.

[17] M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization,* Combinatorica **1**, 169 (1981).

[18] IBM, *Ibm ilog cplex optimization studio. getting started with cplex,* (2009).

[19] G. Optimization, *Gurobi optimizer 5.0,* (2014).

[20] J. R. Quinlan *et al.*, *Learning with continuous classes,* in *Proceedings of the 5th Australian joint Conference on Artificial Intelligence*, Vol. 92 (Singapore, 1992) pp. 343–348.

[21] E. Frank, Y. Wang, S. Inglis, G. Holmes, and I. H. Witten, *Using model trees for classification,* Machine Learning **32**, 63 (1998).

[22] N. Landwehr, M. Hall, and E. Frank, *Logistic model trees,* Machine Learning **59**, 161 (2005).

[23] B. Settles, *Active Learning Literature Survey*, Computer Sciences Technical Report 1648 (University of Wisconsin–Madison, 2009).

[24] K. Hofmann, S. Whiteson, and M. de Rijke, *Contextual bandits for information retrieval,* in *NIPS 2011 Workshop* (2011).

# A

# RAW DATA

The raw results of running the experiments without clustering can be found in Table A.1. The raw results of running the experiments with clustering can be found in Table A.2.

| | | Centralized | Greedy | Random |
|---|---|---|---|---|
| No mutation | 2013-01-0 | 600.957 | 905.828 | 536.539 |
| | 2013-01-1 | 614.249 | 2.1e+08 | 548.845 |
| | 2013-01-2 | 552.139 | 874.586 | 487.819 |
| | 2013-01-3 | 559.900 | 882.235 | 494.320 |
| | 2013-01-4 | 575.626 | 890.481 | 511.477 |
| | 2013-01-5 | 569.589 | 883.771 | 499.604 |
| | 2013-06-0 | 476.346 | 858.102 | 428.180 |
| | 2013-06-1 | 492.982 | 853.860 | 427.125 |
| | 2013-06-2 | 502.175 | 864.736 | 441.255 |
| | 2013-06-3 | 438.938 | 833.732 | 387.080 |
| | 2013-06-4 | 421.535 | 820.217 | 368.972 |
| | 2013-06-5 | 409.344 | 978.585 | 357.246 |
| Random mutation | 2013-01-0 | 600.957 | 905.828 | 536.539 |
| | 2013-01-1 | 614.249 | 1.36e+03 | 548.845 |
| | 2013-01-2 | 552.139 | 874.586 | 487.819 |
| | 2013-01-3 | 559.900 | 882.235 | 494.320 |
| | 2013-01-4 | 575.626 | 890.481 | 511.477 |
| | 2013-01-5 | 569.589 | 883.771 | 499.604 |
| | 2013-06-0 | 476.346 | 858.102 | 428.180 |
| | 2013-06-1 | 492.982 | 853.860 | 427.125 |
| | 2013-06-2 | 502.175 | 864.736 | 441.255 |
| | 2013-06-3 | 438.938 | 833.732 | 387.080 |
| | 2013-06-4 | 421.535 | 820.217 | 368.972 |
| | 2013-06-5 | 409.344 | 978.585 | 357.246 |
| Adaptive mutation | 2013-01-0 | 489.538 | 586.398 | 473.975 |
| | 2013-01-1 | 492.864 | 468.762 | 485.248 |
| | 2013-01-2 | 457.641 | 551.756 | 454.032 |
| | 2013-01-3 | 464.053 | 557.635 | 457.122 |
| | 2013-01-4 | 473.861 | 567.539 | 454.371 |
| | 2013-01-5 | 469.884 | 560.451 | 456.480 |
| | 2013-06-0 | 403.182 | 551.334 | 408.755 |
| | 2013-06-1 | 404.049 | 546.970 | 412.676 |
| | 2013-06-2 | 412.170 | 558.927 | 414.574 |
| | 2013-06-3 | 378.891 | 527.031 | 382.183 |
| | 2013-06-4 | 366.148 | 512.679 | 368.091 |
| | 2013-06-5 | 358.843 | 574.959 | 354.895 |
| Smart mutation | 2013-01-0 | 394.634 | 688.396 | 399.314 |
| | 2013-01-1 | 413.596 | 2.05e+08 | 397.121 |
| | 2013-01-2 | 366.923 | 658.114 | 353.462 |
| | 2013-01-3 | 371.390 | 709.149 | 358.286 |
| | 2013-01-4 | 380.582 | 687.738 | 383.892 |
| | 2013-01-5 | 365.582 | 711.895 | 360.771 |
| | 2013-06-0 | 342.721 | 758.589 | 348.081 |
| | 2013-06-1 | 350.268 | 720.368 | 349.274 |
| | 2013-06-2 | 357.024 | 765.270 | 350.810 |
| | 2013-06-3 | 320.418 | 627.462 | 315.007 |
| | 2013-06-4 | 298.396 | 668.088 | 304.520 |
| | 2013-06-5 | 291.428 | 634.463 | 293.815 |

Table A.1: Raw data

| | | Centralized | Greedy | Random |
|---|---|---|---|---|
| No mutation | 2013-01-0 | 471.751 | 914.094 | 421.415 |
| | 2013-01-1 | 488.164 | 902.023 | 430.456 |
| | 2013-01-2 | 416.118 | 940.329 | 368.001 |
| | 2013-01-3 | 427.168 | 932.861 | 376.314 |
| | 2013-01-4 | 426.062 | 935.973 | 376.133 |
| | 2013-01-5 | 431.798 | 888.190 | 381.492 |
| | 2013-06-0 | 355.391 | 905.807 | 318.671 |
| | 2013-06-1 | 355.250 | 954.693 | 302.454 |
| | 2013-06-2 | 397.189 | 906.419 | 346.188 |
| | 2013-06-3 | 314.429 | 932.891 | 272.619 |
| | 2013-06-4 | 264.824 | 967.999 | inf |
| | 2013-06-5 | 242.982 | 971.733 | 205.045 |
| Random mutation | 2013-01-0 | 471.751 | 914.094 | 421.415 |
| | 2013-01-1 | 488.164 | 902.023 | 430.456 |
| | 2013-01-2 | 416.118 | 940.329 | 368.001 |
| | 2013-01-3 | 427.168 | 932.861 | 376.314 |
| | 2013-01-4 | 426.062 | 935.973 | 376.133 |
| | 2013-01-5 | 431.798 | 888.190 | 381.492 |
| | 2013-06-0 | 355.391 | 905.807 | 318.671 |
| | 2013-06-1 | 355.250 | 954.693 | 302.454 |
| | 2013-06-2 | 397.189 | 906.419 | 346.188 |
| | 2013-06-3 | 314.429 | 932.891 | 272.619 |
| | 2013-06-4 | 264.824 | 967.999 | inf |
| | 2013-06-5 | 242.982 | 971.733 | 205.045 |
| Adaptive mutation | 2013-01-0 | 356.395 | 631.441 | 353.974 |
| | 2013-01-1 | 400.060 | 662.368 | 420.510 |
| | 2013-01-2 | 317.215 | 670.652 | 308.112 |
| | 2013-01-3 | 348.618 | 777.609 | 307.516 |
| | 2013-01-4 | 322.537 | 681.740 | 309.821 |
| | 2013-01-5 | 329.133 | 712.320 | 310.942 |
| | 2013-06-0 | 325.687 | 687.750 | 273.960 |
| | 2013-06-1 | 280.572 | 791.127 | 253.375 |
| | 2013-06-2 | 302.966 | 632.037 | 299.282 |
| | 2013-06-3 | 241.910 | 697.439 | 240.604 |
| | 2013-06-4 | 201.497 | 750.075 | inf |
| | 2013-06-5 | 195.745 | 734.642 | 190.468 |
| Smart mutation | 2013-01-0 | 405.551 | 671.389 | 374.979 |
| | 2013-01-1 | 446.506 | 761.145 | 402.054 |
| | 2013-01-2 | 368.604 | 762.988 | 333.706 |
| | 2013-01-3 | 380.057 | 782.382 | 337.752 |
| | 2013-01-4 | 374.791 | 781.581 | 336.271 |
| | 2013-01-5 | 377.370 | 724.745 | 339.772 |
| | 2013-06-0 | 293.859 | 702.284 | 264.635 |
| | 2013-06-1 | 322.907 | 805.617 | 278.635 |
| | 2013-06-2 | 359.999 | 733.470 | 320.067 |
| | 2013-06-3 | 268.103 | 716.855 | 237.913 |
| | 2013-06-4 | 196.562 | 711.312 | inf |
| | 2013-06-5 | 179.979 | 645.650 | 175.309 |

Table A.2: Raw data clustered