Reconfiguring Electricity Distribution Networks to Minimize Power Loss

NP-hardness and efficient approximation algorithms

Radboud University

Liander N.V.

Author:
Michiel van der Meulen BSc

Supervisors:
dr. Wieb Bosma (RU)
drs. ir. Robin Hagemans (Liander)
ir. Werner van Westering (Liander)

August 20, 2015

Dutch Summary

Dit rapport hoort bij het afstudeer onderzoek dat is uitgevoerd voor de Master Mathematics aan de Radboud Universiteit, in opdracht van Liander NV. Het doel van deze thesis was om een efficiënt, wiskundig algoritme te ontwikkelen dat voor een middenspanningsnetwerk bepaald hoe deze geschakeld moet worden zodat het vermogensverlies in dit netwerk minimaal is. Hierbij wordt onder vermogensverlies verstaan het verlies door de kabels. In dit rapport is een wiskundige definiëring van dit minimalisatieprobleem gegeven, evenals een bewijs van de NP-hardheid van dit probleem. Dit impliceert dat het onwaarschijnlijk is dat er een efficiënt algoritme voor dit probleem bestaat, als gevolg van het "$P \neq NP$" vermoeden.

Sinds 1975 met Merlin & Back als pioniers, zijn er echter tal van efficiënte benaderings algoritmes voor dit probleem ontwikkeld. Er is geen garantie dat een dergelijk algoritme voor een netwerk de beste schakeling teruggeeft, maar er zal wel een goede schakeling teruggegeven worden. Voor deze thesis is een uigebreide literatuurstudie uitgevoerd, waarna een aantal van de ontwikkelde benaderings algoritmes zijn geselecteerd voor verder onderzoek. Dit zijn het *Greedy-Shifting algoritme* van Baran & Wu, het *Greedy-Demeshing algoritme* van Shirmohammadi & Hong, een toepassing van het *Harmony Search Algoritme* gebaseerd op een artikel van Rao et al. en een toepassing van *Mixed-Integer Linear Programming* door Jabr, Singh & Pal. Ook is een nieuw ontwikkelde versie van het *Genetisch algoritme* toegevoegd, evenals een *Brute Force Calculation*, waarin simpelweg alle mogelijke schakelingen worden nagegaan. Deze algoritmes zijn geïmplementeerd in "R", getest op de middenspanningsnetwerken van Texel, Zaltbommel en Zaandam (met uitzondering van het Greedy-Shifting algoritme, aangezien een soortgelijke variant hiervan beschikbaar is in "Vision"), en de resultaten zijn met elkaar vergeleken. Ook zijn het Harmony Search Algoritme en het Genetische Algoritme gecombineerd met het Greedy-Demeshing algoritme, door de uitkomst van de laatste toe te voegen aan de startschakelingen van de eersten. Deze combinaties zijn evenwel getest en vergeleken met elkaar en met de losse algoritmes. Uit de resultaten blijkt dat de combinatie van het Greedy-Demeshing algoritme met het Genetisch algoritme het beste presteert. Dit zal daarom de basis vormen voor de definitieve methode.

Vrijwel alle benaderings algoritmes die te vinden zijn in de literatuur, zijn slechts geschikt voor middenspanningsnetwerken met een enkele aansluiting op een hoogspanningsnetwerk, inclusief het Greedy-Demshing algoritme en het Genetisch Algoritme. Het is echter wenselijk dat de definitieve methode ook geschikt is voor netwerken met meerdere hoogspanning aansluitingen, zodat deze breder inzetbaar is. Daarom is een innovatieve, algemene methode ontwikkeld die een netwerk met meerdere hoogspanning aansluitingen aanpast zodat ieder ontwikkelde benaderings algoritme gebruikt kan worden om ook in dit netwerk de optimale schakeling te benaderen. Deze methode wordt beschreven in dit rapport en is opgenomen in de definitieve methode.

De uitkomsten van de voorgestelde definitieve methode op Texel en Zaandam zijn vergeleken met de huidige schakeling van de corresponderende middenspanningsnetwerken. Met als resultaat dat in Texel een vermogensverlies reductie van 15% gerealiseerd wordt met de voorgestelde schakeling ten opzichte van de huidige, en in Zaandam zelfs 27%. Nu moeten

deze cijfers wat genuanceerd worden, aangezien de algoritmes zoals ze nu geïmplementeerd zijn, slechts in staat zijn om de beste schakeling voor een bepaald moment te benaderen. Aanpassingen zodat de beste schakeling over een periode (dag, seizoen, jaar) benaderd kan worden zullen nog moeten gebeuren, maar dan zullen deze percentages wellicht niet meer gehaald worden. Desalniettemin zijn dit veelbelovende resultaten, die laten zien dat de voorgestelde definitieve methode een goede basis is voor een praktisch toepasbaar programma ter ondersteuning van netwerkplanning activiteiten.

Abstract

The subject of this master Mathematics graduation thesis is finding and proving a mathematical method for minimizing the power loss in Liander medium-voltage distribution networks, by reconfiguring such a network. This minimization problem is defined mathematically as an optimization problem, and a proof is found and given which shows that this problem is NP-hard. As a consequence, it is unlikely that an efficient algorithm could be developed that actually minimizes the power loss in a network. However, many approximation algorithms for this problem have been developed that can efficiently approximate an optimal solution for such a network. In this report, several of these algorithms are described. These algorithms were investigated, improved and/or adjusted at some aspects, implemented in "R" and tested on real Dutch networks such as Texel, Zaltbommel and Zaandam. Results are included in this report as well. Based on these results, a final method for approximating an optimal solution for a certain network is proposed, which combines the *Greedy-Demeshing algorithm* of Shirmohammadi & Hong with a novel application of the *Genetic Algorithm* to this problem. Moreover, many of the developed approximation algorithms found in the literature are only suitable for medium-voltage networks with a single connection to a high-voltage network. In this thesis, an innovative and general method is proposed that adapts any network with multiple high-voltage connections such that any developed appoximation algorithm can be used to approximate a solution for this network as well. This method is adopted in the final method. At last, a comparison of the output of the final method with the actual situation was executed, which resulted in a 15% power loss reduction in Texel, and 27% in Zaandam. It is noted that these results are not definite, but they show that the end product of this thesis gives a promising basis for a practical program that can support network planning activities.

Contents

Chapter 1

Introduction

Liander NV is a regional Dutch utility company that operates in the distribution of electricity and gas in the Netherlands. It is responsible for keeping the distribution networks in a well-working state of operation. In the past, the electricity distribution network was a simple, one-way network, where power was generated in a controllable way at certain power plants, and via the network this power was distributed to the clients. However, today the electricity network tends to become more and more complex because of numerous technological developments. This is due to for instance distributed generation; solar panels, CHP units etc. installed at consumers, making it possible that power is delivered instead of received at certain points in the net, resulting in a reversed current. An other example is the growing use of natural sources such as wind- and solar energy, which are not controllable and introduce huge volatility in the grid.

Due to these developments, the management of the distribution network shows more and a high variety of difficulties compared to the past. One specific example of this is network planning activities; deciding which configuration of (a part of) the network is most suitable for a specific situation, taking into account several factors such as robustness of the network, load balancing over the assets and power loss through the cables. The number of possible configurations can grow to hundreds of thousands for medium sized networks, and even more for the bigger ones. Due to the growing complexity of the distribution networks, these planning activities show much more difficulties than in the past. Today however, planning decisions still are made by the network architects at the Liander control room, based on their experience and intuition. A supportive computer program that can be used to get good initial configuration suggestions or to base decisions on would be very helpful, but is not yet available. The aim of this research project is to investigate different possibilities for the mathematical algorithms of such a program, to compare the performances of these algorithms, and finally to do a well-founded suggestion for an efficient algorithm to base such a program on.

Another motivation for the development of such a program is with a view to the future. Nowadays, actually reconfiguring a network is a complex, time consuming activity, since switches in the network have to be switched on location by professional engineers. However, it is possible that at some point in the future all these switches can be switched remotely, from one central point. This makes reconfiguring a network much easier, and therefore it could be beneficial to reconfigurate the network any time the situation in the network changes. For

instance, if the sun starts to shine on the solar panels in a part of the Liander area, this results in a new power flow pattern, probably inducing a new optimal configuration. This could lead to several reconfigurations in one day, which is not appropriate to plan manually. Then a supportive computer program is needed, to quickly get suitable configuration suggestions for every new situation. This project aims to develop a good basis for such a program.

This research project has been performed as a M.Sc. Mathematics graduation project at the Radboud University in Nijmegen, in corporation with Liander NV. In order to mark out the scope of this project, we focused on reconfiguring an electricity distribution network to minimize power losses through the cables. This objective was chosen for the aspect of sustainability, but also since this generates an unnecessary big expense for Liander NV.

In this chapter, a summary will be given of the electricity networks in the Netherlands in Section 1.1. After this, the problem and the research questions can and will be specified in Section 1.2, and then a quick overview will be given of the rest of this report.

1.1 Electricity networks in the Netherlands

The electricity distribution networks are split in three categories, based on the voltage of the electricity in the cables. From 110 kilovolt (kV) to 380kV is labelled 'high voltage' (HV). Medium voltage (MV) covers voltages between 10 and 20kV, and any network part with voltage up to 400V is labelled low voltage (LV). This covers all the present voltages in the distribution networks.

Transporting a certain amount of power $P = \Delta U \cdot I$ over a distance through a cable with a certain resistance R (where ΔU is the voltage difference between the end points, I is the current through the cable; see Figure 1.1), the energy loss is $P_{\text{loss}} = I^2 R$. So if the voltage difference is increased, a lower current is needed for transporting the same amount of power, hence the loss is lowered. On the other hand, due to consumption amounts and safety reasons, high voltage electricity is normally impractical for small consumers. This justifies the existence of these different categories; HV for long-range transportation over the country, MV for finer distribution, and LV for the finest distribution to households and small consumers. However, exceptions in this model are not uncommon, for instance when big industrial companies are connected immediately to the MV network, or when households generate power into the network with their solar panels.

Quantity	Abreviation	Unit	Abreviation
Voltage	U	Volt	V
Current	I	Ampére	A
Power	P	Watt	W
Resistance	R	Ohm	Ω

Figure 1.1: *Quantities and units in electrical engineering*

Different voltage level networks are connected via transformers. Transformers can change the voltage level of electricity. The switch from HV to MV, or from MV to LV, or voltages within

a level, is performed in stations, in size varying from a wardrobe to a whole football field filled with transformers. To give an indication, in the operating area of Liander there are around 350 HV/MV stations, with circa 45.000 MV/LV connections, and these contain over three million connections to the consumers ([29]).

The different voltage level networks are structured in different, specific ways. The HV network has a meshed structure, which means that it contains cycles, closed rings. Now when a fault occurs, it automatically gets isolated, and then no interruption of the power transport occurs. On the other hand, the LV networks mostly have radial structures (i.e. are forests in graph theoretical terms, see [1]). Although a meshed network structure would result in lower network losses, a radial structure is preferable since disturbances, which occur far more in LV than HV networks, can be found quicker in this topology, and short-circuit currents will be lower.

In an MV network, the situation is more complicated. The underlying structure of this network is meshed, but a radial structure is preferable for the same reasons as in LV. Therefore, several cables and links in this network are equipped with switches that can be opened to shut off the cable/link. Hence with a certain specification for the switches to be opened or closed, a radial structure can be accomplished. The resulting network from such a switch specification is called a *configuration*. The number of switches in a network is high, leading to a high variety of radial (and non-radial) configurations for one MV network.

The main reason why an MV network is build up this way is that when a fault occurs, by reconfiguring the network this fault can be isolated and power supply can be restored to all/many consumers without actually needing to repair the fault. Now when a fault occurs, this restoration will always be priority one. However, when an MV network is in a well operating state, one could reconfigure the network for other objectives. As may be clear, this project focusses on this last situation.

A complete and detailed description of the Liander distribution network can be found in [2].

1.2 Research Questions and Report Overview

We can now define the central problem in this project:

Which feasible configuration of an MV network minimizes the power loss through the cables?

We will call this problem the *Loss Reduction Reconfiguration Problem* (LRRP), and it will be stated more precise and in a mathematical way in chapter 3. For now, we will give a quick overview of this problem, and then state the research questions in this project.

For a network, a configuration is feasible if it satisfies three constraints: radiality, the demand constraint and the capacity constraint. These constraints are the mathematical representation of the conditions that a configuration must satisfy according to **??**, and they will now be described.

The topology of a configuration has the following requirements:

- Every node in the network must be connected to an HV/MV transformer, in order to be able to gain power from, or withdraw power to the HV network.
- No cycles (closed rings, see [1]) are allowed in the network. This makes it easier to allocate and restore a disturbance. Also, when a fault occurs in a cycle, the short-circuit current will be higher. Hence this requirement is imposed also for safety reasons.
- HV/MV transformers are not allowed to be connected to each other. Different transformers constitute different voltage and current frequencies, so when these are connected, an unstable voltage and current will arise as a consequence.

If a configuration meets these requirements, it satisfies the *radiality* constraint.

In reality, the HV/MV transformers in an MV network admit a specific voltage. For the other busses (nodes), a certain power demand or supply occurs at these busses. As a consequence of these values, the power distributes in a specific way through the network, with certain voltages at the nodes and certain currents in the lines. This behavior can be computed by so called *power flow equations*, which will be explained in detail in Chapter 2. Since we need these voltages and currents for the calculation of the power losses, we have to add a constraint to the problem that simulates this distribution. Hence we define the *demand constraint*, which states that the nodal voltages in the configuration satisfy the power flow equations. With this, the line currents can be calculated by Ohm's law (2.3).

Another reason for the need of the nodal voltages and line currents is that the assets have certain capacities. Therefore, the *capacity constraint* is added to check whether the assets can handle the accompanying voltages and currents. If not, the configuration is not suitable for the situation, hence should no longer be considered.

As will be described in Chapter 2, the power loss through a line e with impedance $Z_e = R_e + jX_e$ and current I_e can be calculated as $L(e) = |I_e|^2 \cdot R_e$. So we can state LRRP more precise as:

Which configuration of an MV network that satisfies the radiality, demand, and capacity constraints, minimizes the sum of the line losses $L(e)$ over all lines e of the network?

As mentioned earlier, LRRP will be described mathematically in Chapter 3. However, the main research question of this graduation project can now be expressed:

Can we find an efficient algorithm that solves LRRP?

Here 'efficient' means 'polynomial-time', i.e. for such an algorithm A a polynomial P must exist such that for an input of size s, the algorithm needs at most $P(s)$ steps to get an output. See **??**.

The problem shows some similarity with the Minimum Weight Spanning Tree problem (MWST), where for a weighted graph, a spanning tree is asked in which the total weight is minimized.

This problem is efficiently solvable, meaning there exist fast algorithms that solve this problem for any weighted graph. However, in LRRP the weights of the edges are not constant, but in contrast to MWST they depend on the configuration that is considered. This makes the problem far more complicated. Indeed, as we will see, LRRP is an NP-hard problem, making it unlikely that an efficient algorithm for LRRP exists.

However, interesting results remain to be achieved. During the years, a high variety of approximation algorithms were derived for LRRP. Many publications can be found about this subject, each claiming to have better algorithms than others. With this in mind, the following questions guided the project and were attempted to be answered:

1. Is LRRP efficiently solvable?
2. Can we find/develop fast and good (approximation) algorithms for LRRP on realistic sized networks?
3. Which algorithm is best suitable for networks under Liander management?

This report gives a detailed description of the process and the results of the project. In Chapter 2, background information is given about the physics of electricity in general and in distribution networks. The power flow equations are explained in detail, as well as the calculation of power losses though a line. In Chapter 3, LRRP is described as a mathematical optimization problem, and some useful propositions which will be used further in the report, are stated and proved here. The NP-hardness of LRRP will be proved in Chapter 4. In Chapter 5 several suggestions are given for algorithms to approximate (or solve) LRRP, based on findings in the literature and expansions/adaptations on these findings. However, most of these suggested algorithms are only suitable for distribution networks with a single HV-connection. Chapter 6 states a novel method to adapt networks with multiple HV-connections in such a way that all the suggested algorithms are suitable for them, and proves its correctness. Chapter 7 is about the implementation procedure of the suggested algorithms, what difficulties occurred and how they were solved. Test results and conclusions are given in Chapter 8. Future work and research can be found in chapter 9, and chapter 10 gives the bibliography.

Chapter 2

Physics: Background Information on Electricity

In this chapter background information will be given about the physics involved with alternating current electricity (Section 2.1) and three-phase circuits (Section 2.2). This information is essential in order to get a good understanding of the network problematics, since all electricity is executed in a three-phase alternating current fashion. Full information can be found in [3]. In Section 2.3 detailed information can be found concerning the distribution of electricity in a network and the power flow equations, which directly explains the demand constraint in detail.

2.1 Alternating Current

In a generator, electricity is generated by a mechanically driven rotation of a coil in a magnetic field. This can be modelled as in Figure 2.1. Here a coil with length and width a is positioned in a magnetic field $\overrightarrow{B}$, and it is rotated with a rotation speed ω. In this situation a time

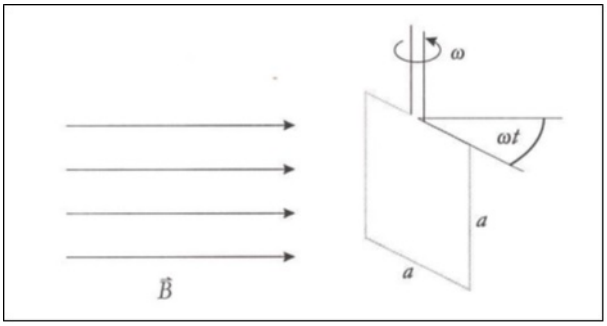

Figure 2.1: *A rotating coil in a magnetic field* ([3])

dependent voltage difference is generated over the endpoints of the coil. We can compute this voltage difference as follows:

Let $\overrightarrow{A}$ be the vector normal to the plane of the coil, with length $|\overrightarrow{A}| = a^2$. Then the

value of the magnetic flux satisfies the following equality:

$$\Phi_{\overrightarrow{B}} = \langle \overrightarrow{B}, \overrightarrow{A} \rangle = |\overrightarrow{B}| \cdot a^2 \cdot \sin(\theta) \tag{2.1}$$

Here θ is the angle between $\overrightarrow{A}$ and $\overrightarrow{B}$. This angle is time dependent since the coil is rotating, so we can write $\theta = \omega t + \psi_u$ with ψ_u the starting phase angle. Now this magnetic flux induces a time dependent potential difference or voltage difference between the two vertical sides of the coil, with value

$$u(t) = \frac{d}{dt}\Phi_{\overrightarrow{B}}(t) = |\overrightarrow{B}| \cdot a^2 \cdot \omega \cdot \cos(\omega t + \psi_u) = U_0 \cos(\omega t + \psi_u) \tag{2.2}$$

where $U_0 = |\overrightarrow{B}| \cdot a^2 \cdot \omega$. Now assume that the endpoints of the coil are connected via a cable with resistance R. Then as a consequence of this voltage difference, an electrical current I will flow from the higher voltage side to the lower voltage side through the cable. Now Ohm's law tells us

$$I \cdot R = |U_1 - U_2| = \Delta U \tag{2.3}$$

with U_1, U_2 the voltages on the end points of the cable. Applying this to our circuit with $\Delta U = u(t)$ we get a time dependent alternating current $i(t)$:

$$i(t) = \frac{u(t)}{R} = \frac{U_0}{Z} \cos(\omega t + \psi_u) = I_0 \cos(\omega t + \psi_i) \tag{2.4}$$

with $I_0 = \frac{U_0}{R}$ and $\psi_i = \psi_u$. We call ψ_u, ψ_i the phase of the voltage, current respectively. In this case where we have only a real-valued resistance, the phases of the voltage and the current will always be the same.

As will become clear later on, calculating with alternating current and voltage is easier when they are expressed as (the real part of) complex numbers. Therefore, the following definitions are made:

$$U(t) = U_0 e^{j(\omega t + \psi_u)} \tag{2.5}$$

$$I(t) = I_0 e^{j(\omega t + \psi_i)} \tag{2.6}$$

Note that j is used for the complex number $\sqrt{-1}$ instead of i. Now $u(t) = \Re(U(t))$ and $i(t) = \Re(I(t))$.

In reality, in an AC network the voltage and current phase will normally not be equal. A phase difference can occur over a certain part of a circuit, due to condensator- or coil-like behavior of assets (components in the network). Say we are in this case, and $\psi_i \neq \psi_u$ are different starting phases of the voltage and the current. We will always assume that the frequencies of the voltage and the current are equal, so we obtain a constant phase difference $\delta = \psi_u - \psi_i$. Therefore, the ratio between $U(t)$ and $I(t)$ will be a constant complex number:

$$\frac{U(t)}{I(t)} = \frac{U_0}{I_0} e^{j(\omega t + \psi_u) - j(\omega t + \psi_i)} = \frac{U_0}{I_0} e^{j\delta} = Z \tag{2.7}$$

This number Z is called the *impedance* of this specific part of the circuit, and is a time-invariant characteristic. If the argument δ of the impedance is zero, the impedance is a real number, corresponding to an ideal resistance as mentioned above. In general, the real part of the impedance is called the *resistance*, and the imaginary part is called the *reactance*. The former is the actual resistance of the asset an specifies the permeability of the asset, where the latter specifies the phase difference caused by the asset. Details can be found in [2] and in [3].

It should be noted that the assumption about equal frequencies is a simplification of reality, resulting in a discrepancy between actual voltage and current values and the calulated ones. However, it is believed that these differences are less than 5%. And since calculations are far better manageable with this assumption, this discrepancy is tolerated.

As mentioned before, the appearance of phase differences is a consequence of the specific properties of condensators and coils. Briefly, the alternating voltage causes alternating loading-unloading behavior in these components. As a consequence, part of the power transported through the network cannot be used in practice, but is going alternately forward and backward in the circuit, in order to alternately load and unload the components. This part is called the reactive power, in contrast to the active power, which is the power that is consumed by client devices in the circuit. We will now make this precise.

The power $p(t)$ present at a certain moment t is the product of the voltage and the current (using cosine product-to-sum identities):

$$\begin{aligned}
p(t) &= u(t) \cdot i(t) && (2.8)\\
&= U_0 I_0 \cos(\omega t + \psi_u)\cos(\omega t + \psi_i) && (2.9)\\
&= U_0 I_0 \frac{\cos(\omega t + \psi_u)\cos(\omega t + \psi_i) + \sin(\omega t + \psi_u)\sin(\omega t + \psi_i)}{2} && (2.10)\\
&\quad + U_0 I_0 \frac{\cos(\omega t + \psi_u)\cos(\omega t + \psi_i) - \sin(\omega t + \psi_u)\sin(\omega t + \psi_i)}{2}\\
&= U_0 I_0 \frac{\cos(\psi_u - \psi_i)}{2} + U_0 I_0 \frac{\cos(2\omega t + \psi_u + \psi_i)}{2} && (2.11)\\
&= \frac{U_0 I_0}{2}\cos(\delta) + \frac{U_0 I_0}{2}\cos(2\omega t + \psi_u + \psi_i) && (2.12)
\end{aligned}$$

with $\delta = \psi_u - \psi_i$ the phase difference. Now the average power is $P = \frac{U_0 I_0}{2}\cos(\delta)$. We call this the *active* power, often noted as P_{eff}. Although the value of the power is actually a sinusoïde, we often work with it as a constant power with value P_{eff}, since we work in a big time scale in where these waves are flattened out. Furthermore, we define $U_{\text{eff}} = \frac{1}{2}\sqrt{2} \cdot U_0$ and $I_{\text{eff}} = \frac{1}{2}\sqrt{2} \cdot I_0$. We call these values the *voltage magnitude* and *current magnitude*, respectively. Together with the starting phases ψ_u and ψ_i, which are called the *voltage angle* and *current angle*, we often regard the voltage as a constant complex voltage $U = U_{\text{eff}} \cdot e^{j\psi_u}$ and the current as a constant complex current $I = I_{\text{eff}} \cdot e^{j\psi_i}$. In the literature this is often noted as *phasor* notation. Remark that $\frac{U}{I} = \frac{U(t)}{I(t)} = \frac{U_0}{I_0}e^{j\delta}$ is the impedance.

For a complex number $y = a \cdot e^{j\theta}$, set $y^* = a \cdot e^{-j\theta}$ the complex conjugate. Set $S = U \cdot I^* = U_{\text{eff}} I_{\text{eff}} e^{j\delta}$. Then $P = \Re(S)$. It appears that the *reactive* power Q takes the average value

$Q = U_{\text{eff}} I_{\text{eff}} \sin(\delta)$. Hence $S = P + jQ$, and it turns out that $|S|$ is the total power that is transported through the circuit component in order to get a usable active power P. We call S the *apparent* power.

It is clear from the definitions that a smaller phase difference results in a smaller reactive (useless) power, hence a smaller difference between the active power and the apparent power. In electrical engineering it is important to take into account the reactive power as much as the active power. So we should work with the complex value S, and not just with the real value P.

Now we can calculate the power loss through a line. If I is the complex current through a line from a to b with voltages U_a, U_b, and $Z = R + jX$ the impedance of this line, then the *apparent power loss* S_{loss} can be calculated simply as the difference between the powers at a and b:

$$\begin{aligned} S_{\text{loss}} &= S_a - S_b && (2.13)\\ &= (U_a - U_b) \cdot I^* && (2.14)\\ &= IZI^* && (2.15)\\ &= |I|^2 Z && (2.16)\\ &= |I|^2 R + j|I|^2 X && (2.17) \end{aligned}$$

using Ohm's law (2.3). Now the reactance X only gives information about the phase difference between the current and the voltages due to the cable, where the resistance R induces the permeability of the cable. Only the latter plays a role for power loss, so the *actual power loss* is $P_{\text{loss}} = |I|^2 R$.

2.2 Three-phase Circuits

Most electrical distribution networks in the Netherlands are organized in a three-phase system. The principle of three-phase electric power is that three conductors carry an alternating current. These currents all have the same voltage- and current magnitude and frequency, but the voltage and current angles are shifted one-third of a period. See Figure 2.2.

The voltages will take the values:

$$\begin{aligned} U_A(t) &= U_{\text{eff}} e^{j\omega t + \psi_u} && (2.18)\\ U_B(t) &= U_{\text{eff}} e^{j(\omega t + \psi_u + \frac{2\pi}{3})} && (2.19)\\ U_C(t) &= U_{\text{eff}} e^{j(\omega t + \psi_u - \frac{2\pi}{3})} && (2.20) \end{aligned}$$

If the impedances over all phases are equal, representing a power demand balanced equally over the phases, the currents on phase A, B, and C will be (taking $\psi_i = 0$, possibly by shifting

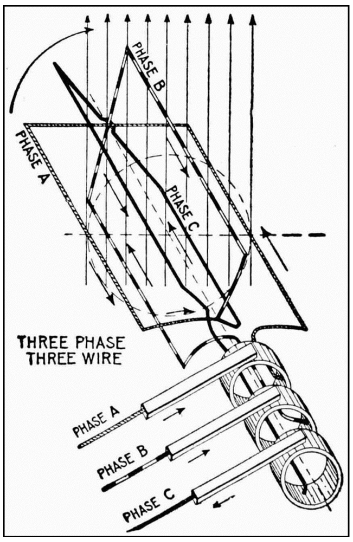

Figure 2.2: *Generation of three-phase electricity* ([3])

in time):

$$I_A(t) = \frac{U_A(t)}{Z} = I_{\text{eff}} e^{j\omega t} \tag{2.21}$$

$$I_B(t) = \frac{U_B(t)}{Z} = I_{\text{eff}} e^{j(\omega t + \frac{2\pi}{3})} \tag{2.22}$$

$$I_C(t) = \frac{U_C(t)}{Z} = I_{\text{eff}} e^{j(\omega t - \frac{2\pi}{3})} \tag{2.23}$$

with Z the value of the impedances. See Figure 2.3. In this situation, we have

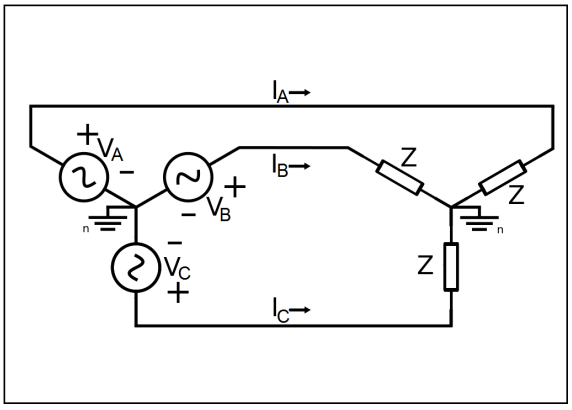

Figure 2.3: *Three-phase circuit* ([3])

$$I_A(t) + I_B(t) + I_C(t) = I_{\text{eff}}(e^{j\omega t} + e^{j(\omega t + \frac{2\pi}{3})} + e^{j(\omega t - \frac{2\pi}{3})}) \tag{2.24}$$

$$= I_{\text{eff}} e^{\omega t}(1 + \cos(\frac{2\pi}{3}) + j\sin(\frac{2\pi}{3}) + \cos(-\frac{2\pi}{3}) + j\sin(-\frac{2\pi}{3})) \tag{2.25}$$

$$= I_{\text{eff}} e^{\omega t}(1 + 2\cos(\frac{2\pi}{3})) \tag{2.26}$$

$$= 0 \tag{2.27}$$

so the circuit satisfies the Kirchhoff Current Law (KCL, stating that the sum of the currents at a node should be zero), hence it is a closed, well-behaving circuit.

(Note that if we work with the phasor notation I_A, I_B, I_C, we get the same outcomes, since then we just omit the factor $e^{j\omega t}$ in the above calculation.)

In reality however, towards the endpoint of the distribution network the power demand is often not balanced equally over the phases, resulting in different current magnitudes over the phases. Therefore, in general we have

$$I_A + I_B + I_C = |I_A|e^0 + |I_B|e^{\frac{2\pi}{3}} + |I_C|e^{-\frac{2\pi}{3}} \neq 0 \tag{2.28}$$

To cope with this, in the LV networks normally a fourth phase is added, possibly connected to the earth. This phase is called the *neutral* phase or *ground* phase, normally represented with an n. The voltage in this phase is zero, so the remaining current can be transported over this phase, in order to get the circuit satisfying the KCL.

In the MV and HV networks though, normally the voltages and currents in the different phases are almost equally balanced again, as an effect of the transformers. This justifies the general custom of modelling these network parts as if the loads are perfectly balanced, as is mentioned in [2].Therefore, we model a three-phase cable in the net as one line with one impedance, which makes it much easier to investigate a network model.

2.3 Power Flow

To compute the nodal voltages, line currents and total power loss as mentioned in Chapter 1 for a certain configuration of a network, we need to know the distribution of the electricity through the network, the so called *power flow*. We will now describe this behaviour and how we can simulate this. Detailed information can be found in [2] and in [4].

We model a network $N = (W, E)$ as a set of nodes W, and a set of connections between these nodes E. See [1]. Assume in a network N we know the impedances of all lines. Say node $k \in W$ is connected with nodes $l_1, ..., l_t \in W$ through lines $e_1 = (k, l_1), ..., e_t = (k, l_t) \in E$, with impedances $Z_{e_1}, ..., Z_{e_t} \in \mathbb{C}$. We define for k the complex nodal current I_k as the sum of the currents flowing away from k through lines $e_1, ..., e_t$, multiplying a current with -1 if it flows into k. For $\alpha \in \{1, ..., t\}$ we have:

$$U_k - U_{l_\alpha} = I_{e_\alpha} \cdot Z_{e_\alpha} \tag{2.29}$$

as is known from Ohm's law (2.3). (Note that if $U_{l_\alpha} > U_k$ then the current flows into k, and indeed we get a minus sign.) Hence

$$I_k = \sum_{\alpha=1,\ldots,t} I_{e_\alpha} \tag{2.30}$$

$$= \sum_{\alpha=1,\ldots,t} \frac{U_k - U_{l_\alpha}}{Z_{e_\alpha}} \tag{2.31}$$

$$= U_k \cdot \left(\sum_{\alpha=1,\ldots,t} \frac{1}{Z_{e_\alpha}} \right) - \left(\sum_{\alpha=1,\ldots,t} \frac{U_{l_\alpha}}{Z_{e_\alpha}} \right) \tag{2.32}$$

Now we define the *bus admittance matrix* Y_{bus} as the symmetric $|W| \times |W|$ matrix with, for $a, b \in W$, the values (set $E[a]$ the lines in E connected to a):

$$(Y_{\text{bus}})_{a,b} = \begin{cases} \sum\limits_{e \in E_s[a]} \frac{1}{Z_e} & \text{if } a = b \\ \frac{-1}{Z_e} & \text{if } a \neq b \text{ and } a \text{ is linked to } b \text{ with line } e \\ 0 & \text{else} \end{cases} \tag{2.33}$$

Then in the previous example we have

$$I_k = \sum_{l \in W} (Y_{\text{bus}})_{l,k} \cdot U_l \tag{2.34}$$

So for node k we have a remaining apparent power S_k with value:

$$S_k = U_k \cdot I_k^* \tag{2.35}$$

$$= U_k \sum_{l \in W} (Y_{\text{bus}})_{l,k}^* \cdot U_l^* \tag{2.36}$$

For every Bus k in the network N we can set up this equation, with two complex unknowns S_k and U_k, hence four real unknowns: The active and reactive powers, and the voltage magnitude and angle. We call these equations the *power flow equations*. Since every equation is complex, it gives us two real equations. Hence, if for every bus we specify two unknowns, the system is fixed by the power flow equations. And with this also the complex currents at the lines by (2.29).

In a distribution network indeed we know two parameters for every bus. For this, we can distinguish three types of nodes. The MV/LV transformers are referred to as load busses, and for such a bus we know the active and reactive power demand or supply (demand goes out of the network, given as negative value, while supply comes into the network, given as positive value). Generators in the network can be synchronized or asynchronized. For the former we know the supplied active and reactive power, hence we consider these as load busses. For the latter, we know the generated active power and the voltage magnitude. An HV/MV transformer is called a slack bus, although nowadays the function is more that of a compensator; it compensates the difference in total power supply, power demand and power loss in a network. For this node we know the voltage magnitude and angle. In the literature these nodes are also referred to as slack bus or infinite bus.

Now for each node two real parameters are assumed to be known, and two remain unknown. However, as mentioned above, these unknowns are fixed by as many power flow equations. Indeed, the behaviour in the network will satisfy these equations. However, as these equations are quadratic, they might have several solutions. How can we know which solution matches the behaviour in the network? It turns out that in case of multiple solutions for the voltages, those voltages will be realized that are as close to each other as possible. This seems natural, since smaller voltage differences cause smaller currents for the same apparent powers, resulting in smaller power losses. So in reality, the most thrifty power flow pattern will occur. Therefore, from now on, in case of multiple solutions for the power flow equations, we will only consider the one that is realized in the network.

To illustrate how we can use these power flow equations, an example cited from [4] is given. Consider the network in figure 2.4. At each cable e_x we have an impedance $Z_{e_x} = Z_x = |Z_x|e^{j\lambda_x}$. And at each node k we have a complex voltage $U_k = |U_k| \cdot e^{j\phi_k}$ with $|U_k|$ the voltage magnitude and ϕ_k the voltage angle, and a complex power $S_k = P_k + jQ_k$ with P_k the real power and Q_k the reactive power.

In the example of Figure 2.4, Bus 1 and 2 are generator busses, Bus 3 and 4 are load

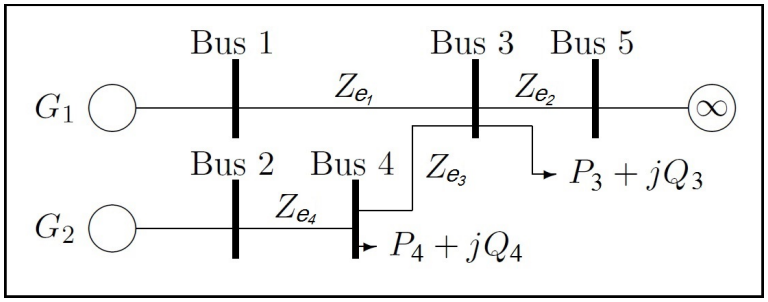

Figure 2.4: *Example of an electricity network* ([4])

busses, and Bus 5 is set as slack bus. For each Bus k we have a power flow equation

$$S_k = U_k \cdot \sum_{l=1}^{5} (Y_{\text{bus}})^*_{l,k} \cdot U_l^* \tag{2.37}$$

We will now make these equations explicit.

The *line admittance matrix* in this example is defined as

$$Y_{\text{line}} = \begin{pmatrix} \frac{1}{Z_1} & 0 & 0 & 0 \\ 0 & \frac{1}{Z_2} & 0 & 0 \\ 0 & 0 & \frac{1}{Z_3} & 0 \\ 0 & 0 & 0 & \frac{1}{Z_4} \end{pmatrix} \tag{2.38}$$

and the oriented incidence matrix of the network is

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 \end{pmatrix} \tag{2.39}$$

Then the bus admittance matrix can be computed as follows, with A^T the transposed matrix of A:

$$Y_{\text{bus}} = A \cdot Y_{\text{line}} \cdot A^T \tag{2.40}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{Z_1} & 0 & 0 & 0 \\ 0 & \frac{1}{Z_2} & 0 & 0 \\ 0 & 0 & \frac{1}{Z_3} & 0 \\ 0 & 0 & 0 & \frac{1}{Z_4} \end{pmatrix} \begin{pmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 \end{pmatrix} \tag{2.41}$$

$$= \begin{pmatrix} \frac{1}{Z_1} & 0 & \frac{-1}{Z_1} & 0 & 0 \\ 0 & \frac{1}{Z_4} & 0 & \frac{-1}{Z_4} & 0 \\ \frac{-1}{Z_1} & 0 & (\frac{1}{Z_1} + \frac{1}{Z_2} + \frac{1}{Z_3}) & \frac{-1}{Z_3} & \frac{-1}{Z_2} \\ 0 & \frac{-1}{Z_4} & \frac{-1}{Z_3} & (\frac{1}{Z_3} + \frac{1}{Z_4}) & 0 \\ 0 & 0 & \frac{-1}{Z_2} & 0 & \frac{1}{Z_2} \end{pmatrix} \tag{2.42}$$

Note that exchanging a 1 with a -1 in a column of A, i.e., changing the orientation of a line, does not effect Y_{bus}. So we can choose an arbitrary orientation on a network for constructing the incidence matrix.

Now the power flow equations become

$$S_1 = U_1 \cdot \left(\frac{U_1 - U_3}{Z_1}\right)^* = \frac{|U_1|^2 - U_1 U_3^*}{Z_1} \tag{2.43}$$

$$\tag{2.44}$$

$$S_2 = U_2 \cdot \left(\frac{U_2 - U_4}{Z_4}\right)^* = \frac{|U_2|^2 - U_2 U_4^*}{Z_4} \tag{2.45}$$

$$\tag{2.46}$$

$$S_3 = U_3 \cdot \left(\frac{U_3 - U_1}{Z_1}\right)^* + U_3 \cdot \left(\frac{U_3 - U_4}{Z_3}\right)^* + U_3 \cdot \left(\frac{U_3 - U_5}{Z_2}\right)^* \tag{2.47}$$

$$= \frac{|U_3|^2 - U_3 U_1^*}{Z_1} + \frac{|U_3|^2 - U_3 U_4^*}{Z_3} + \frac{|U_3|^2 - U_3 U_5^*}{Z_2} \tag{2.48}$$

$$\tag{2.49}$$

$$S_4 = U_4 \cdot \left(\frac{U_4 - U_2}{Z_4}\right)^* + U_4 \cdot \left(\frac{U_4 - U_3}{Z_3}\right)^* \tag{2.50}$$

$$= \frac{|U_4|^2 - U_4 U_2^*}{Z_4} + \frac{|U_4|^2 - U_4 U_3^*}{Z_3} \tag{2.51}$$

$$\tag{2.52}$$

$$S_5 = U_5 \cdot \left(\frac{U_5 - U_3}{Z_2}\right)^* = \frac{|U_5|^2 - U_5 U_3^*}{Z_2} \tag{2.53}$$

Each of these complex equations can be decomposed in two real equations, one for the real part and one for the imaginary part. For S_1 we get

$$S_1 = \frac{|U_1|^2}{|Z_1|} e^{-j\lambda_1} - \frac{|U_1||U_3|}{|Z_1|} e^{j(\phi_1 - \phi_3 - \lambda_1)} \tag{2.54}$$

hence

$$P_1 = \Re(S_1) = \frac{|U_1|^2}{|Z_1|} \cos(-\lambda_1) - \frac{|U_1||U_3|}{|Z_1|} \cos(\phi_1 - \phi_3 - \lambda_1) \tag{2.55}$$

$$Q_1 = \Im(S_1) = \frac{|U_1|^2}{|Z_1|} \sin(-\lambda_1) - \frac{|U_1||U_3|}{|Z_1|} \sin(\phi_1 - \phi_3 - \lambda_1) \tag{2.56}$$

We can repeat this for nodes 2, 3, 4 and 5 to get the remaining power flow equations:

$$P_2 = \frac{|U_2|^2}{|Z_4|}\cos(-\lambda_4) - \frac{|U_2||U_4|}{|Z_4|}\cos(\phi_2 - \phi_4 - \lambda_4) \tag{2.57}$$

$$Q_2 = \frac{|U_2|^2}{|Z_4|}\sin(-\lambda_4) - \frac{|U_2||U_4|}{|Z_4|}\sin(\phi_2 - \phi_4 - \lambda_4) \tag{2.58}$$

$$\begin{aligned} P_3 =& \frac{|U_3|^2}{|Z_1|}\cos(-\lambda_1) - \frac{|U_1||U_3|}{|Z_1|}\cos(\phi_3 - \phi_1 - \lambda_1) + \frac{|U_3|^2}{|Z_3|}\cos(-\lambda_3) \\ & - \frac{|U_3||U_4|}{|Z_3|}\cos(\phi_3 - \phi_4 - \lambda_3) + \frac{|U_3|^2}{|Z_2|}\cos(-\lambda_2) - \frac{|U_3||U_5|}{|Z_2|}\cos(\phi_3 - \phi_5 - \lambda_2) \end{aligned} \tag{2.59}$$

$$\begin{aligned} Q_3 =& \frac{|U_3|^2}{|Z_1|}\sin(-\lambda_1) - \frac{|U_1||U_3|}{|Z_1|}\sin(\phi_3 - \phi_1 - \lambda_1) + \frac{|U_3|^2}{|Z_3|}\sin(-\lambda_3) \\ & - \frac{|U_3||U_4|}{|Z_3|}\sin(\phi_3 - \phi_4 - \lambda_3) + \frac{|U_3|^2}{|Z_2|}\sin(-\lambda_2) - \frac{|U_3||U_5|}{|Z_2|}\sin(\phi_3 - \phi_5 - \lambda_2) \end{aligned} \tag{2.60}$$

$$\begin{aligned} P_4 =& \frac{|U_4|^2}{|Z_4|}\cos(-\lambda_4) - \frac{|U_2||U_4|}{|Z_4|}\cos(\phi_4 - \phi_2 - \lambda_4) + \frac{|U_4|^2}{|Z_3|}\cos(-\lambda_3) \\ & - \frac{|U_3||U_4|}{|Z_3|}\cos(\phi_4 - \phi_3 - \lambda_3) \end{aligned} \tag{2.61}$$

$$\begin{aligned} Q_4 =& \frac{|U_4|^2}{|Z_4|}\sin(-\lambda_4) - \frac{|U_2||U_4|}{|Z_4|}\sin(\phi_4 - \phi_2 - \lambda_4) + \frac{|U_4|^2}{|Z_3|}\sin(-\lambda_3) \\ & - \frac{|U_3||U_4|}{|Z_3|}\sin(\phi_4 - \phi_3 - \lambda_3) \end{aligned} \tag{2.62}$$

$$P_5 = \frac{|U_5|^2}{|Z_2|}\cos(-\lambda_2) - \frac{|U_3||U_5|}{|Z_2|}\cos(\phi_5 - \phi_3 - \lambda_2) \tag{2.63}$$

$$Q_5 = \frac{|U_5|^2}{|Z_2|}\sin(-\lambda_2) - \frac{|U_3||U_5|}{|Z_2|}\sin(\phi_5 - \phi_3 - \lambda_2) \tag{2.64}$$

Now for every node two of the four parameters are known, as described. Substituting these known parameters into the equations gives a system of ten equations with ten variables. Solving these equations exactly is a complex problem, it might be NP-hard itself. Therefore it is widely common to approximate these solutions with the Newton-Raphson method, see [30]. This will also be the method used during this project.

One important remark should be made. In actual distribution networks, distributed generation is a growing factor in the network, which can result in reversed currents and slack busses receiving power instead of delivering it. Although this demands caution for the actual assets, this is not a problem for the theoretical power flow calculations. Generators can be added to the network without any difficulties, and when they supply power back in the direction of the slack bus, the voltages will become higher than the slack bus voltage and the corresponding current angles will simply turn 180 degrees, corresponding to a reversed current.

Chapter 3

Mathematics: Problem Description and Propositions

This chapter contains the mathematical description of the central problem (Section 4.1). Also some interesting propositions are stated and proved in Sections 4.2 and 4.3. These will be used later on in this report.

3.1 The Loss Reduction Reconfiguration Problem

The Loss Reduction Reconfiguration problem (LRRP) is described in Chapter 1 as

Which configuration of an MV network that satisfies the radiality, demand, and capacity constraints, minimizes the sum of the line losses $L(e)$ over all lines e of the network?

We will now describe this problem as a mathematical optimization problem. To define the search space, i.e. the set of search candidates, a mathematical formulation of a configuration for a network is given. After this, the constraints and the objective function are defined mathematically as well.

We can represent an MV network as an undirected graph $N = (W, E)$ with nodes W and edges E (see [1] for full information about graph theory). The nodes W represent the busses; stations and transformers in the net. We distinguish HV/MV transformers connected to the HV net which we traditionally call *slack busses* (although nowadays these nodes actually function as compensators for the difference between the power supply and demand in the network), MV/LV transformers connected to LV networks which we call *load busses* (which, again, can also supply power to instead of demand power of the MV net), and (asynchronised) *generators*. The difference between load busses and generators was made clear in Chapter 2, and these nodes together are known as the *client busses* in the net. Set W_n as the set of slack busses in the network, and W_c the set of client busses, then $W = W_n \cup W_c$.

For the edges E, we distinguish the set of *optional* edges E_o which represent the cables and links with switches. These switches can be opened, resulting in a deletion of the corresponding edge in the network. Other edges are *fixed* edges E_f representing those without switches. So these edges will always be present in the network. Now $E = E_o \cup E_f$.

A certain open/closed specification of the switches in N results in a certain *configuration* of N. Let $A \subseteq E_o$ be the set of optional edges with closed switches. We call A a *switch specification*, and N_A the corresponding configuration. Then $E_o \backslash A$ are the open edges, hence $N_A = (W, E_A = E_f \cup A)$, the subgraph of N with only the fixed edges and the closed optional edges.

Let A be a switch specification for network N. Every node $v \in W$ contains a certain voltage U_v^A, and every line $e \in E_A$ contains a certain current I_e^A. These voltages and currents depend on the power supplies and demands at the nodes in the network, and can be calculated by solving the power flow equations, as described in Chapter 2. But they also depend on the way N_A is configured, since different connections give different power flow pattern. Hence we have to add the index A to the parameters. Note that when a line $e \in E_o$ is opened, the current through this line will always be zero, hence we can always talk about current I_e^A for every $e \in E$.

Cables and links in the network have certain impedances. Say line $e \in E$ has impedance $Z_e = R_e + jX_e$ (with $j = \sqrt{-1}$, standard in electrical engineering). Then the power loss through line e can be calculated as $P_e^{\text{loss}} = |I_e^A|^2 \cdot R_e$. With this, we can describe our objective function as

$$L(A) = \sum_{e \in E_A} |I_e^A|^2 \cdot R_e \tag{3.1}$$

$$= \sum_{e \in E} |I_e^A|^2 \cdot R_e \tag{3.2}$$

Note that if $e \notin A$, then $I_e^A = 0$, so the second equality holds. Now $L(A)$ is the total power loss in N_A, so we want to find A such that $L(A)$ is minimized.

As described in Chapter 1, the topology of a configuration has the following requirements:

- Every node in the network must be connected to an HV/MV transformer, in order to be able to get demanded power from or withdraw remaining power to the HV net.
- No cycles (rings) are allowed in the network. This makes it easier to allocate and restore a disturbance. Also, when a fault occurs in a cycle, the short-circuit current will be higher. Hence this requirement is also for safety reasons.
- HV/MV transformers are not allowed to be connected to each other. Different transformers constitute different voltage and current frequencies, so when these are connected, an unstable voltage and current will arise as a consequence.

Taking these requirements into account, a feasible configuration must be, in graph theoretical terms, a forest with in each subtree exactly one slack bus $v \in W_n$. This is what we from now on mean with a *radial* structure or topology, and we will refer to it as the *radiality* constraint.

As described in Section 2.3, due to power supply and demand in the network, a specific power flow pattern occurs. This pattern can be simulated by the power flow equations. Since we need this pattern for loss calculation and capacity checks, the *demand constraint* is added,

which postulates that the voltages and currents in the network satisfy the power flow equations. These equations can be found in section 2.3.

The *capacity constraint* simply states that the nodal voltages should be between certain bounds and the line currents should be bounded from above, in order to maintain good power quality and avoid overcharging of the assets. Mathematically, this can be described as

$$\forall v \in W \qquad |U_v^{\min}| \leq |U_v^A| \leq |U_v^{\max}| \tag{3.3}$$

$$\forall e \in E \qquad |I_e^A| \leq |I_e^{\max}| \tag{3.4}$$

where the values of $U_v^{\min}, U_v^{\max}$ for any $v \in W$, and $I_e^{\max}$ for any $e \in E$ are assumed as known.

Now we are able to describe the central problem LRRP mathematically. For network $N = (W, E)$ we have the following optimization problem:

$$\text{LRRP}: \min_{A \subseteq E_o} \sum_{e \in E} |I_e^A|^2 \cdot R_e \text{ such that}$$

$$\begin{aligned} &N_A \text{ is a radial network} \\ &(U_{v_1}^A, ..., U_{v_n}^A), (I_{e_1}^A, ..., I_{e_m}^A) \text{ satisfy the power flow equations} \\ &\forall e \in E: \quad |I_e^A| \leq |I_e^{max}| \\ &\forall v \in W: \quad |U_v^{min}| \leq |U_v^A| \leq |U_v^{max}| \end{aligned}$$

3.2 Bijection Cycle Basis - Deleted Edges

Note that a radial configuration of a network with a single slack bus is actually a spanning tree of the corresponding graph. For a connected, undirected graph G, a spanning tree T of G can be constructed by deleting some edges of G. One way of finding all possible spanning trees of a graph is by checking all possible combinations of edge deletions of the graph. However, in this section we will suggest a faster method based on a mathematical proposition.

Before we can do this, we have to define the notion of a cycle basis of G. And therefore we will need the notion of symmetric difference.

Definition 3.1. *Let $G = (W, E)$ be a connected, undirected graph, and let $C_1, C_2 \subset E$ be cycles. Define the* symmetric difference $C_1 \triangle C_2$ *of C_1 and C_2 as*

$$C_1 \triangle C_2 = C_1 \backslash C_2 \ \cup \ C_2 \backslash C_1 \tag{3.5}$$

Note that $C_1 \triangle C_2$ is an even subgraph of G, i.e. a subgraph which is a cycle or a combination of cycles. Furthermore, note that symmetric difference is associative and commutative, so we can define

$$\triangle_{i=1}^n C_i = C_1 \triangle C_2 \triangle \ ... \ \triangle C_n \tag{3.6}$$

If $C_1, ..., C_n$ are cycles, the resulting subgraph will always be a cycle itself or the union of some cycles in G. And since $A \triangle A = \emptyset$, we have that

$$\forall (\alpha_1, ..., \alpha_n) \in \mathbb{N}^n \qquad \triangle_{i=1}^n \alpha_i C_i = \triangle_{i=1}^n (\alpha_i \mod 2) \cdot C_i \tag{3.7}$$

Definition 3.2. *For a connected, undirected graph $G = (W, E)$, let $\mathcal{C}$ be the set of all cycles $C \subseteq E$ in G. Now $\mathcal{C}$ is a subset of the cycle space of G, which is the vector space consisting of all even subgraphs in G, with symmetric difference as operator on the edges (see [1]). In this context, we call cycles $C_1, ..., C_n$* linear dependent *if there exists a non-trivial vector $(\alpha_1, ..., \alpha_n) \in \{0, 1\}^n$ such that*

$$\triangle_{i=1}^{n} \alpha_i C_i = \emptyset \tag{3.8}$$

i.e., assuming $\alpha_1 = 1$, C_1 can be constructed by 'adding up' cycles of $C_2, ..., C_n$. Otherwise, we call $C_1, ..., C_n$ linearly independent*. Now we define a* cycle basis *of G to be a linearly independent subset $C_1, ..., C_n$ of $\mathcal{C}$ of maximal cardinality, so any cycle $C \in \mathcal{C}$ can be constructed out of this basis.*

Note that a cycle basis of G is also a vector basis of the cycle space of G. Hence two cycle bases of G will have the same cardinality. As we will see, this cardinality is $r = m-n+1$, with $n = |V|,\ m = |E|$. For this, first we state another lemma, which is easy to prove inductively and can be found in [1]. Therefore we will not prove it here.

Lemma 3.3. *Graph $G = (V, E)$ is a tree (a connected graph without cycles) if and only if $|E| = |V| - 1$.*

Lemma 3.4. *Let $G = (V, E)$ be a connected, undirected graph , with $|V| = n$, $|E| = m$. Set $r = m - n + 1$. Then a cycle basis of G has r elements.*

Proof. We will proof this lemma with induction on r.
Say $r = 0$. Then $m = n - 1$, hence G is a tree by Lemma 3.3. So $\mathcal{C}$ is empty, hence has an empty cycle basis.
Choose $r \geq 1$, and say the lemma holds for $1, ..., r - 1$. Since $r = m - n + 1 > 0$, G contains a cycle $C \subset E$, since G is connected and by lemma 3.3 not a tree. Choose $e \in C$. Define $G' = (V, E\backslash\{e\})$. Then $|E\backslash\{e\}| - |V| + 1 = m - 1 - n + 1 = r - 1$. Hence, by induction, a cycle basis of G' has $r - 1$ elements. Say $C_1, ..., C_{r-1}$ is such a basis. These cycles are linearly independent in G as well. Furthermore, they do not contain e. Now note that

$$\triangle_{i=1}^{k} X_i \subseteq \bigcup_{i=1}^{k} X_i \tag{3.9}$$

always holds, which implies that there cannot exist $\alpha_1, ..., \alpha_{r-1}$ such that $C = \triangle_{i=1}^{r-1} \alpha_i C_i$. Hence $C, C_1, ..., C_{r-1}$ are linearly independent cycles in G.

Now choose another cycle D in G. If $e \notin D$, then D is a cycle in G'. Therefore, $D, C_1, ..., C_{r-1}$ are linearly dependent, hence $D, C, C_1, ..., C_{r-1}$ are linear dependent as well.
If $e \in D$, then $e \notin D \triangle C$, which is an even graph in G'. This even graph can be decomposed into cycles in G', so each of these cycles is linearly dependent with $C_1, ..., C_{r-1}$, i.e. can be constructed out of $C_1, ..., C_{r-1}$. Hence $D \triangle C$ is linearly dependent with $C_1, ..., C_{r-1}$, so D is linearly dependent with $C, C_1, ..., C_{r-1}$. This implies that $C, C_1, ..., C_{r-1}$ is a linearly independent set of cycles of maximal cardinality, hence a cycle basis of G. And it has $r = m - n + 1$ elements. □

The notion of a cycle basis will become very useful for the Loss Reduction Reconfiguration Problem. This will be expressed in the next proposition.

Proposition 3.5. *Let $G = (V, E)$ be a connected, undirected graph, and set $r = |E| - |V| + 1$. Let $S = \{e_1, ..., e_r\} \subset E$ be such that $G \backslash S = (V, E \backslash S)$ is a spanning tree and let $B = \{C_1, ..., C_r\} \subset \mathcal{C}$ be a cycle basis of G. Then a bijection $f : S \to B$ exists such that for all $e \in S$: $e \in f(e)$.*

Note that $|E \backslash S| = |V| - 1$ by Lemma 3.3, hence $|S| = |E| - (|V| - 1) = r$. This proposition implies that for a network G with one slack bus and a cycle basis B of G, we can construct any radial configuration of G by picking a certain optional edge in every cycle in B that should be opened. This gives a practical method for finding all radial configurations of G in which significantly fewer combinations have to be checked compared to simply checking all possible combinations of edges that can be opened.

In order to prove this proposition, we will make use of the following theorem.

Proposition 3.6 (Hall's Marriage Theorem)**.** *Let $G = (W_1 \cup W_2, E)$ be a bipartite graph with $|W_1| = |W_2|$. For $X \subset W_1$, let $N(X)$ be the set of neighbours of X, i.e. the nodes in W_2 that are connected to some node in X. Then a perfect matching for G exists if and only if for any $X \subset W_1$: $|X| \leq |N(X)|$.*

A matching of a graph $G = (W, E)$ is a subset $M \subseteq E$ such that any node $v \in W$ is covered by at most one edge in M. Furthermore, M is a perfect matching if it covers every node of G. A proof of this theorem can be found in [1], and will not be given here. However, we can now prove Proposition 3.5.

Proof of Proposition 3.5. Define the bipartite graph $H = (B \cup S, E')$ with $(C_i, e_j) \in E'$ iff $e_j \in C_i$. Then a bijection as stated in proposition 3.5 corresponds to a perfect matching of H. By Hall's Marriage Theorem and since $|B| = |S|$, a perfect matching of H exists if for every $X \subseteq B$: $|X| \leq |N(X)|$. We will prove that this is the case by induction on $k = |X|$.

Let $|X| = 1$, so $X = \{C_i\}$ for some i. Then there exists a j such that $e_j \in C_i$, since else C_i is a cycle in $E \backslash S$, contradicting $E \backslash S$ being a tree. So $e_j \in N(X)$, hence $|N(X)| \geq 1 = |X|$.

Now choose $k \in \mathbb{N}$, $2 \leq k \leq r$, and assume the induction hypothesis holds for all $X \subseteq B$ with $|X| < k$. Then choose $X \subseteq B$ with $|X| = k$. Say $X = \{C_1, ..., C_k\}$, possibly after renumbering the elements of B. Define $X' = \{C_1, ..., C_{k-1}\}$, $X'' = \{C_k\}$. By induction, $|N(X')| \geq k - 1$, $|N(X'')| \geq 1$.

If $|N(X')| \geq k$, or if $N(X'') \not\subseteq N(X')$, then $|N(X)| = |N(X') \cup N(X'')| \geq k$, so the hypothesis holds.
Now assume $|N(X')| = k - 1$ and $N(X'') \subseteq N(X')$. We will see that this leads to a contradiction.

Say $N(X') = \{e_1, ..., e_{k-1}\}$ and $N(X'') = \{e_1, ..., e_{l_1}\}$ with $1 \leq l_1 \leq k - 1$ (possibly after renumbering elements of S). Let H' be the subgraph of H on X' and $N(X')$, so $H' = H[X' \cup N(X')]$. Then by the induction hypothesis and Hall's Marriage Theorem, H' admits a perfect matching. For $i = 1, ..., k - 1$, let e_i be matched to C_i (possibly after renumbering).
Now look at $C_k \cup C_1 \cup \ ... \ \cup C_{l_1}$ in G. We can construct a cycle D_1 in this subgraph that does

not contain $e_1, ..., e_{l_1}$ as follows: Start in a node on C_k and walk through C_k. If we reach an edge e_i with $1 \leq i \leq l_1$, there is at least one path avoiding e_i and leading back to C_k, namely (a part of) C_i. Once back on C_k, proceed walking in the same direction of C_k. Since we have finitely many edges, we will eventually pass a node we visited before. Hence we found a cycle that does not contain $e_1, ..., e_{l_1}$.
Now if D_1 does not contain any edge from $e_{l_1+1}, ..., e_{k-1}$, then D_1 does not contain any edge in S. Hence D_1 is a cycle in $E \backslash S$, a contradiction. So let $e_{l_1+1}, ..., e_{l_2}$ be the edges contained by D_1, with $l_1 < l_2 \leq k-1$. But then with the same construction, $C_k \cup C_1 \cup \ ... \ \cup C_{l_2}$ has a cycle D_2 that does not contain $e_1, ..., e_{l_2}$: We can walk through D_1 and when we reach an e_i with $l_1 + 1 \leq i \leq l_2$, at least one path is available that avoids e_i and leads back to D_2, namely (a part of) C_i. If this path contains an e_j with $1 \leq j \leq l_1$, we arrived at C_k and we can walk to the other side of e_j avoiding any edge of $e_1, ..., e_{l_1}$ by using C_j again, and proceed to D_1. Again, there are only finitely many edges, so ultimately a cycle is formed.
And again, D_2 must contain at least one edge of $e_{l_2+1}, ..., e_{k-1}$, leading to the construction of a new cycle D_3. This cannot continue for more than $k-1$ iterations, so we will end up with a cycle D_q in G that does not contain any edge of S. Hence D_q is a cycle is $G \backslash S$, which contradicts $G \backslash S$ being a tree.

As a consequence, we have either $|N(X')| \geq k$, or $N(X'') \not\subseteq N(X')$, which results in $N(X) \geq k = |X|$. And since X was arbitrary, this holds for any $X \subseteq B$ with $|X| = k$.
Hence, by induction, for all $X \subseteq B: \ |X| \leq |N(X)|$. So by Hall's Marriage Theorem, H admits a perfect matching, i.e. there is a bijection $f : S \to B$ such that $e \in f(e)$ for all $e \in S$. □

A radial configuration of a (connected) distribution network with one slack bus is actually a spanning tree of this network. Assuming we have a cycle basis for this network and an arbitrary radial configuration, Proposition 3.5 implies that there exists a one-to-one correspondence between the cycles in the basis and the opened edges. This implies that any radial configuration of a network can be constructed by choosing an edge on every cycle in a cycle basis, and opening this edge. This gives us a method for finding all radial configurations without actually checking all possible switch specifications.

However, also non-radial configurations can be constructed this way. For instance, look at the small network in Figure 3.1. Here Bus 1 is the slack bus, recognizable by the square connected to it. The other busses are load busses, recognizable by the outgoing arrows. Say we have cycle basis $C_1 = \{1, 4, 7, 5, 2\}, C_2 = \{2, 5, 8, 6, 3\}, C_3 = \{9, 12, 10, 7\}, C_4 = \{10, 13, 11, 8\}$. (Indeed this is a cycle basis of the network. We will not check this here.)

Note that some edges are contained in more than one cycle in the basis. As a consequence, if we choose an edge in every cycle in the basis arbitrary, we might choose some edges more than once. And if we then open the chosen edges, we will open less than r edges, hence the configuration can never be a spanning tree, i.e. never be radial.

But even if we choose r different edges, radiality cannot be ensured. In Figure 3.1, choose edge 5 for cycle C_1, edge 8 for C_2, edge 7 for C_3 and edge 10 for C_4. The resulting configuration contains cycle $\{1, 4, 9, 12, 13, 11, 6, 3\}$ and Bus 6 is isolated. Hence no radial configuration.

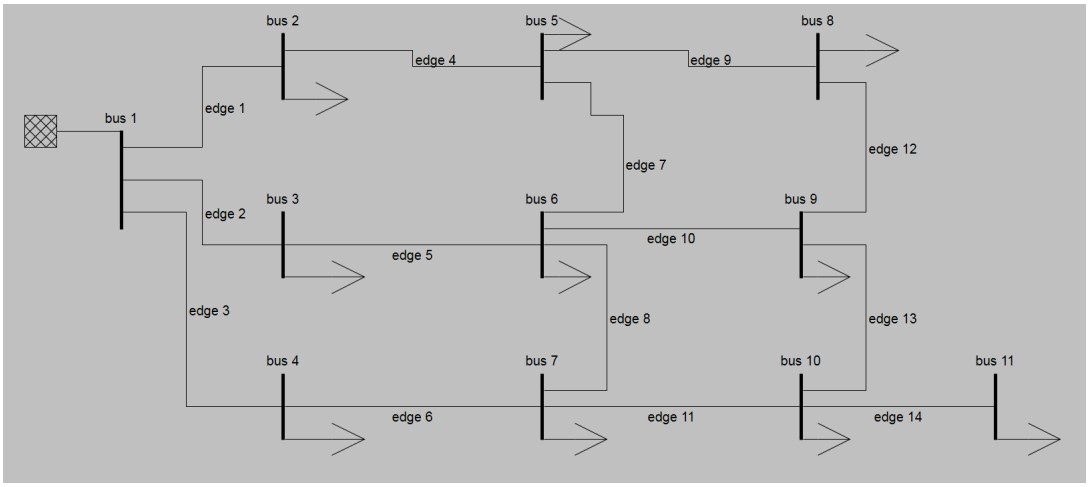

Figure 3.1: *Network 1* ("Vision")

We conclude that by opening an edge for every cycle in a cycle basis, we can construct any radial configuration of a network with one slack bus, but also non-radial configurations can be constructed. We have to take this into account when we use this method for the algorithms.

3.3 Bijection Semi-Ear Decomposition - Deleted Edges

However, instead of looking at a cycle basis of the network, an alternative method might be to look at a *semi-ear decomposition*:

Definition 3.7. *Let $G = (V, E)$ be a connected, undirected graph, and let $\{C_1, ..., C_r\}$ be a cycle basis of G. Define $O_1 = C_1 \subset E$ and for all $i = 2, ..., r$: $O_i = C_i \backslash (\bigcup_{j=1}^{i-1} C_j) \subset E$. Then $\{O_1, ..., O_r\}$ is a* semi-ear decomposition *of G, and $O_1, ..., O_r$ are called* semi-ears.

We call this a semi-ear decomposition since this concept is much alike that of an *ear decomposition* ([1]). Only semi-ears can consist of several connected components, in contrast to an ear, which is connected by definition. Furthermore, a semi-ear decomposition can consist of more than one cycle, where in an ear decomposition only the first ear is a cycle. Another, more critical difference is that a semi-ear can be empty, since a cycle in a cycle basis can be fully contained in the union of other cycles in the basis. Nevertheless, we will work with a semi-ear decomposition rather than an ear decomposition, since the former can be constructed easily for a graph, and the latter not. Note that an ear decomposition is always a semi-ear decomposition.

For a single slack bus network, the fact that deleting an edge in every cycle of a cycle basis can generate also non-radial configurations, is caused by the intersections of the cycles. Indeed, if for two cycles two edges are deleted on the intersection of these cycles, a new cycle remains and a part of the graph is isolated. In a semi-ear decomposition, all these intersections are deleted for all but one of the concerning cycles. And as will be substantiated by the following proposition, this ensures that any configuration created by deleting an edge on every semi-ear will be radial.

Proposition 3.8. *Let $G = (V, E)$ be a connected, undirected graph, and let $D = \{O_1, ..., O_r\}$ be a semi-ear decomposition of G such that no semi-ear is empty. For $i = 1, ..., r$, choose $e_i \in O_i$. Then $G \backslash \{e_1, ..., e_r\}$ is a spanning tree of G.*

Proof. We will prove this proposition by induction on r.

Say $r = 0$. Then G has an empty cycle basis, hence G does not contain any cycle. This means G is a tree, hence the proposition holds.

Now let $r \in \mathbb{N},\ r > 0$. Say the proposition holds for $1, ..., r-1$. Further, let $B = \{C_1, ..., C_r\}$ be the cycle basis used for the construction of D, in this order.

Say O_r consists of k connected components (which are sub paths of C_r, we will call them strings). Look at $H = G - O_r$. We will see that H has k connected components as well: If H has fewer than k connected components, then $k > 1$ and there is a string s of O_r that connects two nodes in one component of H. Hence a cycle $\hat{C}$ in G exists containing this string, and no other string of O_r. This cycle cannot be contained in B, else O_r would not contain s by construction of the semi-ear decomposition. But this cycle is linearly independent of B. This is because C_r is the only cycle in B containing s, but since $k > 1$, C_r contains another string s' not contained by any other cycle in B and also not contained by $\hat{C}$. A contradiction, since B is a cycle basis.
If H has more than k connected components, then G was not connected, a contradiction as well.

Let $H_1, ..., H_k$ be the connected components of H. Note that $C_1, ..., C_{r-1}$ are fully contained in H, since any edge in $C_1, ..., C_{r-1}$ is not in O_r. Also, for $i = 1, ..., r-1$ there is exactly one $j \in \{1, ..., k\}$ such that C_i is contained by connected component H_j of H.
Furthermore, if $\hat{C}$ and $\tilde{C}$ are two cycles in two different connected components of H, then $\hat{C} \cap \tilde{C} = \emptyset$, since else they would be in the same connected component.

Let $B_j = \{C_1^j, ..., C_{r_j}^j\}$ be the cycles in the basis contained by H_j for all $j = 1, ..., k$, in the same order as they are in B. Then B_j is a cycle basis of H_j.
Say for $j \in \{1, ..., k\},\ i \in \{1, ..., r_j\}$ we have C_i^j is actually C_q in B. Then by the above arguments:

$$O_i^j = C_i^j \backslash (\bigcup_{l=1}^{i-1} C_l^j) = C_q \backslash (\bigcup_{p=1}^{q-1} C_p) = O_q \tag{3.10}$$

Indeed, for $l \leq i-1$ there is a $p \leq q-1$ such that $C_l^j = C_p$, since the order of the elements of B_j is consistent with the order of those in B. And if $C_p \notin B_j$, then $C_i^j \cap C_p = \emptyset$, hence removing C_p from C_i^j has no effect.

Now $D_j = \{O_1^j, ..., O_{r_j}^j\}$ is a semi-ear decomposition of H_j, with $r_j \leq r-1$. Hence, by induction, if $e_1^j, ..., e_{r_j}^j$ are the edges chosen for $O_1^j, ..., O_{r_j}^j$, then $H_j - \{e_1^j, ..., e_{r_j}^j\}$ is a tree.

This implies that

$$H - \{e_1, ..., e_{r-1}\} = (H_1 - \{e_1^1, ..., e_{r_1}^1\}) \cup \ ... \ \cup (H_k - \{e_1^k, ..., e_{r_k}^k\}) \tag{3.11}$$

is a forest with k trees.
Now O_r connects all these subtrees in $G - \{e_1, ..., e_{r-1}\} = H - \{e_1, ..., e_{r-1}\} \cup O_r$, since G was connected and every tree is connected. Therefore, $G - \{e_1, ..., e_{r-1}\}$ contains exactly one cycle, since r edges have to be deleted in order to get a tree, as implied by lemma 3.3.
This cycle can be constructed by starting on a point of O_r and walking in a fixed direction. When we reach the end of a string of O_r, we are at some tree $H_j - \{e_1^j, ..., e_{r_j}^j\}$ for some j. There is a path in this tree to a new string of O_r, which leads to another tree. Proceeding this way, we will cross any string of O_r and any subtree $H_j - \{e_1^j, ..., e_{r_j}^j\}$,until we get back to our starting point.

Now O_r is fully contained in this cycle, so opening any edge of O_r opens this cycle, resulting in a spanning tree of G. Hence $G - \{e_1, ..., e_r\}$ is a tree. □

In contrast to a cycle basis, by opening an edge in every semi-ear of a semi-ear decomposition of a connected distribution network with a single slack bus, the resulting configuration will *always* be radial, as implied by Proposition 3.8. However, there is a disadvantage for this method as well. In general, not all radial configurations of a network can be constructed with one semi-ear decomposition. Look again at network 1 in Figure 3.1. The semi-ear decomposition corresponding to the mentioned cycle basis is $O_1 = \{1, 4, 7, 5, 2\}, O_2 = \{8, 6, 3\}, O_3 = \{9, 12, 10\}, O_4 = \{13, 11\}$. Indeed, the non-radial configuration constructed earlier by opening edges 5, 7, 8, and 10 cannot be generated by this ear decomposition. Moreover, any configuration constructed by opening an edge in every semi-ear is radial. But if we open edges 5, 7, 10, and 13, we do get a radial configuration as well, although this configuration cannot be generated with this ear decomposition. So there are radial configurations that cannot be constructed by this semi-ear decomposition.

There is a way to solve this. If we change the order of the cycles in the cycle basis, we can obtain a different semi-ear decomposition, for instance $\hat{O}_1 = \{2, 5, 8, 6, 3\}, \hat{O}_2 = \{1, 4, 7\}, \hat{O}_3 = \{9, 12, 10\}, \hat{O}_4 = \{13, 11\}$. With this semi-ear decomposition, the proposed radial configuration *can* be constructed by picking an edge for each semi-ear.

Indeed, if we consider enough semi-ear decompositions for a connected distribution network with a single slack bus, we are able to construct every radial configuration for this network by opening an edge on every semi-ear in a decomposition. However, as we will see in Chapter 7, 'enough' might not be a practical number.

Chapter 4

NP-hardness of the Loss Reduction Reconfiguration Problem

Finding the network reconfiguration which results in minimal power loss is a hard problem in general. In this chapter we will prove that this problem is actually NP-hard, which implies that the existence of an efficient algorithm that solves LRRP is unlikely. To understand the full meaning of this statement, we give some definitions. Details can be found in [24].

Definition 4.1 (Class P)**.** *A problem* Π *belongs to* class P *if and only if there exists an algorithm* A *and a polynomial* $P : \mathbb{N} \to \mathbb{N}$ *that for any instance* I *of* Π *of input size* n *(in data, for instance cardinalities or dimensions),* A *solves* I *and the number of steps needed for this is smaller than* $P(n)$*. We say that* A *solves* Π *in polynomial time.*

Definition 4.2 (Class NP)**.** *A problem* Π *belongs to* class NP *if and only if there exists an algorithm* A *and a polynomial* $P : \mathbb{N} \to \mathbb{N}$ *that for any instance* I *of* Π *of input size* n *(in data, for instance bits or dimensions) and any certificate* c *of* I*,* A *decides whether or not* c *is a witness of* I *and the number of steps needed for this is smaller than* $P(n)$*.*

To illustrate this, consider the Minimum Weight Spanning Tree problem (MWST). An instance is a graph G with edge weights. One wants to find a spanning tree of G with minimal total weight. This problem belongs to P, since several efficient, polynomial-time algorithms are developed that find a minimal spanning tree of G. For instance Kruskal's algorithm or Prim's algorithm, see [23]. On the other hand, consider Hamilton Cycle (HC). Again an instance is a graph G, and now one wants to know whether G has a Hamilton cycle, i.e. a cycle which visits every node of G exactly once. Now a certificate of G would be a cycle of G. To check whether this is a Hamilton cycle is simple, so HC belongs to NP. However, a polynomial-time algorithm that decides whether or not a Hamilton cycle exists has not yet been found. So it is not clear whether HC belongs to P or not. This describes a very famous open problem in mathematics, stated in the following conjecture:

Conjecture 4.3. $P \subsetneq NP$*, i.e. there exist problems in NP that are not solvable in polynomial time.*

It is widely believed that this conjecture is true. However, no proof has yet been found for this. Now look at the following definition.

Definition 4.4. *A problem* Π *is called* NP-hard *if any problem* Π' *in NP admits a polynomial time reduction to* Π*, i.e. using a subroutine for* Π *that is assumed to give a solution of* Π *in polynomial time, there is a polynomial time algorithm for solving* Π'*.*

So if we assume that a polynomial-time algorithm exists for Π, then we use this algorithm to develop a polynomial-time algorithm for Π' by transforming the input data. Intuitively, we can interpret this as that an NP-hard problem Π is harder than any problem in NP. Indeed, if an efficient algorithm exists for an NP-hard problem Π, then by definition efficient algorithms exist for all problems in NP, hence they are 'not harder' than Π.

Now assume the conjecture is true. Then NP contains problems which are not polynomial-time solvable, and therefore an NP-hard problem Π is not polynomial-time solvable as well, since it is harder than these problems. On the other hand, if the conjecture is false, finding a polynomial-time algorithm for Π would result in polynomial-time algorithms for every problem in NP, proving the converse of the conjecture. In other words, finding such an algorithm would solve this famous open problem immediately.

To prove that LRRP is NP-hard, first we specify a subproblem of LRRP. Then we give a mathematical representation of this subproblem, and then a polynomial-time reduction of PARTITION to this problem is given. Now PARTITION is NP-complete (NP-hard and in NP itself, proof can be found in [24]), so at least as hard as any problem in NP. Therefore, the subproblem of LRRP and so LRRP itself are NP-hard.

4.1 Specification of a Subproblem of the LRRP

We define the Special Loss Reduction Reconfiguration Problem (SLRRP) as the subproblem of LRRP in where we consider only a special kind of networks. Namely those distribution networks with only one slack bus, where all lines have strictly real impedances, and where all client busses are load demand busses with a purely active power demand (and no supply). These distribution networks have certain physical properties which will become useful, and which will be expressed in the following lemma's.

Lemma 4.5. *In a distribution system with one slack bus bus and only client busses with power demand of type constant power, the voltage magnitudes at the client busses will always be smaller than the voltage magnitude of the slack bus, and in case of multiple solutions of the power flow equations, the solution with values closest to the slack bus voltage is realized.*

Lemma 4.6. *In an alternating current circuit with purely real impedances and power demands, no phase differences occur at the nodal voltages and the line currents.*

We will not prove the above lemma's, since they are not strictly mathematical and beyond the scope of this project. But we will give a short explanation of their validity. Lemma 4.5 can be justified by looking at the physical situation in a load demand bus. A certain power S is demanded at a MV/LV transformer in this node. In reaction the voltage U in this node decreases such that the voltage difference between the node and the slack bus increases. Hence the current I flowing into the node increases, until the product $U \cdot I^*$ equals the demanded power S, and the system is balanced. This will happen at the first suitable voltage difference, hence the voltage closest to the slack bus voltage.

Phase differences occur as a consequence of the presence of condensators and coils. This presence is expressed in the imaginary parts of the impedances and power demands. These values being zero imply that no condensator- or coil-like behaviour is present in the network, hence no phase differences will occur. This explains lemma 4.6.

It is clear that once we prove this subproblem SLRRP to be NP-hard, the general problem is NP-hard as well; if we can solve the general problem LRRP in polynomial time, we can also solve the subproblem SLRRP in polynomial time, and hence by NP-hardness any problem in NP.

4.2 Mathematical Representation

In this section we define the Flow-Cost Minimal Spanning Tree problem (FCMST). We will see that this is the mathematical representation of SLRRP. In the next section, we will prove NP-hardness of FCMST.

FCMST: We describe the *Flow-Cost Minimal Spanning Tree* problem ($FCMST$) as the following search problem. An instance I of $FCMST$ consists of

- A directed graph $G = (W, E)$, with $|W| = n, |E| = m$.
- A special node $\omega \in W$ and height $H_\omega \in \mathbb{R}$. Set $W \backslash \{\omega\} = W_c$.
- A subset $E_o \subset E$ of optional edges. Set $E_f = E \backslash E_o$, the fixed edges.
- Weight function $\kappa : E \to \mathbb{R}$, demand function $D : W_c \to \mathbb{R}$, and capacity functions $C^{\min}, C^{\max} : W \to \mathbb{R},\ I^{\max} : E \to \mathbb{R}$.
- An objective function $L : \mathcal{P}(E_o) \to \mathbb{R} \cup \{\infty\}$.

For a subset $O \subset E_o$, compute height vector $h \in \mathbb{R}^n$ and flow vector $f \in \mathbb{R}^m$ as follows:

For $O \subset E_o$, construct *admittance matrix* $Y \in \mathbb{R}^{n \times n}$ ($E[v]$ are the edges in E connected to v):

$$Y_{v,w} = \begin{cases} \sum\limits_{e \in E_f[v]} \kappa(e) & \text{if } v = w \\ -\kappa(e) & \text{if } v \neq w \text{ and } (v,w) = e \in E \backslash O \\ 0 & \text{else} \end{cases}$$

Then h is the vector such that

$$h_\omega = H_\omega \tag{4.1}$$

$$\forall v \in W_c \qquad h_v \cdot (Y \cdot h)_v = D(v) \tag{4.2}$$

and satisfying the *flatness property*: For $v \in V_c$, the value h_v must be smaller than H_ω, and in case of multiple solutions for equation (4.2), the solution with the values as close as possible to the value h_w is realized.
Then $\forall e \in E\ f_e = \kappa(e) \cdot (h_v - h_w)$, where v is the head and w is the tail of e.

Now we have $L(O) = \infty$ if and only if O does not satisfy the following constraints:

1. *radiality*: The resulting graph $G_O = (V, E \backslash O)$ is a tree.

2. *Capacity*: $C^{\min}(v) \leq |h_v| \leq C^{\max}(v)$ for all $v \in V$ and $|f_e| \leq I^{\max}(e)$ for all $e \in E$.

The solution space $S(I)$ of instance I is the set of subsets $O \subset E_o$ such that $L(O)$ is minimal and less than ∞, i.e. O satisfies the radiality and capacity constraints.

Note that this search problem is indeed the mathematical representation of the physical reconfiguration subproblem $SLRRP$ described above. Node ω corresponds to the single slack bus and W_c to the client busses. E_o is the set of lines with switches, hence the optional lines. Then we have for each line e a real-valued admittance $\kappa(e)$ which is the inverse of the resistance. For each client bus v we have a real-valued load demand $D(v)$, for slack bus ω a real-valued voltage H_ω, and we have nodal voltage capacities $C^{\min}, C^{\max}$ and line current capacities $I^{\max}$. Further, we have an objective function L with an $O \subseteq E_o$ as input, which in SLRRP corresponds to the sum of the power losses through the lines in the configuration where O are the opened lines. Now the admittances and the load demands are all taken as real numbers, so by Lemma 2 the vectors h and f representing the nodal voltages and line currents will also be real-valued vectors. And the flatness property corresponds exactly to Lemma 1.

4.3 Reduction to PARTITION

We will now prove the NP-hardness of SLRRP by reducing the decision problem "PARTITION" to it. An instance of this problem is a set of weighted elements, which we want to split up in two subsets of equal weigth. Since PARTITION is NP-complete (NP-hard and in NP) and reducibility is transitive, this proves the claim.

PARTITION: An instance I of PARTITION is a set $A = \{a_1, ..., a_n\}$ and a weight function $w : A \to \mathbb{N}^+$. We want to know whether or not I admits a half-weight partition, i.e. a subset $A' \subset A$ with $w(A')(= \sum_{a \in A'} w(a)) = \frac{1}{2} w(A)$.

For an instance of PARTITION, construct an instance of FCMST as follows. For each $a_i \in A$ construct a node l_i. Then construct three other nodes: t_1, t_2, g. Set $\omega = g$, so $W_c = \{l_1, ..., l_n, t_1, t_2\}$. Then construct two fixed lines $p_1 = (g, t_1), p_2 = (g, t_2) \in E_f$ and for each $i \in \{1, ..., n\}, j \in \{1, 2\}$ the optional line $e_i^j = (t_j, l_i) \in E_o$. Set $E = E_o \cup E_f$. See figure 1.
Now we define the values of the capacity functions, the demand function, the height function

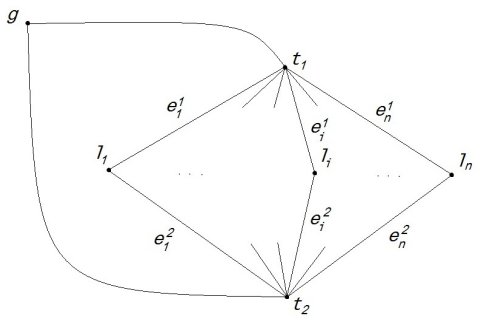

Figure 4.1: *The graph* $G = (W, E)$

.

and the weight function as:

$$
\begin{aligned}
\forall v \in W \qquad & C^{\min}(v) = 0 \\
& C^{\max}(g) = 2 \\
& C^{\max}(t_{1,2}) = \frac{3}{2} \\
\forall i = 1, ..., n \qquad & C^{\max}(l_i) = 1 \\
\forall i = 1, ..., n,\ j = 1, 2 \qquad & I^{\max}(e_i^j) = \frac{1}{2} w(a_i) \\
& I^{\max}(f_{1,2}) = \frac{\alpha}{2} \\
\forall i = 1, ..., n \qquad & D(l_i) = -\frac{1}{2} w(a_i) \\
& D(t_{1,2}) = 0 \\
& h_\omega = 2 \\
\forall i = 1, ..., n,\ j = 1, 2 \qquad & \kappa(e_i^j) = w(a_i) \\
& \kappa(p_{1,2}) = \alpha
\end{aligned}
$$

with $\alpha = \frac{1}{2} w(A)$. We can set the objective function L to any function we want, satisfying the mentioned property, since we will see that it will not play an actual role in the prove. Here we set $L(O) = 0$ if $O \subseteq E_o$ satisfies the radiality and capacity constraints, and $L(F) = \infty$ else.

Proposition 4.7. *An instance I of* PARTITION *admits a half-weight partition if and only if the constructed instance of* FCMST *has a non-empty solution space, i.e. there is an* $F \subset E_o$ *satisfying the constraints.*

This proposition implies that deciding whether a feasible configuration exists for a network, is already an NP-hard problem. Indeed, a subroutine that solves this problem is indirectly capable of solving PARTITION according to the proposition.

The underlying idea of the suggested construction and the proof is that a radial configuration of the constructed graph G corresponds to a partition of A. Indeed, if a configuration is radial, then each node l_i is connected to either t_1 or t_2. Then the impedances, voltages and power demands are chosen in such a way, that for $i = 1, 2$, the flow (current) through line p_i is exactly half the sum of the weights of the elements connected to t_i. And by chosing the capacities of p_1, p_2 equal to a quarter of the total weight of A, both sums must be equal to half of the total weight in order to satisfy the capacity constraint. So if a half-weight partition of A exists, then a feasible $O \subseteq E_o$ exists. And on the other hand, if no half-weight partition exists, then any $O \subseteq E_o$ satisfying one constraint will never satisfy the other constraint.

Proof. "⇒": Let $\delta \subset \{1, ..., n\}$ be an index set such that for $A_\delta = \{a_i | i \in \delta\}$ we have $w(A_\delta) = \alpha = \frac{1}{2}w(A)$. Define $S_1 = \{e_i^1 | i \in \delta\}, S_2 = \{e_i^2 | i \notin \delta\}$, and set $S = S_1 \cup S_2 \subset E_o$. *Then S satisfies the radiality constraint.*
The network G_S has two branches from g. We can compute the values of the height vector of the nodes connected to g via p_1 by solving system (4.2):

$$(Y \cdot h)_v = \sum_{(v,x)\in E} (h_v - h_x)\kappa((v, x)) \tag{4.3}$$

$$\Rightarrow (Y \cdot h)_{l_i} = w(a_i)(h_{l_i} - h_{t_1}) \tag{4.4}$$

$$\Rightarrow h_{l_i} \cdot (Y \cdot h)_{l_i} = w(a_i)h_{l_i}^2 - w(a_i)h_{t_1}h_{l_i} \tag{4.5}$$

$$= D(l_i) \tag{4.6}$$

$$= -\frac{1}{2}w(a_i) \tag{4.7}$$

$$\Rightarrow 0 = w(a_i)h_{l_i}^2 - w(a_i)h_{t_1}h_{l_i} + \frac{1}{2}w(a_i) \tag{4.8}$$

$$= h_{l_i}^2 - h_{t_1}h_{l_i} + \frac{1}{2} \tag{4.9}$$

since $w(a_i) \neq 0$. So for all l_i connected to t_1 only two values for the nodal voltages are available, namely the roots of (4.9). But by the flatness property, all voltages will take the same root as value, namely the one closest to h_ω. Say this root is r, then we have $h_{l_i} = r$ for all $i \in \delta$.
Now we look at t_1:

$$(Y \cdot h)_{t_1} = \kappa(p_1)(h_{t_1} - h_g) + \sum_{i\in\delta} \kappa(e_i^j)(h_{t_1} - h_{l_i}) \tag{4.10}$$

$$= \alpha(h_{t_1} - 2) + \sum_{i\in\delta} w(a_i)(h_{t_1} - r) \tag{4.11}$$

$$= \alpha(h_{t_1} - 2) + \alpha(h_{t_1} - r) \tag{4.12}$$

Now we must have $h_{t_1} \cdot (Y \cdot h)_{t_1} = D(t_1) = 0$, so $h_{t_1} = 0$, or $(Y \cdot h)_{t_1} = 0$. But if $h_{t_1} = 0$ then h_{l_i} will become complex due to (4.9). So $(Y \cdot h)_{t_1} = 0$. With (4.12) this results in $h_{t_1} = \frac{r}{2} + 1$. Now we can substitute this in (4.9):

$$\frac{r^2}{2} - r + \frac{1}{2} = 0 \tag{4.13}$$

$$\Rightarrow r = 1 \tag{4.14}$$

$$\Rightarrow \forall i \in \delta : h_{l_i} = 1, \text{ and } h_{t_1} = \frac{3}{2} \tag{4.15}$$

Now we can compute the flow vector values as follows:

$$f_{e_i^1} = W(e_i^1)(h_{l_i} - h_{t_1}) \tag{4.16}$$
$$= -\frac{1}{2}w(a_i) \tag{4.17}$$
$$f_{p_1} = W(p_1)(h_{t_1} - h_g) \tag{4.18}$$
$$= -\frac{1}{2}\alpha \tag{4.19}$$

So in this component of the network no capacities are exceeded. Note that we can compute h and f in the other component in exactly the same way by replacing δ with $\{1, ..., n\}\backslash\delta$, resulting in the same values. So also in the other component no capacities are exceeded. *Hence S satisfies the capacity constraint.* Hence $L(S) = 0$, so S is contained in the solution space.

"$\Leftarrow$": Let $S \subset E_o$ be such that $L(S) = 0$. Then S satisfies the radiality constraint, hence each l_i is connected to either t_1 or t_2 in G_S. Set $\delta_1 = \{i|e_i^1 \in S\}, \delta_2 = \{i|e_i^2 \in S\}$, and $A_1 = A_{\delta_1}, A_2 = A_{\delta_2}$. Then A_1, A_2 is a partition of A.
For $e_i^j \in S$ we have

$$|(Y \cdot h)_{l_i}| = |w(a_i)(h_{l_i} - h_{t_j})| = |f_{e_i^j}| \leq I^{\max}(e_i^j) = \frac{1}{2}w(a_i) \tag{4.20}$$
$$|h_{l_i}| \leq C^{\max}(l_i) = 1 \tag{4.21}$$
$$\text{but } h_{l_i} \cdot (Y \cdot h)_{l_i} = D(l_i) = -\frac{1}{2}w(a_i) \tag{4.22}$$

So $h_{l_i} = 1,\ (Y \cdot h)_{l_i} = -\frac{1}{2}w(a_i)$ or $h_{l_i} = -1,\ (Y \cdot h)_{l_i} = \frac{1}{2}w(a_i)$. Now together with (4.10) we have for $j = 1, 2$:

$$0 = D(t_j) \tag{4.23}$$
$$= h_{t_j} \cdot (Y \cdot h)_{t_j} \tag{4.24}$$
$$= h_{t_j}(\alpha(h_{t_j} - 2) + \sum_{i \in \delta_j} w(a_i)(h_{t_j} - h_{l_i})) \tag{4.25}$$
$$\Rightarrow \quad h_{t_j} = \frac{2\alpha + \sum_{i \in \delta_j} w(a_i)h_{l_i}}{\alpha + w(A_j)} \text{ or } h_{t_j} = 0 \tag{4.26}$$

Look at the solution $h_{l_i} = 1\ \forall i = 1, ..., n$ and $h_{t_j} = \frac{2\alpha + w(A_j)}{\alpha + w(A_j)},\ j = 1, 2$ of (4.22),(4.25). Since $w(A_j) \geq 0$, we have $\frac{2\alpha + w(A_j)}{\alpha + w(A_j)} \leq 2$. And this is the height vector with values as close as possible to $h_g = 2$, . So by the flatness property, this is the realized solution.
And we have $|h_{t_j}| \leq C^{\max}(t_j) = \frac{3}{2}$ for $j = 1, 2$, since S is a feasible switch specification. So $w(A_1), w(A_2) \leq \alpha$. But $w(A_1) + w(A_2) = w(A) = 2\alpha$, hence $w(A_1) = w(A_2) = \alpha$. Hence I admits a half-weight partition. □

Now with the above proposition we can formulate a correct many-one reduction from PARTITION to FCMST as follows:

1. Let $I =< A, w >$ be an instance of PARTITION.

2. Construct the instance I^* for FCMST as described above.

3. Use the assumed subroutine for FCMST to compute an $F \in S(I^*)$.

4. If such an F exists, then $S(I^*)$ is not empty, i.e. I^* has a feasible solution. Then I admits a half-weight partition by proposition 4.7. Return "I is yes-instance".

5. Else, $S(I^*)$ is empty, so no feasible solution exists. Hence I does not admit a half-weight partition by proposition 4.7. Return "I is no-instance".

The construction in step 2 can be done in linear time and the reduction is correct by the proposition. This proves that FCMST is an NP-hard problem. Therefore SLRRP is an NP-hard problem, and so the general problem LRRP is NP-hard as well.

Notice that we did not make use of the objective function for loss calculation specifically. Now if we change the problem to for instance load balancing or reliability, we only change the objective function. Therefore the given proof will be feasible for these problems as well. Hence the argument given above actually proves a more general theorem, namely that any distribution network reconfiguration problem is NP-hard.

Chapter 5

Approximation Algorithms for the Loss Reduction Reconfiguration Problem

Although the Loss Reduction Reconfiguration problem is by NP-hardness not likely to admit a polynomial-time algorithm that solves it, many approximation algorithms for this problem have been developed. For this project an extensive literature study was executed to investigate the developments in this field, which can be found in [32]. In 1975, Merlin and Back where the first to publish about profound ideas for network configurations in urban power distribution systems ([5]), and they lifted this field to a higher, more recognized level. Since then, scientists all over the world worked on network reconfiguration for loss reduction, with a large variety of publications as a result.

In this chapter several suggestions are given for (approximation) algorithms to solve LRRP. These suggestions are a selection of algorithms found in the literature, based on claimed results, comparison to other methods and clarity of the publications. Then eventually adaptations or extensions are added to these algorithms, and the idea of a 'brute force' calculation was added to the selection.

Remark: One important remark must be made. In many of the suggested algorithms it is assumed that the input network has only one slack bus. The descriptions and the flowcharts given will not work in case of multiple slack busses, since the algorithms are based in some way on Proposition 3.5 or 3.8. Only the Mixed Integer Linear Programming algorithm can also handle multiple slack bus networks as it is suggested now. However, in Chapter 6 a method is given to adapt a multiple slack bus network in such a way that all the suggested algorithms are suitable for this network. So for this chapter we will assume that a network has a single slack bus.

5.1 Edge Shifting Algorithm

One straight-forward approximation algorithm was developed by Baran and Wu ([6]) in 1989. It assumes for a network N to have an initial switch specification S. This switch specification is adjusted by replacing the open edges in S for their neighbours, and checking whether this

increases or decreases the total power loss. The replacement that induces the highest loss reduction is executed, and then the process is repeated. See Figure 5.1.

The loss reduction in step 2b can be computed with a simple quadratic equation, as described

Edge Shifting algorithm		
input	Network $N = (W, E)$ with impedances, powers, etc. And a starting switch specification $S \subseteq E$.	
1	Compute the power flow in N_S	
2	For all $e \in S$, do	
	a	Close e, specify arising cycle C in N_S
	b	Open $f \in C$ such that loss reduction ΔL_{ef} is maximal. Specify new network N_{ef} with e closed and f open
3	Find (e^*, f^*) among all pairs (e, f) such that $N_{e^*f^*}$ exceeds no capacities, and $\Delta L_{e^*f^*}$ is maximal	
4	If $\Delta L_{e^*f^*} \leq 0$, **return** S as the new switch specification	
5	Else, set $S = (S \cup \{f\}) \setminus \{e\}$ and go to step 2	

Figure 5.1: Edge Shifting algorithm

in [6]. Once no strict loss reduction is found, the algorithm stops and returns the resulting switch specification.

This is a fast, simple approximation algorithm that needs only one power flow calculation, which is a time-consuming calculation. However, an initial configuration is needed, and it cannot be assured that the process will converge to a global optimum. Only a local optimum can be assured. Furthermore, this algorithm can return only one switch specification, which is constructed in a deterministic way, hence the output will always be the same for a fixed input.

At Liander NV, the distribution networks are modelled in a program called "Vision". Now Vision has a tool which gives for a network, a new configuration in which the power loss is reduced. It turns out that this tool is based on an algorithm almost the same as the Edge Shifting algorithm. Only in Vision, no capacities of the nodes and lines are checked during the process, which can result in an unfeasible configuration as output.

Nevertheless, the Edge Shifting algorithm will not be implemented in this project, although we will compare the results of the Vision tool with the results of the final, most suitable method.

5.2 Greedy Demeshing Algorithm

The second approximation algorithm also dates back to 1989. It was developed by Shirmohammadi and Hong ([7]) and is based on the fact that in a meshed network (all switches closed) the power flow is balanced in the most thrifty way, with as little power loss as possible. By opening those edges that effect this pattern the least, it is tried to find the configuration that looks as much as the meshed pattern as possible.

The algorithm works with a greedy principle. Initially, al switches are closed, and then

in an iterative process the optional edge is opened which effects the current power flow as little as possible. This continues until enough edges are opened to make the configuration radial. See Figure 5.2.

Note that when an edge is not on a cycle, opening this edge results in a disconnected

Greedy Demeshing algorithm	
input	Network $N = (W, E)$ with impedances, powers, etc.
1	Set $S = \emptyset$, and compute the power flow in $N_S = N$
2	Set all edges that are not on a cycle in N_S permanent as non-optional
3	Compute the *optimal line currents:* $I^S_{e_1}, \ldots, I^S_{e_m}$
4	Find optional line e^* such that $I^S_{e^*}$ is minimal, and set $S^* = S \cup \{e^*\}$
5	Compute the power flow in N_{S^*}
6	If capacities are exceeded, set e^* temporarily as a non-optional line, go to step 4
7	Else, set $S = S^*$
8	If $\|S\| = \|E\| - \|W\| + 1$, then N_S is radial. **Return** S as the near-optimal switch specification
9	Else, set all temporary non-optional lines optional again and go to step 2

Figure 5.2: Greedy Demeshing algorithm

network. Since we assume that N has only one slack bus, this counters the radiality constraint. Therefore it is justified to consider only optional edges that are contained by a cycle.

In [7], a method is described to calculate the optimal line currents, which can be done by solving a linear system. These line currents are based on the nodal currents resulting from the power flow equations, computed in the network with only the resistances, and not the reactances. As proven in [7], by taking these currents into account for the opening selection, the power flow pattern in the meshed network is effected as less as possible. Now if enough edges are opened, the network will be radial. Since capacities are checked in every iteration, the resulting configuration is feasible by construction.

The Greedy-Demeshing algorithm is a fast and simple algorithm as well, needing a relatively small number of iterations. However, as with the Edge Shifting algorithm, only convergence to a local optimum can be assured. And the process is deterministic as well, so a certain input will always result in one and the same configuration.

5.3 Harmony Search Algorithm

The *harmony search algorithm* is an optimization algorithm of Zong Woo Geem, inspired by the process of, say n, musicians with different instruments when making a pleasant harmony. The idea is to update a harmony memory in an iterative process. A new harmony, which is a specification for every instrument which note should be played, is constructed based on experience (by a certain probability, picking a note for an instrument that is in the harmony) and improvisation (by a certain probability, adjusting this note a bit up or down the bandwidth of the instrument). Then is checked whether this new harmony is better than the worst harmony in the memory. If so, the latter is replaced by the former, and the process repeats. In this way the harmony memory gets better and better, and when a maximum iteration is reached, the best harmony in the memory is given as output.

In [12], an implementation is given of the harmony search algorithm for the power loss reconfiguration problem, by Rao et al. A harmony S corresponds to a radial configuration, and the fitness function is given as

$$L(S) = \sum_{e \in E} (|I_e^S|^2 \cdot R_e + \beta_1 \max\{0, |I_e^S| - |I_e^{\max}|\}) \tag{5.1}$$

$$+ \sum_{v \in W} (\beta_2 \max\{0, |U_v^{\min}| - |U_v^S|\} + \beta_3 \max\{0, |U_v^S| - |U_v^{\max}|\}) \tag{5.2}$$

Here $\beta_1, \beta_2, \beta_3$ are large numbers, counting as penalty factors for every capacity that is exceeded in N_S. So the fitness of a configuration is the total power loss plus penalties for capacity exceeding.

However, in [12] it is not clear at all how the instruments should be chosen. Therefore, based on the publication and on Propositions (3.5) and (3.8) in chapter 3, two suggestions are made for this. One can take the cycles in a cycle basis of the network as instruments, and the edges on the cycles as notes. Or one can take the ears in a semi-ear decomposition as instruments, again with the edges as notes. Now a harmony, which is a selection of one edge in each cycle/semi-ear, corresponds to that configuration of N in which the chosen edges are the opened edges. In the first case, by Proposition (3.5) any radial configuration of a network N can be constructed as a harmony in the Harmony Search Algorithm. However, as mentioned, also non-radial configurations can be constructed. Therefore an extra subroutine has to be build into the algorithm to check whether a harmony corresponds to a radial configuration or not. And if not, this harmony must be abandoned. See Figure 5.3.

In the second case, by Proposition (3.8) a harmony will always correspond to a radial configuration. However, as mentioned, not all radial configurations can be constructed with one semi-ear decomposition. Therefore, more than one decomposition must be considered to get a complete search space. See Figure 5.4.

In both versions, $HMCR$ and PAR are two numbers between 0 and 1, representing the *harmony memory consideration rate* and the *pitch adjustment rate*, respectively. The first gives the probability that an edge must be chosen from the harmony memory instead of random from the full bandwidth, where the second gives the probability that this edge must be replaced by its neighbour on the corresponding cycle or semi-ear.

Due to the improvisation aspect, the process of the Harmony Search Algorithm ensures convergence to a global optimum. Moreover, this algorithm provides the opportunity to return not only the best configuration found, but more than one good configuration can be returned. This is highly advantageous when we want to use this method as a tool for netplanners, since they would like to have a list of switch specifications of which they can choose the most suitable one, based on their considerations. However, disadvantage is that this algorithm needs a power flow calculation in every iteration. Since the number of iterations needed for good results can be high, this makes the method slower than the previous ones.

In [13], the given algorithm is extended in the presence of distributed generation. Given

Harmony Search Algorithm (Cycle Basis version)		
input	Network $N = (W, E)$ with impedances, powers, etc.	
1	Compute a cycle basis $B = \{C_1, \dots, C_r\}$ of N	
2	Fill *Harmony Memory* (HM) with random radial switch specifications S:	
	a	Set $S = \emptyset$
	b	For each cycle C_i in B, choose edge e on C_i and add to S
	c	If N_S is radial, add S to HM, compute fitness $f(S)$
	d	If HM is not totally filled, go to step 2a
3	Set $x = 0$, $S = \emptyset$	
4	Construct new switch specification: For $i = 1, \dots, r$ do	
	a	If *Random()*<*HMCR*, choose e_i from all edges in HM chosen for C_i
	b	Else, choose e_i from C_i arbitrary
	c	If *Random()*<*PAR*, adjust e_i by choosing an edge f on C_i next to e_i. Set $e_i = f$
	d	Add e_i to S
5	If N_S is not radial, set $S = \emptyset$ and go to step 4	
6	If $L(S)$ is smaller than the highest fitness in HM (belonging to, say, S^*), delete S^* from HM and add S to HM	
7	Set $x = x + 1$, $S = \emptyset$	
8	If $x < x_{max}$, go to step 4	
9	Else, **return** the best switch specification in HM (the switch specification with the smallest fitness	

Figure 5.3: Harmony Search Algorithm (version 1)

a certain number of b distributed generators on fixed places in the network, a harmony is extended with b instrument values, namely the generated powers at each distributed generator. This makes it possible to examine what are the best places for distributed generation, but also how to configure the network and maybe tune the generators in a given situation with fixed distributed generators. This will not be taken into account during this project, but this might be interesting for future research.

5.4 Genetic Algorithm

The *genetic algorithm* is an optimization algorithm inspired by the idea of evolution and genetic improvement. In an iterative process, new generations of individuals are created from old ones via the principle of evolution, where the fitness of the individuals gets better and better. Two individuals from the old generation, represented by a string of genes called a chromosome, are chosen to mate and reproduce. This selection is based on their fitness, the higher the fitness, the more likely an individual is to mate. Then these individual strings are both cut in two parts and interchanged, producing two new individuals. Finally, by a certain probability, mutation takes place at some genes. When the new generation size has reached the population limit, it becomes the old generation from which again a new generation is build. This repeats until a maximum generation number is reached, and then the best individual in the newest generation is returned.

This algorithm shows much similarity with the harmony search algorithm. New individuals are constructed based on experience (cross-over of old individuals) and improvisation (mutation). In [31] an argumentation can be found that indeed the harmony search algo-

Harmony Search Algorithm (Semi-Ear Decomposition version)			
input	Network $N = (W, E)$ with impedances, powers, etc.		
1	Compute a cycle basis $B = \{C_1, \dots, C_r\}$ of N, and set $T = \emptyset$		
2	Compute enough semi-ear decompositions $D^1, \dots, D^k$ out of B		
3	For $j = 1, \dots, k$, say $D^j = \{O_1, \dots, O_r\}$, set *Harmony Memory* (HM) $= \emptyset$ and do:		
	a	Fill HM with random radial switch specifications S:	
		i	Set $S = \emptyset$
		ii	For each semi-ear O_i in D^j, choose edge e on O_i and add to S
		iii	Add S to HM, compute fitness $f(S)$
		iv	If HM is not totally filled, go to step 2a
	b	Set $x = 0$, $S = \emptyset$	
	c	Construct new switch specification: For $i = 1, \dots, r$ do	
		i	If *Random()*<*HMCR*, choose e_i from all edges in HM chosen for O_i
		ii	Else, choose e_i from O_i arbitrary
		iii	If *Random()*<*PAR*, adjust e_i by choosing an edge f on O_i next to e_i. Set $e_i = f$
		iv	Add e_i to S
	d	If $L(S)$ is smaller than the highest fitness in HM (belonging to, say, S^*), delete S^* from HM and add S to HM	
	e	Set $x = x + 1$, $S = \emptyset$	
	f	If $x < x_{max}$, go to step 3a	
	g	Else, add to T the best switch specification in HM (the switch specification with the smallest fitness	
4	**Return** the switch specification in T with the smallest fitness		

Figure 5.4: Harmony Search Algorithm (version 2)

rithm, which was developed when the genetic algorithm was already mature and widely used, is just a subclass of the genetic algorithm. More about this later in this section.

In [10], an implementation of the genetic algorithm is given for the power loss reconfiguration problem. In [11], this implementation is improved on some details. A short description of the described algorithm follows. However, this will not be the version we will work with.

In a network $N = (W, E)$, because of the radiality constraint, each feasible configuration has a fixed number of switches set open. Say this number is r, and $m = |E|$. Then an individual is defined as a switch specification S, represented by a binary string $b = (b_1, ..., b_l)$ with length $l = r \cdot \lceil \log_2(m) \rceil$. Here $b_1, ..., b_{\lceil \log_2(m) \rceil}$ give the binary representation of the edge number of the first open switch S_1, $b_{\lceil \log_2(m) \rceil + 1}, ..., b_{2\lceil \log_2(m) \rceil}$ the edge number of the second open switch S_2, and so on.

Almost identical to the Harmony Search Algorithm, the fitness function will be the penalty function $f(S) = \frac{1}{L(S)}$ with

$$L(S) = \sum_{e \in E} (|I_e^S|^2 \cdot R_e + \beta_1 \max\{0, |I_e^S| - |I_e^{\max}|\}) \tag{5.3}$$

$$+ \sum_{v \in W} (\beta_2 \max\{0, |U_v^{\min}| - |U_v^S|\} + \beta_3 \max\{0, |U_v^S| - |U_v^{\max}|\}) \tag{5.4}$$

as earlier. We want to minimize $L(S)$, hence maximize $f(S)$. In contrast to the Harmony Search Algorithm, in the Genetic Algorithm we specifically want to maximize the fitness,

since higher fitness of an individual gives a higher probability of mating and reproducing.

Now the principle of the genetic algorithm can be applied to these switch specification strings, corresponding to the individuals, and the fitness function. However, the implementation of this method in "R" shows extremely bad behaviour. It will be clear that a new string constructed by gluing two parts of two old strings representing radial configurations, does not have to represent a radial configuration itself, or a well-defined switch specification at all, since a slight change in a binary string of a number can result in a big change of the number itself. As it turns out, the probability that a new individual represents a radial configuration is very small, making this method very slow and inefficient, and not suitable for this project.

However, we can apply the method for the Harmony Search Algorithm also to the Genetic Algorithm. Indeed, if we assign every cycle in a cycle basis or every ear in a semi-ear decomposition to a gene place, we can choose a gene to be an edge from the corresponding cycle or semi-ear. Then an individual is a string of length $r = |E| - |V| + 1$ corresponding to a switch specification S for the network. The resulting algorithms are described in Figure 5.5 and 5.6.

Here COR and MR are the *cross-over rate* and the *mutation rate*, respectively. The

Genetic Algorithm (Cycle Basis version)				
input	Network $N = (W, E)$ with impedances, powers, etc.			
1	Compute a cycle basis $B = \{C_1, \dots, C_r\}$ of N			
2	Fill *Old Generation* (OG) with random radial switch specifications S:			
	a	Set $S = \emptyset$		
	b	For each cycle C_i in B, choose edge e on C_i and add to S		
	c	If N_S is radial, add S to OG, compute fitness $f(S)$		
	d	If OG is not totally filled, go to step 2a		
3	Set $x = 0$			
4	Set *New Generation* (NG)= 0 and construct NG from OG:			
	a	Choose A, B from OG with chance proportional to their fitness		
	b	If *Random()*<*COR*, cut A and B in two parts A', A'', B', B'' at the same cutting point. Then set $A = A' \cup B'', B = B' \cup A''$		
	c	For $S = A, B$, do		
		i	If *Random()*<*MR*, adjust some e_i in S by chosing edge f on C_i next to e_i and set $e_i = f$	
		ii	If N_S is radial, add S to NG and compute fitness $f(S)$	
	d	If NG is not totally filled, go to step 4a		
5	$x = x + 1$, OG=NG			
8	If $x < x_{max}$, go to step 4			
9	Else, **return** the best switch specification in OG (the switch specification with the highest fitness)			

Figure 5.5: Genetic Algorithm (version 1)

first gives the probability that two individuals mix their chromosomes for producing new individuals, or just 'clone' themselves. The second gives the probability that mutation takes place at some genes, which will adjust the gene a bit up or down the bandwidth. In [11] it is suggested that the mutation rate should be depending on the development in the algorithm, in the following way: Set $f_{min}(k)$ the minimal value of the fitness values f_i in generation k.

Genetic Algorithm (Semi-Ear Decomposition version)					
input	Network $N = (W, E)$ with impedances, powers, etc.				
1	Compute a cycle basis $B = \{C_1, \dots, C_r\}$ of N, and set $T = \emptyset$				
2	Compute enough semi-ear decompositions $D^1, \dots, D^k$ out of B				
3	For $j = 1, \dots, k$, say $D^j = \{O_1, \dots, O_r\}$, set *Old Generation* (OG) $= \emptyset$ and do:				
	a	Fill *Old Generation* (OG) with random radial switch specifications S:			
		i	Set $S = \emptyset$		
		ii	For each cycle O_i in B, choose edge e on O_i and add to S		
		iii	Add S to OG, compute fitness $f(S)$		
		iv	If OG is not totally filled, go to step 2a		
	b	Set $x = 0$			
	c	Set *New Generation* (NG)$= 0$ and construct NG from OG:			
		i	Choose A, B from OG with chance proportional to their fitness		
		ii	If *Random()*<*COR*, cut A and B in two parts A', A'', B', B'' at the same cutting point. Then set $A = A' \cup B'', B = B' \cup A''$		
		iii	For $S = A, B$, do		
			1	If *Random()*<*MR*, adjust some e_i in S by chosing edge f on C_i next to e_i and set $e_i = f$	
			2	Add S to NG and compute fitness $f(S)$	
	d	If NG is not totally filled, go to step 3ci			
	e	$x = x + 1$, OG=NG			
	f	If $x < x_{max}$, go to step 3c			
	g	Else, add to T the best switch specification in HM (the switch specification with the smallest fitness			
4	**Return** the switch specification in T with the smallest fitness				

Figure 5.6: Genetic Algorithm (version 2)

Then the mutation rate in the constructing generation $k+1$ is suggested to be

$$r_m(k+1) = \begin{array}{ll} r_m(k) - r_{\text{step}} & \text{if } f_{\min}(k) \geq f_{\min}(k-1) \\ r_m(k) & \text{if } f_{\min}(k) < f_{\min}(k-1) \\ r_{\text{final}} & \text{if } r_m(k) = r_{\text{final}} \end{array} \tag{5.5}$$

with in [11] the values $r_m(0) = 1, r_{step} = 0.001, r_{final} = 0.05$. So if the whole generation gets better, the mutation rate goes down. This suggestion will be taken into account during the implementation of the Genetic Algorithm.

Note that the same disadvantages hold for these algorithms as for the versions of the Harmony Search Algorithm. According to Propositions (3.5) and (3.8), in the cycle basis version also non-radial configurations can be represented, whereas in the semi-ear decomposition version more than one decomposition is needed to cover all radial configurations. Again, the algorithms are adapted to these circumstances. Furthermore, as with the Harmony Search Algorithm, due to the mutation factor the process in this algorithm will also converge to a global optimum. And again, the Genetic Algorithm can easily return more than one good configuration. However, again for every iteration a power flow calculation is needed, which makes the method rather slow.

As mentioned earlier, and as will be clear from the descriptions, the Harmony Search Algorithm and the Genetic Algorithm show much similarity. And both seem very well suitable

for the Loss Reduction Reconfiguration Problem, since the influence of one opened edge on the total power loss is dependent of the other opened edges, but in a rather smooth way. So it will be interesting which algorithm shows better behaviour when tested on realistic networks.

5.5 Mixed Integer Linear Programming

Jabr et al. transform in [14] the power loss configuration problem into a linear convex optimization problem. These kind of problems are well-known and good methods to solve them are available.

In [14] it is assumed that every edge $e \in E$ of network $N = (W, E)$ contains a switch, with switch specification $s_e \in \{0, 1\}$. We assume that we know the power demand $P_k + jQ_k$ at the load busses, the real powers and voltage magnitudes at the generator busses, the voltages at the slack busses (of which can be multiple!), and the impedance $Z_e = R_e + jX_e$ at each line e. Now we define g_e and b_e as the real and imaginary part of the admittance at e, hence $g_e = \frac{R_e}{|Z_e|^2}, b_e = \frac{-X_e}{|Z_e|^2}$. Now say $W = W_n \cup W_g \cup W_l$, the set of slack busses, generator busses and load busses respectively. The Loss Reduction Reconfiguration Problem is then formulated in [14] as (using notation as in Chapter 2 and 3, and $N(k)$ the set of neighbours of k):

$$\min \sum_{e \in E} |I_e|^2 R_e \text{ subject to} \tag{5.6}$$

$$s \in \{0,1\}^{|E|},\ \phi, |U| \in \mathbb{R}^{|W|} \tag{5.7}$$

$$N_s \text{ is radial} \tag{5.8}$$

$$\forall k \in W: \qquad |U_k^{\min}| \leq |U_k| \leq |U_k^{\max}| \tag{5.9}$$

$$\forall k \in W_g \cup W_l: \sum_{l \in N(k)} s_{kl}(|U_k|^2 g_{kl} - |U_k||U_l|(g_{kl}\cos(\phi_k - \phi_l) + b_{kl}\sin(\phi_k - \phi_l))) = P_k \tag{5.10}$$

$$\forall k \in W_l: \qquad \sum_{l \in N(k)} s_{kl}(-|U_k|^2 b_{kl} + |U_k||U_l|(b_{kl}\cos(\phi_k - \phi_l) - g_{kl}\sin(\phi_k - \phi_l))) = Q_k \tag{5.11}$$

$$\forall (k,l) \in E: |I_{kl}|^2 = s_{kl}\left(\frac{|U_k - U_l|^2}{|Z_{kl}|^2}\right) \tag{5.12}$$

$$= s_{kl}((g_{kl}^2 + b_{kl}^2)(|U_k|^2 + |U_l|^2 - 2|U_k||U_l|\cos(\phi_k - \phi_l)))$$

$$\leq |I_{kl}^{\max}|^2 \tag{5.13}$$

Here $s_{kl} = 1$ implies that edge (k, l) is closed, hence contained in the configuration. In this way, the power flow study is otiose, since constraint (5.10) and (5.11) are actually the power flow equations as in chapter 2. So the demand constraint is satisfied. (5.8) represents the radiality constraint, and (5.9), (5.13) the capacity constraint. In [14] it is proven that this optimization problem is equivalent to the following convex, almost linear optimization problem (details of the transformation can be found in [14]):

$$\min \sum_{e\in E} |I_e|^2 R_e \text{ subject to} \tag{5.14}$$

$$\forall (kl) \in E: \ \beta_{kl}, \beta_{lk} \in \{0,1\}; \forall e \in E: \ Y_e, T_e \in \mathbb{R}; \forall k \in W: \ u_k \in \mathbb{R} \tag{5.15}$$

$$\forall k \in W_g \cup W_l: \ \sum_{l\in N(k)} \beta kl = 1 \tag{5.16}$$

$$\forall k \in W_n, \ l \in N(k): \ \beta kl = 0 \tag{5.17}$$

$$\forall (kl) \in E: \ 0 \leq \beta_{kl} + \beta_{lk} = s_{kl} \leq 1 \tag{5.18}$$

$$\forall k = 1, ..., n: \ \frac{|U_k^{\min}|^2}{\sqrt{2}} \leq u_k \leq \frac{|U_k^{\max}|^2}{\sqrt{2}} \tag{5.19}$$

$$\forall (kl) \in E: \ 0 \leq u_k^{kl} \leq \frac{|U_k^{\max}|^2}{\sqrt{2}} s_{kl} \tag{5.20}$$

$$\forall (kl) \in E: \ 0 \leq u_l^{kl} \leq \frac{|U_l^{\max}|^2}{\sqrt{2}} s_{kl} \tag{5.21}$$

$$\forall (kl) \in E: \ 0 \leq u_k - u_k^{kl} \leq \frac{|U_k^{\max}|^2}{\sqrt{2}} (1 - s_{kl}) \tag{5.22}$$

$$\forall (kl) \in E: \ 0 \leq u_l - u_l^{kl} \leq \frac{|U_l^{\max}|^2}{\sqrt{2}} (1 - s_{kl}) \tag{5.23}$$

$$\forall k = 0, ..., n: \ \sum_{l\in N(k)} s_{kl}(g_{kl} u_k^{kl} \sqrt{2} - g_{kl} Y_{kl} - b_{kl} T_{kl}) = P_k \tag{5.24}$$

$$\forall k = 1, ..., n: \ \sum_{l\in N(k)} s_{kl}(-b_{kl} u_k^{kl} \sqrt{2} + b_{kl} Y_{kl} - g_{kl} T_{kl}) = Q_k \tag{5.25}$$

$$\forall (kl) \in E: \ |I_{kl}|^2 = (g_{kl}^2 + b_{kl}^2)((u_k^{kl} + u_l^{kl})\sqrt{2} - 2Y_e) \leq |I_{kl}^{max}|^2 \tag{5.26}$$

$$\forall (kl) \in E: \ 2u_k^{kl} u_l^{kl} \geq Y_{kl}^2 + T_{kl}^2, \ Y_{kl} \geq 0 \tag{5.27}$$

Equations (5.19), (5.24), (5.25), (5.26) can be obtained by substituting

$$u_k = \frac{|U_k|^2}{\sqrt{2}} \tag{5.28}$$

$$Y_{kl} = |U_k||U_l| \cos(\phi_k - \phi_l) \tag{5.29}$$

$$T_{kl} = |U_k||U_l| \sin(\phi_k - \phi_l) \tag{5.30}$$

into (5.9)-(5.13). Equations (5.20)-(5.23) are used to distinguish between open and closed edges. (5.16)-(5.18) represent the radiality constraint. Equation (5.27) is the only nonlinear constraint. However, in [14] a method of Ben-Tal and Nemirovski is used that approximates this constraint by a system of linear constraints. Details can be found in [25], and we will only state the results here.

Choose $k \in \mathbb{N}$. Define $2(k+1)$ variables $a_0, ..., a_k, b_0, ..., b_k$. Then the cone defined by equation

$x_3 \geq \sqrt{x_1^2 + x_2^2}$ can be approximated by the polyhedron defined by

$$a_0 \geq |x_1| \tag{5.31}$$
$$b_0 \geq |x_2| \tag{5.32}$$
$$\forall\ j = 1, ..., k: \quad a_j = \cos(\frac{\pi}{2^{j+1}})a^{j-1} + \sin(\frac{\pi}{2^{j+1}})b_{j-1} \tag{5.33}$$
$$\forall\ j = 1, ..., k: \quad b_j \geq |-\sin(\frac{\pi}{2^{j+1}})a^{j-1} + \cos(\frac{\pi}{2^{j+1}})b_{j-1}| \tag{5.34}$$
$$a_k \leq x_3 \tag{5.35}$$
$$b_k \leq \tan(\frac{\pi}{2^{j+1}})a^k \tag{5.36}$$

The maximal error of this approximation is

$$\epsilon(k) = \frac{1}{\cos(\frac{\pi}{2^{k+1}})} - 1 \tag{5.37}$$

and in [14] is suggested to take $k = 11$, resulting in an error $\epsilon(k) \approx 6 \times 10^{-7}$. Now constraint (5.27) can be described as

$$\frac{u_k^{kl} + u_l^{kl}}{\sqrt{2}} \geq \sqrt{r_k^2 l + (\frac{u_k^{kl} - u^k l_l}{\sqrt{2}})^2} \tag{5.38}$$
$$r_{kl} \geq \sqrt{Y_{kl}^2 + T_{kl}^2} \tag{5.39}$$

which both can be linearized as described. Note that this linearization, for $k = 11$, adds $4|E|(k+1)$ variables and $4|E|(k+2)$ constraints. Now an implementation of this algorithm could be as described in figure 5.7.

The advantage of this method is that we can use available linear mixed-integer optimization

Mixed Integer Linear Programming	
input	Network $N = (W, E)$ with impedances, powers, etc.
1	Construct constraint matrix A for N
2	Use a *mixed integer linear optimization* tool to solve the corresponding program
3	**Return** the resulting switch specification

Figure 5.7: Mixed Integer Linear Programming

methods to solve the problem. Very good methods exist, and the field is still developing. Also, most of these methods converge to global optima. Further, the transformation given is exact, no stochastics are involved. A disadvantage is the high complexity of this method. It does not seem flexible for adaptations or extra requirements, and it will return only one solution.

5.6 Brute Force Calculation

The most basic algorithm for solving the Loss Reduction Reconfiguration Problem exactly would be to calculate the power loss in any radial configuration, and then return the configuration in where this loss is minimized. We will call this a Brute Force Calculation. Since

LRRP is NP-hard, this is certainly not a polynomial-time algorithm, which implies that the running time depends at least exponentially on the size of the input network N. However, it might be the case that the size of a standard MV network in Liander management is not too big, so that the Brute Force Calculation will give an output within acceptable time. If so, this algorithm would be highly preferable over the previous ones, since it will certainly return the global optimum. Moreover, it can return a multiple number of good configurations, namely the best configurations of the network. Therefore this method will be implemented as well, and the running time will be compared to those of the other methods.

For the implementation of the Brute Force Calculation again we will use Proposition (3.5). We want to investigate all radial configurations of an input network N. However, we do not know in advance which configurations are radial and which are not. So instead of checking all possible configurations on radiality, we will calculate a cycle basis B of N. Then we can decrease the search space by investigating only those configurations that can be constructed from B by opening one edge on every cycle in B. As Proposition (3.5) implies, all radial configurations will be investigated this way. See Figure 5.8 for a full description of the implementation.

Brute Force Calculation		
input	Network $N = (W, E)$ with impedances, powers, etc.	
1	Compute a cycle basis $B = \{C_1, \dots, C_r\}$ of N and set *memory* $M = \emptyset$	
2	For each combination $S = \{e_1, \dots, e_r\}$ with $e_i \in C_i$, do:	
	a	Check radiallity of N_S
	b	If N_S is radial, compute the power flow in N_S
	c	If no capacities are exceeded, compute the total power loss $L(S)$ in N_S and add this to M
3	**Return** S in M with the lowest total power loss	

Figure 5.8: Brute Force Calculation

Chapter 6

Expanding the Algorithms to Multiple Slack Bus Situations

All of the suggested methods in Chapter 5, except for the Mixed-Integer Linear Programming, are developed for single slack bus networks. When more than one slack bus is present, these algorithms will fail to achieve satisfying results. The critical aspect is that these algorithms focus on opening all cycles in a network. However, as Proposition 3.5 implies, this results in a spanning tree of the network, which is only a radial configuration if the network has one slack bus. In case of multiple slack busses, opening the cycles is not enough. Then also paths from one slack bus to another must be opened. Hence an output configuration will never be radial.

The functionality of these algorithms is decreased extremely due to this, since we are also interested in larger networks with several slack busses. With the current algorithms, we have to divide such a network into subnetworks by separating the slack busses before we can use the algorithms. And this division cannot be based on these algorithms, but has to be done manually.

Therefore a method is proposed that adapts a multiple slack bus network in such a way that all the suggested algorithms can be used to find or approximate the optimal configuration of this network. Moreover, this optimal configuration corresponds to the optimal configuration of the original network in a clear way. The method is based on a mathematical proposition which will be stated and proved in Section 6.1. In Section 6.2 it will be described how this method can be used for the suggested algorithms.

6.1 Mathematical Construction

Let $N = (W, E)$ be a network with $W = W_n \cup W_c$, W_n the set of slack busses, and W_c the set of client busses. Say $|W_n| = n'$, so N has n' slack busses. Set $E = E_f \cup E_o$, the set of fixed and optional lines respectively. Furthermore, assume the input data such as impedances, voltages for the slack busses, powers for the load busses and voltage magnitudes/active powers for the generator busses are present.

Now construct network $\hat{N} = (\hat{W}, \hat{E})$ out of N by adding one artificial load bus a to W_c,

and for any slack bus $v \in W_n$, add a fixed line $f_v = (v, a)$ to E_f. For load bus a, set active and reactive power demands equal to zero, so $P_a, Q_a = 0$. And for all added edges f_v, set the impedance $Z_{f_v} = 0 + j \cdot \infty$, so the admittance $A_{f_v} = \frac{1}{Z_{f_v}} = 0$. Then we have the following proposition:

Proposition 6.1. *For network N, and network $\hat{N}$ constructed out of N as described above, the power flow in N and $\hat{N}[W]$ will be equal. That is, for any $v \in W$, if U_v, S_v are the voltage and power in v in N, and $\hat{U}_v, \hat{S}_v$ are the voltage and power in v in $\hat{N}$, then $U_v = \hat{U}_v$ and $S_v = \hat{S}_v$. Furthermore, the total power loss in N and $\hat{N}$ is equal.*

Proof. Say N has k slack busses $v_1, ..., v_k$, and set $f_{v_i} = f_i$ for all $i = 1, ..., k$. Let $\mathcal{A}$ be the admittance matrix of N. Then the admittance matrix of $\hat{N}$ will be

$$\hat{\mathcal{A}} = \begin{pmatrix} & & -A_{f_1} \\ & & \vdots \\ & \mathcal{A} & -A_{f_k} \\ & & 0 \\ & & \vdots \\ & & 0 \\ -A_{f_1} \cdots - A_{f_k} & 0 \cdots 0 & (A_{f_1} + ... + A_{f_k}) \end{pmatrix} \tag{6.1}$$

$$= \begin{pmatrix} & & 0 \\ & \mathcal{A} & \vdots \\ & & 0 \\ 0 \cdots 0 & & 0 \end{pmatrix} \tag{6.2}$$

Now let $\hat{U} = \begin{pmatrix} \hat{V} \\ \hat{U}_a \end{pmatrix}$ and $\hat{S} = \begin{pmatrix} \hat{T} \\ \hat{S}_a \end{pmatrix}$ be the vectors of voltages and powers in $\hat{N}$, with $\hat{V}$ and $\hat{T}$ the vectors of voltages and powers in nodes in W, and $\hat{U}_a, \hat{S}_a$ the voltage and power in a. Then

$$\hat{\mathcal{A}} \cdot \hat{U} = \begin{pmatrix} \mathcal{A} \cdot \hat{V} \\ 0 \end{pmatrix} + (0 \dots 0) \cdot \hat{U}_a \tag{6.3}$$

$$= \begin{pmatrix} \mathcal{A} \cdot \hat{V} \\ 0 \end{pmatrix} \tag{6.4}$$

So for $w \in W$, we have the power flow equation in $\hat{N}$:

$$\hat{S}_w = \hat{T}_w = \hat{U}_w \cdot (\hat{\mathcal{A}} \cdot \hat{U})^*_w \tag{6.5}$$

$$= \hat{V}_w \cdot (\mathcal{A} \cdot \hat{V})^*_w \tag{6.6}$$

since $\hat{V}$ is the vector of voltages in nodes in W. But this gives exactly the same equations as the power flow equations in N, with voltage vector U and power vector S, since $S_w = U_w \cdot (\mathcal{A} \cdot U)^*_w$ and the input voltages and powers are equal. Furthermore, we have the equation for a:

$$\hat{S}_a = \hat{U}_a \cdot (\hat{\mathcal{A}} \cdot \hat{U})^*_a \tag{6.7}$$

$$= \hat{U}_a \cdot 0 \tag{6.8}$$

$$= 0 \tag{6.9}$$

This equation is correct since the demanded power in a is set to zero. Moreover, this equation is correct for any U_a, and since this is equation is completely free from the other equations, it adds no condition at all. So the power flow equations for $\hat{N}$ on the nodes in W are equal to the equations for N, hence will result in the same voltages and powers in any node.

Therefore, for an $e \in E$, the current I_e through e in N will be equal to the current $\hat{I}_e$ in $\hat{N}$. And for $i \in 1, ..., k$, $\hat{I}_{f_i} = A_{f_i}(\hat{U}_{v_i} - \hat{U}_a) = 0$ by Ohm's Law. So

$$\sum_{e \in \hat{E}} |\hat{I}_e|^2 \cdot \Re(Z_e) = \sum_{e \in E} |\hat{I}_e|^2 \cdot \Re(Z_e) + \sum_{i=1}^{k} |\hat{I}_{f_i}|^2 \cdot \Re(Z_{f_i}) \tag{6.10}$$

$$= \sum_{e \in E} |I_e|^2 \cdot \Re(Z_e) + 0 \tag{6.11}$$

since the current and the resistance for any f_i is zero. So the total power losses in N and $\hat{N}$ are equal. □

Note that for a configuration N_A of N, where A is the set of closed optional edges and $B = E_o \backslash A$ the set of opened optional edges, the power flow in N_A will be equal to the power flow in N if all edges in B get an infinite impedance, so an admittance of value 0. Hence this proposition implies not only that the power flow and power loss in N and $\hat{N}$ will be equal, but also in any configuration of N and the corresponding configuration of $\hat{N}$.

6.2 Application to the Algorithms

The reason for the failure of most of the suggested algorithms when applied to multiple slack bus networks, is that these algorithms only open the cycles present in the network. Paths between slack busses do not have to be opened this way, resulting in a non-radial configuration. However, when a multiple slack bus network is adapted as suggested, any path between two slack busses in the old network is turned into a cycle in the new network via node a. And since the two edges in this cycle linked to a are fixed, opening this cycle corresponds to opening the path. So in the new, adapted network, opening all cycles is sufficient to attain a radial configuration in the original network, no matter how much slack busses are present. And for this new network, we can use Propositions 3.5 and 3.8 again.

For example, look at Figure 6.1. This network contains one cycle. Two edges have to be opened to attain a radial configuration, one in path $(1, 3, 4, 2)$ and one in path $(1, 5, 6, 7, 2)$. For the Greedy-Shifting algorithm, such a radial configuration is assumed as initial configuration. But when one of the opened edges is closed, no cycle will arise, hence it is not clear which edges could be opened. For instance, say edges $(3, 4)$ and $(5, 6)$ are opened. Then closing $(3, 4)$ allows $(1, 3)$ or $(2, 4)$ to be opened for radiality, but no other edge.

In the Greedy-Demeshing algorithm, one of the edges will be opened, and then the algorithm stops, returning a non-radial configuration. And even if it continues, it is not clear immediately which edges are candidates to be opened again. For the Harmony Search Algorithm and the Genetic Algorithm, the cycles or semi-ears are the sets from which the edges are chosen to be opened, in this case the one cycle in the network. Therefore, no radial

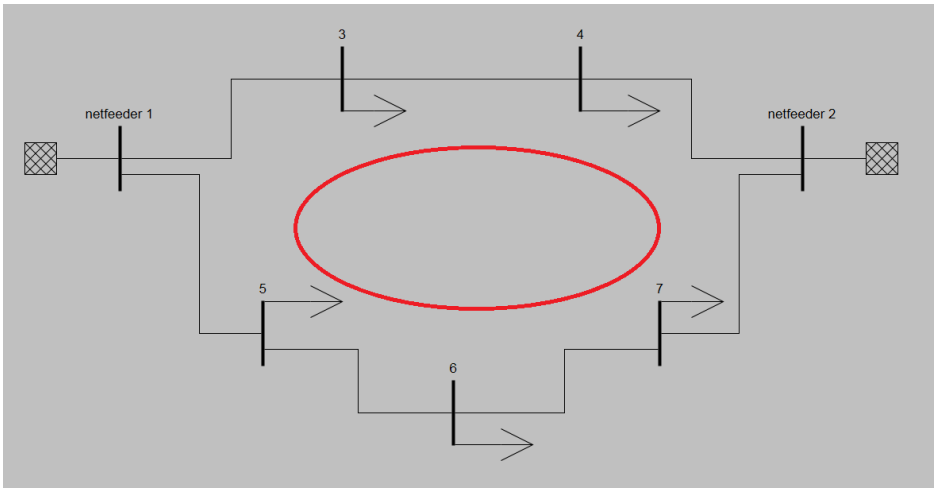

Figure 6.1: *A multiple slack bus network* ("Vision")

configurations will be considered. The same holds for the Brute Force Calculation.

Now look at Figure 6.2. Here the network construction as described for proposition (6.1) is applied. In this case a cycle basis contains two cycles. Both will be opened by the Greedy-Demeshing algorithm, the Harmony Search Algorithm, the Genetic Algorithm and the Brute Force Calculation. And since the additional edges $(1, a), (2, a)$ are fixed, a resulting configuration in the new network will be a tree where all slack busses are connected via a, which corresponds to a radial configuration in the original net. Also for the Greedy-Shifting algorithm, closing an edge in a radial configuration results in a cycle, which corresponds to an actual cycle or a path between slack busses. Then the shifting procedure can be applied successfully.

Note that Proposition 6.1 also holds for a single slack bus network. So a procedure to make the Greedy-Shifting algorithm, the Greedy-Demeshing algorithm, the Harmony Search Algorithm, the Genetic Algorithm and the Brute Force Calculation suitable for single- and multiple slack bus networks would be simply to start with the above construction (see Figure 6.3), and then run the algorithm. From now on we will assume that all the suggested algorithms are extended with this procedure, which we will call the *Network Adapter*. As described above, in this way the algorithms work with radial configurations of the original network, or work towards such a configuration. And by Proposition (6.1), the voltages, currents and powers in the original part of a configuration of the new network will be the same as the corresponding configuration in this original network, as well as the total power loss. So minimizing the power loss without capacity excess in the new network is equivalent to minimizing the power loss without capacity excess in the original network. Hence the extended algorithms will return/approximate the configuration of the input network with minimal power losses.

Two remarks must be made. First, note that in Proposition 6.1 we work with infinite impedances, and we assume that the inverse of this is zero. Mathematically this works out fine. However, in actual calculations this can lead to difficulties. Therefore, we will take in

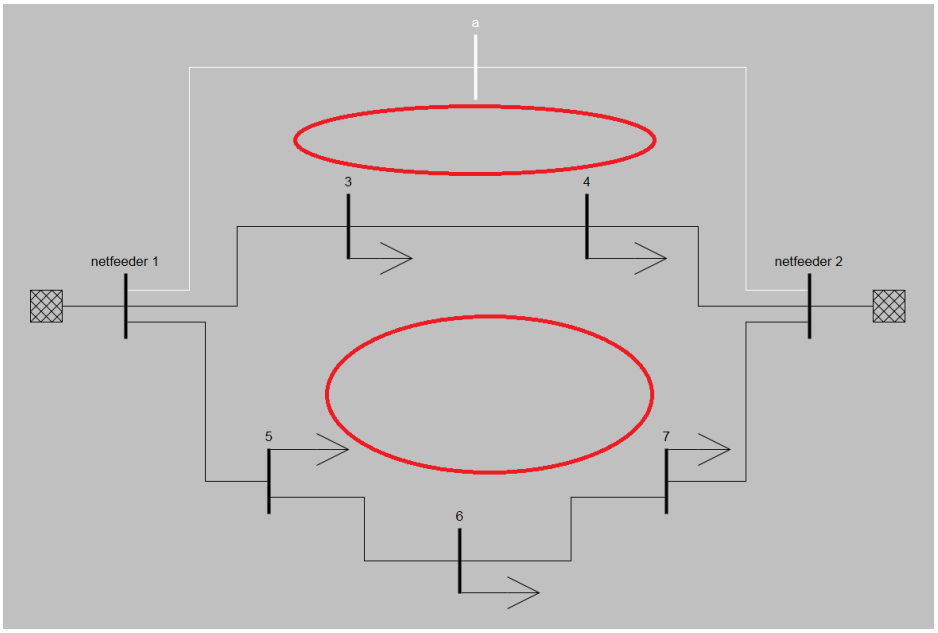

Figure 6.2: *Addition of an artificial node* ("Vision")

the actual implementations a relatively high number β, and set the impedances as $Z = 0 + j\beta$ instead of $0 + j\infty$ (and for the capacities similar substitutions as well). Then the admittances become $\frac{1}{Z} = 0 - \frac{j}{\beta}$ instead of 0. This leads to a difference between the total power losses in the original and the adapted network. However, if we take β large enough (for instance 1.000.000 times the highest impedance magnitude present in the original network) then this difference becomes insignifficant, no more than 0.0001 %. In practical situations, this is good enough.

Second, this method can be used when other objective functions are considered as well.

Network Adapter		
input	Network $N = (W = W_n \cup W_c, E = E_o \cup E_f)$	
1	Add *artificial node* a to W_c as a load bus: $W_c = W_c \cup \{a\}$	
	Set active and reactive power demands at a as: $P_a = 0, Q_a = 0$. Set voltage boundaries at a as: $U_a^{min} = 0, U_a^{max} = \beta$ with β a high real number	
2	For all $v \in W_n$ do	
	a	Add fixed line $f_v = (v, a)$ to E_f: $E_f = E_f \cup \{f_v\}$
	b	Set impedance and current capacity of f_v as: $Z_{f_v} = 0 + j\beta, I_{f_v}^{max} = \beta$
3	Return N	

Figure 6.3: Network Adapter

Indeed, for other objectives the radiality constraint remains the same, hence can be ensured with this method. And since for other objectives normally the current magnitude through a line is a factor in the objective calculation of this line, the objective through an artificial added line will be zero, since the current is zero. Therefore, the objective value for the total configuration will not be effected by this adaptation.

Chapter 7

Adaptations and Decisions during the Implementation Process

The step from theoretical algorithm to a well working program in "R" revealed several difficulties and defects for almost all of the suggested methods. The Greedy-Demeshing algorithm is very straight-forward, and did not cause any problems as it is described in chapter 5. However, all the other implementations did. This chapter will survey the fundamental, mathematical problems that occurred, and, when found, the solutions that are brought up to this.

During the implementation procedure, two imaginary test networks were used. Test net 1 can be found in Figure 7.1, and contains nine nodes of which one slack bus, and ten edges. Hence two lines must be opened in order to obtain a radial configuration (implied by Lemma 3.3). Test net 2 contains 50 nodes and 60 lines. Node 1 and optionally node 2 are slack busses, so 11 or 12 lines respectively have to be opened in this network (implied by lemma 3.3). See Figure 7.2. In both networks, multiple values for the impedances, voltages and power supplies/demands were chosen, to check whether the implementations worked as expected. These are small, imaginary networks, used for fast testing of correctness of the

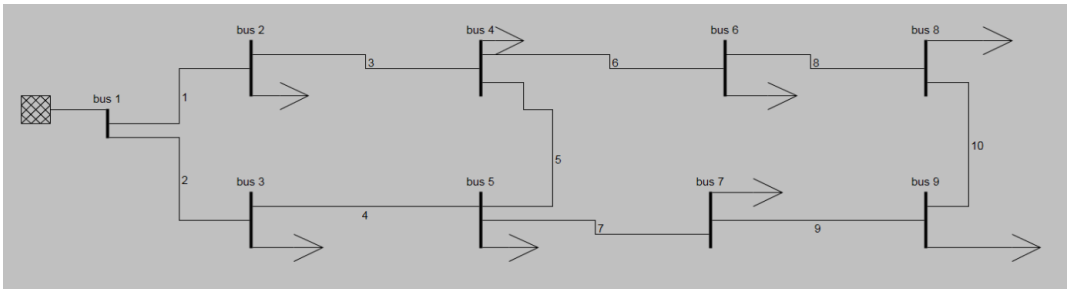

Figure 7.1: *Test Network 1* ("Vision")

implementations. Almost all defects in this chapter were revealed while the corresponding algorithm was tested on one of these test networks.

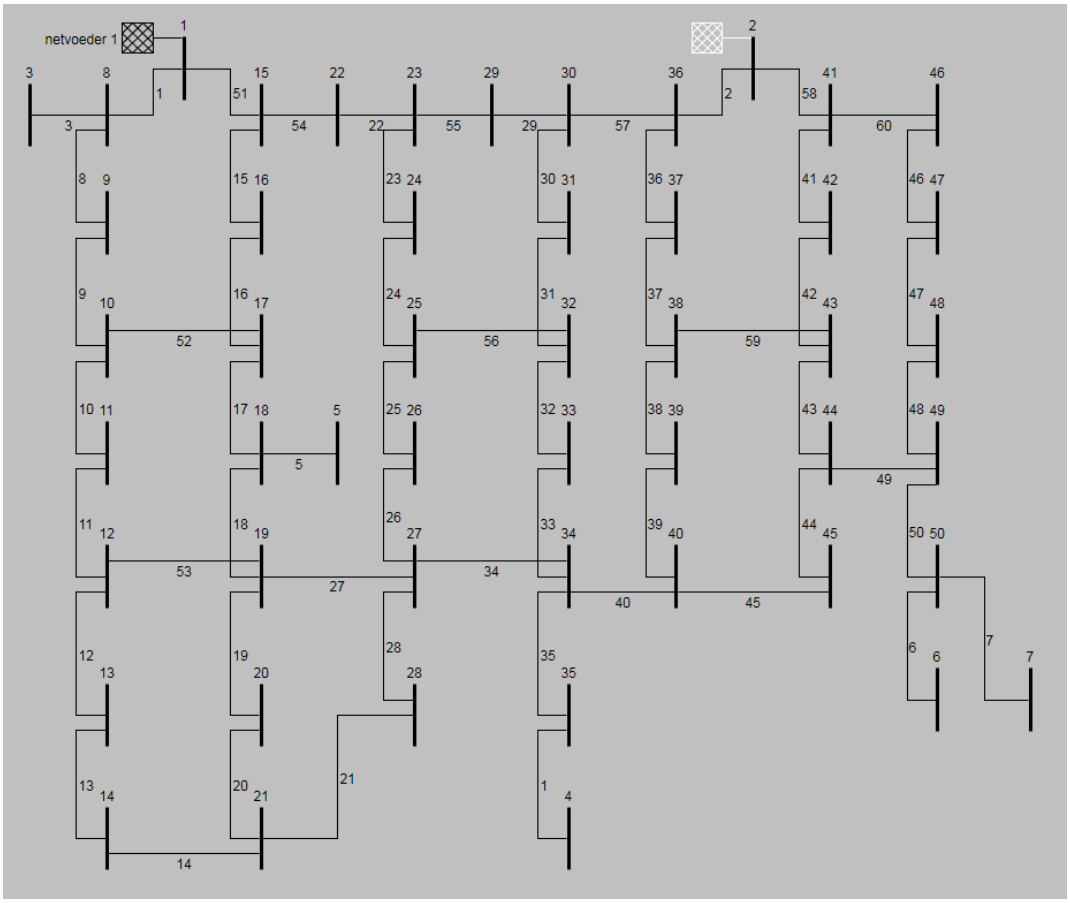

Figure 7.2: *Test Network 2* ("Vision")

7.1 Drawback Semi-Ear Decomposition

For the Harmony Search Algorithm and the Genetic Algorithm, a version was suggested that makes use of semi-ear decompositions. As mentioned in Chapter 3, out of a semi-ear decomposition only radial configurations will be constructed, but not all radial configurations. Therefore, in Chapter 5 it was suggested that several semi-ear decompositions out of one cycle basis should be considered, to cover all possible radial configurations.

However, it turns out that the number of semi-ear decompositions needed to cover all possible radial configurations of one network could be very high. This became clear when these versions of the Harmony Search Algorithm and the Genetic Algorithm were tested on test network 2. Also, this will be mathematically substantiated by Lemma 7.1. Moreover, it is not clear at all for a network and a cycle basis of this network, which semi-ear decompositions are actually needed to cover all these configurations. Another difficulty of the semi-ear decomposition version is that a semi-ear could be empty, which always results in bad behaviour of the algorithms, since in that case not enough edges will be opened. Together with the fact that checking radiality of a configuration is rather simple, this makes the cycle basis versions of the Harmony Search Algorithm and the Genetic Algorithm highly preferable over the semi-ear decomposition versions. Therefore, the semi-ear decomposition versions will no

longer be considered, and so they will not be further tested and compared to other methods.

The next proposition shows that the number of semi-ear decompositions needed could grow impracticably big, by presenting example networks in which this happens. In short, for any number $k \in \mathbb{N}$ there exists a network in which k decompositions are needed to cover all radial configurations. Moreover, this network only has $2(k+1)$ nodes and k independent cycles, and it has a realistic structure. So the examples given are not difficult, artificial networks, but similar to MV (sub)networks that could occur in reality.

Lemma 7.1. *For all $k \in \mathbb{N}$ there exists a network N_k with a cycle basis $B = \{C_1, ..., C_k\}$ for which at least k different semi-ear decompositions are needed in order to cover all radial configurations, i.e., in order to be able to construct all possible spanning trees by deleting one edge in every semi-ear of a certain decomposition.*

Proof. For $k \in \mathbb{N}$, define N_k to be the network with $2k+2$ nodes and $3k+1$ edges as shown in Figure 7.3. Say all edges are optional. For $1 \leq i \leq k$, set $C_i = \{3i-2, 3i-1, 3i, 3i+1\}$. We will prove that N_k needs at least k different semi-ear decompositions in order to cover all radial configurations by induction on k.

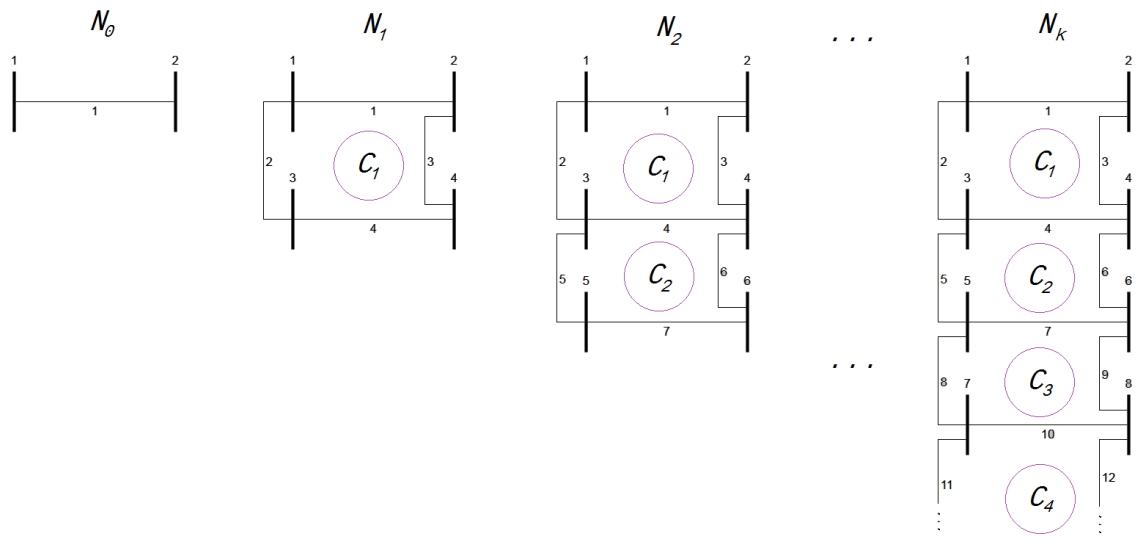

Figure 7.3: *Construction of N_k*

N_0 is already radial, so needs no decomposition to construct a radial configuration. N_1 is a cycle, and has a unique semi-ear decomposition, namely itself. And indeed, this decomposition is enough to cover all possible radial configurations.

Now assume $k \geq 2$ and N_{k-1} needs at least $k-1$ decompositions. Look at N_k. In C_1, at least one edge must be opened to make the network radial. And if two edges are opened in C_1, one of them must be edge 4, since else node 1 or node 2 is isolated (or both). Three or four edges opened in C_1 will never result in a radial configuration.

Now we will consider two cases; one of edges 1, 2, or 3 is opened, or these three edges are all closed. In the first case, edge 4 can be either opened or closed, where in the seconds case it must be opened.

First, look at a radial configuration R of N_k where edge 1,2 or 3 is opened, say e. Set $N' = N_k - e$. Then $\{C_2, ..., C_k\}$ is a cycle basis of N', and R induces a radial configuration for N'. Moreover, a radial configuration of N' is equivalent to a radial configuration of N_{k-1}, since they have the same structure when we look at their cycles, and only edges on cycles must be opened in order to make a configuration radial.

By induction, N_{k-1} needs at least $k-1$ different semi-ear decompositions in order to cover all possible radial configurations. This means N' needs at least $k-1$ different decompositions as well, that is, $k-1$ different orders of $C_2, ..., C_k$ to construct a semi-ear decomposition of N'. So for all radial configuration of N_k with edge 1,2 or 3 opened, if we choose equivalent orders of $C_2, ..., C_k$ and always add C_1 as the last cycle to construct the decompositions of N_k, all these configurations (where edge 1, 2 or 3 is opened) are covered. Note that it is even enough for a specific order of $C_2, ..., C_k$ to add C_1 anywhere after C_2 to get the same decompositions of N_k. Moreover, note that all these decompositions are needed, else we could also use fewer decompositions for N_{k-1}, which is not true by assumption.

Now look at a radial configuration where edges 1,2 and 3 are closed. Then edge 4 is opened. But edge 5, 6 or 7 then has to be opened as well. Say edge e' is opened. If C_1 is used after C_2 for a decomposition, then edge 4 and e' are contained in the semi-ear corresponding to C_2. Hence this configuration cannot be constructed via such a decomposition. Therefore, we need at least one new order of $C_1, ..., C_k$ for a decomposition with C_1 before C_2. Hence, we need at least k decompositions in total for N_k.

Note that after adding a decomposition where C_1 was used before C_2, we cannot remove another decomposition, since with any of the old decompositions we can choose a unique radial configuration with edge 4 and, say, edge 1 opened. This configuration cannot be constructed with the new decomposition, so we still need the old one as well. □

7.2 Minimum Weight Cycle Basis

A network usually admits several cycle bases. For instance, test network 1 has basis $B_1 = \{(1,3,5,4,2),(5,6,8,10,9,7)\}$, $B_2 = \{(1,3,5,4,2),(1,3,6,8,10,9,7,4,2)\}$ and
$B_3 = \{1,3,6,8,10,9,7,4,2),(5,6,8,10,9,7)\}$. As Proposition 3.5 implies, all radial configurations can be constructed by opening one edge in every cycle of a cycle basis, independent of which cycle basis is used. And the search space of the Harmony Search Algorithm (cycle basis version), the Genetic Algorithm (cycle basis version) and the Brute Force Calculation is actually the set of all possible combinations of edges that can be constructed this way. We shall call such a combination an *edge-open combination*. Note that by Proposition 3.5 any combination that generates a radial configuration can occur, but also combinations that do not generate radial configurations.

For basis B_1, the total number of edge-open combinations that can be made this way is

$5 \cdot 6 = 30$. For B_2 this is $5 \cdot 9 = 45$ and for B_3 this is $9 \cdot 6 = 54$. So out of B_2 at least 15 combinations generate non-radial configurations, since if a combination generates a radial configuration, by Proposition 3.5 it must be available in B_1 as well. And out of B_3 even more, namely 24. This means that by using B_2 or B_3 for the Harmony Search Algorithm, the Genetic Algorithm, or the Brute Force Calculation, more irrelevant edge-open combinations can and will be considered than by using B_1.

Initially, a simple subroutine was used for the implementations of HSA, GA and BFC

Simple Cycle Basis Finder		
input	Network $N = (W, E = \{e_1, \dots, e_m\})$	
1	Set $F = \emptyset$, and set basis $B = \emptyset$	
2	For $i = 1, \dots, m$ do	
	a	$F = F \cup \{e_i\}$
	b	If graph $G = (W, F)$ contains a cycle C, set $B = B \cup \{C\}$ and set $F = F \backslash \{e_i\}$
3	Return B	

Figure 7.4: *Simple Cycle Basis Finder*

to construct a cycle basis of the input network $N = (W, E)$. This subroutine starts with graph $G = (W, \emptyset)$ and iteratively adds an edge to G. If a cycle arises, this cycle is saved in the basis and the last edge is deleted again. In this way, a spanning tree is constructed, and for every edge not in the tree, exactly one cycle containing this edge is present in the resulting basis. Hence these cycles are independent, and the cardinality of the basis is $|E| - |W| + 1$, so by Lemma 3.4 this is indeed a cycle basis of N. See figure (7.4).

This is a fast and simple subroutine, and it is easy to implement it such that orientation of the cycles is preserved. However, this subroutine turns out to return bases with very big cycles. For instance, in test network 1 it will return cycle basis B_2, and in test network 2 this gets more extreme. More about this later.

Bigger cycles in the cycle basis lead to bigger intersections between pairs of these cycles. And it is exactly this intersection that causes edge-open combinations which generate non-radial configurations. Indeed, if for two cycles, both edges are chosen in the intersection of these cycles, a non-radial configuration will arise. Moreover, this configuration can be constructed in two ways, choosing the first edge for the first and for the second cycle, and vice versa for the second edge. Now when the intersections get bigger, the number of these possibilities gets higher, leading to more irrelevant combinations to be considered.

Already for test network 2, which is a very small network compared to real MV networks, this gives problems. HSA and GA show bad behaviour concerning convergence and running time, since many of the considered configurations are non-radial. Also BFC suffers from this; for test network 2 it has to consider around 2.4×10^{11} combinations, of which more than 99% is surely irrelevant, as we will see later on.

So we actually want a subroutine that finds the cycle basis in where the cardinalities of the pairwise intersections between the cycles are minimized. No literature was found on this, and it is beyond the scope of this project to find such an algorithm. However, a fine alterna-

tive would be to use a cycle basis in which the length of the cycles is minimized. Small cycles will generally lead to small intersections. And in 1987, Horton developed a polynomial-time algorithm to find such a basis, see [26]. Formally, we want to find the cycle basis in which the sum of the cardinalities of the cycles is minimized. So if we give every edge in the network weight equal to 1, we want to find the cycle basis of minimal total weight. Horton proved that such a basis always consists of *Horton cycles*, fundamental cycles of a shortest path tree of the graph, of which at most $|W| \cdot |E|$ exist. So the algorithm first finds all these Horton cycles, and then builds a basis out of these cycles by adding a cycle to the basis-to-be-build if and only if it is independent of the already added cycles. Hereby it considers the cycles from small to big. We will call this algorithm *Horton's Algorithm*, see Figure 7.5.

Using this new subroutine for test network 2, a cycle basis arises that admits around 2.0×10^9

Hortons Algorithm		
input	Network $N = (W = \{w_1, \dots, w_n\}, E = \{e_1, \dots, e_m\})$	
1	Set $M = \emptyset$	
2	For $i = 1, \dots, n$ do	
	a	Compute the Shortest-Path-Tree $T \subseteq E$ for w_i
	b	For each $e \in E \backslash T$, let C be the cycle in $T \cup \{e\}$, and set $M = M \cup \{C\}$
3	Say $M = \{C_1, \dots, C_q\}$, ordered by increasing size. Set $B = \emptyset$	
4	For $i = 1, \dots, q$ do	
	a	If C_i is independent of B, then set $B = B \cup \{C_i\}$
	b	If $\|B\| = m - n + 1$, then **return** B

Figure 7.5: *Horton's Algorithm*

edge-open combinations, almost five times fewer than with the basis from the Simple Cycle Basis Finder. As we know from Proposition (3.5), this new cycle basis can generate all radial configurations of N, so indeed more than 99% of the combinations constructed with the old cycle basis is irrelevant.

As a consequence of these results, Horton's Algorithm is used for the implementations of HSA, GA and BFC. And already on test network 2 the behaviour of these algorithms greatly improved.

7.3 Construction of Random Configurations using MWST

For the cycle basis version of the Harmony Search Algorithm, an initial harmony memory is generated by randomly constructing configurations. Each configuration is constructed by randomly picking an edge for every cycle in the specified cycle basis, and then checking if it is radial or not. If not, the configuration is abandoned and a new one is created. Also the initial generation of the cycle basis version of the Genetic Algorithm is generated this way. When these algorithms were tested on test network 2, using Horton's Algorithm introduced in the previous section, the number of edge-open combinations needed to make enough radial configations to fill the initial harmony memory/generation was over twice the size of this memory/generation. So over a half of the considered combinations did not generate a radial configuration. And when these algorithms were applied to realistic networks with hundreds of nodes, it became clear that that this was a serious problem. In the first 500 combinations

checked, at most 3 of them generated radial configurations. Clearly when the networks get bigger, the probability that a configuration, created by randomly picking an edge for each cycle, is radial decreases rapidly. This results in an extremely slow start of the algorithms, which is clearly undesirable.

To avoid this slow start, a new method for constructing the initial harmony memory/generation is suggested, which turns out to be successful in practice. A radial configuration of a network is actually a spanning tree of this network, assuming the network is adapted as proposed in Chapter 6. Now finding a minimum weight spanning tree in a weighted graph is an efficiently solvable problem as mentioned earlier. Fast algorithms such as Prim's algorithm are developed for this. So constructing a random radial configuration of a network $N = (W, E)$ can be achieved quickly by adding random weights to the edges in E, and then computing a minimum weight spanning tree of this graph with such an algorithm. It turns out that if for every edge a natural number between 1 and $|E|$ is chosen as weight randomly, the resulting minimum weight spanning trees will show much variation and will not have too much similarity. With this method, only a number of configuration constructions equal to the size of the initial harmony memory/generation is needed to fill them, since by Chapter 6 every configuration constructed is surely radial. And since any radial configuration can be attained, and a wide variety of configurations will be attained, the results will be as good as with the original method. Moreover, this suggested method is much faster than the original one.

One important remark must be made. In the suggested method no difference between fixed and optional edges is taken into account. Fixed edges cannot be opened, hence the must be contained by the minimal spanning tree. For the method as described above, this cannot be ensured at all. However, a simple adaptation will solve this problem, and is based on the following lemma.

Lemma 7.2. *Let $G = (W, E)$ be graph with for every $e \in E$, weight $w_e \in \mathbb{N}$. Assume that for every cycle $C \subseteq E$ of G, there is at least one edge $e \in C$ such that $w_e > 0$. Then for any minimum weight spanning tree $T \subseteq E$ of G and for all $e \in E$ with $w_e = 0$: $e \in T$.*

Proof. Let T be a minimum weight spanning tree of G, and let $e \in E$ be an edge with weight $w_e = 0$. Assume $e \notin T$. Then $T \cup \{e\}$ contains a cycle C of G, since T is a spanning tree. And $e \in C$. Furthermore, by assumption C contains an edge $e' \neq e$ with $w_{e'} > 0$. Now $e' \in T$, since $e' \neq e$, $e' \in C$ and $C \subseteq T \cup \{e\}$. Set $T' = (T \cup \{e\}) \backslash \{e'\}$. Then T' is a spanning tree of G as well.

Say $w(T) = \sum_{e \in T} w_e$ is the weight of T. Then

$$w(T') = w(T) + w_e - w_{e'} \tag{7.1}$$
$$< w(T) \tag{7.2}$$

But then T is not a minimum weight spanning tree. A contradiction, hence $e \in T$ must hold. And since T and e were chosen arbitrary, the claim holds for any minimum weight spanning tree T of G and any weight zero edge $e \in E$. □

For an MV network N, we can assume that any cycle C of N contains an optional line. If not, then C cannot be opened, hence any configuration on N contains C and therefore will

not be radial. This will never be the case.

Therefore the following adaptation on the method above is suggested. For any edge $e \in E$, if e is fixed, give e weight zero. Else, give e an arbitrary natural number between 1 and $|E|$ as weight. Then compute a minimum weight spanning tree of the network. See Figure 7.6. Now by assumption any cycle in N has an optional edge, so an edge with weight greater than zero. Therefore by Lemma 7.2, for a minimum weight spanning tree T of N, all edges with weight zero are contained by T. These are exactly all the fixed edges, as desired.

Random Radial Configuration Finder	
input	Network $N = (W, E = E_o \cup E_f)$
1	For all $e \in E_f$, give e weight $w_e = 0$.
2	For all $e \in E_o$, let q be a random number between 1 and $\|E\|$. Give e weight $w_e = q$.
3	Compute a minimum weight spanning tree $T \subseteq E$ of N with the assigned weights.
4	Return T

Figure 7.6: *Random Radial Configuration Finder*

7.4 Drawback Mixed Integer Linear Programming

Although the idea behind the Mixed Integer Linear Programming algorithm seems very promising, the implementation of this method causes serious problems. For this project, two different "R" packages for solving mixed integer linear programs were tried for the implementation, namely "lpSolve" ([27]) and "Rglpk" ([28]). In both cases, a program was written that builds the constraint matrix corresponding to constraints (5.16)-(5.36) out of the input network, in the format desired by the packages. Then the optimization tool was applied to this matrix, yielding a solution.

In both cases the output for test network 1 was already far from good. If all load busses in test network 1 have equal power demand, the optimal configuration is clearly reached when edges 5 and 10 are opened. However, with the first package the output was invariably to open edge 1 and 5, where the output using the second package never even generated a radial configuration. And a short test on test network 2 gave similar results.

This failure could be caused by several reasons:

1. Publication [14] contains errors.
2. The method was not implemented properly.
3. The used "R" packages contain errors.

Reason 1 could be plausible. However, the article is very detailed, hence every substitution can be followed and checked, and no error has been detected by the author. Reason 2 could be the cause as well. But again, exhaustive investigation of the implementation as well as re-implementing the method did not give any positive results. Furthermore, e-mails were send to the authors of [14], but no useful reactions were received at the time of writing this report.

Reason 3 does not seem likely, since two different, independent packages were used.

So a well-working state of operation of MILP was not achieved in this project. As a consequence, the Mixed Integer Linear Program algorithm has been abandoned for further investigation in this project, and will no longer be taken into account as a candidate for solving/approximating LRRP.

Chapter 8

Results and Conclusions

After the implementation procedure, four candidate algorithms remained, besides the Greedy-Shifting algorithm from "Vision": the Greedy-Demeshing algorithm, the Harmony Search Algorithm (cycle basis version), the Genetic Algorithm (cycle basis version) and the Brute Force Calculation. These algorithms were tested and compared to each other on several networks, in order to find which algorithm is most suitable for the MV networks under Liander management.

In this chapter, the results of these tests will be shown, and the conclusions that are based on these results will be stated. In section 8.1 the test results and conclusions with respect to test network 2 (Figure 7.2) will be discussed. After this, the (remaining) implementations were tested on a part of the MV network of the Dutch town of Zaltbommel, a network consisting of 211 nodes and for which good "R" data were available. Based on these results a final method was determined, which happens to be a combination of two algorithms. This will be described in Section 8.2. However, the data of Zaltbommel were not available in "Vision", so these results could not be compared to the Greedy-Shifting method or to the current configuration. Therefore, a final comparison was executed on the MV network of the Dutch city of Zaandam and the Dutch island Texel. The MV network of Zaandam contains 408 nodes and 8 slack busses, and the MV network of Texel is a single slack bus network with 228 nodes. Moreover, these networks are simulated in detail in "Vision", so this gave a good opportunity to compare the final method with the existing tool in "Vision", and to investigate the current configurations in these nets. Moreover, this gave a nice challenge to turn "Vision" data into "R" data, which was handled successfully.

8.1 Test Network 2: Drawback Brute Force Calculation

Test network 2 (see Figure 7.2) was used to test the correctness of the implementation during the process, but also for comparing the different algorithms. For this, the following artificial values were added to the network, which were randomly chosen:

- Node 1 is the slack bus, with voltage magnitude 10000V and angle 0°.
- The other nodes are load busses, with active power demand 10000W and reactive power demand 12000VAr.
- All lines have impedance $1 + j0.1\Omega$.

- The optional lines are lines 1, 9, 10, 12, 16, 17, 19, 21, 22, 24, 26, 27, 29, 31, 33, 34, 37, 39, 40, 42, 44, 45, 47, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, and 60. These were chosen randomly, but evenly spread over the network.
- Capacities are chosen wide, voltage magnitudes between 0V and 10000V for every node, current magnitudes under 1000A for every line.

The Brute Force Calculation gave as output a list of the 20 best configurations, which can be found in figure 8.1.

Each row represents a feasible configuration, namely those edges that are opened in this

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	Loss
52	53	19	26	21	31	33	39	42	59	47	11893.78
52	53	19	26	21	31	33	37	42	59	49	11956.25
52	53	19	26	21	31	33	37	42	44	49	11987.33
52	53	19	26	21	31	33	37	42	39	49	12049.55
52	53	19	26	21	31	33	37	42	45	49	12049.55
52	53	19	26	21	56	33	39	42	59	47	12099.03
52	53	19	26	21	31	33	39	42	59	49	12117.54
52	53	19	26	21	56	33	37	42	59	49	12225.80
52	53	19	26	21	56	33	37	42	44	49	12256.88
52	53	19	26	21	31	56	39	42	59	49	12309.60
52	53	19	26	21	31	56	37	42	59	49	12316.38
52	53	19	26	21	31	33	37	59	44	49	12317.68
52	53	19	26	21	56	33	37	42	39	49	12319.09
52	53	19	26	21	56	33	37	42	45	49	12319.09
52	53	19	26	21	31	56	37	42	44	49	12347.64
52	53	19	26	21	31	33	37	42	59	47	12370.53
16	53	19	26	21	31	33	39	42	59	49	12391.69
52	53	19	26	21	31	56	37	42	39	49	12410.21
52	53	19	26	21	31	56	37	42	45	49	12410.21
52	53	19	26	21	31	33	37	59	44	47	12423.70

Figure 8.1: *Output Brute Force Calculation on Test Network 2*

configuration, followed by the total power loss in this configuration. So the total power loss in this network under the above circumstances is minimized at 11894W, since the Brute Force Calculation considers all feasible configurations. However, the running time critically decreases the status of the Brute Force Calculation, since this calculation took over 50 hours.

Applying the Greedy-Demeshing algorithm to the network with the above data, a configuration was returned in which a power loss was achieved of 12693W. Note that this configuration does not belong to the 20 best configurations, since the power loss is over 250W above number 20 in Figure 8.1. However, the running time of the Greedy-Demeshing algorithm on this network situation is less than a second, which is fast.

The Harmony Search Algorithm and the Genetic Algorithm are non-deterministic algorithms, hence will not always give the same configuration as output. Moreover, both algorithms contain parameters which can be tuned (such as the harmony consideration rate, the cross-over rate and the population space) and will affect the performance and (for the population space, the memory space and the number of iterations) the running time of the algorithms. Several combinations of values for these parameters were applied to the algorithms and tested on the described network situation a 100 times. Based on maxima, minima and averages, the optimal values for the parameters were determined. The complete test results will not be shown here, but the optimal values that are found will be given now:

- For the Harmony Search Algorithm:
  - Harmony memory space = 10
  - Harmony memory consideration rate = 0.75
  - Pitch adjustment rate = 0.1
- For the Genetic Algorithm:
  - Population space = 10
  - Cross-over rate = 0.7
  - Initial mutation rate = 0.15, final mutation rate = 0.05, and mutation rate step size = 0.004

With the parameters chosen as above, the best results were achieved. Note that the number of iterations cannot be optimized: A higher number of iterations always results in better output, but also in larger running-time. So this parameter will remain as a free variable, which can be decided by the user based on the time that is affordable for the computation. The test results achieved when using the above parameter values can be found in Figure 8.2, for 500 iterations and for 1000 iterations. Note that in the Genetic Algorithm a construction of a new generation takes 10 iterations, hence 50 and 100 generations are constructed here.

As can be seen, both algorithms mostly give better results than the Greedy-Demeshing

	Power loss after 500 iterations			Power loss after 1000 iterations		
	Average	Minimum	Maximum	Average	Minimum	Maximum
Results Genetic Algorithm:	12404.28	11893.78	13132.40	12248.32	11893.78	12920.65
Results Harmony Search Algorithm:	12210.66	11893.78	13019.42	11987.48	11893.78	12590.36

Figure 8.2: *Results 100 times GA and HSA on test network 2 with best found parameters*

algorithm. Moreover, with 1000 iterations HSA scored better than GD every time. But the running time has to be taken into account as well; for 500 iterations both HSA and GA need around 10 seconds for one computation, and for 1000 iterations around 20 seconds.

Based on therunning times, the Brute Force Calculation does not seem a good method for reducing the power loss in Dutch MV networks. If we take into account that the running time of the algorithm scales badly with the size of the input network, the time needed when Zaltbommel or Zaandam is used as input will be far from practical, even if the algorithm is

speeded up or if better computers are used. To give an example, the number of edge-open combinations to be considered for Zaandam, with a cycle basis constructed by Horton's algorithm, is around 1.6×10^{46}. Even if every iteration would take the same amount of time (which is absolutely not the case), the time needed for Zaadam would be over $50^4 = 6250000$ hours. Therefore we abandon this method as a candidate, and we will no longer consider it in our tests and comparisons.

A slight difference can be seen between the results from the Harmony Search Algorithm and the Genetic Algorithm. Overall the first scores a little bit better than the second. However, this difference is too small to base conclusions on, so none will be made at this point.

8.2 Zaltbommel: Combining Algorithms

To check if the remaining methods are suitable for big, realistic networks, they were tested on a part of the MV network of Zaltbommel, a Dutch town. See Figure 8.3. This subnetwork contains 211 nodes with one slack bus, several synchronized generators (hence load busses with negative demand) and load busses. There are 221 links and cables, which means that 11 of them have to be opened for radiality.

The network data for Zaltbommel, connectivity, impedances, capacities and power profiles

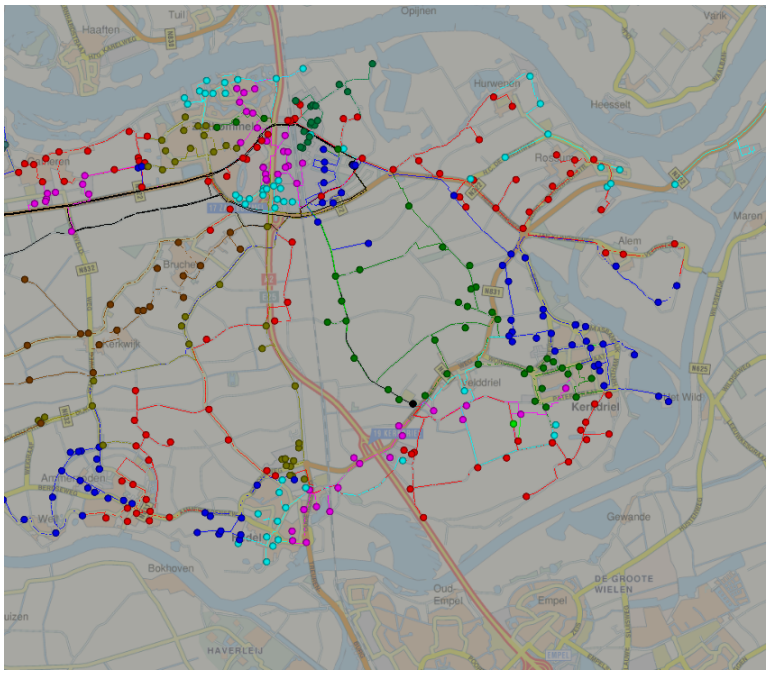

Figure 8.3: *Zaltbommel and a schematic rendering of the MV net* ("Vision")

for the nodes are available as "R" data, which makes this network very suitable as testing network for this project. For the comparison, three scenarios were made for the network, with different power supplies and demands at the load busses, based on the profiles. To these scenarios, the Greedy Demeshing algorithm was applied, resulting in a specific configuration

for each scenario. After this, the Harmony Search Algorithm and the Genetic Algorithm were both applied twenty times to each scenario, each computation consisting of 2000 iterations. During the processes, after every hundred iterations, the power loss of the best configuration so far was saved. When all computations were executed, for each scenario and each batch of hundred iterations, the average of the twenty HSA computations were calculated and for the twenty GA computations as well. The results for scenario 1 can be found in Figure 8.4. Similar graphs for scenario 2 and 3 can be found in the appendix.

The results from scenario 1, 2 and 3 show similar and very clear results: HSA shows

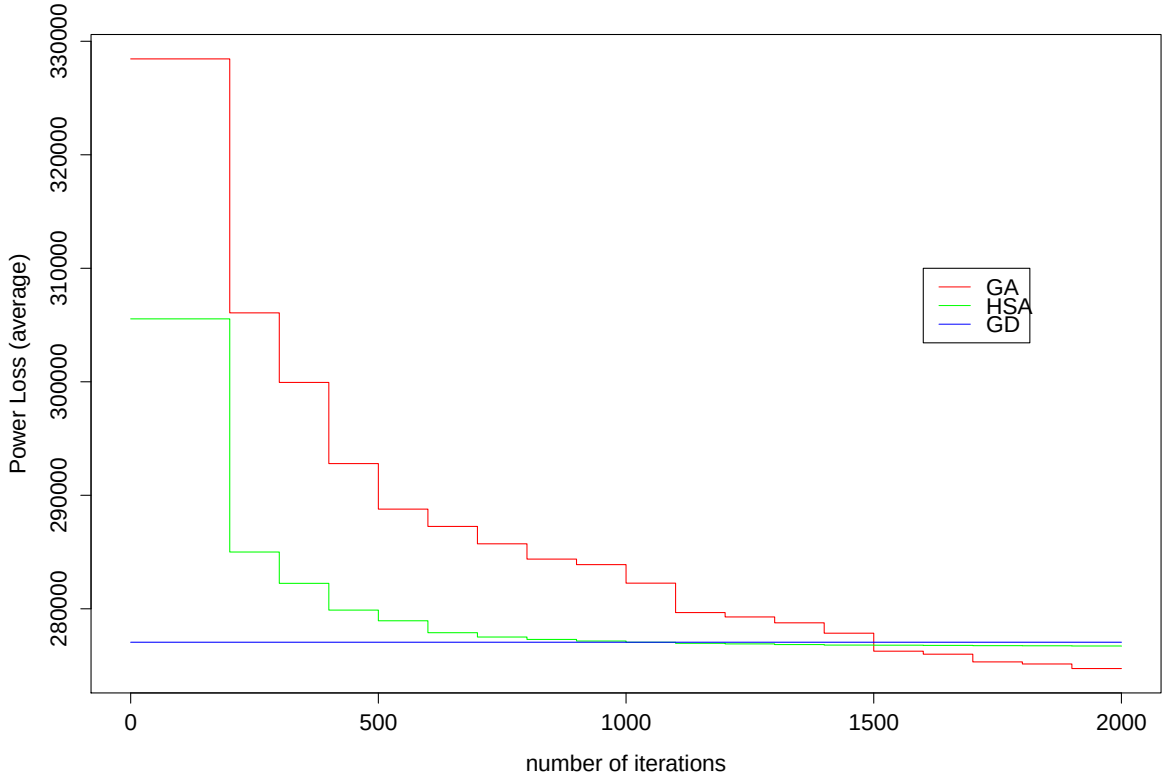

Figure 8.4: *results testing on Zaltbommel, scenario 1*

better performance in general in the first few hundred iterations. However, the decrease of the minimal power loss becomes smaller much faster than by GA. And at some point, after at most 1500 iterations, GA shows better results. When we take GD into account as well, it turns out that HSA scores badly on improving the power loss once it reaches losses lower than the loss found by GD. GA, on the other hand, seems to have no problems with this at all.

We are also interested in the running time of the algorithms. The computation of the Greedy Demeshing algorithm for this network with these scenarios is under 10 seconds. On the other hand, the time needed for the Harmony Search Algorithm and the Genetic Algorithm to finish 2000 iterations for the three situations is between 6 and 6.5 minutes. And the time needed for these two algorithms is about linear with the number of iterations.

At this point, the idea arose to combine the fast, deterministic Greedy-Demeshing algorithm with the slower meta-heuristic algorithms. For both HSA and GA an initial memory/generation is created at the start of a computation. Then around 1000 to 1500 iterations are needed to reach results similar to GD. What if we first compute a starting configuration with GD, and then put this configuration in the initial memory of a HSA computation/ initial generation of a GA computation. These two combinations were implemented and tested on the three scenarios of the Zaltbommel subnetwork as well, and the results can be found in Figure 8.5 for scenario 1, and in the appendix for scenario's 2 and 3. For scenario 1, these results are plotted, together with the results of the single algorithm computations, in Figure 8.6.

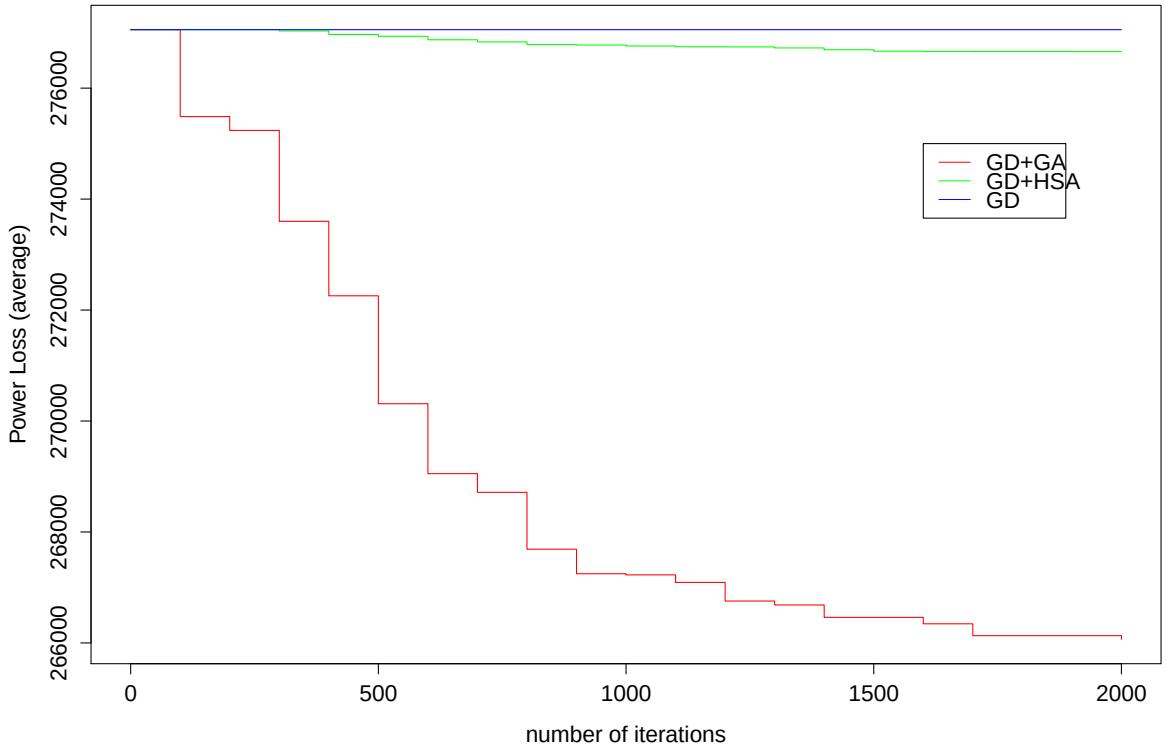

Figure 8.5: *results testing on Zaltbommel, scenario 1*

We can see that the combined algorithms start with similar behaviour at iteration 0 as the single algorithms somewhere between iteration 1000 and 1500. Clearly adding a good configuration found by a quick computation of GD can speed up HSA and GA extremely by making the first 1000/1500 iterations otiose. Furthermore, the observations made earlier are confirmed as well: The Genetic Algorithm decreases the minimal power loss found much faster than the Harmony Search Algorithm under the results of the Greedy-Demeshing algorithm. Based on these results and observations, a final method for a program that approximates LRRP is proposed, which will be described in the next section.

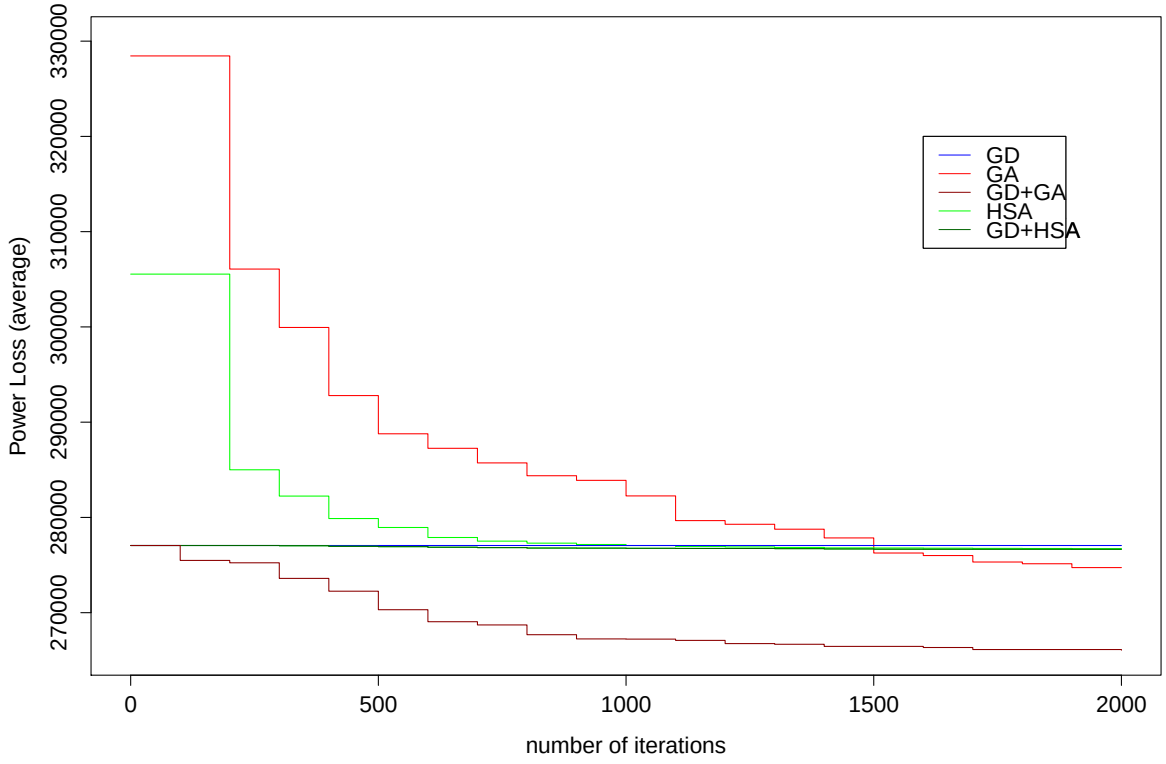

Figure 8.6: *results testing on Zaltbommel, scenario 1*

8.3 Proposal for a Final Method

Based on the results on test network 2 and on a subnetwork of Zaltbommel, a final method is proposed that finds for an MV network a configuration with small total power losses. After the implementation procedure and the testings on test network 2, the Greedy-Demeshing algorithm, the Harmony Search Algorithm and the Genetic Algorithm remained as serious candidates. The Greedy-Demeshing algorithm gives a good result in a relatively short time, but as can be seen in the testings on Zaltbommel, the Harmony Search Algorithm and the genetic Algorithm can improve this once enough iterations are executed. However, by combining GD with HSA and GA, the same results can be achieved in a much smaller number of iterations. Therefore, such a combination will be proposed as the most suitable method for approximating the Loss Reduction Reconfiguration Problem.

On the subnetwork of Zaltbommel it was clear that the improvement speed of the Harmony Search Algorithm decreased faster than that of the Genetic Algorithm. As a consequence, GA finds better configurations than HSA in less iterations. Therefore, the combination of the Greedy-Demeshing algorithm with the Genetic Algorithm is proposed as final method, the most suitable algorithm for finding feasible configurations of a network with small power losses. With this method, for an input network, first this network is adapted as described in Chapter 6. Then the Greedy-Demeshing algorithm quickly computes a starting configuration

of the network, as described in Section 5.2. This configuration forms the basis of generation 0 of the Genetic Algorithm, which is described in Section 5.4. With the method from Section 7.3, generation 0 is filled with randomly generated configurations. Then a cycle basis of the network is computed based on Horton's Algorithm, see Section 7.2. With the cycles in this basis as genes, new generations are build out of generation 0. When a (user determined) maximum number of iterations is reached, the algorithm stops and returns the best (or a number of best) configuration(s) found. See Figure 8.7.

Some additional features can be added to this algorithm. Suggestions are:

Final Method	
input	Network $N = (W, E)$ with impedances, powers, etc.
1	Run *Network Adapter* (figure 6.3) on N.
2	Run the *Greedy-Demeshing* algorithm (figure 5.2) on N. Let S_0 be the resulting switch specification.
3	Compute a cycle basis B of N with *Hortons Algorithm* (figure 7.5)
4	Set $G_0 = \{S_0\}$, and use *Random Radial Configuration Finder* (figure 7.6) to fill G_0 as generation 0 for the *Genetic Algorithm*.
5	Apply the *Genetic Algorithm* (*cycle basis version,* figure 5.5) to N with cycle basis B and initial generation G_0. Let S be the resulting switch specification.
6	**Return** S

Figure 8.7: *Final Method for solving LRRP*

- Normally a user might have a reasonable guess for good configurations, for instance the actual configuration, if it exists. Such a guess could be used for generation 0 as well as the output of GD. This would make generation 0 even better, probably resulting in better performance. Therefore a feature that makes this adding possible seems preferable.

- For some links or cables it might be preferable/necessary that they are opened. Others might be needed to be closed. A useful feature would be the option to add this information to the input. This narrows the search space and makes it more likely that the output configuration is actually a suitable configuration.

- The method could also be used for planning future connections in an MV net. If a number of possible connections is present, of which only a small number can be executed, one would like to know which connection has the best impact. By adding all these connections to the input data of the existing network, with realistic impedances and capacities, one could see which of the possible connections is actually used in the output configuration. These are the connections that would make power loss reduction possible.

- As mentioned in Chapter 1, Liander is not only interested in which configuration minimizes the power loss, but also for instance which configuration is most reliable, or has the best power balance. The Genetic Algorithm is very suitable for changing the objective function. And if line currents or nodal voltages/currents are needed for this objective function (which is the case for the examples), the Genetic Algorithm is probably even relatively fast approximation algorithm. In case of the Greedy-Demeshing algorithm, this transition to other objective functions is a bit more complex. The idea behind GD

is that the situation is at its best if all edges are closed, and by opening the right edges, this situation is effected as little as possible. For reliability and power balance this is not immediately clear, and this is beyond the scope of this project.

8.4 Texel and Zaandam: Performance of the Final Method

For the subnetwork of Zaltbommel that was used to test and compare the different methods, no good "Vision" data is available. Hence no good comparison could be made of the final method with the power losses in current configurations or in configurations resulting from the "Vision" tool, based on the Greedy-Shifting method. However, for an area of North-Holland, good "Vision" data is available. For two networks, of the island Texel (228 nodes, 1 slack bus) and the city Zaandam (406 nodes, 8 slack busses), this data was transformed in "R" data suitable for the final method. This made it possible to compare the results of the final method with actual power losses and with the "Vision" tool. See Figure 8.8.

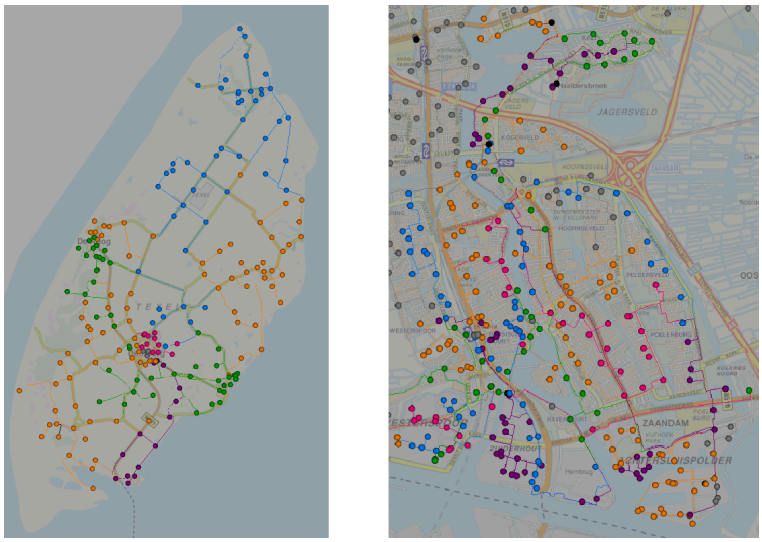

Figure 8.8: *The MV network in Texel (left) and Zaandam (right)* ("Vision")

Figure 8.9 shows the results of this comparison. On Texel a power loss reduction of 15% can be established, and in Zaandam almost twice as much, 27%. This difference is probably caused by the number of slack busses in the networks. The network of Texel contains one slack bus, where the network of Zaandam contains eight slack busses. Moreover, the (adapted, as described in Chapter 6) network of Zaandam admits a cycle basis of 49 cycles, where in Texel this number is 23. This makes the network of Zaandam far more complex than the network of Texel, with a much larger search space.

As can be seen in Figure 8.9, the final method gives much better results than the "Vision" tool. Furthermore, in the network of Zaandam several line currents exceed the corresponding capacities in the configuration suggested by this tool. We conclude that the final method is highly preferable over the existing "Vision" tool.

	MV network of Texel	MV network of Zaandam
Power loss in actual configuration	89538.38 W	95334.17 W
Power loss with Vision tool (Greedy-Shifting algorithm)	89210.65 W (0.5% reduction)	89627.98 W (6% reduction)
Power loss with final algorithm (GD+GA)	76127.57 W (15% reduction)	69989.04 W (27% reduction)

Figure 8.9: *Results comparing on Texel and Zaandam*

These results need to be nuanced. The final method is, as all the implemented algorithms, suitable for computing a good configuration based on one specific power demand/supply pattern in a network. However, in a network, the power demands and supplies normally vary in time. So an optimal configuration on one moment could be a bad configuration on another moment. More general, the power loss for a specific configuration normally varies in time. The results in Figure 8.9 are based on one specific power demand/supply pattern, namely 25% of the maximal load pattern. With this pattern, a power loss reduction of 15% and 27% can be established, but the reduction over a day, a month or a year is probably less. Nevertheless, considered that power losses in MV networks in 2014 led to 747.570.000 tons of CO_2 emission and costs of 40 million euros ([29]), even if a quarter of the results in Figure 8.9 can be established in reality, serious profits could be achieved. We conclude that the suggested final method forms a good fundament for creating a computer program usable in real situations for reducing power loss in the MV networks.

Chapter 9

Discussion

The final method for approximating the Loss Reduction Reconfiguration Problem as it is developed now, is believed to be capable to function as a basis for a network planning program. This holds especially for future situations, when a network can be reconfigurated quickly from one central point. Then any time the situation in the network changes, the new voltages and power supplies/demands can be used as input and the method will approximate the optimal configuration for this situation.

Because of the improvement described in Chapter 6, the method can be used for any kind of MV network, single- and multiple slack bus networks. This is a huge improvement with respect to existing algorithms, especially since the improvement is very general, and can be applied to any other algorithm for LRRP, even new ones. Therefore, this supports future innovation in this field, since with this method an algorithm for single slack bus networks is immediately an algorithm for multiple slack bus networks.

As claimed in Chapter 1, the adaptability of the final method to other objectives is taken into account as well. For the Greedy-Demeshing algorithm, this cannot be ensured in general, since this algorithm is based on the fact that the underlying, meshed structure of a network is optimal for the objective. This is the case for minimizing power loss, but it is probably not for every other objective. However, in the final method the Greedy-Demeshing algorithm serves only as a 'kick start' for the Genetic Algorithm. This greatly improves the perfomance of the method, but it is not indispensable. And the Genetic Algorithm on the other hand, is suitable for any other objective function in where the line currents and/or the nodal voltages in a configuration are needed (note that if these values are not necessary, then the demand constraint becomes futile and we consider a totally different problem). Indeed, we can simply change the fitness function in the Genetic Algorithm without adjusting anything in the process of the algorithm. Results in Chapter 8 show that this process based on evolution is the most suitable one among all the considered precesses and algorithms for searching optimal configurations. This probably does not change when the objective function changes.

It should be noted that by other objectives as mentioned above we understand objectives for networks in well-working operation state, so when no unisolated faults are present. An interesting question would be whether the final method could also function as a good basis for a program which can support by isolating a fault in a network and restoring the service

in this network. In this case, the objective function could be to restore service to as much consumers as possible in as little switch exchanges as possible. Or maybe even, in case of multiple possible reconfigurations for fault isolation and service restoration, to find among these reconfigurations the one with minimal power loss, or optimal load balancing, etcetera. This question goes beyond the scope of this thesis, and is not investigated.

As mentioned above, the final method is capable of serving as a good basis for a supportive program for netplanning activities, also for today. However, some adjustments or extensions are needed in order to achieve this. The main shortcoming of the method as it is now, is that it is only able to approximate the optimal configuration for one specific moment, in where to power demands/supplies and the voltages are constant. However, openening or closing a switch today has to be executed on location by a professional engineer, hence reconfigurating a network is a time consuming activity. Moreover, today switches are mechanical instruments which can only be switched a limited number of times before they crash. Therefore, it is desirable for present-day usage that an optimal configuration for a certain period instead of a moment is approximated, for instance for a day or a season. This desired extension is adopted as future work. However, it should be noted that this disadvantage also holds for the existing tool in "Vision" based on the Greedy-Shifting algorithm. So the final method remains superior over this tool.

Another shortcoming for operation is the running time. As can be found in Chapter 8, the final method takes several minutes for an MV network, but this must be reduced to seconds to make this interesting as a tool for network architects, especially in future situations. However, by using more powerful computers and improving the implementations of the algorithms, this seems achievable. This is another essential part of future work.

What makes the final method so suitable as a supportive tool for planning activities is that it can easily return not just one configuration, but the ten best configurations found can also be returned, or even the hundred best. This is highly advantageous over other algorithms, including the existing tool in "Vision", since in reality there are numerous practical limitations for network reconfigurations. Lines with switches which cannot be switched because the switch is currently unreachable or is broken, are not uncommon. Therefore the best configuration found could be impractical to execute, so it is useful to have other good alternatives.

Mathematical options for improvement are present as well. As argued in Chapter 7, it was not achieved to get a well-working implementation of the Mixed Integer Linear Programming algorithm. It could be that this algorithm is capable of better performance than the final method. However, this algorithm is also deterministic, so it will always return one and the same configuration for a specific network. Moreover, MILP does not seem suitable for other objectives as well. So these disadvantages with respect to the final method will certainly remain. However, maybe MILP could function as a better kickstarter than the Greedy-Demeshing algorithm, or maybe using both gives even better results. Therefore, it would still be useful to attain a well-working implementation of MILP.

Also the semi-ear decomposition versions of the Genetic Algorithm and the Harmony Search Algorithm were abandoned, based on mathematical grounds. But the author does not exclude the possibility that these versions could be improved. For instance, a method that could

decide which orders of a cycle basis give enough semi-ear decompositions to cover all possible radial configurations would make these versions interesting again. Especially if the number of order needed remains practical.

Another point of improvement that follows from Chapter 7 is that we actually want a cycle basis of a network that minimizes the sum of the pairwise intersections of the cycles. Horton's algorithm however, only gives a cycle basis in where the sum of the length of the cycles is minimized. Although an optimal basis for the second problem is often also an optimal basis for the first problem, this is not always the case. Hence this desire remains open as well.

However, the last two improvements become futile if another, more fundamental question is solved, based on Propositions 3.5 and 3.8. These propositions show a frustrating phenomenon for finding a radial configuration of a network. We have a method based on Proposition 3.5 which can construct any radial configuration of the network, only also non-radial configurations can occur with this construction. On the other hand, we have a method based on Proposition 3.8 that only constructs radial configurations. However, not all radial configurations can be constructed in this way. An extremely desirable result would be a method which can construct a configuration if and only if this configuration is radial. Implementing such a method in the Genetic Algorithm replaces the need for checking a configuration to be radial, making the algorithm more fundamental and faster.

One last point of discussion should be attended. The Genetic Algorithm contains several parameters, which are now tuned with test network 2. It is possible that the resulting values are no longer a good approximation of the optimal parameters for realistic networks. Therefore, finetuning on a real network could lead to better performance. Moreover, in the implementation of the final method, a very basic version of the Genetic Algorithm was used. However, the Genetic Algorithm is already an old algorithm, and many versions of this algorithm where developed. It might be that such a more complex version is more suitable for approximating LRRP. This is an interesting point of future investigation.

Chapter 10

Future Work

Based on the discussion in Chapter 9, the following subjects are suggested as future work:

- Adapt the method for calculations over a period of time. Now the algorithm returns a configuration that approximates the optimal solution for one specific moment. However, we also want to know good configurations for a certain period, for instance a day or a season.
- Speed up the final method. By using a more powerful computer, but also by implementing the algorithms more efficiently.
- Expand the method to other objective functions. For networks in a well-working state, but maybe also for networks in where a fault has occured. The second case lead to a significantly different problem about service restoration, so this will be more complex than the first case, in where the problem remains almost the same.
- Find a method that constructs every radial configuration of a network, and no more. This boils down to a desired proposition with a meaning somewhere between Proposition 3.5 and Proposition 3.8, since the former gives a method that finds all radial configurations but also more, where the latter only finds radial configurations, but not all.
- Get the Mixed Integer Linear Programming algorithm in a well-working operation state.
- Improve the semi-ear decomposition versions of HSA and GA.
- Find a method that returns a cycle basis of a network in where the sum of the pairwise intersection lengths of the cycles is minimized.
- Finetuning the Genetic Algorithm and investigating different versions of this algorithm.

Chapter 11

Bibliography

[1] J.A. Bondy, U.S.R. Murty, *Graph Theory*, Springer. (2008)

[2] P.M. van Oirsouw, *Netten voor distributie van elektriciteit*, Phase to Phase, Arnhem. (2012)

[3] R.A. Serway, J.W. Jewett, *Physics for Scientists and Engineers*, Cengage Learning. (2008)

[4] J.L. Kirtley Jr., *6.061 Introduction to Power Systems, Class Notes chapter 5, introduction to load flow*, MIT, Department of Electrical Engineering and Computer Science. (2003)

[5] A. Merlin, H. Back, *Search for a minimal loss operating spanning tree configuration in an urban power distribution system*, 5th power system computation conference. (1975)

[6] M.E. Baran, F.F. Wu, *Network reconfiguration in distribution systems for loss reduction and load balancing*, IEEE Transactions on Power Delivery, Vol 4, No 2. (1989)

[7] D. Shirmohammadi, H.W. Hong, *Reconfiguration of electric distribution networks for resistive line losses reduction*, IEEE Transactions on Power Delivery, Vol 4, No 2. (1989)

[8] V. Glamocanin, *Optimal loss reduction of distribution networks* IEEE Transactions on Power Delivery, Vol 5, No 3. (1990)

[9] G. Schäfer, *Discrete Optimization*, CWI, Networks and Optimization Group. (2013)

[10] Nara, Shiose et al., *Implementation of genetic algorithm for distribution systems loss minimum reconfiguration*, IEEE Transactions on Power Systems, Vol 7, No 3. (1992)

[11] J.Z. Zhu, *Optimal reconfiguration of electrical distribution network using the refined genetic algorithm*, Elsevier Science B.V. (2002)

[12] Rao, Narasimham et al., *Optimal network reconfiguration of large-scale distribution system using harmony search algorithm*, IEEE Transactions on power systems, Vol 26, No 3. (2011)

[13] Rao, Ravindra et al., *Power loss minimization in distribution system using network reconfiguration in the presence of distributed generation*, IEEE Transactions on power systems, Vol 28, No 1. (2013)

[14] R.A. Jabr, R. Singh, B.C. Pal, *Minimum loss network reconfiguration using mixed-integer convex programming*, IEEE Transactions on Power systems, Vol 27, No 2. (2012)

[15] S. Dimitrijevic, N. Rajakovic, *An innovative approach for solving the restoration problem in distribution networks*, Elsevier B.V. (2011)

[16] Aoki, Satoh et al., *Voltage drop constrained restoration of supply by switch operation in distribution systems*, IEEE Transactions on power delivery, Vol 3, No 3. (1988)

[17] A. Zidan, E.F. El-Saadany, *A cooperative multiagent framework for self-healing mechanisms in distribution systems*, IEEE Transactions on smart grid, Vol 3, No 3. (2012)

[18] P.M. De Oliveira-De Jesus, M.A. Alvarez, J.M. Yusta, *Distribution power flow method based on a real quasi-symmetric matrix*, Elsevier B.V. (2013)

[19] A. Trias, *The holomorphic embedding load flow method*, IEEE PES General Meeting. (2012)

[20] M.L. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Springer. (2008)

[21] M.E. Baran, F.F. Wu, *Optimal sizing of capacitors placed on a radial distribution system*, IEEE Transactions on power delivery, Vol 4, No 1. (1989)

[22] V.M. Da Costa, N. Martins, J.L.R. Pereira, *Developments in the Newton-Raphson power flow formulation based on current injections*, IEEE Transactions on power systems, Vol 14, No 4. (1999)

[23] Guido Schäfer, *Discrete Optimization*, Utrecht University. (2013)

[24] M.R. Garey, D.S. Johnson, *Computers and Intractability*, A Series of Books in the Mathematical Sciences, San Francisco. (1979)

[25] A. Ben-Tal, A. Nemirovski, *On polyhedral approximations of the second-order cone*, Math. Oper. Res., Vol 26, No. 2. (2001)

[26] J.D. Horton, *A polynomial-time algorithm to find a shortest cycle basis of a graph*, SIAM journal of computing, 16:359-366. (1987)

[27] https://cran.r-project.org/web/packages/lpSolve/lpSolve.pdf

[28] https://cran.r-project.org/web/packages/Rglpk/Rglpk.pdf

[29] Alliander Jaarverslag 2014

[30] J. Marsden, A. Weinstein, *Calculus II*, Springer-Verlag, New York. (1985)

[31] D. Weyland, *A Rigorous Analysis of the Harmony Search Algorithm*, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Switzerland. (2012)

[32] M.T. van der Meulen, *Literature Review on Distribution Network Reconfigurations*, Radboud University, Nijmegen. (2015)

Appendix A

Additional Results on Zaltbommel

The Greedy-Demeshing algorithm, Harmony Search Algorithm and Genetic Algorithm were tested and compared on Zaltbommel, with three scenarios. Results with the first scenario can be found in chapter 8. For scenario 2 and 3, the results can be found in the following figures. Figure A.1 and A.2 give the results of the single algorithms, and figures A.3 and A.4 give the results of the combined algorithms.

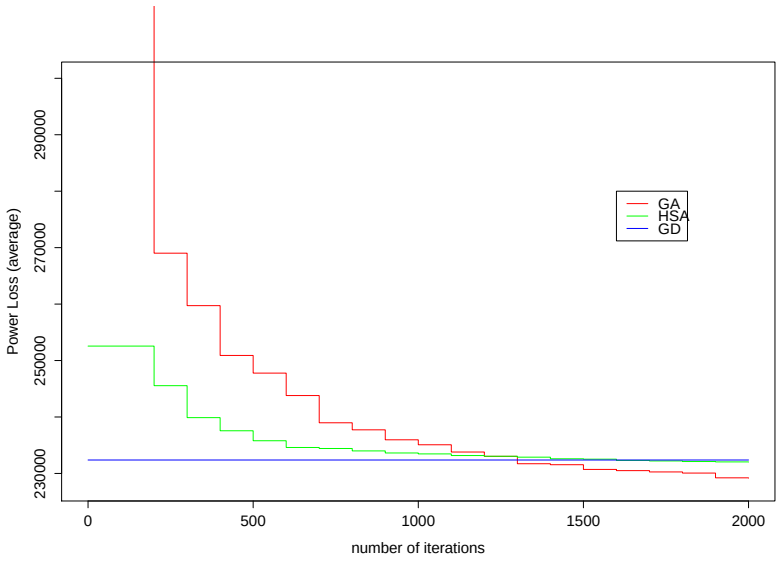

Figure A.1: *results testing on Zaltbommel, scenario 2*

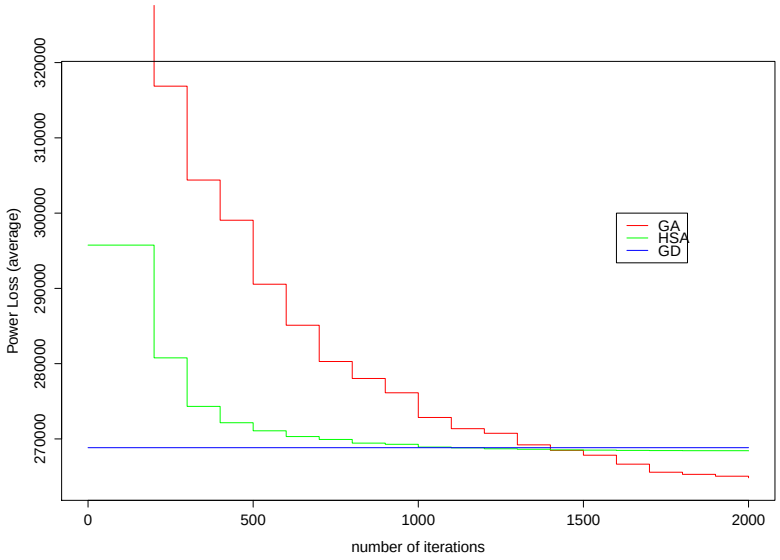

Figure A.2: *results testing on Zaltbommel, scenario 3*

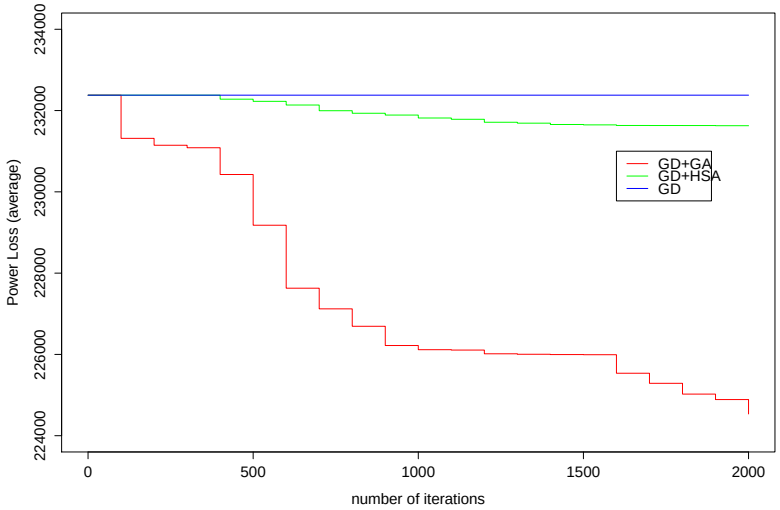

Figure A.3: *results testing on Zaltbommel, scenario 2*

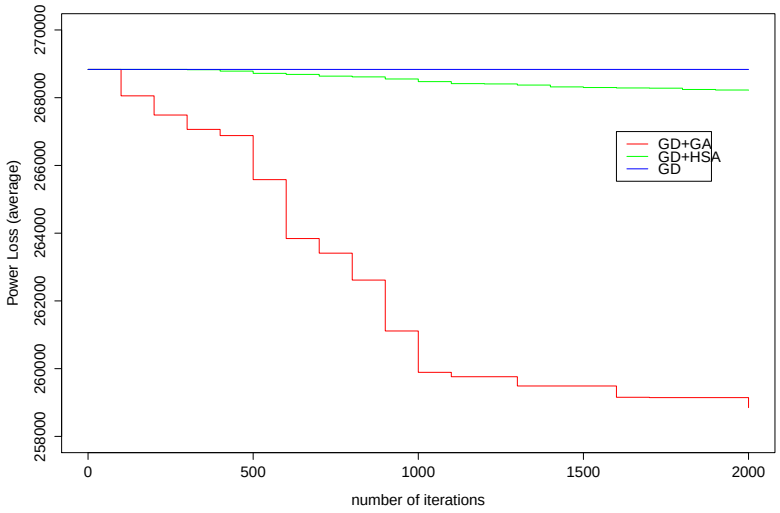

Figure A.4: *results testing on Zaltbommel, scenario 3*