Utrecht University

Master graduation project

Storage placement in electricity distribution networks: the SLOPER model

Author:
Stephan Leemhuis

Supervised by:
dr. ir. Marjan van den Akker,
Utrecht University

ir. Gabriël Bloemhof,
KEMA

August 31, 2011

ICA-0483311

Abstract

Storage systems can have a beneficial effect on operational problems in electricity distribution networks. However, they are very expensive and have not been applied much in electricity grids in Europe. When determining whether storage systems can have a positive influence on network operation, the cost of long term investments have to be balanced to the cost of short term operational benefits. Furthermore the planning optimization involves number, locations and sizes of the storages. To solve this problem, the SLOPER model (**S**torage **L**ocation **OP**timization **E**fficient **R**outine) was developed, which identifies where storage systems could be placed to benefit most, while also providing the optimal storage control strategy. It makes use of a simulated annealing approach with a linear programming model to determine the added value of the storage systems to the network. The SLOPER model proves to be an interesting approach to solving the storage location problem, and a promising approach to solving investment problems in electricity networks in general.

Contents

Chapter 1

Introduction

In this chapter we introduce the storage location problem. First of all, some principles of electrical engeneering are introduced. In the next section, the beneficial effects of storage systems are explained. This information is combined into a project aim and an approach to solving the problem. The chapter concludes with a short overview of the contents of this report.

In the storage location problem, the goal is to place storage systems in an electricity grid to add net value to the network. This means that the cost of investing in storage systems should be outweighed by the benefits of these storage systems. A solution to the storage location problem is a plan in which is determined where storage systems should be placed and of what exact type they should be.

1.1 The electricity grid

The electricity grid that provides us with electric power can be viewed in a similar simple fashion. Power plants produce power which is transported along a network of lines to smaller, disconnected networks that distribute the power. The *transportation network* is operated at a high voltage, to which the power plants and some high demand industries are linked. The *distribution networks* are lower voltage networks, that are connected to residential and small business consumers. The distribution networks are connected to the transportation network by *transformers*, which transform the high voltage power to a lower voltage.

Electricity networks are usually shown as *single line diagrams*. Although in reality there should be a circuit to allow the flow of an electric current, the second line of this circuit is omitted from this schematic to decrease the complexity. Because of this it seems as if current is 'created' at the power plant, is transported towards the consumer, where it is consumed and 'disappears' from the network. Although this is physically impossible, it is an easy way of displaying the dynamics of the network.

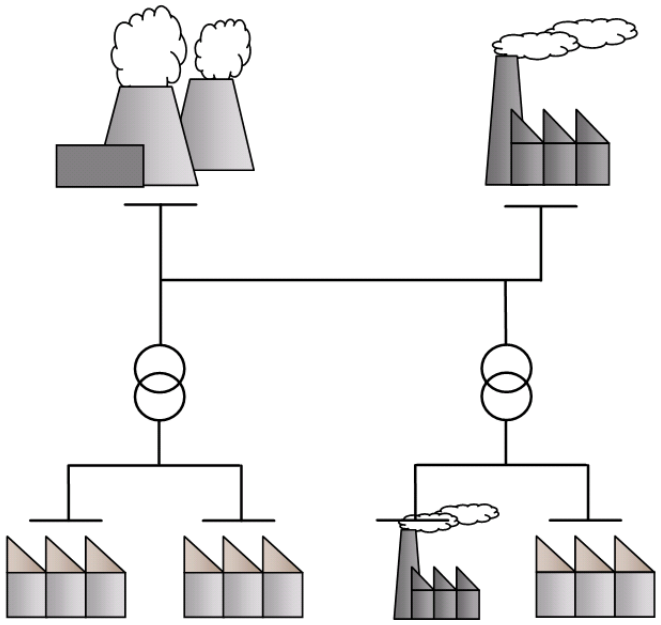


Figure 1.1: A single line diagram: a simple representation of an electricity grid. The upper half is the high voltage network, with a power plant at the left and a high voltage consumer at the right. The bottom half is the low voltage network, with domestic and business consumers.

1.1.1 Physics of electricity

The simple idea of electricity is that electrical charge flows through power lines. This flow is known as *current* and is measured in *Ampere*, which is an amount of charge (Coulomb) passing through a line per second. The potential of this current is expressed in *Volt*, which is a measure for the energy transported by this electrical charge. The power, which is the energy that is transported along the line per second, can be obtained by multiplying the voltage with the current

$$W = V \times I$$

where V is the voltage and I is the current. The power is expressed in *Watt*, a measure for energy (Joule) per second. The total energy transported can be obtained by multiplying the power with the period of time during which this power is transported. This total energy is often expressed in kilowatt hour (kWh), which is the energy transported during one hour with a power of 1000 Watt (3600 seconds $\times$1000 Watt = 3600000 J = 3.6 MJ).

The *resistance* R of a line can be calculated by

$$R = \rho \times \frac{l}{A}$$

where ρ is the electrical resistivity of the material the line is made of, l is the length of the line in meters, and A is the cross-section area of the line in square

meters. From this formula it can be seen that a longer or narrower line will have a higher resistance. This resistance, which is expressed in *Ohm*, is a measure for the opposition of the passage of an electric current through this line. This relation is given by Ohm's law

$$R = \frac{V}{I}$$

The larger the potential of the electric current is, the larger the electric current through the line will be. The resistance of two lines *in series* can simply be added to calculate the resistance of these two lines combined. If the lines are used *in parallel*, the combined resistance can be calculated by

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}$$

Using lines in parallel always decreases the combined resistance.

When power is transported along a line, part of it is lost due to the resistance of the line and will be converted into thermal energy. This loss of power can be obtained from rewriting the laws introduced above. When transporting power along a line, the voltage of the electric current will decrease according to Ohm's law:

$$\Delta V = I \times R$$

This decrease in voltage can be translatead to a decrease in power by multiplying with the flow of current:

$$W_{loss} = \Delta V \times I = I^2 \times R$$

As can be seen, this loss in power increases quadratically with the increase of the flow of current [8]. This is why the transportation network is operated at a high voltage: to reduce the flow of current through lines and thus to reduce losses.

In a *load-flow calculation*, a power network is studied during normal operation. Input for such a calculation is a network with given physical properties and given demands at each of the consumers positions (or given generations in case of decentral power generation). One position in the network will generate or consume an amount of power to compensate for the total net demand or production of the rest of the network. This position is known as the *slack bus*. With the load-flow calculation the flow of current through each of the lines and the voltage at each point in the network can be determined. These calculations are often used to plan for future expansion of the networks, or to determine the best way of operating the network. When too much electricity is transported along a line, the loss of energy will increase the temperature of the line, which may cause overheating and damaging. Also, with a larger flow of current the voltage drop will be higher, and the voltages at the position of the consumer might be out of the tolerated ranges. Because net operators have to avoid these types of problems, load-flow calculations can be used to analyze the possible solutions to these types of operational problems.

1.1.2 Storage systems

There are many different types of storage systems, which can have different types of functions in power networks. Since storage systems are often custom-built

according to the required specifications, the number of possible storage systems is almost endless. However, there is a number of properties shared between all systems. They all store electric energy by converting it to another kind of energy, which is converted back whenever electric energy is required again. For example, energy can be stored as thermal energy (as in molten salt systems), kinetically (flywheels), chemically (as in hydrogen fuel cells), or as gravitational energy (by pumping water upwards in a reservoir. The total amount of energy that can be stored is an important parameter of storage systems. The amount of energy that is stored in a storage system is called the *state of charge*. Another important physical limit is the amount of energy that can be converted per unit of time. Both charging and discharging of the systems is limited, and the speed at which both are done can influence both the efficiency and lifetime of the storage system.

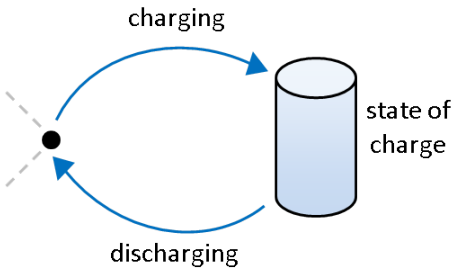


Figure 1.2: A schematic drawing of the physics of a storage system.

It is easy to understand how storage systems work, but what are the reasons to use them? In general, there are four beneficial effects that can be achieved by placing storage systems in an electricity grid

- Prevention of overloading
- Prevention of under- and overvoltage
- Stabilization in case of network problems (short circuits, blackouts)
- Trading

In the following sections we will show some examples of these positive effects.

Energy delivery and overloading prevention

The first important reason to use storage systems is to prevent overloading of power lines. During peak hours (typically the early morning and early evening), demands can rise to such a level that the capacity of the power lines is not sufficient, which is called overloading. Operating power lines at this level will cause damage because the generated heat cannot drain fast enough. This damaging will require the line to be replaced earlier, or can lead to burn out of the line, causing a (temporary) blackout in an area. A storage system can be charged during hours of low demand and can discharge during peak hours to prevent this

overloading. The energy deliverd by the storage system will reduce the energy deliverd through the power lines and thus prevent the overloading. Making use of storage systems in this case will only be profitable when the delivery lines are often overloaded and the delivery lines have sufficient capacity during the off-peak to allow charging of the storage system.

Storing decentral generated energy

With a growing focus on 'green' energy, the function of the electricity grid also changes. In the classical power grid, there were only electricity producers (coal/gas/oil fired plants, nuclear plants) and consumers (homes and companies). Nowadays, Combined Heat Power (CHP) systems, photovoltaic (PV) systems, windmills and other systems also produce a part of the required electricity. The difference here is that these systems can be placed anywhere in the network, hence the name decentral generated energy.

While the production of the classical production facilities could easily be regulated, this cannot be done for the new energy sources. Especially PV systems, which are often applied in residential areas, will produce most of their energy during the middle of the day, at which time power consumption in these areas usually is low. The excess generated power could maybe be used in other areas, but this requires transporting it along a higher voltage network. The losses that are involved with the transportation and transformation of energy however will cost a significant amount of the power generated. A storage system can in this case store the energy in the distribution network, and use this later to fulfill a net demand in the network. This idea works for any type of decentral energy generation. In fact, more and more renewable energy sources already are equipped with storage systems to make their power production more constant.

Voltage drops

When power is transported along a network, the resistance of the lines cause a decrease in voltage along this line, which is proportionate with the amount of current passing through the line. During times of high demand, this voltage drop might become so large that the voltage at the location of the consumer has dropped to a level that is below a lawfully acceptable level. Similarly, in the case of a net decentralized generation, the voltages might be higher than the tolerated maximum. A storage system can in the first case raise the local voltage by discharging, or in the second case decrease the voltage by charging.

Blackouts and short circuits

As mentioned before, storage systems can be useful for temporarily delivering power in case of a blackout. Power critical businesses, such as hospitals or datacenters, often have permanently spinning flywheels, from which energy can quickly be drawn in case of a blackout, to provide the systems with power while starting up reserve aggregates.

Also in electricity networks storage systems can have a function in case of a short circuit. Because the resistance along the path of the short circuit is much lower, this will cause a large short-circuit current in the direction of the short circuit. Of course, the security systems will quickly disconnect the part

of the network with the short circuit from the rest of the network when the large currents induced by the short-circuiting are detected. This however will not happen fast enough to prevent the voltage drop in the rest of the network, which might trigger other safety systems. As mentioned before, storage systems can help in preventing undervoltage. In this case, the power delivered by the storage system will be used to feed the short-circuiting, since the power will be directed towards the short-circuit location. In this way, the power for the short-circuit current does not have to be transported through the whole network, thus preventing the large voltage drop. Although this sounds like a waste of energy, it can prevent the ripple-effect of safety systems jumping into action throughout the whole network.

Trading

Because both the demand and the production of electricity depend on the hour of the day, the price of electricity will fluctuate accordingly. Storage systems can make a profit from these fluctuations by charging when prices are low and discharging when prices are high. This however requires the predicition of energy prices, which is a non-trivial problem. Moreover, the electrical energy has to be converted to a storable form of energy, and later on has to be converted back to electrical energy. The loss of energy occuring in the conversion processes makes the profit margins for trading of energy even smaller. Previous studies have shown that investing in storage systems for the sole purpose of trading is not profitable. However, the possibility of trading with the excess of the storage system capacity is an advantage that makes investing in storage systems more attractive.

1.2 Project aim

Electricity distribution networks were initially used for electric lighting in the early 1900's. Nowadays, electric energy is used for a large variety of household applications, and accordingly electricity consumption for households has increased. Solving the problems caused by the increase in demand traditionally was done by adding lines to the network in order to increase the capacity. However, this increased capacity is only required during the peak hours, and is not used during the rest of the time. Investing in extra lines is usually very expensive, especially when these power lines are below ground and streets have to be closed for construction. The aim of the project is to prove the value of storage systems in a network by solving these operational problems by making use of storage systems.

In the previous section, we mentioned there are four beneficial effects of storage systems. Prevention of overloading, keeping the voltage within the limits and trading are positive effects occuring during normal operation of the network. The fourth advantage, stabilization in case of network problems, is not considered because it does not offer an advantage during normal operation. Goal is to develop a prototype software system that can determine how storage systems can be used to solve overloading and voltage problems. It should be able to determine where storage systems should be placed, what type of storage

systems should be used and what the trading potential of the storage systems is. It should be possible to compare different *storage configurations*, which are placements plans of storage systems, and visualize the benefits of each of these configurations.

1.3 Approach

During the SWI 2010, the Study Group Mathematics with Industry, a model had been developed to determine the effects of decentral power generation. In particular, it was designed to determine the maximum amount of square meters of solar panels that could be placed in a distribution network without causing a line to overload. In the example distribution network, a number of houses with given load profiles was given, together with the solar insulation at each of the houses. The model made use of a loadflow calculation for each period of time to determine where overloading would take place. These loadflow calculations were formulated as a linear problem, which could be solved by linear programming.

Linear programming is a mathematical optimization technique that finds a solution to a system of linear equations in such a way that an objective function is either maximized or minimized. A linear problem consists of three parts:

- A set of variables
- A set of constraints
- An objective function

The values the variables can take are restricted to lower and upper bounds. Their values are further restricted by the constraints. These constraints have the general form

$$\textit{linear combination of the variables} = \textit{constant}$$

In this paper some constraints will be written with a *less-than-or-equal* ($\leq$) or a *greater-than-or-equal* ($\geq$) sign, these can be rewritten into the normal form above. The objective of a linear program is also a linear combination of the variables, and has to be either minimized or maximized.

A solution to a linear problem is an assignment of values to the variables, where this assignment of values obeys the lower and upper bounds for the variables, and obeys the restrictions set by the constraints. The optimal solution to a linear problem is a solution with the lowest possible (in case of a minimization) or the highest possible (in case of a maximization) value of the objective expression. Linear programming is a collection of techniques that is able to find the optimal solution to a linear problem. The technique of solving a linear problem has been around for over seventy years. Since then it has been extensively used in many applications, and the methods have been optimized for efficiency [1].

In this paper a method consisting of two models is proposed to solve the storage location problem. The first model is called the *loadflow series model* and is based on the model developed at SWI 2010. It will try to find a *storage operation plan* to minimize overloading of lines and to respect voltages bounds

as much as possible. Also, it will use the excess capacity of the storage systems and the network to trade. By doing so, the model will show the added value of a storage configuration in a given network.

The second model is called the *SLOPER model*, which attempts to find the storage configuration with the highest value. To do this, the SLOPER model will make use of the loadflow series model to analyze the added value of different storage configurations. It will look for a balance between investment cost and added value to determine the best solution to the storage location problem. The storage configurations that are examined are identified through *local search*.

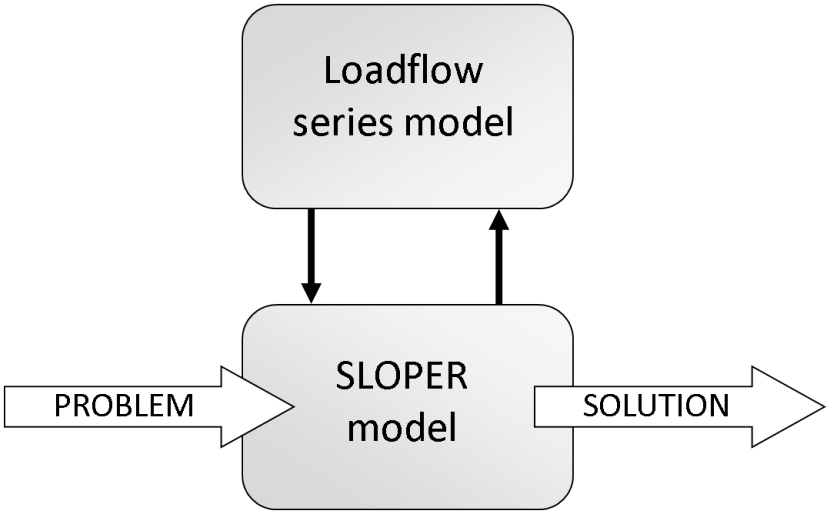


Figure 1.3: A schematic of the working of the SLOPER model.

1.3.1 Literature

Traditionally, investment decisions in electricity networks have been made based on expert judgement, but is more and more supported by large scale network calculations. Loadflow calculations have long been used to determine the effects of increasing demands, and probablistic models have been used to analyze risks in networks. Storage systems have been included in calculations [6] but mostly in time-series simulations or other stochastic calculations. Models to calculate the optimal power flow using storage systems, based on three phase alternating current loadflow calculations, have been developed as well [2, 4]. The idea behind the loadflow series model in that sense is not unique, but is different since the loadflow calculation is simplified to solving a DC system (ignoring phase angles which are used in DC loadflow calculations) [5].

So far, few publications have been dedicated to applying search routines to find solutions for investment problems. It is customary for experts to identify the key investment scenarios and use calculations to study the effects of these investment plans. In one study, evolutionary algorithms were used to find maintenance schedules for a group of power plants [7]. Another study uses non-linear integer programming to determine the size of a storage system in a microgrid

[3]. Also, KEMA has succesfully applied evolutionary algorithms to identify key positions for circuit breakers. Based on this application of search techniques, KEMA developed a tool to find solutions of network problems through adding power lines, adjusting transformer tap changer and the placing of storage systems. This tool makes use of three phase alternating current loadflow calculations. Since the iterative character of these calculations severely limits the speed at which calculations can be completed, the proposed model makes an interesting approach to solving the storage location problem.

1.3.2 Report content

In this chapter, the storage location problem was introduced, together with the approach for solving this problem. In the next chapter the loadflow series model, based on the model developed at SWI 2010, is explained. The following chapter will explain two approaches for solving the storage location problem, one of which is the SLOPER model. The next chapter is about the implementation of the models and the results from a number of tests for the SLOPER model. In the final chapters, it is discussed how well the model works and what improvements could be made.

This report has been written for two audiences: on the one side readers with an electrotechnical background, and on the other side readers with a background in mathematics/computing science. If some parts of the paper are an explanation of a subject that is common knowledge in the readers field, it is probably meant for another audience.

Chapter 2

The loadflow series model

In the previous section it was explained that solving the storage location problem is done in two steps. In this chapter we introduce a linear programming model for calculating a loadflow series for a network with fixed storage systems. First of all, the parameters that make up the problem instance are introduced. Next, the decision variables in the LP formulation are explained. The relations between these variables are then derived from the basic physical laws of electricity. The chapter concludes with a summary of the loadflow series model.

2.1 Problem parameters

An instance of the storage location problem at least consists of a power network, load patterns, and storage systems. The topology of the network is represented as a *graph*, which is a set of vertices V and a set of edges E. Next, we divide time into a discrete set of intervals called *timeframes*. This set will be called T. Finally, we have a set of available storage systems (also called storage types) called S.

2.1.1 Discrete time

The set T consists of small periods of time which we are call timeframes. Timeframes can have different durations, which will sometimes affect the properties of the network during such a timeframe. For example, the maximum amount of energy that can be stored in a storage system depends on the length of the timeframe, while the maximum allowed current through a line is constant. Therefore we introduce a parameter L_t, which is the duration of timeframe t in hours. In the model, we assume that all parameters and the system behaviour during a timeframe is constant.

2.1.2 Network topology

In this problem a single line power network is represented as a graph, which consists of vertices and edges. A vertex (also called node) is a point in space with a certain location which is in this case defined by a (x,y)-coordinate. It is not hard to see that a site of electricity production/consumption, a site where a storage is located or a connection point or busbar can all be modeled by a

vertex. For each vertex in the network, we define the parameter $P_{i,t}$, which is the fixed production of power at vertex i during timeframe t. The value of $P_{i,t}$ is determined by the amount of decentralized power generation and the demand from loads. If there is only a load at this vertex, this value will be negative, if there is only a distributed power generator, such as a PV system or windmill, it will be positive. If neither exists at vertex i, $P_{i,t}$ will be zero for all values of t, this is also called a *connection point*. The parameter $P_{i,t}$ is the fixed production of power, whereas the actual production of power at a vertex is also influenced by storage systems and/or grid connections.

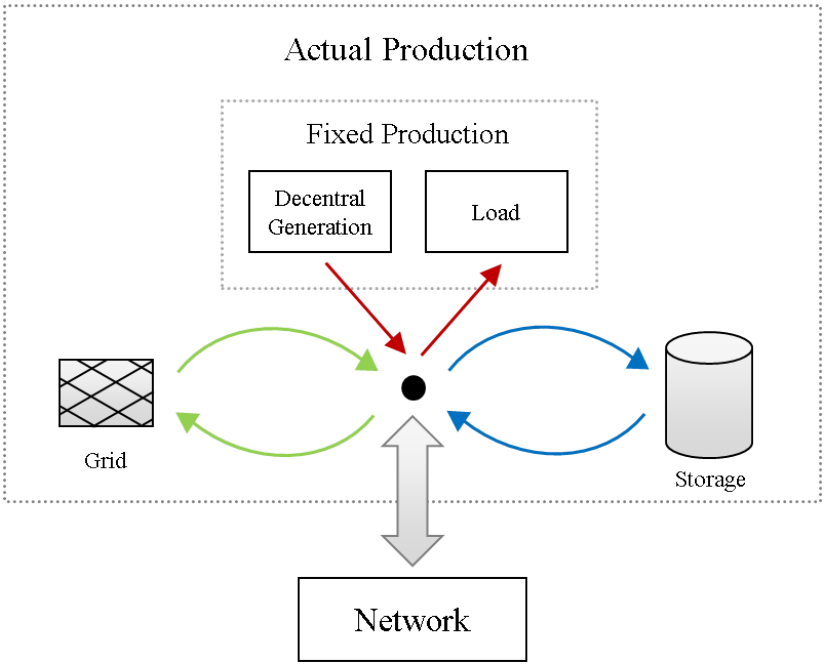


Figure 2.1: Difference between fixed production and actual production.

An edge is a connection between two of these vertices and is often represented by a combination of two vertices: $e_{i,j}$. Any line, cable or connector between two of these locations can be modeled as an edge. If there are multiple connections between two vertices, these are grouped and treated as a single edge such that the graph is not a *multigraph*. For each edge $e_{i,j}$ we have the resistance of this line $R_{i,j}$ and the rated current $I_{i,j}^{max}$, which is the maximum flow of current along that can be transported along the line indefinitely without damaging the line.

2.1.3 Storage systems

Apart from the network topology and the timeframes, we have a set of storage types. A storage type should not be confused with a *storage technology*, such as a flywheel or a liquid-metal system. Storage types are descriptions of the exact

properties of storage systems. Each type of storage system has a maximum charging rate S^+, a maximum discharging rate S^- and a maximum state-of-charge S^c. Since charging and discharging a storage system requires converting power from alternating current to direct current and vice versa, there usually is a loss factor on these operations. We will call the loss factor for charging the storage system e^+ and the loss factor for discharging the storage system e^-. Moreover, storage systems usually suffer from self-discharge, a process that reduces the state-of-charge even though the storage system is at rest. The per hour self-discharge rate of a storage system will be called e^c. Finally, a storage system also has a cost that is required to build it, which we will call C_s.

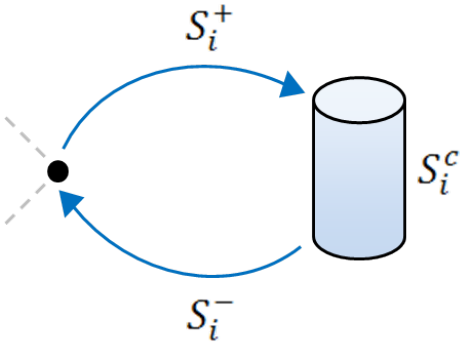


Figure 2.2: Limits on charging, discharging and state-of-charge for storage systems.

2.2 Basic model

The parameters treated in the previous section represented physical, non-changing values that describe the network. Since we want to calculate the loadflow series in the network, we have to define a set of variables to represent the flow of current and the voltages in the network and find the dependencies between them. Electricity cannot be 'directed' as you could do with discrete forms of transport such as trucks or packages. In fact, the flow of current is subject to both Kirchoff's current law and Kirchoff's voltage law. Also, adding storages to the model makes the actual production at each of the vertices variable, since we do not know yet when the storage systems will charge or discharge.

2.2.1 Kirchoff's current law

Let $f_{i,j,t}$ be the flow of current from vertex i to vertex j during timeframe t. If this value is larger than 0 current flows from i to j, if it is negative current flows from j to i. Also, $f_{i,j,t} = -f_{j,i,t}$.

Kirchoff's current law states that the current entering a point in an electricity network must also leave this point, so that there is no net buildup of current. We can state that the current leaving the vertex minus the current entering the vertex should equal the production of current at this vertex. For the graph

representation of the single line diagram we can translate this into

$$\forall t \in T, k \in V : \sum_{i \in V} f_{k,i,t} = q_{k,t}$$

where $q_{k,t}$ is the actual production of current at vertex k during timeframe t. We will come back to this actual production $q_{k,t}$ later.

2.2.2 Kirchoff's voltage law

Kirchoff's voltage law states that the net voltage drop around a loop in the network should be zero. This can be directly translated into a constraint for cycles in the graph network. Suppose we have a cycle consisting of edges $e_{0,1}, e_{1,2}, \ldots, e_{i-1,i}, e_{i,0}$. We have defined the voltage drop along an edge as $\Delta v_{i,j,t}$, which results in

$$\forall t \in T : \Delta v_{i,0,t} + \sum_{j=0}^{i-1} \Delta v_{j,j+1,t} = 0$$

for each cycle in the graph.

Since we know the resistance of each line, we can calculate the voltage drop along a line from the flow of current using Ohm's law:

$$\forall i, j \in V \times V, \forall t \in T : \Delta v_{i,j,t} = f_{i,j,t} \times R_{i,j}$$

Of course the voltage drop will be a voltage rise when the current flows from j to i. We can express the voltage at a vertex by the voltage at another vertex and the total voltage drop between them

$$\forall i, j \in V \times V, \forall t \in T : v_{j,t} = v_{i,t} - \Delta v_{i,j,t}$$

This means we can express the voltage at any of the vertices in terms of the voltage at the grid connection points and the flow of currents. For a vertex in a cycle this means that the voltage at this vertex is defined by the voltage drops along both sides of the cycle, and in this way Kirchoff's voltage law is automatically enforced.

2.2.3 Network operational limits

Apart from these physical properties of the network, we also want to assure that the currents and voltages in the network remain within certain limits. In the basic model the flow of current cannot be larger than the rated current, so

$$\forall i, j \in V, \forall t \in T : -I_{i,j}^{max} \leq f_{i,j,t} \leq I_{i,j}^{max}$$

($\forall i, j \in V$ means for each pair of vertices i and j in the total set of vertices, $\forall t \in T$ means for all timeframes t in the total collection T).

If we impose limits on the value of $v_{i,t}$ we can let the model assure that power is delivered at vertex i within a certain voltage range. By doing so, the model will not allow under- or overvoltage and will have to compensate for these by charging or discharging its storage systems. This can be represented as

$$\forall i \in V, \forall t \in T : V_N^{min} \leq v_{i,t} \leq V_N^{max}$$

2.2.4 Production of energy at vertex

In the previous section we defined a variable $q_{i,t}$ which is the net production of current at vertex i during timeframe t. This total net production is defined by the fixed production from the loads and decentralized power generation, and the production of current by charging or discharging a storage system and taking energy from a grid connection. How these values affect the total net production of current at a vertex is explained below.

If we impose limits on the value of $v_{i,t}$ we can let the model assure that power is delivered at vertex i within a certain voltage range. By doing so, the model will not allow under- or overvoltage and will have to compensate for these by charging or discharging its storage systems. This can be represented as

$$\forall i \in V, \forall t \in T : V_N^{min} \leq v_{i,t} \leq V_N^{max}$$

Storage

When a storage system is located at a vertex there is the possibility to charge or discharge the storage system. Let $s_{i,t}^+$ be the amount of current used for charging and $s_{i,t}^-$ be the amount of current used for discharging the storage system at vertex i during timeframe t. Let $s_{i,t}^c$ be the amount of energy stored at vertex i at the end of time t (state-of-charge). Of course these variables have to be in the allowed ranges

$$\forall i \in V, \forall t \in T :$$
$$0 \leq s_{i,t}^+ \leq S_{i,t}^+$$
$$0 \leq s_{i,t}^- \leq S_{i,t}^-$$
$$0 \leq s_{i,t}^c \leq S_{i,t}^c$$

Note that the physical quantity of $s_{i,t}^+$ and $s_{i,t}^-$ are current, and the physical quantity of $s_{i,t}^c$ is energy. The stored amount of energy obeys a simple inventory formula

$$\forall i \in V, \forall t \in T : s_{i,t}^c = s_{i,t-1}^c + (s_{i,t}^+ - s_{i,t}^-) \times L_t \times V_N$$

(L_t is the length of timeframe t and V_N is the nominal voltage of the network). The net production of current at a vertex is now described by the following formula

$$\forall i \in V, \forall t \in T : q_{i,t}^{storage} = s_{i,t}^- - s_{i,t}^+$$

With the current set of formulas, the inventory formula for timeframe 0 will be

$$s_{i,0}^c = s_{i,0}^+ - s_{i,0}^-$$

because there is no timeframe -1. This means that the storage is initially empty, and the storage systems will not be able to alleviate the violation of constraints of the system during the first few timeframes, because there is no energy stored. This might make it impossible to solve the network problems by placing storage systems. A way to prevent this from happening is to make the stored amount *cyclic*, which means that the state-of-charge at the **beginning** of timeframe T_0 is equal to the state-of-charge at time T_{final}

$$s_{i,0}^c = s_{i,t}^c + s_{i,0}^+ - s_{i,0}^-$$

In this way each of the storages can have an initial state-of-charge, but the state-of-charge at the end of the simulation defines the state-of-charge at the beginning of timeframe 0. This means that the amount of energy that is discharged during the whole time period is also charged at some point, and the storage system does not result in a net production of current.

In the inventory formulas it is assumed that the efficiency of converting the energy in the storage system is 100% and that the state-of-charge of a storage system is maintained without any losses. Since this is not the case in reality, have to introduce the efficiency factors for converting energy and maintaining storage. We introduce efficiency factors e_i^+ and e_i^- as the conversion efficiencies for charging and discharging the storage system at vertex i, which are values between 0 and 1. Furthermore, we introduce efficiency factor e_i^c which is the percentage of the state-of-charge that is left after keeping storing an amount of energy for one hour (also a value between 0 and 1). The inventory formula now can be written as

$$\forall i \in V, \forall t \in T : s_{i,t}^c = (e_i^c)^{L_t} \times s_{i,t-1}^c + (e_i^+ \times s_{i,t}^+ - \frac{1}{e_i^-} \times s_{i,t}^-) \times L_t \times V_N$$

The efficiency of maintaining storage depends on the length of the timeframe, but since both the length of the timeframe and the efficiency are given this value can be calculated beforehand. The charged power is reduced by multiplying it with its efficiency factor to account for energy loss during conversion. The discharged power is multiplied with the inverse of the discharging efficiency to keep it in the inventory formula. Because all efficiencies are accounted for in the inventory formula, the formula for the total net production of the vertex is not affected.

Grid production

In the network we also have to consider grid connections, which are sources of energy that can be accessed without any limits. These can be thougth of as mathematical simplifications of a transformer delivering power from a higher voltage grid. Because of this link to a higher (constant) voltage network and the way a transformer functions, we can safely assume the voltage at the point of the grid connection to be constant. The current produced by the grid connection is a measure for the power taken from the higher voltage grid. Since the fixed production of the decentral generation and loads are known, the only way energy can disappear from the grid is by the inefficiency of storage systems. If we add the amount of power taken from the grid in the minimizing objective function, we are effectively minimizing the losses in the system. If we use different per unit prices for the power taken from the grid at different times we will even see some trading happening. The amount of current produced by the grid connection at vertex i at time t will be called $g_{i,t}$. The price per unit of energy taken from the grid connection at vertex i at time t will be known as $k_{i,t}$. The production of current at a vertex is now expanded to

$$\forall i \in V, \forall t \in T : q_{i,t}^{grid} = g_{i,t}$$

Loads and decentral generation

The other contribution to the total net production of a vertex comes from decentralized power generation and the demand of loads. These two are combined in the fixed power production $P_{i,t}$. The fixed power production is a given, constant value, which has to be translated into an actual production in ampere I. In order to calculate this, we need to know what the actual power production at this vertex is. The general formula for determining the actual power produced P is

$$P = P_N \left(\frac{V}{V_N} \right)^p$$

where V_N is the nominal voltage, P_N is the nominal production of power, and p is the voltage exponent. We can rewrite this formula into an expression for the actual current production

$$I = \frac{P}{V} = P_N \times V^{p-1} \times (V_N)^p$$

We will discuss how to model *constant current production* and *constant resistance production*, which are most often used in modeling loads and generators.

Constant current production Suppose that the voltage exponent $p = 1$. The expression for the actual production of current simplifies to

$$I = \frac{P_N}{V_N}$$

which shows that the current does not depend on the actual voltage. Since we know both P_N and V_N in advance, the current produced at the vertex is constant, hence it is named constant current. When using constant current production, the production of current at a vertex is

$$q_{i,t}^{fixed} = \frac{P_{i,t}}{V_N}$$

Constant resistance production Suppose that the voltage exponent $p = 2$. The expression for the actual production of current now simplifies to

$$I = P_N \times \frac{V}{(V_N)^2}$$

which we rewrite to

$$I = V \times \frac{P_N}{(V_N)^2}$$

Since the last part of this equation is constant, it is clear that with an increasing voltage the production of current will increase proportionally. This behaviour is according to Ohm's law

$$I = \frac{V}{R}$$

These equations show that the constant part of the first expression $\frac{P_N}{V_N^2}$ is actually the inverse of the resistance. Therefore this type of modelling is called constant resistance. When using constant resistance production, the production of current at a vertex is

$$q_{i,t}^{fixed} = v_{i,t} \times \frac{P_{i,t}}{(V_N)^2}$$

2.2.5 Violating the constraints

The model described so far works perfectly fine for networks without any problems. However, since we are looking at networks in which problems occur, it might very well be that the given network has overload, undervoltage or overvoltage problems. In this case we have to violate some constraints to find some sort of solution to the problem. Since Linear Programming solvers will only find feasible solutions and stop whenever looking for a solution whenever infeasibility is proven, no solution will be found for networks with operation problems.

We can solve this issue by making use of *soft constraints*. The idea is to allow the model to violate some of the constraints, but the violation will have a negative effect on the objective function. In this way the model can 'buy' the right to violate the constraints and will be able to find a solution that normally would not be valid.

For each of the edges, we define a relaxing variable $x_{i,j}$ which is the extra current that is allowed to pass through line from i to j at any time. This variable allows the model to increase the rated current along this line. For each of the vertices, we define a relaxing variable y_i, which is the increase in the voltage range at vertex i. This variable allows the model to have voltages below or above the minimum or maximum nominal voltage. We use these variables in the bounds of the variables for the flow of current and the voltage

$$\forall (i,j) \in V, \forall t \in T : -I_{i,j}^{max} - x_{i,j} \leq f_{i,j,t} \leq I_{i,j}^{max} + x_{i,j}$$

$$\forall i \in V, \forall t \in T : V_N^{min} - y_i \leq v_{i,t} \leq V_N^{max} + y_i$$

Since we do not want to overload lines or deviate from the nominal voltage ranges, we want to encourage the model to stay within these limits by adding the relaxing variables to the objective function with large cost factors. The larger these cost factors are, the harder the model will try to prevent overloading and under- and overvoltage. The cost factor for increasing the rated current of a line by one Ampere will be called X, the cost factor for increasing the minimum and maximum nominal voltage for a vertex by one Volt will be called Y.

2.3 Model summary

In this section we summarize the total model into a *linear programming model*, consisting of an objective function, a set of constraints, and a set of variable bounds.

Objective

The objective consists of two parts. The first part is to minimize the energy taken from all of the grid connections multiplied by their prices. The second part is to minimize the cost involved with violating the constraints: for each line minimizing the violation of the rated current, and for each vertex minimizing the violation of the voltage bounds.

$$\min \sum_{i \in V} \sum_{t \in T} g_{i,t} \times k_{i,t} \times V_N + \sum_{(i,j) \in E} X \times x_{i,j} + \sum_{i \in V} Y \times y_i$$

Constraints

The sum of the currents leaving the vertex should be equal to the total net production of current of the vertex.

$$\forall t \in T, k \in V : \sum_{i \in V} f_{k,i,t} = q_{k,t}$$

The voltage drop along each edge is the product of the flow of current along the edge and the resistance of the edge.

$$\forall i, j \in V \times V, \forall t \in T : \Delta v_{i,j,t} = f_{i,j,t} \times R_{i,j}$$

The voltage at a vertex is defined by the voltage drops along the adjacent edges and the voltage at the bordering vertices.

$$\forall i, j \in V \times V, \forall t \in T : v_{j,t} = v_{i,t} - \Delta v_{i,j,t}$$

The total net production of current of a vertex is the sum of the net production of current of the load, grid connection and storage at this vertex (constant current load)

$$\forall i \in V, \forall t \in T : q_{i,t} = g_{i,t} - s^+_{i,t} + s^-_{i,t} + \frac{P_{i,t}}{V_N}$$

or

The total net production of current of a vertex is the sum of the net production of current of the load, grid connection and storage at this vertex (constant resisance load)

$$\forall i \in V, \forall t \in T : q_{i,t} = g_{i,t} - s^+_{i,t} + s^-_{i,t} + v_{i,t} \times \frac{P_{i,t}}{V_N^2}$$

The total energy stored in a storage system at a vertex at the end of timeframe t is the total energy stored at the end of timeframe t-1 increased with the energy charged during timeframe t and decreased with the energy discharged during timeframe t, taking into account losses in conversion of charging and discharging, and losses for storing the energy during timeframe t

$$\forall i \in V, \forall t \in T - \{T_0\} : s^c_{i,t} = (e^c_i)^{L_t} \times s^c_{i,t-1} + (e^+_i \times s^+_{i,t} - \frac{1}{e^-_i} \times s^-_{i,t}) \times L_t \times V_N$$

The total energy stored in a storage system at the end of the simulated time period should be the same as the total energy stored at the beginning of the simulated time period.

$$\forall i \in V : s^c_{i,T_0} = (e^c_i)^{L_{T_0}} \times s^c_{i,T_{|T|}} + (e^+_i \times s^+_{i,T_0} - \frac{1}{e^-_i} \times s^-_{i,T_0}) \times L_{T_0} \times V_N$$

Bounds

The flow of current along each of the edges may never be larger than the rated current of the edge.

$$\forall (i, j) \in V, \forall t \in T : -I^{max}_{i,j} - x_{i,j} \leq f_{i,j,t} \leq I^{max}_{i,j} + x_{i,j}$$

The voltage at each of the vertices may never be larger than the maximum nominal voltage and never be smaller than the minimum nominal voltage.

$$\forall i \in V, \forall t \in T : V_N^{min} - y_i \leq v_{i,t} \leq V_N^{max} + y_i$$

A storage system can never be charged more than its maximum charging power, it can never be discharged over its maximum discharge power and can never contain more or less stored energy than the minimum and maximum capacity of the storage system.

$$\forall i \in V, \forall t \in T :$$

$$0 \leq s_{i,t}^{+} \leq S_{i,t}^{+}$$

$$0 \leq s_{i,t}^{-} \leq S_{i,t}^{-}$$

$$0 \leq s_{i,t}^{c} \leq S_{i,t}^{c}$$

Chapter 3

Model for optimization of storage location: the SLOPER model

In the previous chapter a linear programming model for the calculation of the loadflow series was explained. In this section this model will be made to allow changes in the configuration of the storages. From the adapted model it is only one small step to an Integer Linear Programming (ILP) model, which solves the storage location problem to optimality. Solving this problem to optimality however requires enumerating (almost) all of the possibilities, which will take too long for any real instance. Therefore a local search heuristic is developed to avoid enumerating all possible storage configurations and keep the calculation manageable even for larger instances. The approach using local search has a number of big advantages, one of them is the transformation of the model into a form that can be solved much faster.

3.1 Integer Linear Programming model

In the previous section, we modeled a network with fixed storage systems. Since we want to examine different storage configurations, we could build a new model for each storage configuration we wish to examine. This however is not a very efficient approach: it is best having to build the model only once. Problem is that in the storage location problem, we do not know in advance where the storage systems will be positioned. To solve this, we build the model as if there is a storage system positioned on every available location. The dimensions of the storage system (maximum charge rate, maximum discharge rate and storage capacity) will be changed to simulate the different storage systems. If we want to simulate a position without a storage system, we set all the dimensions to zero, which results in a free but useless storage system. We will assume that there can be only one storage system at each position.

Next, we have to add a part to the model to allow switching between the different storage systems at each location. For this, we make use of *integer*

variables. These variables can only take integer values between the given lower and upper bound. We create an integer variable $d_{s,i}$ for each combination of a storage type and a location where a storage system can be placed. These variables will be allowed to take the values of 0 and 1. The variable $d_{s,i}$ will be zero if there is no storage system of type s at vertex i, and will be one if there is such a storage system. We only want to allow solutions with one storage system per position, which we ensure as follows:

$$\forall i \in V : \sum_{s \in S} d_{s,i} \leq 1$$

Because each of these variables is integer, only one can take the value of 1, or they can all be zero.

Since we have now enforced that there will be at most one storage system at a position, we can easily express the dimensions of the storage system as the sum of the dimensions of each type multiplied by this integer variable. For the storage capacity this can be written as

$$\forall t \in T : s^c_{i,t} = \sum_{s \in S} d_{s,i} \times S^c_s$$

where S^c_s is the capacity of a storage system of type s. In the same way, this can be done for the maximum charge and maximum discharge rates. Since we know that only one of the variables $d_{s,i}$ can be one at the same time, the dimensions of the storage system will be defined as the dimensions of the storage system that has its corresponding integer variable set to one. If there is no storage system at this position, the expression above will evaluate to 0, and we effectively have no storage system.

The investment costs for placing all the storage systems is a simple linear expression built from the integer variables and the costs of buying a storage system of a certain type:

$$\sum_{s \in S} C_s \times \sum_{i \in V} d_{s,i}$$

This expression can be added to the objective so that we now also minimize the investment cost.

Solving a problem with integer variables sounds a lot easier, since these variables can only take a few different values. However, because these values are discrete, the mathematical solution method used for linear programming (LP) problems does not apply for integer linear programming (ILP) problems. The method to find the optimal solution to the problem is to assign all possible values to the integer values, and then solve the remaining LP problem with the same method. However, this requires enumerating a large number of possible storage configurations. At each location we can place each of the storage types or choose not to place a storage type. This leads to a total of $(|S|+1)^{|V|}$ different storage configurations, a number which grows exponentially when increasing the number of storage types or the number of locations. Such a rapid growth in possible combinations and therefore solving time is typical for a *combinatorial problem*. Although most solvers have efficient methods to significantly reduce the number of possible configurations that need to be examined, the solution time will remain to grow exponentially.

As a second problem, we cannot incorporate storage losses in this model, because we cannot describe the effect of the integer values on the self-discharge and conversion loss parameters in a linear way. Changing these storage parameters requires making changes to the model, so the ILP solver program has to be interrupted to do this. We can overcome this problem by assuming that all storage systems have no self-discharge or conversion losses. Since losses are a significant characteristic of the different storage systems however, the model will favour storage systems that would normally have high losses. This problem could be overcome by not only adding a virtual storage to every node, but adding a virtual storage system *of each type* to each node. However, this will increase the number of variables rapidly and will increase solving time even more.

To overcome the problems of the exponential growth of the problem and to be able to incorporate storage losses, we will try a different approach called *local search*.

3.2 Local Search

Local search is a name for a collection of search heuristics that work along a general principle:

1. Start with an initial solution to the problem as the current solution
2. Randomly move from the current solution to another solution in the *neighbourhood space*
3. If the new solution is better than the best found solution so far, remember this solution as the new best solution
4. Decide to accept or reject the new solution according to the *decision heuristic*. If it is accepted, the current solution will be this new solution
5. Repeat step 2 until some *stopping criterion* is reached
6. Report the best found solution

In the short description, three new terms were introduced. The *neighbourhood space* is a set of solutions that can be reached from a given solution by applying one of the possible changes to the given solution. Usually these changes are relatively small and will not have a large effect on the quality of the solution, hence the name neighbourhood space. Which solutions belong to the neighbourhood space of a solution is part of the design of the algorithm and might have a big impact on the quality of the solution found at the end of the routine.

The *decision heuristic* is largely determined by the chosen local search method. The decision to accept or reject the new solution could depend on the quality of the new and old solution, on the previously visited solutions (as in Tabu Search) or on the progress of the search progress and the quality of the new solution compared to the old (as in Simulated Annealing). However, all decision heuristics return a simple 'yes' or 'no' which implies whether to keep the new found solution or to reject it.

The *stopping criterion* can also be based on different parameters. It could decide to terminate the search progress after a number of states has been visited, or after a number of visited states without finding a better best solution. If the quality of the best solution (or an estimate) is known, it is also possible to terminate when the quality of the found solution is within a certain range of this best known quality.

Local search is an efficient technique for two main reasons. First of all, it does not require the enumeration of all possible solutions to the problem, but it only searches a small part of the solution space. When local search is used, it is no longer guaranteed that you will find the optimal solution, as you would when using the ILP formulation. Since the number of possible solutions to the storage location problem can become enormous (a network of 100 nodes with 10 different storage systems has 1.37×10^{104} solutions), enumerating all solutions is an impossible task.

The second advantage of local search is based on the thought that applying a small change to the solution will allow a fast recalculation of the subproblem. This implies that the solution of the problem before the change should hold some information about the new situation. Recalculating a solution to a linear programming problem after changing some of the constraints or variable bounds is relatively easy, and will take a lot less time than calculating the solution without any prior knowledge about solutions. Because small changes will allow for fast recalculation, local search seems very suitable for the storage location problem.

How do we implement this local search heurstic for the storage location problem? As with the ILP model, we will create storage variables for each available location, and change the dimensions of the storage systems to simulate the different storage systems. We will start with an initial solution which can be either provided by the user or, by default, is a network without any storage systems. We do not introduce variables that indicate the existence of a storage system at a vertex, we simply adapt the bounds of the variables for the charge, discharge and capacity. Solving the model means finding a loadflow series for a single storage configuration. Therefore we can also change the constraints that control the storage losses when changing the configuration. In the ILP model this was not possible, because solving the model meant solving the storage location problem, not just a single storage configuration as is the case with the local search.

The quality of the solution in this case is the score of the objective function of the model, which is a measure for the losses in the network, the cost of investment, the penalties of violating the bounds and possibly the effect of trading.

3.2.1 Neighbourhood space definition

The neighbourhood space is defined by a number of simple actions that can be taken to transform the current solution into a new solution. These moves are selected randomly and therefore each have a probability to be selected. The probability of selecting these moves has an influence on the search process and will be determined later on.

- **Add**: A storage system is added to a position. A random location that does not contain a storage system is selected. Next, a random storage type is selected. A storage system of the selected type is simulated at the location by changing the variable bounds of the storage variables in the model, and the new score is calculated. Of course this move cannot be applied if there are no available locations for storage systems left.

- **Remove**: A storage system is removed from the network. A random location that contains a storage system is selected. The storage system is removed from the model by setting the variable bounds of the storage variables to zero, and the new score is calculated. Of course this move cannot be executed if there are no storage systems in the network.

- **Change**: The type of a storage system is changed. First, a location that contains a storage system is selected. A random storage type that is not the same as the current storage type is selected. The corresponding variable bounds in the model are changed and the new score is calculated. This move cannot be executed if there would only be one single type of storage system.

- **Swap**: Two locations are selected and swap their storage systems. The first selected location will always contain a storage system, while the second is a random location which may or may not contain a storage system. If the second location does not contain a storage system, we effectively move the storage system from the first position to the second. Locations that are *adjacent* (connected by an edge) are given a higher probability of being selected than other locations. The storage systems are swapped by swapping the bounds on their variables, and the new solution is calculated. This move cannot be executed if there are no storage systems in the network.

3.2.2 Decision heuristic

The decision to accept or reject a new solution will be based on the quality on the score of the new solution. We consider two different types of decision mechanisms.

The most simple decision heuristic available is the one applied in *hill climbing*. With this heuristic, a new solution is accepted only if it is an improvement over the current solution. Therefore the current solution will always be the best solution found so far. This simple approach will sometimes get the algorithm stuck in a *local optimum*, that is a state of which all neighbourhood states have a worse score. It will not always be the case that this local optimum is also the *global optimum*, which is the best solution in the whole search space. Because of this, the success of hill climbing heavily depends on the definition of the neigbhourhood space. If the neighbourhood space is well defined, with little to no local optima, hill climbing will perform well.

The second decision heuristic considered in the search algorithm is applied in *simulated annealing*. With simulated annealing we calculate the probability that the new solution is accepted or rejected as follows:

$$p(accept) = \begin{cases} 1 & \textbf{if } score_{new} \leq score_{prev} \\ e^{-\frac{(score_{new} - score_{prev})}{T}} & \textbf{otherwise} \end{cases}$$

where T is the 'temperature' of the search progress. As in the hill climbing heuristic, improvements will always be accepted, but now deteriorating solutions also have a chance of being accepted depending on the difference in score and the temperature. During the search process, the temperature will drop with every number of steps accepted by the decision heuristic, making it harder for deteriorations to be accepted. The drop of the temperature is an analogy for a metallurgic process called annealing, where a metal is slowly cooled to obtain a better crystal structure. With simulated annealing, the rate of cooling determines both the speed of convergence to an optimum as well as the quality of the solution found. Slower cooling means slower convergence, but will in general provide better solutions.

Using local search for solving the storage location problem has two great advantages. Firstly, the linear programming model has the ability to quickly calculate the solution to a slightly changed model. When the model has been solved initially and changes to the model are made by adapting the bounds of variables, the solution to this new model can be calculated using the previous solution. Although the old solution might be infeasible to the changed model, a feasible solution can be created using some simple mathematical operations. This first feasible solution is likely to be close to the best solution of the new model, since most of the model remained the same. Because local search always moves from one solution to another using small changing operations, recalculating the new solution is likely to happen much faster than finding a solution without knowing anything from the previous solutions. This advantage would not be apparent when using *genetic algorithms*, since these work with a pool of solutions.

Secondly, the model used for calculating the loadflow series in the local search heuristic does not contain integer variables. Because the model is completely linear, there is a corresponding dual problem to the initial problem. In the next section it will be explained how the model can be reduced and transformed, and we can make use of the dual problem to find solutions even faster.

3.3 Model transformation

In the linear model described in the previous chapter, we used variables to quantify the current through lines and the voltages at each of the nodes, as well as charging and discharging of storage systems and their state-of-charge. The constraints were used to ensure that none of the lines was overloaded and to limit the range of voltages at the nodes. Also, constraints were used to assure that the state-of-charge of the storage systems was always in its limits and to ensure that all physical laws were respected. These variables and constraints together make up the linear programming model.

A linear programming model can be efficiently solved to optimality by linear programming solver programs. The models are solved through expressing the

constraints in a matrix and the objective function and ranges of the constraints into vectors. Any linear programming problem can be transformed into this normal form, which looks like

$$\text{Minimize } c^T x$$

$$\text{subject to}$$

$$Ax \leq b$$

$$x \geq 0$$

In this form, x is the vector of variables, c is the vector of costs assigned to each variable (c^T is its transpose), A is the matrix containing all the constraints and b is the vector of limits to the constraints. The dimensions of the matrix are $m \times n$, where m is the number of rows and n is the number of columns. The number of variables in the problem is equal to n, while the number of constraints is m.

This system of equations is then solved by numerous mathematical operations on this matrix to transform it into a representation of the optimal solution. The different methods that exist for doing this transformation are all based on the simplex method. The speed at which the simplex method converges to the optimal solution largely depends on the number of constraints in the model, while the number of variables hardly has an influence. So, reducing the number of constraints in the model would lead to a faster computable solution. However, removing any of the constraints mentioned could lead to solutions in which the variables have values which they should not have. Removing constraints from this problem is therefore not an option.

3.3.1 Duality

If we model the linear programming problem in the way we did above it is called the primal problem. We can transform this primal problem in the dual problem, which has the following normal form

$$\text{Maximize } b^T y$$

$$\text{subject to}$$

$$A^T y \geq c$$

$$y \geq 0$$

It can clearly be seen that the original vector of variables x is replaced with another vector of variables y. Although the vectors b and c and the matrix A exist in the primal as well as in the dual problem, it is hard to see what the connection between the two is. The strong duality theorem states that the optimal solution to the primal problem will have the exact same value as the optimal solution to the dual problem, also written as

$$c^T x^* = b^T y^*$$

where x^* is the optimal assignment of values to the variables in x and y^* is the optimal assignment of values to the variables in y. More importantly, the

optimal solution to the dual problem y^* can be transformed into the optimal solution to the primal problem x^*.

So transforming our problem into the dual problem will give us a different problem, which still happens to be linear. Because it is still linear we can solve it using the same solver. The optimal solution to the dual problem can be transformed into the optimal solution to the primal problem. However, this does not yet seem to make our problem any easier.

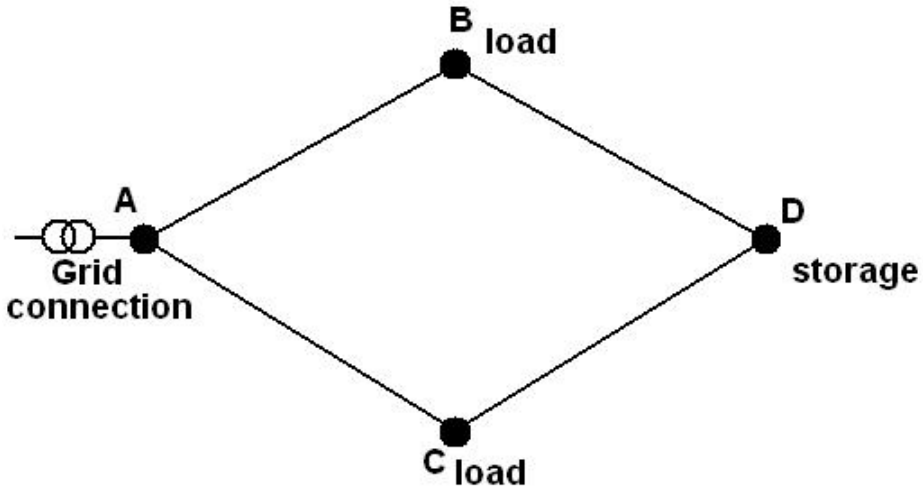
The advantage of transforming the primal problem into the dual problem in this case can be seen from the formulation of the dual problem. The dimensions of the matrix A in the primal problem were $m \times n$, which resulted in a problem in the normal form with m constraints and n variables. In the dual problem, the constraint matrix is A^T, the transpose of the matrix A. Since the transpose of a matrix is its reflection in its main diagonal, the matrix A^T will be an $n \times m$ matrix. This leads to a linear programming problem with n constraints and m variables, which is the other way around as in the primal problem. If we can reduce the number of variables in the primal problem, we reduce the number of constraints in the dual problem.

As was stated before, the speed of solving a linear programming problem depends on the number of constraints in the problem. So reducing the number of variables in the primal problem will allow us to solve the dual problem faster. In the next section it will be epxlained how we can reduce the number of variables in the problem without essentially changing the problem.

3.3.2 Reducing the number of variables

The transformation of the model in this section has one simple goal: reducing the number of variables to be able to solve the model faster. The resulting model will be the same in terms of the optimal solution, but is less intuitive to explain. The number of variables will be reduced by expressing one variable into another, thus eliminating the need for one of them. This will be done through a matrix transformation. It is important to note that the matrix in this section have nothing to do with the matrix of the linear programming model in the previous section.

In a basic loadflow calculation, the goal is to calculate the current through each line and the voltage at each point. Both of these are quantified by the demand and production at each of the locations. So for deterministic loads and productions, there can only be one outcome to the values of the current flows and the voltages. By including storages we give the model an extra opportunity to alter the demand and production at each of the nodes, resulting in different voltages and currents in the network. However, the dependency of the currents and voltages on the production and generation does not change. This means that the currents and voltages can be directly derived from the fixed demand and production and the production and demand from the storage systems. We will see this in the following example.



We consider the small example network above during a single period of time. Vertex A has a connection to the grid, while a storage system is positioned at vertex D. There are two current constant loads in this network, located at vertices B and C. We will try to express the flow of current through the edges and the voltages at each of the vertices in terms of the production of the storage system. The production of current by each of the loads is -10 (so they both use 10 ampere). The resistance of each of the lines is 1 Ohm. The grid connection at vertex A will ensure a constant local voltage of 230 Volt (the nominal voltage of the network).

We have variables for the flow of current along the four edges (f_{AB}, f_{AC}, f_{BD}, f_{CD}), variables for the voltage at the four vertices (v_A, v_B, v_C, v_D), and a variable for the flow of current through the grid connection g_A. Also, we have the production of current by the storage system s_D to yield a total of ten variables. The production of current at the vertices and the voltage drop along the lines are not considered as separate variables, instead they will be expressed in terms of these ten variables above. First we describe the production of current at each of the nodes using Kirchoff's Current Law

$$g_A - f_{AB} - f_{AC} = 0$$
$$f_{AB} - f_{BD} = 10$$
$$f_{AC} - f_{CD} = 10$$
$$f_{BD} + f_{CD} + s_D = 0$$
which we rewrite to
$$f_{BD} + f_{CD} = -s_D$$

Next, we link the voltage at each of the nodes to the voltage drop along each of the edges using Ohm's Law

$$v_A - f_{AB} \times R_{AB} - v_B = 0$$
$$v_A - f_{AC} \times R_{AC} - v_C = 0$$

$$v_B - f_{BD} \times R_{BD} - v_D = 0$$
$$v_C - f_{CD} \times R_{CD} - v_D = 0$$

Since the resistances of each of the lines is 1 Ohm, we can omit the resistances from the equation. Finally we have to restrict the voltage at the point of the grid connection.

$$v_A = 230$$

Below we have expressed these nine equations in a matrix format. Every row is a constraint, while every column is used for a variable

$$\begin{bmatrix}
-f_{AB} & -f_{AC} & & & & & & & g_A \\
f_{AB} & & -f_{BD} & & & & & & \\
& f_{AC} & & -f_{CD} & & & & & \\
& & f_{BD} & f_{CD} & & & & & \\
-f_{AB} & & & & v_A & -v_B & & & \\
& -f_{AC} & & & v_A & & -v_C & & \\
& & -f_{BD} & & & v_B & & -v_D & \\
& & & -f_{CD} & & & v_C & -v_D & \\
& & & & v_A & & & &
\end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 10 & 0 \\ 10 & 0 \\ 0 & -s_D \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 230 & 0 \end{bmatrix}$$

We can separate the variables from their coefficients to obtain on either side of the equals sign the product of a numerical matrix with a vector of variables, as can be seen below

$$\begin{bmatrix}
-1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{pmatrix} f_{AB} \\ f_{AC} \\ f_{BD} \\ f_{CD} \\ v_A \\ v_B \\ v_C \\ v_D \\ g_A \end{pmatrix} =
\begin{bmatrix} 0 & 0 \\ 10 & 0 \\ 10 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 230 & 0 \end{bmatrix}
\begin{pmatrix} 1 \\ s_D \end{pmatrix}$$

Since the left matrix is square and the equations are independent, we can apply Gauss-Jordan elimination to transform the system of equations. The left matrix will then become a unit matrix as can be seen below

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{pmatrix} f_{AB} \\ f_{AC} \\ f_{BD} \\ f_{CD} \\ v_A \\ v_B \\ v_C \\ v_D \\ g_A \end{pmatrix} =
\begin{bmatrix} 10 & -0.5 \\ 10 & -0.5 \\ 0 & -0.5 \\ 0 & -0.5 \\ 230 & 0 \\ 220 & 0.5 \\ 220 & 0.5 \\ 220 & 1 \\ 20 & -1 \end{bmatrix}
\begin{pmatrix} 1 \\ s_D \end{pmatrix}$$

We can again rewrite this to a set of equations to recover expressions for the variables for the flow of current, voltages and grid production, linearly expressed in terms of a constant and the storage production behaviour.

$$f_{AB} = 10 - 0.5 \times s_D$$
$$f_{AC} = 10 - 0.5 \times s_D$$
$$f_{BD} = 0 - 0.5 \times s_D$$
$$f_{CD} = 0 - 0.5 \times s_D$$

$$v_A = 230$$
$$v_B = 220 + 0.5 \times s_D$$
$$v_C = 220 + 0.5 \times s_D$$
$$v_D = 220 + s_D$$

$$g_A = 20 - s_D$$

Because the matrix on the left was a unit matrix, we have obtained expressions for each of the variables on the left in terms of a constant and the storage production. This means that we can replace the variables in the original model by these expressions. For example, in the original model we limited the flow through a line by stating that

$$-I_{AB}^{max} \leq f_{AB} \leq I_{AB}^{max}$$

but this can now be replaced by

$$-I_{AB}^{max} \leq 10 - 0.5 \times s_D \leq I_{AB}^{max}$$

In the same way, we can eliminate the variables for the voltage. The expression for the grid production can be used in the objective function.

It was mentioned before that the matrix we obtain is square and the equations were independent, which are the requirements for Gauss-Jordan elimination. We can easily show that it will always be a square matrix. Let n be the number of nodes in the graph, m the number of edges in the graph and g the number of grid connections. In the initial matrix form, we have a variable for the voltage at each node, the flow along each edge, and the production of each grid connection. This comes down to a total of $n + m + g$ variables. Next, we have a constraint for the net production at each node, a constraint for the voltage drop along each edge, and a constraint for the voltage at each grid connection. So there are a total of $n + m + g$ constraints. Hence, the matrix will always be square. It is important to note that the number of storage systems does not influence the size of this matrix, since the production of these storage systems is kept at the right side of the equation.

In a similar way, we can eliminate the variables for the state-of-charge, by introducing a variable for the initial charge. This initial charge is the state-of-charge at the beginning of the first time period. For example, we can express the state-of-charge at the end of the first and second time period as

$$s_0^c = s_{initial}^c + s_0^+ - s_0^-$$

$$s_1^c = s_0^c + s_1^+ - s_1^-$$

and then replace s_0^c in the second equation by the right hand side of the first equation, resulting in an expresion for the state-of-charge at the end of the first time period

$$s_1^c = s_{initial}^c + s_0^+ - s_0^- + s_1^+ - s_1^-$$

As can be seen, the expression is based solely on the initial state-of-charge and the charging and discharging during the previous and current time periods. In this way the state-of-charge can be expressed at any time, and the variables that were constrained to the capacity of the storage system could now be replaced by these expressions. In this way, also the variables for the state-of-charge can be removed from the model. This process can also be done in matrix form as has been done with the other variables.

Transforming the full model is achieved in two steps. First, all the flow of current and voltage variables are removed from the system by applying Gauss-Jordan elimination on the set of equations for each timeframe. Since the variables for the different timeframes are independent (their only connection is through the inventory formula for the storage systems), this can be done for each timeframe separately. Second, the storage inventory formulas are transformed to remove the state-of-charge variables from the model.

How many variables do we remove from the system in this way? Suppose we have a distribution network of N nodes, E edges and S storage systems, and we are calculating the loadflow series for a period of T timeframes. In the full model described in the previous chapter, there was a variable for the voltage drop along an edge and the transported current along this edge. Also there was a variable for the voltage at each node and the production of current at each node. For each position where a storage system could be placed, whe had to introduce three variables: one for the charging of the storage system, one for the discharging of the storage system, and one for the state-of-charge. Each of the variables mentioned exists for every existing timeframe. If we add all this together, we would have a total of $(2N + 2E + 3S) \times T$ variables. When the number of variables would be reduced as was done in this section, we would only remain with the variables for the charging of the storage system and the discharging, so a total of $2ST$ variables.

Since most distribution networks are *radial* or *looped*, the number of edges in the network will roughly be the same as the number of nodes, so $E \approx N$. Because we are trying to solve the storage location problem, we have to introduce a storage system at every position in the network, so $S = N$. With the old model we would end up with approximately $7NT$ variables, while after the reduction only $2NT$ variables remain. This is a reduction of more than 70% of the variables in the primal problem, and thus a reduction of 70% of the constraints in the dual problem.

Chapter 4

Implementation and results

In the previous two chapters we explained the loadflow series model and proposed two different models for the storage optimization problem. In this section it will be explained how these models were implemented and how they perform. The first part is about the implementation of the loadflow series, where some choices to increase performance are explained, and about the SLOPER model. The second section is about the performance of the SLOPER model. Here the parameters for the local search routine are optimized to make it converge faster to the optimal solution.

The SLOPER model is written in Java and makes use of IBM ILOG CPLEX Optimization Studio to solve the linear programming problem. The program connects to CPLEX through the Concert library, a modelling layer provided by ILOG. A Java Library for linear programming was developed to connect the model to the Concert library, such that with a minimum effort other solvers could be used to solve the loadflow series model.

The loadflow series model was implemented with a number of options for model construction. In general, this model could be constructed in three ways:

- **Fixed storage configuration:** The model is constructed as described in chapter 2, with the variable reduction described in section 3.3 included. The variables however are still introduced to the model so the results of the loadflow calculations can be examined by the user. The model is constructed in this way to allow the user to view all the numbers in the loadflow calculations, and experience how the storage systems work to minimize the objective.

- **Integer storage model:** The model is constructed with storage systems at each position in the network. The variable reduction as described in section 3.3 is applied. Furthermore, integer variables for each type of storage system at each vertex were introduced as described in section 3.1. Because of these integer variables the storage location model and the loadflow series model are one single model.

- **Local search storage model:** The model is constructed with storage systems at each position in the network. The variable reduction as described in section 3.3 is applied. The constraints that described the inven-

tory formulas for the storage systems at each position are stored, so that the storage efficiency coefficients could be altered during the local search.

The variable reduction as described in section 3.3 was only implemented such that the number of variables for flow of current and voltage were reduced. The state-of-charge variables were not removed from the model, since this did not give any advantage in terms of speed, and made the changing of storage efficiency coefficients much more difficult.

To validate the results from the loadflow series model, a comparison of the loadflow calculation was made with Plexos, a tool developed by Plexos Systems to find optimal production plans for multiple power plants. This program also makes use of a linear model to analyze the flow of power through the network. Apart from some explainable differences, the models found the same flow of power through the networks.

The SLOPER model was implemented to use the loadflow series model without making any assumptions about it. Communication between the two models is kept to a minimum: the SLOPER model proposes a storage configuration, and the loadflow series model returns the performance value of this configuration. The loadflow series model is constructed as the local search storage model described above. Whenever the SLOPER model proposes a change in the storage configuration, it will adapt the bounds of the variables for charging, discharging and state-of-charge of the given position to the values corresponding to the new storage system. Also, the inventory constraints for the state-of-charge are changed to match the new storage efficiency coefficients. The loadflow series model is solved and the new score is returned to the SLOPER model. If the change is rejected, the loadflow series model is ordered to revert the changes made, but will not be solved again. This is left for the next step to save time.

4.1 Parameter calculation

To get the local search algorithm to converge to the optimum as fast as possible, the parameters of the local search method were optimized. This was done on two small but very different networks.

The first network (which wel will call the simple network) has a mainly radial topology with some cycles in the largest distribution lines. The problems all occur in the radial parts of the network, making it relatively easy for an expert to spot good positions for the storage systems. To solve the network problems, two different storage systems are available. One is relatively small in terms of maximum power and storage, the other one is significantly larger but also more expensive.

The second network (which we will call the cyclic network) is an example of a looped distribution network. It was specifically designed to have more than one local optimum in the search space, and is also not as straightforward to analyze. For solving the problems in this network the same two types of storage systems are available, although the prices are a little bit different.

In this section we will explore the parameters of the local search algorithm and see what their effect is on finding the optimal solution for the two given

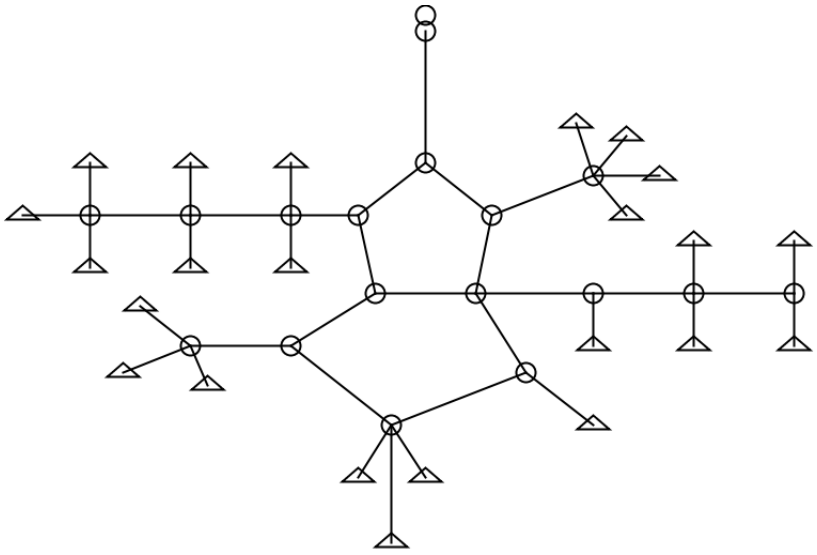


Figure 4.1: The simple network: the grid connection is depicted by the two overlapping circles, the loads are depicted by triangles.

networks. This means we have to know the optimal solutions to both of the networks. To this end the storage systems in the networks are 100% energy efficient, so no energy is lost during conversion or by self discharge. Because of this, the optimal solution can be calculated using the integer linear programming model.

Starting temperature estimation

As was mentioned in the previous chapter, the probability of accepting a new solution is given by the following formula

$$p(accept) = \begin{cases} 1 & \textbf{if } score_{new} \leq score_{prev} \\ e^{-\frac{(score_{new} - score_{prev})}{T}} & \textbf{otherwise} \end{cases}$$

In this equation, T is often referred to as the temperature. The higher this temperature is, the larger the probability that a deterioration is accepted. During the process of searching this temperature will steadily drop so that the local search algorithm will move to better performing cases. We chose to work with an exponential temperature decay a, meaning that $T_{new} = a \times T_{old}$.

The starting value of T has a large influence on the search process: when chosen too low, it will not examine enough states and is likely to end up in a local optimum, but when chosen too high, the algorithm might take longer to converge than is actually required.

To find a good starting value for T, first of all the optimal solution had to be found for both of the networks. This was done by the integer storage model. Next, we examined the performance of the local search algorithm when using

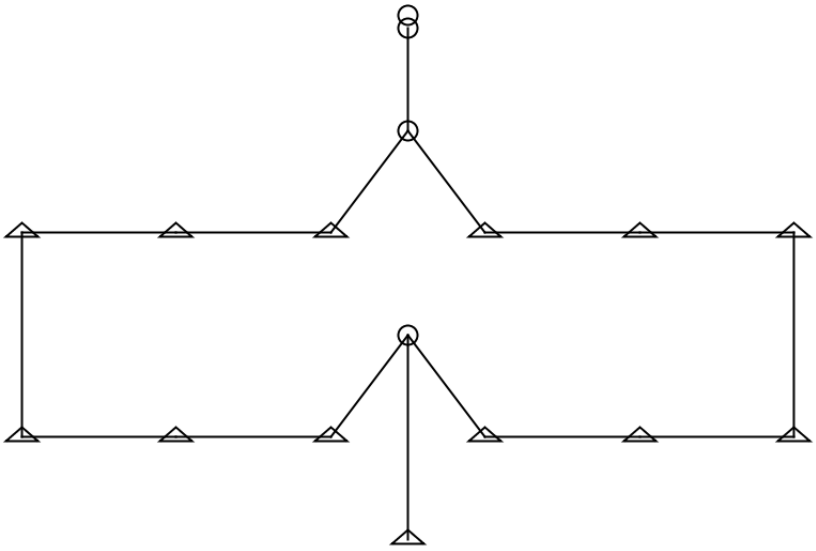


Figure 4.2: The cyclic network: the grid connection is depicted by the two overlapping circles, the loads are depicted by triangles.

hill climbing (only accept improvements, $T = 0$), to get an idea of the number of iterations required to end up in a local optimum. Based on these results, a large scale of values for T were tested.

Each test result is based on 500 runs of the SLOPER model, using the given value of T. After each new state accepted by the algorithm, T was decreased by one percent ($a = 0.99$). The number of iterations was determined by using a hill climbing algorithm (which only accepts improvements) to solve the problem. After 200 iterations, the algorithm would usually be stuck in a local optimum, so therefore the number of iterations was chosen to be 200. Whenever the optimal solution was found, the calculation was terminated as well to save time. For each set of runs, the average score of the solutions found by the SLOPER model were calculated. Also, the number of times the optimal solution was found was counted.

Simple network

optimum = 44672

T =	**Average score**	**# Optimum found**
0	51309	469
100	56283	464
1000	50749	471
10000	48540	479
20000	47722	482
30000	46431	485
40000	46329	481
42500	46377	478
45000	46389	467
47500	46403	472
50000	46648	460
100000	51702	361

Cyclic network

optimum = 37699

T =	**Average score**	**# Optimum found**
0	41920	246
100	41642	247
1000	39647	271
10000	38314	403
12500	38275	390
15000	38150	402
17500	38106	410
20000	38143	393
22500	38108	397
25000	38074	408
27500	38016	400
30000	37964	396
32500	37983	396
35000	38043	388
37500	38004	379
40000	38155	375
42500	38151	366
45000	38259	376
47500	38305	352
50000	38278	362
100000	38840	321

Move probabilities

Another important parameter in the local search algorithm is the neighbourhood space. The probability of moving in a certain direction in this space is determined by the probabilities of the moves that can be chosen to alter the solution. In the implementation of the SLOPER model, there are 5 different possible moves:

- Add a random storage system to a random location.
- Remove the storage system from a random location that has a storage

system.

- Change the type of storage system at a random location that has a storage system.
- Swap the storage system at one location with another random location that may or may not contain a storage system (this action can therefore be either a moving or a swapping action).
- Swap the storage system at one location with another random **adjacent** location that may or may not contain a storage system.

First of all, for the last three moves it was tested what would happen if the probabilities of selecting this move were zero, to determine the importance of each of these moves for the convergence of the local search algorithm. Excluded from this are the *add* and *remove* moves, which are the only moves that can influence the number of storage systems in the network. Next, experiments with different sets of probabilites were executed to determine the best combination of probabilities for both of the networks. Each set of probabilities was tested 500 times with $T = 40000$ and $a = 0.99$. The maximum number of iterations was set to 500, and whenever the optimal solution was found, the calculation was terminated to save time. For each set of runs, the average number of steps required to converge to the global optimum was calculated (this number of steps was 500 when the optimum was not found).

p_{add}	p_{remove}	p_{change}	p_{swap}	$p_{swap-adj}$	#moves Simple	#moves Cyclic
0.2	0.2	0.2	0.2	0.2	94	133
0.25	0.25	0.25	0	0.25	103	198
0.25	0.25	0	0.25	0.25	125	161
0.25	0.25	0.25	0.25	0	379	173
0.15	0.15	0.1	0.2	0.4	113	153
0.15	0.15	0.1	0.4	0.2	103	130
0.2	0.2	0.05	0.3	0.25	97	131
0.2	0.2	0.05	0.35	0.2	96	126
0.2	0.2	0.05	0.4	0.15	106	131
0.2	0.2	0.1	0.2	0.3	102	140
0.2	0.2	0.1	0.3	0.2	95	132
0.2	0.2	0.1	0.35	0.15	96	129
0.25	0.25	0.05	0.1	0.35	102	174
0.25	0.25	0.05	0.35	0.1	108	129

Network tests

The number of iterations required to find the optimum is not clear, and probably depends on specific network properties. Therefore it was chosen to stop the search if the best solution had not been improved for a number of iterations. How many iterations without improvement are required is to be tested in this section.

Based on the initial temperatures it is difficult to find a good estimate for a starting temperature. A larger starting temperature is usually safer, but the calculation will take longer to converge. For the following calculations, the average price of a storage system was used as a starting temperature. For the

probabilites of the different moves it was chosen that $p_{add} = p_{remove} = 0.2$, $p_{change} = 0.05$, $p_{swap} = 0.35$ and $p_{swap-adj} = 0.2$, since this worked well on the cyclic network and hardly had any effect for the simple network.

To get an idea of the speed at which the different problems could be solved, a third and larger network was also tested with these settings. This network is twice as large as the simple network, and was created by replicating the simple network twice and adding it together: for this reason we will call it the double network. In the double network, apart from the two storage systems available in the simple network, two additional storage types were added. Both systems are relatively small and cheap, to ensure that a large amount of relatively good solutions is available. The optimal solution of this network could not be calculated, because the ILP model runs out of memory.

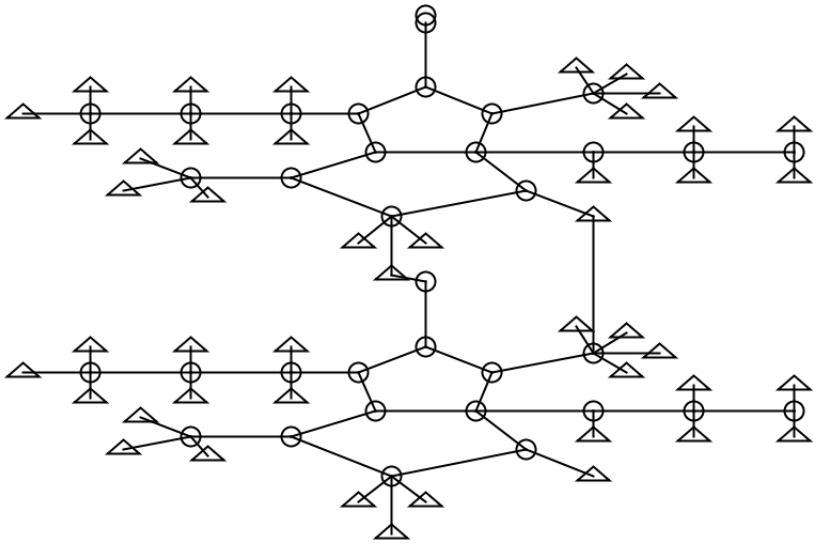


Figure 4.3: The double network: the grid connection is depicted by the two overlapping circles, the loads are depicted by triangles.

To test the effect of the stopping criterium on the found solution, the SLOPER model was stopped after 100, 200, and 500 iterations without change to the optimum. This was done for each of the networks and was repeated 100 times. The results are shown in the table below.

		100	**200**	**500**
Simple network $T = 85000$ $a = 0.99$	Avg. runtime (s)	14.06	23.85	47.74
	Avg. # iterations	194.6	320.0	647.8
	Avg. score	52403	46415	44672
	How often optimum found	76%	91%	100%
Cyclic network $T = 27500$ $a = 0.99$	Avg. runtime (s)	7.98	14.05	29.55
	Avg. # iterations	164.9	288.7	616.2
	Avg. score	38309	37874	37700
	How often optimum found	75%	91%	99%
Double network $T = 47375$ $a = 0.99$	Avg. runtime (s)	236.93	392.68	967.29
	Avg. # iterations	305.1	477.1	1219.4
	Avg. score	116787	88763	82409
	Standard deviation	55646	4731	483

Chapter 5

Conclusion

In this report a model was proposed to determine where to place storage systems in a given electricity grid, and what type of storage systems these should be, such that the most possible value was added to the network. It was assumed that all demands of loads and decentral generation were deterministic, but varied over time. Adding value to the network could be achieved by trading of energy, minimizing losses, or decreasing overload or voltage limit problems.

The loadflow series model was developed to calculate this added value for a given network and a given placement of storage systems called a storage configuration. The model was formulated as a linear programming problem, which could be solved to optimality efficiently by making use of the simplex method, a well-known linear programming algorithm. The loadflows calculated by the loadflow series model were validated with Plexos, a program created to find optimal production planning for multiple power plants. The developed loadflow series model proves to be an interesting and efficient approach to calculating optimal storage behaviour.

The SLOPER model was developed to make use of the loadflow series model to compare different storage configurations and find the best solution to the storage location problem. It makes use of simulated annealing to move through the field of solutions and most of the time managed to find the optimal solution for the tested networks.

Although tests on real life networks still have to be executed, the approach of using a linear loadflow series model combined with a local search heurstic looks promising for this problem.

Chapter 6

Discussion

In this paper, the approach to solving the storage location problem consists of two models: the loadflow series model and the SLOPER model. These two models both have an influence on the solution returned by the proposed method. In this section it will be discussed what the influence of the models is on the solution to the storage location problem, and how the models could be adapted to find better solutions. Apart from this, it is discussed how the model could be accelerated even more, and also what possible extensions could be made to better analyze the advantages of storage systems.

Loadflow series model

The loadflow series model determines the quality of a given storage configuration, and in this way can determine the optimal solution of a given problem. The linear loadflow calculation that forms the basis of the loadflow series model is based on direct current physics (without considering phase angles), which is quite different from the usual loadflow calculations methods based on three-phase alternating current physics. The loadflow series model was developed in this way to increase the efficiency of the calculation, such that more solutions can be examined by the SLOPER model during a period of time.

The objective function of the loadflow series model defines the quality of the proposed storage configuration, and in this way determines which solutions are better than others and what results are returned to the user by the SLOPER model. Therefore it is important that the loadflow series model calculates the added value of storage systems accurately. In the proposed model, the value of a storage configuration is based on the investment cost, the cost of buying and profit of selling energy at the grid connections, and the penalties of overloading lines and breaking voltage limits. However, there are more effects that influence the quality of a storage configuration that are currently not considered.

Currently, the solver can choose to pay a price to increase the maximum allowed current through a line or to increase the allowed voltage range. The idea here is that this type of expansion of the limits holds the most resemblance to the traditional manner of solving network problems: expanding the network by adding extra lines. If the penalties on increasing the limits are high, the objective function will be best if hardly any of the penalties are broken. The

biggest problem with this approach is that the model can choose to increase the limits by only a small amount, paying a relatively small amount for a normally expensive operation.

A way around this problem is to implement a more complex aging model for the lines to get a better measure for the effect of overloading these lines. The amount of aging could be determined by the temperature of the line, which again depends on the flow of current and the cooling of the line. The penalty paid then does not resemble the cost of investing in an extra line, but the depreciation of the overloaded line.

Alternatively, instead of choosing to extend the limits of voltages and currents, the model could allow turning off fixed demand and production. If there would be no fixed demand and production at all, the voltage limits would always be respected and the lines would never be overloaded. For a penalty, the fixed production could be reduced from 100% up to 0%, always enabling a feasible loadflow. It is probably even be easier to put a price on not delivering energy instead of breaking voltage limits. The only disadvantage is that this approach only works for constant current loads, with constant resistance loads the equation is no longer linear.

Another important influence on the performance of storage systems lies in the charging and discharging behaviour of a storage system. This operational behaviour has an important effect on the aging of the components, which will change the parameters of the storage system. Often the capacity of storage systems will decrease with the aging of the storage system, which influences its capacity to add value in the future. Also, a storage system that has aged more will have less remaining value and should be replaced earlier, so effectively it is a bigger investment than a system that has aged less. These effects are not considered in the proposed model.

Another effect not considered in the loadflow series model are net losses. When current is transported along a line, due to its resistance part of the energy is converted into heat. Especially in distribution networks the net losses form a considerable part of the total loss of energy. Although the transportation of current in the model leads to a drop of voltage, the constant current loads and the charging and discharging of storage systems are not influenced by the voltage at a given position in order to keep the model linear. If net losses were introduced in the model, this could stimulate the storage systems to use their excess capacity for peak shaving, whereas now the excess capacity is fully used for trading.

Especially when working on larger, real life networks with more time periods, both calculating the quality of a single storage configuration will take longer and the number of possible storage configurations increases, causing an increase in required computational effort. Apart from the possible expansions to the loadflow series model to get a more accurate quality measure for storage configurations, there is also a possible method to increase the speed of the model.

The simplest but possibly most effective measure is the clustering of timeframes. When attempting a calculation with a time period of a whole year, subdivided into timeframes of one hour, the number of timeframes is 8760.

However, the largest part of these timeframes will have such fixed demand or decentral generation that the use of storage systems is required to keep the currents and voltage within limits. Only a small percentage of these timeframes is critical and will influence the decision about the placement of storage systems. To determine which timeframes are critical and which are not, we can look at the data produced in the variable reduction calculation in Section 3.3. Here we expressed flow of current through lines and voltages in terms of a constant and the storage behaviour. If the flow of current and the voltages during a timeframe are all within bounds without any storage behaviour, we can assume that this timeframe is not critical. Subsequent non-critical timeframes could then be merged together, possibly up to a maximum timeframe length, to reduce the number of timeframes. The fixed production and energy prices during this timeframe will be the average of these numbers during the distinct timeframes. This technique could decrease the number of timeframes drastically, and would hardly harm the accuracy of the calculation. The only small downside is that some of the trading potential is lost because the energy prices are constant during this period.

SLOPER model

The SLOPER model determines which of the solutions in the search space are visited. Currently, it moves randomly through the search space, with the only restriction being that only one storage system can be placed at a position. Although storage systems have not been applied much in Europe, they might be in the future. Therefore a useful expansion would be the possibility to have storage systems already placed within the network. These systems should not be moved by the SLOPER model, but do have an influence on the loadflow series model. Alternatively, the storage system is not fixed but has already been invested in, and can be placed on other positions in the network for a small cost of moving the system.

Another interesting extension would be to allow the SLOPER model to explore some of the traditional ways of solving the network problems. One of the simplest to implement would be to change the voltage at the grid connections. This resembles one traditional way of solving voltage problems in networks with large voltage drops by changing the taps on the transformer. Since this operation requires the transformer to be taken off the grid, it is not an operation that can be done whenever required. The voltage at the grid connections therefore should still remain constant, but could be chosen at another level than the nominal voltage which is default. As mentioned before, the option to choose different voltage levels at the transformer is already included in the model.

Modelling the tap changer on the transformers is easy, but might save the need to add storage systems to increase the voltage. Adding lines to the network is another traditional way of solving network problems, but is a little harder to model. It is easiest to assume that lines can only be added between two nodes that are already connected, because in this way the network topology is not changed. With the addition of a line, the new rated current and the new resistance of the corresponding edge can be calculated. Changing the rated current can be achieved by changing the bounds of the constraint, but changing

the resistance will be more complicated since it is included deeply into the model by the reduction of variables. Especially for edges that are within a loop it becomes more difficult, since there it also affects the flow of currents and not only the voltages. A simplification could be to assume the resistance does not change by adding lines.

The SLOPER model should model that adding a single line might be very expensive (since it might require digging in a busy street), but adding a second line as well is relatively cheap. The cost of expansion might even be different per edge. When adding lines to the network is feasible, it can be included in the SLOPER model as a possible action to move to a neighbouring solution. In this way, the SLOPER model becomes a much more powerful tool in finding the best way to solve operational network problems. Not only will it find solutions solely using either storage systems or traditional approaches, but also solutions where both storage systems, tap changes and network expansions are utilized to solve problems. Especially this combining of solution methods to solve operational network problems would make the model extremely valuable.

Bibliography

[1] Bazaraa, M.S., Jarvis, J.J., Shirali, H.D., *Linear Programming and Network Flows* Wiley, 1990.

[2] Mani Chandy, K., Low, S.H., Topcu, U., Xu, H., *A Simple Optimal Power Flow Model with Energy Storage* Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena.

[3] Chen, S.X., Gooi, H.B., *Sizing of Energy Storage System for Microgrids* PMAPS, 2010.

[4] Gayme, D., Topcu, U., *Optimal power flow with distributed energy storage dynamics* Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena.

[5] Idema, R., Lahaye, D.J.P., Vuik, C., *Load flow literature survey* Delft University of Technology, Delft, 2009.

[6] Kl ockl, B., *Multivariate Time Series Models Applied to the Assessment of Energy Storage in Power Systems* PMAPS, 2008.

[7] Sakowicz, B., Anders, G.J., Kamiński, M., Napieralski, A., *Application of Evolutionary and Hybrid Algorithms to Optimize Investments Strategies in Large Power Plants* PMAPS, 2008.

[8] Young, H.D., Freedman, R.A., *University Physics, 11th Edition* Addison-Wesley, 2004.