Approximating optimal solutions for power outages on the electrical grid by applying heuristic methods

On reducing the running time of a non-polynomial time model on the medium voltage grid

Thesis MSc Mathematics MFoCS

Author:
Jelte Zwetsloot BSc

Supervisors:
dr. Wieb Bosma (RU)
Michiel van der Meulen (Alliander N.V.)
Frans Campfens (Alliander N.V.)

August 28, 2019

Foreword

In September 2016 I started the Master Mathematics at Radboud University Nijmegen in the specialization called Mathematical Foundations of Computer Science (MFoCS). Intuitionistic Mathematics, Computer Algebra, several Cryptology courses and several Optimization courses are courses that I really enjoyed. Since the course Discrete Optimization excited me in particular, I wanted to do my thesis project on a subject that had some similarities to Discrete Optimization.

When I started the Master Mathematics, I knew I wanted to do an external research internship for my thesis. It seemed interesting and necessary to me to see and experience what a mathematician can accomplish in a more practical environment compared to the university. It was Alliander that offered me an interesting and challenging project in the area of optimization.

Doing the internship has been a very valuable experience and I am glad I took the opportunity. To name a few of the things I did: I have studied heuristic methods, learned the basics programming and modeling in R, have had unexpected setbacks during the project which I have overcome and most importantly, I have experienced what it is like to be part of a team of enthusiastic and hard working people. Working together with both my supervisors as well as with other Alliander employees has given me a lot of positive energy.

This final product, the thesis you are reading right now, is the product of hard work, enthusiasm, and a lot of support from various people. It is a cliche, but I could not have finished this thesis without the help of many others. In particular, I would like to thank the following people. First of all, Michiel van der Meulen from Alliander, for being an awesome day-to-day supervisor because, among other things, he was always there when I had something to ask, when I wanted to share some cool result or simply when I wanted to have lunch and chill for a bit. It has been a joy working together with Michiel.

Also a lot of thanks to dr. Wieb Bosma, not only for being my supervisor from the university for this thesis project, but also for helping me through the study Mathematics as a whole. Wieb has been my supervisor for all projects during my mathematics studies, except for the course "Modellenpracticum" in the Bachelor Mathematics, and even then Wieb was the second reader of my group's project.

Next, thanks goes out to my third supervisor, Frans Campfens from Alliander. The two-weekly meetings with Frans made me see the project on a different

level than purely mathematical.

In February I had a meeting with dr. J. Hurink from the University of Twente to talk about heuristic methods and the specific problem Alliander assigned me with. This meeting was very valuable in understanding how I should design and apply heuristic methods to a problem. Thanks to dr. J. Hurink for enthusiastically helping me, even though I had never been a student of his.

I want to thank my friends and fellow students Hannelore Heuer, Teun van Nuland and Luuk Verhoeven. Thanks to Hannelore for being an awesome thesis writing buddy, thanks to Teun for helping me with finding the complexity proofs and for listening with interest to whatever I had to say about the project, and thanks to Luuk for also always listening with interest and being there whenever I had something to ask about R or anything else mathematics related.

Lastly, I would like to thanks my two sisters Hannah and Marie and parents Henk and Paulina for all the mental support. You guys rock.

Contents

Chapter 1

Introduction

In this chapter Alliander and one of its subsidiary companies, *Liander*, are introduced. Secondly, we will take the first look at the *Optimal Mix Problem* and the *Optimal Mix Model* that Alliander works on. The Optimal Mix Problem and Model will be the central subjects of this report. Afterwards, the research questions and report overview are given.

1.1 Alliander N.V. and its parties

Alliander is a Dutch overarching company consisting of several independent divisions which operates in the energy and gas sector in the Netherlands. The best known and largest division that is a part of Alliander is Liander, the Distribution System Operator (DSO). It is responsible for keeping the distribution networks of electricity and gas in a well-working state for a substantial part of the Netherlands. One of the main goals for Liander is to ensure that clients, companies and institutions have access to electricity and gas at all times. Various projects inside Alliander help them to accomplish this goal. One of these projects, called the Optimal Mix Model, is considered in this thesis.

1.2 The Optimal Mix Model

There are three kinds of electricity grids in the Netherlands: high voltage (HV) (110 kilo volt (kV) to 380kV), medium voltage (MV) (10 to 20 kV) and low voltage (LV) (up to 400V). Liander does the management and maintenance of the medium and low voltage grid for a large region of the Netherlands. Malfunctions occur on the grid from time to time, such that clients experience a power outage. Liander aims to minimize this downtime, such that as many clients as possible experience as little downtime as possible, ensuring customer satisfaction. Also, lawmaking regulates the price a distribution system operator is allowed to ask based on the number and total duration of power outages. In other words: if Liander minimizes the downtime, they work on customer satisfaction and on

maximizing their revenue. Considering that Alliander in general needs to make cuts in their budget, it is even more critical for Liander to address the malfunctions.

80% of the electricity downtime for clients is caused by malfunctions on medium voltage networks. Therefore specific projects have been designed to only take the medium voltage grid into account.

The Optimal Mix Problem is one of the problems that can be analyzed and solved in order to make a large impact on the minimizing of the downtime on the medium voltage grid. The Optimal Mix Problem is an optimization problem that looks at the best way, downtime- and money-wise, of implementing (new) techniques on the medium voltage grid. Alliander strived to solve the Optimal Mix Problem and did so by developing the Optimal Mix Model, a model that can be used to simulate malfunctions on (theoretical instances of) the medium voltage grid and to determine the optimal mix of techniques to be implemented on the grid.

The Optimal Mix Model has been under development since July 2017 and has become an advanced model by now. The model is able to determine the optimal mix for a specific medium voltage network while taking all aspects of this broad problem into account. Even though the Optimal Mix Model is able to do all this, it is not perfect yet. One of the aspects that can be improved on, is its running time. A calculation for one part of the grid can easily take longer than 4 days. That is undesirably long, considering that Alliander needs to run the Optimal Mix Model around 350 times to have their whole grid covered. It would be valuable for Alliander if this calculation would take substantially less time, such that it is easier to respond to changes on the grid in the future without having to wait multiple days. However, the model should stay reliable: when the calculation time is shortened, qualitative solutions should still be found. The goal of this report is to find the best ways to minimize the running time of the model while still finding 'good' solutions by using heuristic optimization techniques.

1.3 Research Questions and Report Overview

The general research question is stated as follows:

Can the Optimal Mix Model be improved by implementing heuristic optimization techniques in order to substantially lower the running time while closely approximating qualitative solutions?

This question will be answered by treating the following research questions:

1. How can the Optimal Mix Problem be mathematically described?
2. How does the Optimal Mix Model solve the Optimal Mix Problem?
3. What is the mathematical complexity of the Optimal Mix Problem?

4. How can heuristic optimization techniques be used to improve the running time of the Optimal Mix Problem while closely approximating qualitative solutions?

5. What heuristic optimization techniques can be used to improve the running time of the Optimal Mix Problem while closely approximating qualitative solutions?

6. How well do heuristic optimization techniques improve the running time of the Optimal Mix Problem while closely approximating qualitative solutions?

This report covers these subjects in the following way. Firstly, Chapter 2 fully describes the Optimal Mix Problem in all its relevant details, by giving an exposition of all components and aspects of a network, by giving the definition of a mix and the by showing the financial aspects of the Optimal Mix Problem.

Question 1 is answered in Chapter 3 by giving the mathematical formalization of the Optimal Mix Problem. An mathematical interpretation of the medium voltage network is given, followed by the introduction of a binary notation for mixes and a formal definition of the financial functions. The chapter concludes with stating the Optimal Mix Problem as an optimization problem.

In Chapter 4 lies the answer to Question 2, where the Optimal Mix Model will be explained. An overview of the model is given, where four different phases are distinguished. The second part of this chapter consists of describing the objective function that is used in the Optimal Mix Model, that uses simulations to determine the duration of power outages.

In the following chapter, Chapter 5, Question 3 will be considered by giving a proof of the complexity of the Optimal Mix Problem. In particular, the simulation model from the previous chapter is elucidated in more detail, where an optimization problem in itself is described. This optimization problem is proved to be NP-hard. Since that optimization problem is part of the objective function of the Optimal Mix Problem, this makes Optimal Mix Problem (at least) NP-hard too.

The next chapter covers the 4^{th} question. Chapter 6 is dedicated to describing how heuristic optimization techniques are aimed to be used to adjust the current Optimal Mix Model, where the critical concept is to let an algorithm re-use information generated in earlier steps by that algorithm. The chapter also gives an analysis of the Optimal Mix Problem by looking at the problem in itself and the results of the Optimal Mix Model on two test subjects.

Chapter 7 treats question 5 by proposing multiple heuristic algorithms, with the details from Chapter 6 as theoretical basis, and by describing what versions of the algorithms are implemented in the end. Six algorithms are proposed initially, of which four are implemented and tested. Another algorithm is proposed during the implementation process, to boost the results by the other algorithms.

The final question, Question 6, is answered in Chapter 8, where the results on three test subjects are presented, and Chapter 9, where all the results are discussed. The quality-wise best algorithms approximate the Optimal Mix by

more than 95% on all three of the test subjects, with a speed-up factor of at least 8. The best algorithm with respect to running time reaches a speed-up factor of at least 40 on all test subjects, with the approximation of the optimum for two of the test subjects on at least 95% and on 89% for the third test subject.

In the concluding Chapter 10, the general research question will be answered and an short summary of the thesis is given once more.

The report will be finished by considering future work based on this research in the final Chapter 11.

Chapter 2

The Optimal Mix Problem

This chapter is dedicated to explaining the Optimal Mix Problem. First, the structure of the medium grid is considered. Secondly, power outages caused by malfunctions on cables in the grid are explained. Next, we will take a look at various techniques that are used to reduce downtime. The financial aspects are viewed afterwards. This chapter finishes by formally stating the Optimal Mix Problem, where all of the above comes together.

Before all that, note that there are many different aspects and details to this very broad problem. Since this paper is concerned with the mathematical side of things, some details will be omitted. Only those details that are required to fully describe the problem mathematically will be mentioned.

2.1 DATA

All data and data structures that are mentioned in this chapter lie in some database that Alliander supervises. For example, the term fail frequency is defined in this chapter, and the fail frequency for a specific cable is determined by an Alliander team called Condition Models. The condition modeling team stores their results in a database, such that other teams can find and implement them. This happens for all kinds of information that need some kind of calculating. Mentioning all different teams corresponding to the different definitions and terms can unnecessarily impede the process of easily explaining the Optimal Mix Problem. Therefore it is assumed in this paper that all data that lie in these databases is legitimate and is collected in one big Alliander database, called *DATA*.

2.2 Structure of the medium voltage grid

The medium voltage grid is the collection of all smaller medium voltage networks. Spread over the Netherlands, Liander controls about 350 medium voltage networks. Each network contains at least a *substation*, (multiple) *secondary substations* and cables connecting the (secondary) substations. A network is connected to the high voltage grid at the substation. Each substation contains a device that transforms high voltage into medium voltage, a HV-MV transformer. When exclusively reviewing a medium voltage network, the substation can be seen as the power source of the network. From this substation the electricity is distributed over multiple *routes*: collections of paths over cables that originate from the substation, where all paths have the same cable starting the path from the substation. These routes bring the electricity to the secondary substations. Here the electricity once again is transformed, this time by a MV-LV transformer, and now is distributed over routes on the low voltage grid where the electricity is transported to the clients.

Each medium voltage network is *radial*, which means that no cycles occur. When considering a medium voltage network as an undirected graph, where the vertices are the (secondary) substations and where the edges are the cables connecting the (secondary) substations, then radiality means that this graph has a tree structure. Do note that a route can have *branches*, since this does not imply the presence of cycles. Radiality implies that every client has a unique path of supply. This makes it easier to locate a malfunction whenever a power outage occurs. Furthermore, it becomes safer for mechanics in repairing the network and solving a power outage when a network is radial, since the direction of the flow of the electricity is unambiguously determinable. The importance of radiality is demonstrated in an example in the next section, Section 2.3.

Even though no cycles occur, two secondary substations in different branches of a route, or even in two separate routes, can be connected to each other by a cable with a so called *normally open point*. A normally open point is a switch in the cable that is in open position, creating a gap. This gap makes it that no electricity flows through cables with normally open points. Electricity only starts flowing through this point when the switch is flipped into closed position. A normally open point can be very convenient. For example, suppose that a cable fails and the secondary substation connected to it no longer receives electricity. Suppose that this secondary substation is also connected to a cable with a normally open point. Then we could disconnect the failing cable from the network and flip the switch of the normally open point into closed position to redirect the power flow, such that the secondary substation is now provided with electricity again. After the failing cable is repaired, the normally open point, that is now closed, can be easily switched open again to regain the initial configuration of the network. With such a normally open point in a cable, radiality is again preserved, as no electricity will flow through this specific cable as long as the normally open point is open.

A specific medium voltage network can be displayed schematically by representing its substation, its secondary substations and its cables, often called

sections, in a graph. For an example with three routes and two normally open points, see Figure 2.1.

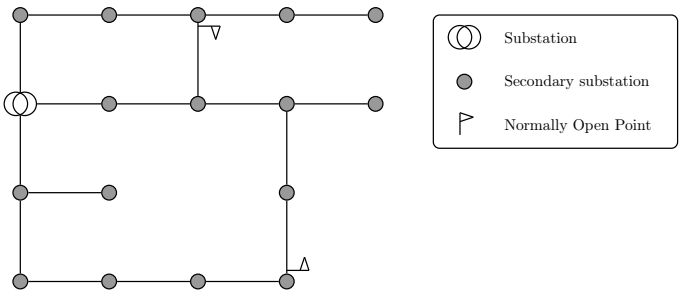


Figure 2.1: Schematic example of a network with three routes

2.3 Malfunctions on the medium voltage grid

Malfunctions can occur in various fashions. For this report and for the Optimal Mix Problem, the focus will be on malfunctions in cables, since these malfunctions are the most common. Whenever a malfunction in a cable occurs, a short circuit flow will emerge between the substation and the malfunction. The short circuit triggers a safety switch. The safety switch creates a gap in the cables, called a *temporarily open point*. This disconnects a number of secondary substations from the substation: a power outage starts for all the clients connected to one of those secondary substations. The goal is to provide all secondary substations with electricity again as soon as possible. This is done by finding the 'bad' cable, isolating it and redirecting the electricity.

The isolating is done by mechanics who turn the switches of the bad cable into open position at its connections with the secondary substations. Now, no electricity is transported through the isolated cable. Electricity is redirected by closing certain open points, such that secondary substations are provided with electricity again. When there are still powerless secondary substations left after previous actions, a power generator will be installed by mechanics, to also provide the clients connected to these powerless stations with electricity again. Now the power outage is solved for all clients.

Figure 2.2 shortly describes these phases during a power outage in an example.

Note that after a power outage Liander always aims to return to the original operating situation, here represented by Figure 2.1, by repairing the damaged components. However, going from the final configuration where the power outage is solved (Figure 2.2e) to the original configuration (Figure 2.1) has no effect on the outage time anymore and hence will not be described here.
An important aspect that is not included in this example, is the time that every

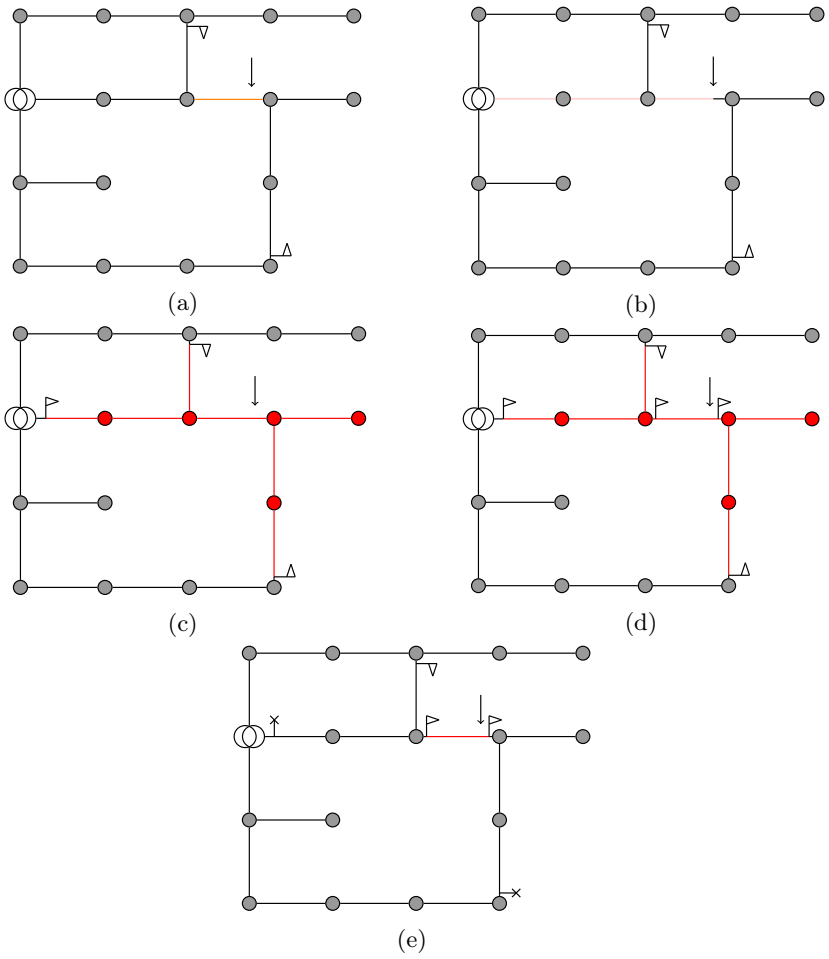


Figure 2.2: (a) A fault occurs in the orange section. (b) A short circuit (in pink) emerges between the fault and the substation. (c) A safety switch near the substation is triggered and flips open resulting in a temporarily open point isolating the fault and connected secondary substations. (d) The fault is isolated by mechanics placing temporarily open points (e) The bottom normally open point and the most left temporarily open point (created by the safety switch) are closed by mechanics: all secondary substations are receiving electricity from the substation again.

step takes. In reality it takes 42 minutes for the mechanics to arrive on average, and then 10 minutes more per secondary substation they visit. Add to this that the mechanics have to locate the malfunction too, and might need to place a power generator, which is time consuming. All of this contributes to the length of the power outage.

2.4 Customer Minutes Lost and other definitions

To understand Section 2.5, some additional definitions are first needed.

2.4.1 Customer Minutes Lost

As stated in the introduction, one of the goals for Liander is to minimize the length of a power outage and the number of power outages. To make this measurable the definition of *Customer Minutes Lost (CML)* is introduced. Customer Minutes Lost is the sum of minutes of power outage per customer. Example: if a power outage occurs and one secondary substation, with 25 customers connected, regains power after 10 minutes and another secondary substation, with 35 customers connected, regains power after 15 minutes, then the total CML for this power outage is $25 \cdot 10 + 35 \cdot 15 = 250 + 525 = 775$ CML.

Determining the Customer Minutes Lost shall be the main issue for the Optimal Mix Problem, as becomes clear in the following chapters.

2.4.2 Additional notions and definitions

- One set of cables connecting two secondary substations is called a *section*.
- The *capacity* of a section in the maximum power that is allowed to flow through a section in standard operation, without damaging the cable.
- The name of a network is often the name of the substation of that network. For that reason, we will refer to either a network when mentioning a substation or the actual substation. Context makes clear which of the two is meant. For example: 'substation $A.5$ has twelve sections' means 'network $A.5$ with substation $A.5$ has twelve sections'.
- *Optional* and *fixed sections* are the two types of sections that occur in medium voltage networks. The optional sections are sections that contain a normally open point and hence are not utilized in normal operation. The fixed sections are all sections that do not contain a normally open point.
- A *full trail* is a longest possible path originating from the substation over fixed sections, while visiting a fixed section and secondary substation at most once. By 'longest' it is meant that the path has reached its largest form: adding another section is not possible. Mathematically, a full trail is a path from the substation to a leaf over the fixed sections sections graph. The number of full trails is therefore equal to the number of leaves of the fixed sections graph. Note that one route can contain multiple full trails, and that different full trails can have sections and secondary substations in common. See Figure 2.3 for an example.

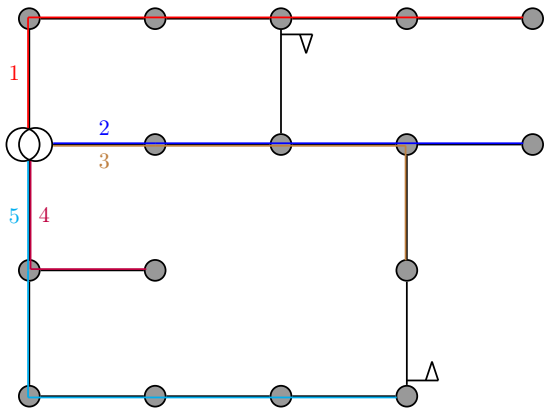


Figure 2.3: Full trails in a network

2.5 Techniques to reduce downtime or solve power outages

The goal is to minimize the duration of power outages for as many clients as possible, that is, to minimize the CML. This is done by isolating the malfunction that caused the power outage and redirecting power flow as quickly as possible. Generally, it is not immediately known in what section the malfunction occurred. As long as this remains unknown, the 'bad' section can not be completely isolated. On top of that, it would take long to solve the power outage by trial and error. In reality, there are many techniques to assist in this process. These techniques can induce certain properties to the locations they are installed, of which the following four properties are critical for the Optimal Mix Problem:

1. A *remotely checkable point* is a point in the network that can be remotely checked for passage of short circuits. This means that it can be determined in the headquarters whether a short circuit passed the remotely checkable point, without a mechanic visiting the point.

2. A *locally checkable point* is a point in the network that can be locally checked for passage of short circuits. This means that it can be determined whether a short circuit passed the point when a mechanic visits the point.

3. A *remotely controllable point* is a (secondary) substation whose sections can be switched open or closed by pressing a button in the headquarters. This means that mechanics do not need to visit a remotely controlled secondary substation whenever one of its section needs to be switched open or closed.

4. An *autonomously secured point* is a connection of a section to a secondary substation in the network that forms protection for the network against

short circuits by recognizing them. When an autonomously secured point recognizes a short circuit, the section automatically switches open to isolate the malfunction that causes the short circuit.

Next follows a list of techniques that can be used to assist in the process of isolating a malfunctioning cable and redirecting electricity:

Fault Passage Indicator

A *Fault Passage Indicator (FPI)* is a device that notes and shows on a local display whether a short circuit passed or not. One is installed in each secondary substation on each cable. So, each section contains two FPIs (one at each secondary substation it is connected to). Mechanics use FPIs to check whether a short circuit has passed to determine at what side of the secondary substation the malfunction occurred. In other words, an FPI induces local checkability.

Communicable Fault Passage Indicator

A *Communicable Fault Passage Indicator (CFPI)* is an FPI, with the extra property of automatically sending a text message to the headquarters when a short circuit has passed. This is more efficient than a FPI, since the mechanic does not need to visit a CFPI to know whether a short circuit has passed. A CFPI can be placed on a cable in a secondary substation and induces remote checkability.

Smart Cable Guard

A *Smart Cable Guard (SCG)* supervises a certain number of sections by predicting and locating malfunctions. An SCG exists of two different parts: the master device and the slave device. These devices are placed in two different secondary substations. All sections between these secondary substations become supervised by the SCG. That means three things. Firstly, all malfunctions that occur on this track are immediately noted and localized. Whenever a malfunction occurs on any of the supervised sections, a message containing the malfunction and section information is immediately sent to the headquarters. In other words, an SCG induces remote checkability to all sections and substations it is installed on. Secondly, an SCG detects whether a short circuit has passed the supervised trajectory, even when the malfunction causing the short circuit does not lie in the trajectory. This way, the precise location is not known by the SCG, but it is known whether the malfunction happened behind or in front of the SCG. Finally, the SCG can predict, with an average accuracy of 71%, that a malfunction is about to occur in a certain section. The SCG will also report this to the headquarters, so that preventive actions can be taken, and the malfunction can be averted.

The SCG has many placement rules, such as that the slave device can not

be placed at a leaf of the radial network, or that there is a maximum length of the trajectory that the SCG can be placed on, or that the devices can not be placed over certain types of cables. (For all specifics, see [1].) For this report, all the placement rules are assumed to be there and we will not go into any of its details, as the details are of no value for this research. How this assumption is done, will become clear in Chapter 3.

Another assumption for SCG is the following. An SCG has a maximum length, and an SCG over a longer trajectory is always more efficient than over a shorter trajectory, since the costs are equal and more sections are supervised on a longer trajectory. For that reason, it is assumed that a certain trajectory will not be considered for the SCG, when a longer trajectory containing the smaller trajectory is possible for the SCG.[1]

Normally Open Point

The *Normally Open Point (NOP)* has already been introduced in Section 2.2. As noted before, an NOP is a gap in a section, that can be closed by flipping a switch, located at the NOP. (When the NOP is remotely controllable, it can also be closed from the headquarters.) This way, radiality is maintained, and there is still the option to easily redirect electricity through the section it is installed on. An NOP is placed in a section at a secondary substation. This means that one section can technically have two NOPs, one at each side of the section. Note that it would be impractical to actually have two NOPs on one section, since two actions are then needed to utilize the section instead of one.

An NOP can not be closed to redirect electricity in all cases. When a NOP is closed, more electricity will flow through certain sections. If the capacity for one of the sections would be exceeded, the NOP will not be closed.

Circuit Breaker

The *Circuit Breaker (CB)* is a safety switch that responds to short circuits. Whenever a short circuit runs through a CB, the CB will pop open, creating a gap in the cable it is installed on, also called a temporarily open point. This way only the secondary substations behind the temporarily open point will be part of the power outage, while all other secondary substations remain in action. A CB thus reduces the size of power outage and reduces the total downtime caused by the malfunction. A circuit breaker pops open almost immediately when a malfunction occurs. A CB can be easily closed again when needed. In the example in Figure 2.2c a CB was assumed to be the mentioned safety switch.

When multiple CBs are installed on a network, we only want the CB in

[1] In reality there are two types of SCG, namely the Smart Cable Guard Partial Discharge (SCG PD) and the Smart Cable Guard Error Position Localization (SCG EPL). The SCG PD is the SCG as described here. The SCG EPL lacks the option of predicting a malfunction, but can be implemented over a much longer distance than the SCG PD. For this report it is chosen to only consider SCG PD, since its effectiveness is bigger (based on results of the Optimal Mix Model), since it is clearer for further definitions, algorithms and such, and since the proposed methods in this report can be easily extended to also include SCG EPL.

the short circuit that is closest to the malfunction to pop open, such that the least amount of secondary substations become powerless. This is achieved by setting a timer on each CB. This is done such that CBs closer to the substation will respond to a short circuit after a longer period, and CBs further away will respond faster. The possibilities are limited however. For example, if the timer is set to a minute, then a short circuit will run for a full minute and cables will be damaged. Due to such technicalities only a maximum of three circuit breakers are allowed per full trail. This is called the circuit breaker rule **(CB rule)**. The CB-rule will be important for the feasibility of placements of components later on. Figure 2.4 shows an example of what CB is triggered when a fault occurs.

When a CB is triggered, it can be reasoned from headquarters which CB was triggered, by analyzing the new voltage and power current values at the substation. Therefore it is said that the triggered CB induces *indirect remote checkability*, since it is now known that the malfunction occurred behind the CB.

Technically, a CB can be placed at either side of a section, right after or right in front of a secondary substation. In reality it is more efficient to place the CB on the side that is closest to the substation, since it eases the search for the malfunction when that CB is triggered. For that reason, it is assumed for this project that new CBs will always be placed on the section side that is closest to the substation.

A CB induces autonomous security on the point it is placed.

Intelligent Secondary Substation

An *intelligent Secondary Substation (iSS)* is a secondary substation that can be remotely controlled: all ingoing and outgoing sections at an intelligent Secondary Substation can be opened and closed by pressing a button in the headquarters. In other words, the connections between sections and an iSSs are remotely controllable points. It sometimes happens that an iSS is placed next to a normally open point. In this case, the normally open point remains in open position, and now it can be remotely closed.

The second iSS-property, is that it is possible to remotely check the FPIs on the connected sections. In other words, iSSs also induce remote checkability.

The combination of being remotely controllable and being remotely checkable, is extremely useful when a malfunction occurs. In 10 minutes the malfunctioning cable can be isolated from a part of the network by opening one of the correct sections in the iSS. The part of the network that is isolated from the malfunctioning cable, can now be be provided with electricity again. If the iSS placement is chosen smartly, this can even been done by remotely closing a normally open point that is connected to the same iSS. All in all, correct iSS placements can save a lot of time.

The effectiveness of an iSS (at an NOP) depends heavily on the capacity of sections. As mentioned at the NOP description, a NOP can not always be closed, because the capacity of sections is also taken into account. An iSS becomes very ineffective when installed at a location electricity is hard to redirect

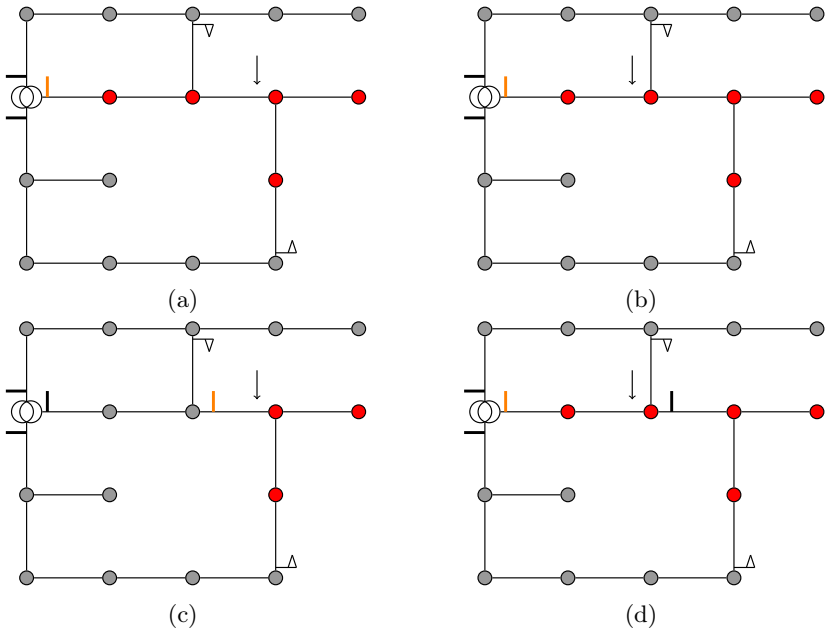


Figure 2.4: Example with one fault in a section and different circuit breaker placements. The arrow indicates where the fault occurs, the thick bars represent the circuit breakers, the orange bar represents the triggered circuit breaker and the red secondary substations represent the secondary substations that become disconnected from the substation.

because of the requirement to not exceed the capacity.

A secondary substation can be transformed into an intelligent substation. However, not all secondary substations can be upgraded into an intelligent secondary substation due to some technicalities. For more info on these technicalities, see [1]. It is taken into account in the actual model which secondary substations can be upgraded and which can not.

2.6 More notions and definitions

- Two types of open points have been mentioned so far: normally open points (NOPs) and temporarily open points (TOPs). An *open point (OP)* is defined to be a gap in a section that can be closed. Consequently, a TOP and a NOP both are an open point. The main difference is that a NOP is open in normal operations and a TOP is only created whenever a malfunction has occurred. Occasionally it does not matter whether we are dealing with a TOP or a NOP, but it does matter that an open point is present. Whenever we refer to an open point, such is the case. When referring to all open points, we refer to the set of all TOPs and NOPs.

- For a specific component type there are *suitable* and *unsuitable* locations. For an iSS for example, not every secondary substation can be upgraded into an intelligent one. Such a secondary substation is said to be an unsuitable location. A secondary substation that can be upgraded into an intelligent one is called a suitable location for the iSS. In general, when a component of a specific type can be newly installed on a location, then that location is called a suitable location. If it can not be installed on a location, the location is called unsuitable. Note that a location that already has an iSS is called an unsuitable iSS location, since a new iSS can not be installed there. For an example per component, see figures 2.5-2.7.

- All substations are either *remotely controllable* or *non-remotely controllable*. A remotely controllable substation can be seen as an intelligent substation: its connections with sections can be remotely opened and closed. It is similar to the intelligent secondary substation in that sense. A non-remotely controllable substation can not open or close its connections with section remotely. It is also assumed in the Optimal Mix Problem that a non-remotely controllable substation can not be upgraded into a remotely controllable substation.

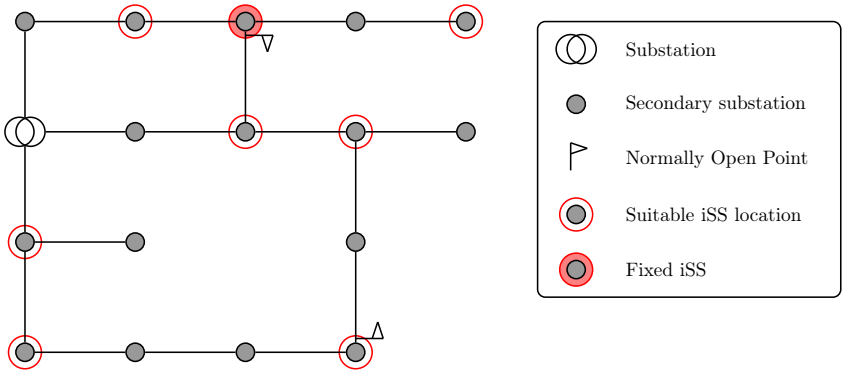


Figure 2.5: Suitable and fixed iSS locations

- A *component mix* (or *component locations mix*) is a combination of specific suitable locations on a specific network for a specific type of component. For CB mixes, the CB-rule has to be met. Otherwise, the CB mix is not feasible.. For an example per component, see figures 2.8a, 2.8b and 2.8c.

- A *mix* (or *locations mix*) is a combination of SCGs, CBs and iSSs on specific suitable locations on a specific network, such that the CB-rule is met. Do note that other techniques, such as CFPIs and NOPs, are not taken into the mix. The focus for the Optimal Mix Problem lies only on SCGs, CBs and iSSs. For an example, see Figure 2.8d.

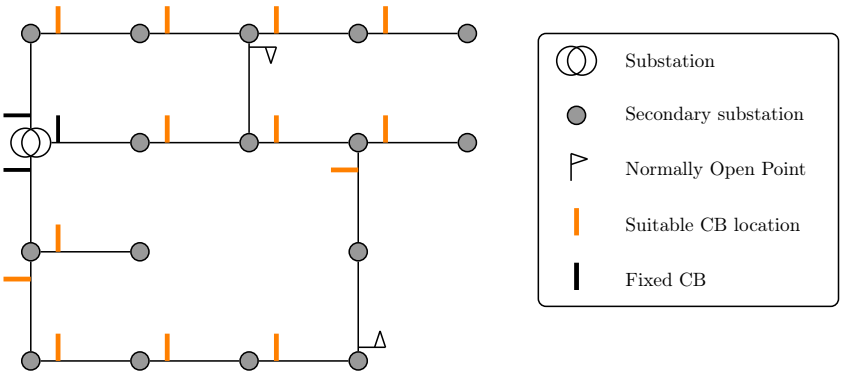


Figure 2.6: Suitable and fixed CB locations

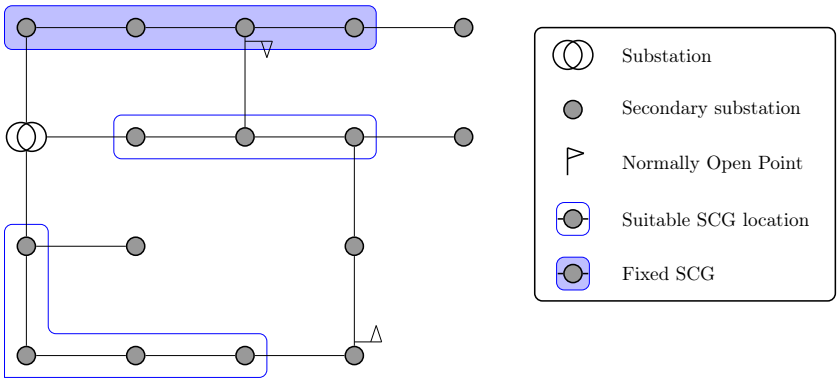


Figure 2.7: Suitable and fixed SCG locations

- The *fixed SCGs, CBs and iSSs* are the SCGs, CBs and iSSs that are already present in the network. They are not taken into a mix, since a mix only contains components that can be added to the current state of the network.
- The *empty mix* is the mix that represents the current state of a network component-wise, where only the fixed SCGs, CBs and iSSs are present. It is called empty because no new components are added to the empty mix yet.
- The *fail frequency of a section* is the probability that a malfunction will occur on that section over a period of one year.
- The *expected CML* of a certain mix is the CML expected over a period of one year with that mix installed on the network.
- The *expected CML reduction* of a mix is the difference between the expected CML of the considered mix is and the expected CML of the empty

mix. It is used when determining the quality of a mix.

- A *superfluous CFPI* in a mix is a CFPI that induces remote checkability on a point where an SCG or iSS also induces remote checkability. In this case, the CFPI no longer adds value to the network. For example: a CFPI in an iSS is superfluous, since the iSS already knows whether a short circuit passed or not. Note that a CB does not make a CFPI superfluous, since a CB only induces indirect checkability.

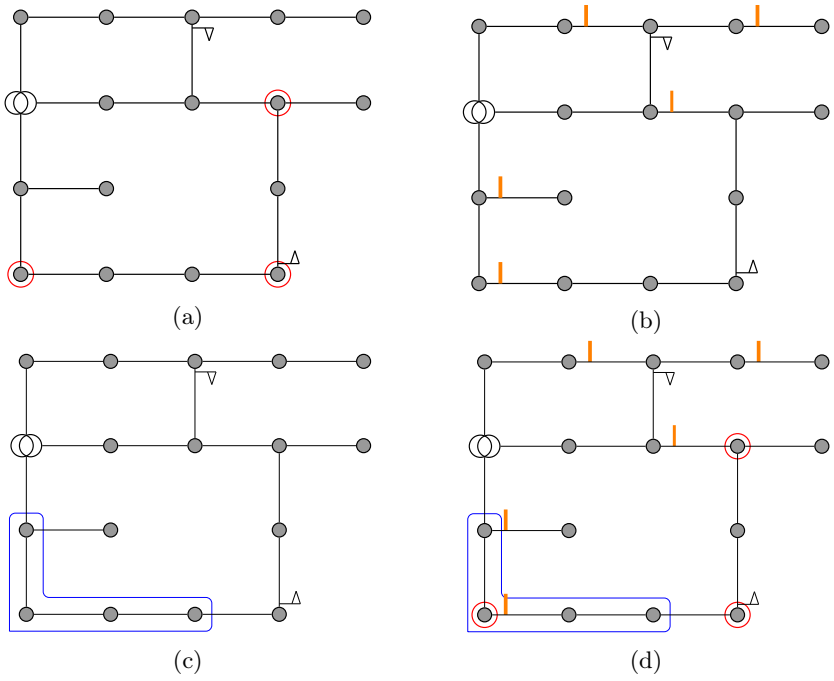


Figure 2.8: (a) An intelligent Secondary Substation mix (b) A Circuit Breaker mix (c) A Smart Cable Guard mix (d) An example of a mix

2.7 Financial aspects

The quality of a mix is expressed as the profit that a mix provides. This may seem contradictory with the fact that Liander N.V. also wants to focus on reducing customer minutes lost. To explain this, assume we place an iSS on all secondary substations. Then all the faults that occur on the network can be fixed in a matter of minutes, so the downtime would be pretty minimal. On the other hand, now we have generated a huge pile of costs for installing and maintaining all new iSSs, and all in all Liander N.V. is now making losses, which is the exact opposite of what we want to achieve. Therefore, we need to take

the financial aspects into account. Since reduced customer minutes lost can be expressed as a profit, it is chosen to show the quality of mixes through their profit. In the next paragraph it is explained how this financial evaluation of mixes is done.

The financial evaluation of a mix consists of three parts: the costs of the components, the revenue of the expected CML reduction and the revenue of the superfluous CFPIs. The evaluations are done by looking at all costs and revenues that occur over a period of 40 years[2]. This means that the costs for components consist of maintenance costs over 40 years and installment costs. An overview of the costs per component is given in Table 2.1. All costs are in euros. All maintenance costs are over a period of 40 years. The costs for the iSS varies based on the type of secondary substation that it is installed on.

Component	Installment costs	Maintenance costs	Total costs
iSS	$60000 - 75000$	12000	$72000 - 87000$
SCG[3]	5000	25000	30000
CB	13000	1000	14000

Table 2.1: Component costs

For the CML reduction revenue the following is done. First the expected CML reduction is determined. Per expected CML reduced, 1 euro is taken into the yearly CML reduction revenue. The total CML reduction revenue over 40 years is then obtained by multiplying the yearly CML reduction revenue by 20.[4] An overview of all revenues are given in table 2.2. In general, the revenue for a mix consists of expected CML reduction revenue for 90% and consists of superfluous CFPI revenue for 10%.

Revenue type	Total revenue
Superfluous CFPI	10000
Expected CML reduction	$20 \times$ expected CML reduction

Table 2.2: Mix revenues

For a mix on a certain network, each financial segment can be determined separately. Taking the sum of these segments gives the financial evaluation of a mix.

In operation, it is easy to determine the costs and superfluous CFPI revenue. Determining the expected CML reduction revenue is quite difficult, but it can

[2]Liander N.V. does all of its financial calculations and predictions over a period of 40 years into the future, and therefore this is also done for the optimal mix problem.

[3]In reality a SCG has a lifespan of 10 years. This is taken into account by adding additional installment costs for three SCGs to the maintenance costs. Together, this makes a sensible estimate of the SCG costs for the 40 year span.

[4]20 is the factor calculated by Alliander for which the total net present value over 40 years of 1 reduced CML is taken, see [1].

be done. For now it is sufficient to know the terminology. The details of the expected CML reduction calculation follow in the next chapters.

2.8 The Optimal Mix Problem

All necessary methods and definitions have now been explained, in order to formulate the optimal mix problem.

The Optimal Mix Problem (OMP)
For a substation, find the mix on that network that maximizes the profit.

So, the goal is to find the most profitable mix per network. This mix is called the optimal mix. Note that most profitable mix largely coheres with a mix that gives a large expected CML reduction, since a big part of the financial evaluation depends on expected CML reduction.

Chapter 3

Mathematical formalization

In this chapter we formalize the Optimal Mix Problem in a mathematical manner. In the first section the (mathematical) notation is introduced. The Optimal Mix Problem will be mathematically defined in the second part.

3.1 Notation and additional definitions

In order to define the Optimal Mix Problem, some notation and additional definitions are required. In order, first the definitions of the topological properties of a network will be given, then the definitions of the mixes and to finish this section the financial aspects are treated. Before all that, let us start by defining the set `DATA` to be the set that represents the big Alliander database DATA from the previous chapter. For a single network, `DATA` can be categorized into two different types of information:

1. *Topological information.* This describes the structure of the network: present (secondary) substations and their locations, the number of sections and how they are connected to the (secondary) substations, NOP presence on sections, and everything that can be concluded from this, such as the number of routes (originating from the substation).

2. *Part information.* This describes the properties that all the present parts have: type of (secondary) substations, fail frequencies of sections, capacity of sections, and so on.

`DATA` will be used to obtain specific instances of the Optimal Mix Problem that represent real-life networks supervised by Liander, and it will also be used as input for the Optimal Mix Model (as will be shown in Chapter 4).

3.1.1 The network

A network N on the medium voltage grid can be mathematically represented by a triple $N = (G, F_{\text{comp}}, L_{\text{comp}})$, where G is an undirected graph, F_{comp} is the set of fixed components and CFPIs and L_{comp} is the set of suitable component locations. This section defines the triple formally for a network N. Starting with defining an undirected graph $G = (U, S)$ with vertices U and edges S for a network N. The set U then represents the (secondary) substations of N, with $u_1 \in U$ representing the power source, and S represents the sections of N connecting the (secondary) substations. To make this more precise define S_o to be the set of optional sections and S_f to be the set of fixed sections, such that $S_o \cap S_f = \emptyset$ and $S_o \cup S_f = S$. Then $G_f = (U, S_f)$ is the undirected graph representing N with its utilized sections in normal operation, the fixed sections. Note that G_f is a tree. $G = (U, S) = (U, S_o \cup S_f)$ is the undirected graph representing N with its utilized sections in normal operation and its sections with an NOP installed on them, the optional sections. For an example, see Figure 3.1 and 3.2. Here, $U = \{u_1, \ldots, u_{17}\}$, $S_f = \{s_1 \ldots, s_{16}\}$, and $S_0 = \{s_{17}, s_{18}\}$.

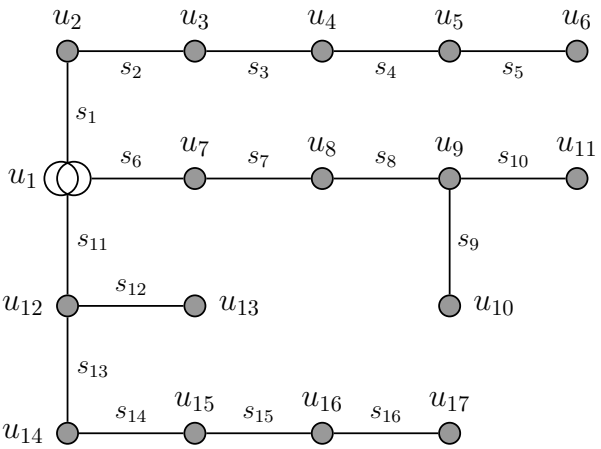


Figure 3.1: $G = (U, S_f)$

Next, define the subsets of the fixed components of N for iSS, SCG and CB and define the set of CFPIs of N:

- $F_{\text{iSS}} \subseteq U$ is the set of fixed iSSs.
- $F_{\text{SCG}} \subseteq U \times U$ is the set of fixed SCGs.
- $F_{\text{CB}} \subseteq S \times U$ is the set of fixed CBs.
- $F_{\text{CFPI}} \subseteq S \times U$ is the set of CFPIs.

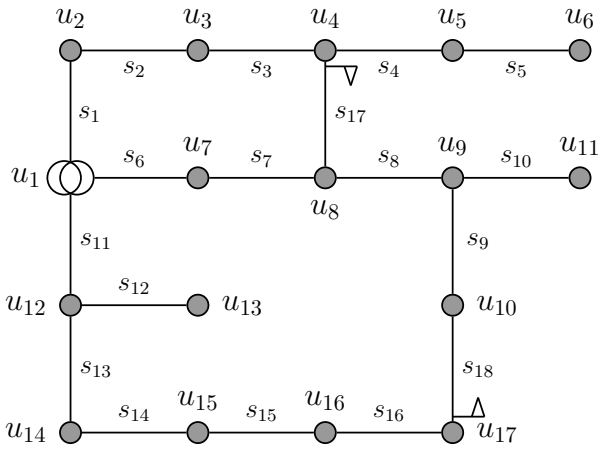


Figure 3.2: $G = (U, S_f \cup S_o)$

With this, define $F_{\text{comp}} = (F_{\text{iSS}}, F_{\text{SCG}}, F_{\text{CB}}, F_{\text{CFPI}})$ to be the 4-tuple of all fixed components and CFPIs. The substation of N, or simply N, is said to be remotely controllable when $u_1 \in F_{\text{iSS}}$.

Now, to define the suitable component locations of N, introduce the functions $f^{\text{iSS}} : \texttt{DATA} \to U$, $f^{\text{SCG}} : \texttt{DATA} \to (U \times U)$ and $f^{\text{CB}} : \texttt{DATA} \to (S \times U)$. These function determine the suitable locations per component type for a specific network based on the rules and information that lies in `DATA`. Note that $F_{\text{iSS}} \cap f^{\text{iSS}}(G) = \emptyset$ (and equivalently for SCG and CB) since a fixed component location never is a suitable component location. Using these functions, the suitable component locations of N can be defined:

- $f^{\text{iSS}}(G) = L_{\text{iSS}} \subseteq U$ is the set of suitable iSS locations.
- $f^{\text{SCG}}(G) = L_{\text{SCG}} \subseteq U \times U$ is the set of suitable SCG locations.[1]
- $f^{\text{CB}}(G) = L_{\text{CB}} \subseteq S \times U$ is the set of suitable CB locations.

With this, define $L_{\text{comp}} = (L_{\text{iSS}}, L_{\text{SCG}}, L_{\text{CB}})$ to be the 3-tuple of all suitable component locations.

Finally, define the following two functions for N that give the capacity for each section in S and the number of clients connected to each secondary substation in U:

- $\texttt{cap} : S \to \mathbb{R}$ is the section capacity function
- $\texttt{clients} : U \backslash \{u_1\} \to \mathbb{N}$ is the number of connected clients function

A medium voltage network N can now be mathematically represented by the 5-tuple $N = (G, F_{\text{comp}}, L_{\text{comp}}, \texttt{cap}, \texttt{clients})$. For an example (where the capacity

[1] If $p = (u_1, u_2) \in L_{\text{SCG}}$, then u_1 specifies the 'master device' location and u_2 specifies the 'slave device' location.

and number of client functions are omitted in the graph for the sake of clarity), see Figure 3.3.

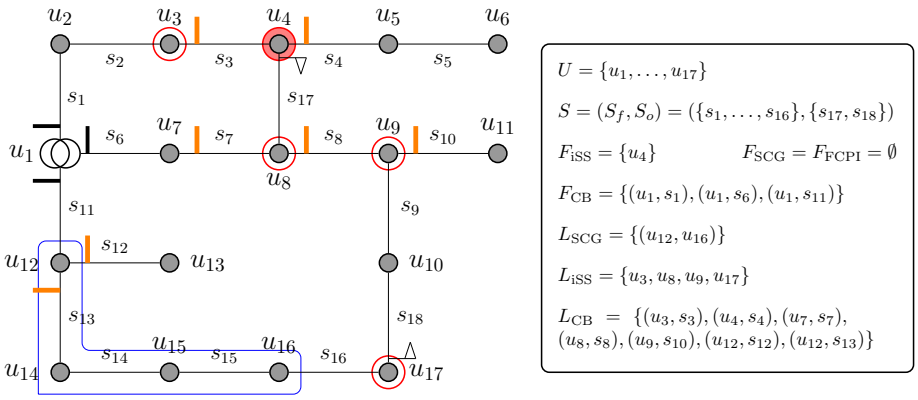


Figure 3.3: Example of a network $N = (G, F_{\text{comp}}, L_{\text{comp}}, \texttt{cap}, \texttt{clients})$ (with `cap` and `clients` omitted)

3.1.2 Mixes

A mix (on a network N) is a set of new non-fixed components and their (suitable) locations, such that the CB-rule is satisfied. To represent a mix mathematically, a binary notation with certain properties is introduced in this section. Start with defining a suitable locations vector for each of the component types. The vectors hold exactly all the locations from their corresponding suitable location sets:

- Let $Q^{\text{iSS}} \in U^{|L_{\text{iSS}}|}$ be the suitable iSS locations vector such that each location in L_{iSS} appears exactly once in Q^{iSS}.
- Let $Q^{\text{SCG}} \in (U \times U)^{|L_{\text{SCG}}|}$ be the suitable SCG locations vector such that each location in L_{SCG} appears exactly once in Q^{SCG}.
- Let $Q^{\text{CB}} \in U^{|L_{\text{CB}}|}$ be the suitable CB locations vector such that each location in L_{CB} appears exactly once in Q^{CB}.

As an example, take the network N as defined in Figure 3.2. For this network, $Q^{\text{iSS}} = (u_3, u_8, u_9, u_{17})$, $Q^{\text{SCG}} = (\quad)$ and $Q^{\text{CB}} = ((u_3, s_3), (u_4, s_4), (u_7, s_7), (u_8, s_8), (u_9, s_{10}), (u_{12}, s_{12}), (u_{12}, s_{13}))$.

With the above, a mix with only iSSs can be defined to be an element a of the set $\{0, 1\}^{|L_{\text{iSS}}|}$, where $a_i = 1$ means that an iSS is placed on suitable iSS location Q^{iSS}_i in mix a and where $a_i = 0$ means that no iSS is placed on suitable iSS location Q^{iSS}_i in mix a. This notation works for iSSs and SCGs, but not for CBs, since it does not take the CB-rule into account. This can be solved.

To take the CB-rule into account, first consider the term full trail again. For a network $N = (G, F_{\text{comp}}, L_{\text{comp}}, \texttt{cap}, \texttt{clients})$, a path T on the graph G is a

full trail if it is a path from the substation vertex u_1 to a vertex that is a leaf in the graph G_f. For a full trail T, define T_Q to be the set of suitable CB locations that occur in the full trail and define T_F to be the set of fixed CBs that occur in the full trail:

- $T_Q = \{i | Q_i^{\text{CB}} \text{ is in } T\}$
- $T_F = \{(s,u) \in F_{\text{CB}} | (s,u) \text{ is in } T\}$

Then a CB mix $c \in \{0,1\}^{|L_{\text{CB}}|}$ is a feasible CB mix whenever $\sum_{i \in T_Q} c_i + |T_F| \leq 3$ for all full trails T, since the CB-rule allows a maximum of 3 CBs per full trail.

Now component mixes per type are defined as follows:

- $A = \{0,1\}^{|L_{\text{iSS}}|}$ is the set of all feasible iSS mixes.
- $B = \{0,1\}^{|L_{\text{SCG}}|}$ is the set of all feasible SCG mixes.
- $C = \{c \in \{0,1\}^{|L_{\text{CB}}|} | \ \forall \text{ full trails } T : \left(\sum_{i \in T_Q} c_i + |T^F|\right) \leq 3\}$ is the set of all feasible CB mixes.

With this binary notation, the following holds. For an iSS mix $a \in A$, an iSS is placed on location Q_i^{iSS} if and only if $a_i = 1$, and equivalently, an iSS is not placed on location Q_i^{iSS} in iSS mix $a \in A$ if and only if $a_i = 0$. This is similar for the SCG and CB mixes.

Now, the set of all feasible mixes is defined by $M = A \times B \times C$ and the empty mix is defined by $m_0 = (\mathbf{0}, \mathbf{0}, \mathbf{0})$. Since mixes are defined on a specific network N and since it is often clear what network is worked with, the set of all feasible mixes M is obviously equal to the set of all feasible mixes on N. When it is needed to be more precise, the network will be specified in the notation. In that case, the set of all feasible mixes on N is denoted by M_N.

For an example of an unfeasible mix and a feasible mix with correct notation, see Figure 3.4 and Figure 3.5 on the next page. Note that a mix can also be represented by a binary vector of length $|A| + |B| + |C|$ where the first $|A|$ coordinates correspond to the iSS placements, the following $|B|$ coordinates correspond to the SCG placements and the final $|C|$ coordinates correspond to the CB placements. A loose component c can then be defined as a binary vector of length $|A| + |B| + |C|$ and of weight 1 ($c_i = 1$ for exactly one $i \in \{1, \ldots, |A| + |B| + |C|\}$). The type of c then is iSS if $c_i = 1$ for $1 \leq i \leq |A|$, SCG if $c_i = 1$ for $|A|+1 \leq i \leq |A|+|B|$ and CB if $c_i = 1$ for $|A|+|B|+1 \leq i \leq |A|+|B|+|C|$. For example, the mix in Figure 3.5 would then be $m = (0,1,0,1,1,0,1,1,0,1,1,0)$.

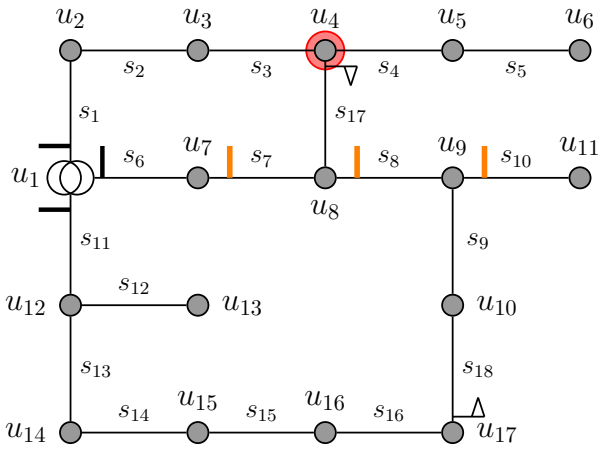


Figure 3.4: Example of an unfeasible mix $m = (\mathbf{0}, \mathbf{0}, (0,0,1,1,1,0,0))$ with $F_{\text{iSS}} = \{u_4\}$, $F_{\text{SCG}} = F_{\text{FCPI}} = \emptyset$, $F_{\text{CB}} = \{(u_1, s_1), (u_1, s_6), (u_1, s_{11})\}$, $Q^{\text{iSS}} = (u_3, u_8, u_9, u_{17})$, $Q^{\text{SCG}} = (\ )$ and $Q^{\text{CB}} = ((u_3, s_3), (u_4, s_4), (u_7, s_7), (u_8, s_8), (u_9, s_{10}), (u_{12}, s_{12}), (u_{12}, s_{13}))$.

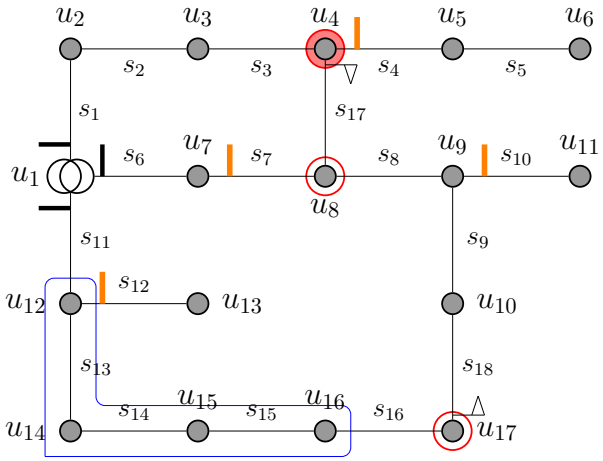


Figure 3.5: Example of a feasible mix $m = ((0,1,0,1),(1),(0,1,1,0,1,1,0))$ with $F_{\text{iSS}} = \{u_4\}$, $F_{\text{SCG}} = F_{\text{FCPI}} = \emptyset$, $F_{\text{CB}} = \{(u_1, s_1), (u_1, s_6), (u_1, s_{11})\}$, $Q^{\text{iSS}} = (u_3, u_8, u_9, u_{17})$, $Q^{\text{SCG}} = (\ )$ and $Q^{\text{CB}} = ((u_3, s_3), (u_4, s_4), (u_7, s_7), (u_8, s_8), (u_9, s_{10}), (u_{12}, s_{12}), (u_{12}, s_{13}))$.

Operations on mixes

For later purposes it is useful to have a notation for removing or adding a specific component. The binary notation for mixes gives us a way to handle mixes

and components, but this is not yet sufficient. By seeing a mix as a binary vector of length $|A|+|B|+|C|$ and a component as a binary vector of the same length and weight 1, the addition of a component c to a mix m can be seen as the binary `OR` operation. The removal of a component c is a bit more difficult, since it depends on the presence of a component in a certain mix how to do this. To make this all clear, the following definitions and notation shall be used:

Let c be a component and let m be a mix, then:

- $m + c$ denotes adding component c to mix m. If c is already present in mix m, then $m + c = m$.
- $m - c$ denotes removing component c from m. If component c is not present in mix m, then $m - c = m$.

3.1.3 Financial aspects

Now the notation for the financial aspects. We will start with the general notion of the profit and then go deeper into each of the functions.

- $w : M \to \mathbb{R}$ is the profit function
- $r : M \to \mathbb{R}$ is the revenue function
- $c : M \to \mathbb{R}$ is the cost function
- $r_{\text{CFPI}} : M \to \mathbb{R}$ is the superfluous CFPI revenue function
- $r_{\text{CML}} M \to \mathbb{R}$ is the expected CML reduction revenue function
- $v_{\text{red}} : M \to \mathbb{R}$ is the expected CML reduction function
- $v : M \to \mathbb{R}$ is the expected CML function

The revenue function r consists of two parts: the expected CML reduction revenue (r_{CML}) and the superfluous CFPI revenue (r_{CFPI}):

$$w(m) = r(m) - c(m) \tag{3.1}$$
$$r(m) = r_{\text{CML}}(m) + r_{\text{CFPI}}(m) \tag{3.2}$$
$$= 20 \cdot v_{\text{red}}(m) + r_{\text{CFPI}}(m) \tag{3.3}$$
$$= 20 \cdot (v(m) - v(m_0)) + r_{\text{CFPI}}(m) \tag{3.4}$$

Here $v(m)$ is the expected CML for mix m and $v(m_0)$ is the expected CML for the empty mix. r_{CFPI} can be expressed in more detail:

$$r_{\text{CFPI}}(m) = q_{\text{CFPI}}(m) \cdot 10^4 \tag{3.5}$$

where $q_{\text{CFPI}}(m)$ is the number of superfluous CFPIs in mix m and 10 000 is the revenue per superfluous CFPI. The following definitions are needed to express v in more detail:

- $\tilde{v} : M \times S \to \mathbb{R}$ is the function that gives the expected total CML for the a specific network using mix m whenever a malfunction in section s occurs
- $\tilde{\tilde{v}} : M \times S \times U \to \mathbb{R}$ is the function that gives the expected CML using mix m when a malfunction in section s occurs for all clients in total connected to secondary substation u
- $t : M \times S \times U \to \mathbb{R}$ is the function that gives the number of minutes that no electricity runs through secondary substation u
- p_s is the fail frequency of section s
- b_u is the number of clients connected to node u

The above relate as follows:

$$v(m) = \sum_{s \in S} \tilde{v}(m, s) \cdot p_s \tag{3.6}$$

$$= \sum_{s \in S} \left(\sum_{u \in U} \tilde{\tilde{v}}(m, s, u) \right) \cdot p_s \tag{3.7}$$

$$= \sum_{s \in S} \left(\sum_{u \in U} t(m, s, u) \cdot b_u \right) \cdot p_s \tag{3.8}$$

Note that $\frac{\tilde{\tilde{v}}(s,m,u)}{b_u} = t(s, m, u)$. Generally, the expected CML function as stated in (3.6) will be used. The deeper levels of expected CML function as stated in (3.7) and (3.8) are insightful as to how the calculation goes, and are only used in specific scenarios.

Finally, the cost function c can be expressed as follows:

$$c(m) = \langle c^{\text{iSS}}, a \rangle + \langle c^{\text{SCG}}, b \rangle + \langle c^{\text{CB}}, c \rangle \tag{3.9}$$

where $m = (a, b, c) \in M$ and $c^{\text{iSS}} \in \mathbb{R}^{|A|}$ is the iSS cost vector, where the cost c_i^{iSS} corresponds to the installment and maintenance costs of an iSS on location Q_i^{iSS}. The similar holds for c^{SCG} and c^{CB}, with these costs derived from `DATA` (see also Table 2.1). So $c(m)$ gives exactly the costs of a mix m.

Combining (3.1), (3.2), (3.5), (3.6) and (3.9), the profit function can be stated in the following way:

$$
\begin{aligned}
w(m) &= r(m) - c(m) \\
&= r_{\mathrm{CML}}(m) + r_{\mathrm{CFPI}}(m) - c(m) \\
&= 20 \cdot (v(m) - v(m_0)) + r_{\mathrm{CFPI}}(m) - c(m) \\
&= 20 \cdot \left(\sum_{s \in S} \tilde{v}(m, s) \cdot p_s - \sum_{s \in S} \tilde{v}(m_0, s) \cdot p_s \right) + q_{\mathrm{CFPI}}(m) \cdot 10^4 \\
&- \langle c^{\mathrm{iSS}}, a \rangle + \langle c^{\mathrm{SCG}}, b \rangle + \langle c^{\mathrm{CB}}, c \rangle \\
&= 20 \cdot \left(\sum_{s \in S} (\tilde{v}(m, s) - \tilde{v}(m_0, s)) \cdot p_s \right) + q_{\mathrm{CFPI}}(m) \cdot 10^4 \\
&- \langle c^{\mathrm{iSS}}, a \rangle + \langle c^{\mathrm{SCG}}, b \rangle + \langle c^{\mathrm{CB}}, c \rangle
\end{aligned}
$$

3.2 Formally Defined: Optimal Mix Problem

With Section 3.1 providing the needed definitions, the Optimal Mix Problem can be mathematically stated as follows:

The Optimal Mix Problem (OMP)

Let N be a network. Find m_{opt} in M_N such that

$$w(m_{\mathrm{opt}}) = \max_{m \in M_N} w(m) \tag{3.10}$$

Here, again, M_N is the set of all feasible mixes on N and w is the profit function. (3.10) is a nice summary of what has been stated so far about OMP. In order to maximize the profit, it is needed to minimize expected CML while also minimizing the component costs and maximizing the superfluous CPFIs. And in order to minimize the CML we need the notion of a medium voltage network and its topology, and so on.

An important notion to make is that OMP is a combined optimization problem and is equivalent to the following:

The Optimal Mix Combined Problem (OMCP)

Let N be a network. Find m_{opt} in $A_N \times B_N \times C_N$ such that

$$w(m_{\mathrm{opt}}) = \max_{(a,b,c) \in A_N \times B_N \times C_N} w(a, b, c) \tag{3.11}$$

It is clear that OMP and OMCP are equivalent. Noting the combined optimization problem property of OMP is useful for analyzing and constructing possible optimization algorithms. Simply said, OMP being a combined optimization

problem means that OMP exists of multiple optimization problems, namely the optimal iSS-mix problem, the optimal SCG-mix problem and the optimal CB-mix problem. All problems can be attacked separately to find an optimal mix of that specific component (so where no components of other types are added to the network). For OMP however, we want to combine the three separate problems to obtain *the* optimal mix, which is a iSS-, SCG-, and CB-mix. How the combined optimization problem property can be used to solve OMP will be elaborated in Chapter 7.

The critical subject that has not been mentioned so far, is the method that is used to calculate $w(m)$. This shall be treated in more depth in the next chapter.

Chapter 4

The Optimal Mix Model

The Optimal Mix Model is developed by Alliander in order to solve the Optimal Mix Problem. The current version of the model will be explained in this chapter by looking at two different aspects. Firstly, the outline of the model as a whole will be considered. Secondly, the objective function that determines the profit per mix is treated.

4.1 Outline

Figure 4.1 is a visualization that represents the Optimal Mix Model. We will treat the blocks in this visualization phase by phase. There are four phases in the model to be considered: determination of suitable locations, generation of mixes, selection of mixes and evaluation of mixes. These phases are colored yellow, green, blue and red respectively in the visualization.

4.1.1 Determination of suitable locations

INPUT: `DATA`
OUTPUT: Q^{iSS}, Q^{CB} *and* Q^{SCG}

This is where the algorithm determines what locations are suitable for a component to be placed. All these locations and their properties are retrieved from `DATA` (as defined in Section 3.1). To be specific, for the iSS options all the secondary substations are reviewed, for the CB options all sections and their connections to secondary substations are reviewed and for the SCG all pairs of secondary substations are reviewed. This action is represented by the functions $f^{\text{iSS}} : \texttt{DATA} \to U$, $f^{\text{SCG}} : \texttt{DATA} \to (U \times U)$ and $f^{\text{CB}} : \texttt{DATA} \to (S \times U)$ (as defined in Section 3.1.1). As specified before, these functions are assumed to exist and to give us the correct output. In reality, these functions check and filter all

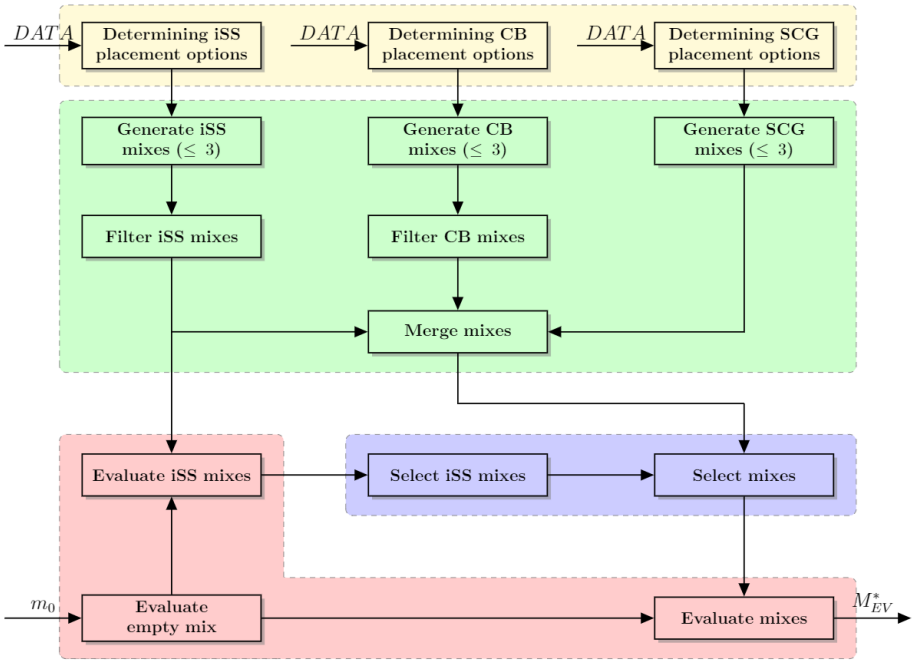


Figure 4.1: Visualization of the Optimal Mix Model

locations on all kinds of component specific rules. These rules are not relevant for the remainder of this thesis, and therefore will not be stated here.

Next, the algorithm orders all suitable locations per component and outputs a list of suitable locations per component, Q^{iSS}, Q^{CB} and Q^{SCG} respectively (as defined in Section 3.1.2). The determination of suitable locations is colored yellow in Figure 4.1.

4.1.2 Generation

INPUT: Q^{iSS}, Q^{CB} *and* Q^{SCG}
OUTPUT: A^*, B^*, C^* *and* M^*

This phase, represented by the green blocks in Figure 4.1, has A^*, B^*, C^* and M^* as output, where you might expect to see A, B, C and M, the sets of mixes as defined in Section 3.1.2. This deserves some elaboration.

The Optimal Mix Model uses a brute force approach to determine the optimal mix. The model does not evaluate every single mix however. Some mixes are unfeasible thanks to the CB-rule that only allows mixes with at most three CBs per long trail. These unfeasible mixes are generated by the model and need to be filtered out. Secondly, some simple eliminations, fabricated by experience

and expertise, are applied to rule out mixes that often have a low chance of being an optimal mix in order to lower the running time. Consequently, not all mixes are reviewed. Therefore we need the notion of a reduced set of mixes, and therefore A^*, B^*, C^* and M^* are introduced. A definition per reduced set follows below. The eliminations also become clear from these definitions.

- $A^* = \{a \in A \mid \text{rules } (i), (ii) \text{ and } (iii) \text{ hold for } a\}$ with rules:
  - (i) a contains at most three iSSs per route
  - (ii) a does not contain two neighbouring iSSs
  - (iii) a contains at least one iSS at an NOP. Consequently, if a contains only one iSS, then the iSS is located at an NOP
- $B^* = \{b \in B \mid b \text{ contains at most three new CBs per route}\}$
- $C^* = \{c \in C \mid c \text{ contains at most three SCGs per route}\}$
- $M^* = A^* \times B^* \times C^*$

Note that B satisfies the CB-rule and therefore B^* does too. Also note that using rules (i), (ii) and (iii) generally gives better mixes, but does not assure that the optimal mix is not excluded.

All in all, in the generation phase the model generates the reduced sets of mixes per component A^*, B^* and C^*, as well as the reduced set of all mixes M^* with the use of the placement location vectors Q^{iSS}, Q^{CB} and Q^{SCG}.

In Figure 4.1 we can distinguish three types of generation blocks. Firstly, we have the block that says *Generate mixes (≤ 3)*. Here, all (not necessarily feasible) mixes per component are created with at most three components added to the empty mix.

Secondly, in the block called *Filter*, the current sets of mixes are filtered by applying the CB-rule, for feasibility of CB mixes, and rules (i), (ii) and (iii) for relaxation of iSS mixes. Note that the set of SCG mixes does not need to be filtered. The output of these blocks are A^*, B^* and C^* respectively.

Finally, the reduced sets are combined in the block *Merge mixes* to give M^* as output.

4.1.3 Selection

INPUT: M^*, A^*_{EV}[1]
OUTPUT: A^*_{SEL} *and* M^*_{SEL}

The selection phase, represented by the color blue in Figure 4.1, contains another elimination method to lower the running time. It consists of two steps. The first step happens in the block named *Select iSS mixes*. Here, it selects the 30 most promising mixes from the set of evaluated iSS mixes, A^*_{EV} (see Section 4.1.4), and puts these in the set A^*_{SEL}. Additionally, $a = \mathbf{0}$ is always included in A^*_{SEL}. Next, in *Select mixes*, mixes from M^* are selected and put in the set M^*_{SEL} such that $(a, b, c) \in M^*_{\text{SEL}}$ if and only if $a \in A^*_{\text{SEL}}$. All in all, the Optimal Mix Model chooses only to evaluate those mixes that have one of the thirty most promising iSS mixes in it.

4.1.4 Evaluation

INPUT: $m_0 \in$`DATA`, A^*, M^*_{SEL}
OUTPUT: A^*_{EV} *and* M^*_{EV}

The evaluation phase is where the brute force occurs. The phase is represented by the color red in Figure 4.1 and consists of three blocks. In each block, the same principle is applied: the profit function $w : M \to \mathbb{R}$, also called the evaluation or objective function, is applied to all the mixes in the input. For *Evaluate empty mix*, *Evaluate iSS mixes* and *Evaluate mixes* the input is A^*, m_0 and M^*_{SEL} respectively. All mixes are evaluated separately. In other words, brute force is applied to the input. The output is a list of all relevant results per mix, such as the profit, the cost and revenue, the expected CML and expected CML reduction etcetera.

After the model finishes *Evaluate mixes*, the algorithm is finished and a list of mixes is obtained, containing the optimal mix from the set M^*_{SEL}. As stated before, it is not certain that the actual optimal mix is found, since some empirical based eliminations are applied during the process. The found mix is still a very good approximation of the optimal mix, since the eliminations are based on expertise and experience. However, the running time of the model still leaves a lot to be desired. On average it takes 3 to 4 days to run the model for a single substation. Considering the 350 substations that Liander maintains, it can take more than 3 years to complete the search for all optimal mixes. This is unacceptably long for operation practices. It appears that the hard part of the Optimal Mix Model lies in evaluating the mixes, even while some smart filtering and selection happens. It would be beneficial to lower the running time, or, in other words, to lower the number of mixes that are evaluated, and still obtain mixes that approximate the optimal mix very well.

The above gave rise to start a project on improvement possibilities of the

[1] See Section 4.1.4

Optimal Mix Model and ultimately gave rise to this paper on optimization of the model. Before we will take a look at improvement options, we will first go into more detail of the objective function, in order to fully understand the Optimal Mix Model.

4.2 The objective function

Recall the objective function for the Optimal Mix Problem, the profit function $w : M \to \mathbb{R}$:

$$w(m) = r(m) - c(m) \tag{4.1}$$
$$= un(m) + v_{red}(m) - c(m) \tag{4.2}$$

where $un : M \to \mathbb{R}$ is the superfluous CFPI function, $v : M \to \mathbb{R}$ is the expected CML function and $c : M \to \mathbb{R}$ is the cost function (as defined in Section 3.1.3). It is quite easy to calculate $c(m)$ and $un(m)$ for each mix: for $c(m)$, use Formula 3.9, and for $un(m)$, use 3.5, where q_u can easily be counted. The difficulty starts when trying to determine $v(m)$. Alliander has developed an elaborate method that is able to calculate the CML for a certain mix when a certain section malfunctions by accurately simulating the *malfunction attack scheme* by Liander, called the *malfunction simulation model (MSM)*. Broadly, MSM takes a certain mix m and section s as input, and returns $\tilde{v}(m, s)$, the CML for mix m when section s malfunctions. This functionality of MSM is sufficient knowledge for now. The next chapter goes into more depth on the malfunction attack scheme and MSM.

Now that we have a way of determining the CML for a mix per malfunctioning section, we can represent the current objective function w_{current} in a visualization, see Figure 4.2.

The red block in this figure corresponds to the red blocks in Figure 4.1. A red block takes a list of mixes as input, evaluates them one by one using w_{current}, and then outputs a list of evaluated mixes. An evaluated mix m_{EV} consists of the profit of the corresponding mix m, together with all the information that was determined in the process of calculating the profit.

w_{current} is represented by the numbered blocks on the right. The following happens here:

1. Choose a section s in the set of sections S.
2. Simulate a malfunction in section s on mix m using the malfunction simulation model.
3. The malfunction simulation model determines the CML per customer for a power outage on mix m caused by a malfunction in section s.
4. The malfunction simulation model sums the CML over all customers, to obtain $\tilde{v}(m, s)$, the CML in total for a power outage on mix m caused by a malfunction in section s.

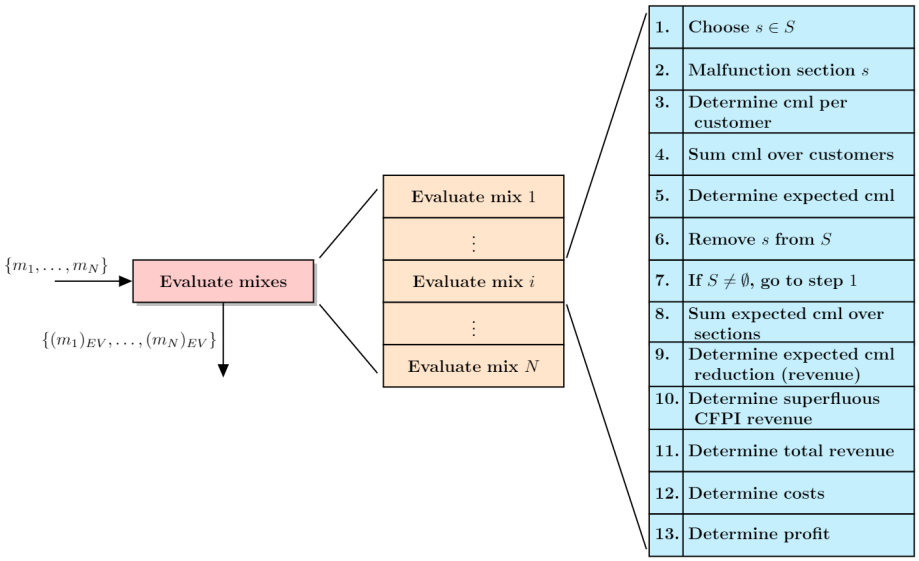


Figure 4.2: Visualization of the current objective function w_{current}

5. Multiply $\tilde{v}(m, s)$ by the fail frequency p_s of s to obtain the yearly expected CML for a power outage on mix m caused by a malfunction in section s.

6. Remove s from the set of sections S.

7. If S is not empty, then there are still sections in mix m to simulate a malfunction on. Repeat the above when such is the case. If S is empty, all possible malfunctions have been simulated: continue to the next step.

8. Sum all the yearly expected CML per section, to obtain $v(m)$ the yearly expected CML for mix m for the whole substation, giving $v(m)$.

9. Compare $v(m)$ to $v(m_0)$ to determine $v_{red}(m)$, the yearly expected CML reduction for the whole network. The yearly expected CML reduction revenue is equal to $20 \cdot v_{red}(m)$.

10. SCGs, iSSs and CBs in m induce remotely checkability on points where CFPIs may be installed. Count all these superfluous CFPIs to correctly determine the revenue $un(m)$.

11. Calculate $r(m)$ by using formula 3.2.

12. Calculate $c(m)$ by using formula 3.9.

13. Calculate $w(m)$ by using formula 3.1.

Summarizing the above, the objective function uses the malfunction simulation model to determine the CML per malfunction on a section, and then the

rest follows quite easy from the correct information in `DATA` together with the formulas from chapter 3.

4.3 Sections to malfunction

At last, an important remark needs to be made concerning the objective function and the model as a whole. In Figure 4.2 it states that a malfunction is simulated on all sections in S per mix. In reality, the Optimal Mix Model evaluates the mixes on route level, and later combines the results on route level to gain the results on substation level. It is implemented this way, since the addition of one component does not influence the expected CML for each single route in the substation. Evaluating on route level skips a lot of unneeded simulations. The results for a mix on route level can be combined using a certain method to gain the results for mixes on substation level. This method includes comparing results of iSSs that influence more than one route.

Since the above does not change the output or functionality of the model, we will not go into detail of the comparing method. The remark is mentioned however, because it offers a bit more insight to how precise the model works, and this helps in understanding how the algorithms are defined later on in Chapter 6.

Chapter 5

Complexity of the Optimal Mix Problem

The Optimal Mix Model solves the Optimal Mix Problem, but has a long running time for relatively small sizes of input in doing so. This could indicate two things. The Optimal Mix Model could be not efficiently programmed while the Optimal Mix Problem is actually solvable in polynomial time. The other, more likely, possibility is that the Optimal Mix Problem can simply not be solved efficiently, and therefore finding the Optimal Mix becomes very hard. When a problem is not efficiently solvable, it gives ground to using approximation algorithms instead.

This chapter is dedicated to analyze the mathematical complexity of the Optimal Mix Problem by looking at the complexity class `NP`. First, a definition for the class `NP` is given. Afterwards, the decision variant of the Optimal Mix Problem is defined. Thirdly, the Malfunction Simulation Model is explained in more detail to show that it is in fact an algorithm that solves an `NP`-hard optimization problem. The chapter concludes with a intuitive proof that the Optimal Mix Model is harder than `NP`.

5.1 Decision Problem Variant of the Optimal Mix Problem

The complexity classes used in this chapter say something about decision problems. If a problem is not a decision problem, it might be possible to define a decision variant of that problem. Then, if that decision problem falls in to a certain complexity class, this complexity also says something useful about the original problem. How this is done exactly will become clear later in this chapter.

Since the Optimal Mix Problem as stated in 3.10 is not a decision problem, it is first needed to define the decision variant of the Optimal Mix Problem.

The variant is worded as follows:

The Optimal Mix Decision Problem (OMDP)

Let N be a network and let $Z \in \mathbb{R}$. Is there an $m \in M_N$ such that

$$w(m) \geq Z \tag{5.1}$$

The decision problem 5.1 asks whether a certain profit bound Z on a certain network N can be met. If for an instance $I = (N, Z)$ there is a mix $m \in M_N$ such that $w(m) \geq Z$, then the answer to the instance I is "yes". If such an m does not exist, then the answer to the instance I is "no".

5.2 Complexity classes

In order to finally classify OMDP, it is needed to take a look at the definitions of complexity classes. The complexity of a problem says something about how easy it is to solve that problem. Two of these classes for decision problems are `P` and `NP`. Decision problems in the class `P` can be described as "easy to solve efficiently". When a problem is in `NP`, it may already be a bit more difficult to solve. To say something about the complexity of the decision variant of the Optimal Mix Problem, a way to work with the class `NP` is given in this section, as well as for the term `NP`-hardness.

5.2.1 Complexity class `NP`

Informally, the complexity class `NP` can be defined to be "the class of all decision problems Π that, under reasonable encoding schemes, can be solved by polynomial time nondeterministic algorithms", as cited in [2] by Garey and Johnson. In [2] a nondeterministic algorithm is described as an algorithm with a *guessing stage* and a *checking stage*. Given a problem instance I, the nondeterministic algorithm first guesses a possible solution S. During the checking stage I and S are used as input and the algorithm computes either until the computing halts and the answer "yes" is given (S indeed is a solution to I), until the computing halts and the answer "no" is given (S is not a solution to I) or the computing never halts. As said by Garey and Johnson, the "no" case and "never halting" case do not need to be distinguished.

With the above, to prove that a decision problem Π is in `NP`, it is sufficient to show that an answer S to a random instance I of problem Π can be declared as a "yes"-answer or a "no"-answer by a polynomial time algorithm. In particular, to prove that a decision problem Π is in `NP`, it is sufficient to be able to verify with a polynomial time algorithm whether a possible solution S to Π, is a "yes"-answer or not. If such a polynomial time algorithm does not exist, the decision problem Π is not in `NP`.

5.2.2 NP-completeness and NP-hardness

A decision problem Π is said to be NP-complete whenever $\Pi \in$NP and for all other decision problems $\Pi' \in$NP, there is a polynomial time reduction algorithm that reduces Π' to Π (see [2]). This means that when a problem is NP-complete, it is at least as hard as all other problems in NP. NP-completeness is an useful term when considering the whole of NP. For example, whenever it is discovered that a certain NP-complete problem Π can be solved in polynomial time, then all problems in NP can be solved in polynomial time, since all problems in NP can be reduced by a polynomial time algorithm to Π. It is said that an NP-complete problem can not be solved in polynomial time, unless P=NP (see [2]).

A problem that is at least as hard as an NP-complete problem, is NP-hard. Trivially, all NP-complete problems are NP-hard. A problem that can not be proved to be in NP, but that can be proved to be at least as hard as an NP-complete problem, also is an NP-complete problem.

A problem Π is proved to be NP-hard when a polynomial time reduction from an NP-hard problem Π' to Π can be given. With this, it should be clear to see that an NP-hard problem can not be solved in polynomial time unless P=NP, similar to NP-complete problems.

5.3 The Malfunction Simulation Model

The Optimal Mix Model makes use of the Malfunction Simulation Model. Viewing this model in more detail helps to analyze the complexity of OMP and OMDP. As stated in the previous chapter, MSM is a model build to simulate the malfunction attack scheme in order to determine CML for a power outage caused by a malfunction in a specific section. The malfunction attack scheme of Liander prescribes what actions to take in case of a real power outage on the medium voltage grid. Therefore, simulating this scheme is a correct way of determining the expected CML. In this section the malfunction attack scheme is broadly introduced and how MSM simulates the malfunction attack scheme is shortly explained.

Whenever a power outage occurs, the goal is to isolate the malfunctioning section and provide the clients with energy again as fast as possible. How this should be done is precisely described in the malfunction attack scheme (see [1]). The scheme describes five phases:

0. The affected area is determined by open points and autonomous security.
1. The affected area is reduced on the basis of remote checkability and remote controllability.
2. The affected area is reduced on the basis of remote checkability.
3. The malfunctioning cable is found and isolated by using local checkability. Afterwards, the affected area is reduced on the basis of the isolated malfunction.

4. The remaining affected area is provided of power again by switching load and/or deployment of generators.

Phase 0 is the initial phase where the area affected by the power outage is determined on the basis of which CB was triggered and on basis of the locations of the open points. No action is taken in this phase, therefore it is called phase *0*.

Phases 1 and 2 are the phases where the functionalities of installed components are used, such as the remote controllability of the iSS or the remote checkability of the SCG. It is possible for the malfunctioning cable to already be isolated after phase 1 or phase 2. When a route does not have certain components installed, less actions can be taken in this phase, and more shall be done in the next phase.

Local checkability is used in phase 3 on the area that is now still affected by the power outage. Each secondary substation is locally checkable, meaning that mechanics can check a secondary substation to see whether a short circuit has passed or not. At the end of phase 3, the malfunctioning cable is isolated.

There may still be powerless secondary substations after phase 3. Phase 4 is used to switch certain flows of electricity or to deploy generators such that these secondary substations are also provided with electricity again, without affecting other clients.

Phases 0 and 4 are the easier phases to simulate. The harder parts of the malfunction attack scheme to simulate are phases 1, 2 and 3, which all are optimization problems in itself. The goal of phases 1, 2 and 3 is to utilize the least number of actions while reducing the affected area as largely as possible and while maximizing the rest-capacity of the network. This problem is called the *Switch Planner Problem* and shall be explained in the next section.

5.4 The Switch Planner Problem (SPP)

As stated in the previous section, the Switch Planner Problem (SPP) occurs in three phases in the malfunction simulation model, where the difference per phase lies in the information and methods available. This section is dedicated to explaining SPP by giving new definitions, by defining the variables needed for SPP, by giving an example of an SPP instance and finally by giving SPP as a mixed integer linear program. First, a global sketch of SPP is given. here

The idea of SPP is as follows. For the initial situation of SPP, an open point will be placed closest to the malfunction. Here, 'closest' is based on the current information, such as possible remote checkability. It is known where this open point will be placed, and therefore it is known what part of the affected area will become disconnected from the malfunction. The goal is to provide this part of the affected area with electricity again, and to do this in the least amount of actions as possible, such as the malfunction attack scheme describes. The additional goal is to maximize the rest-capacity of the 'open point to be placed'

that disconnects the malfunction from a part of the network, such that future actions may become easier to execute. This makes SPP a minimizing problem, where the number of actions and the negative rest-capacity of the open point to be placed are minimized. The following subsections explain this in more detail.

5.4.1 Definitions

SPP is defined for only the relevant part of a network. Here, relevant means relevant for a specific malfunction. On only this relevant part of the network, actions are performed. To mathematically define this relevant part correctly, the following defitions are needed:

$$H = \{\text{Sections of the (relevant part of the) network}\} \tag{5.2}$$

$$H^1 = \{h \in H | h \text{ lies in affected area, OPs on edge of area included}\} \tag{5.3}$$

$$H^2 = \{h \in H | h \text{ lies in unaffected area, OPs on edge of area excluded}\} \tag{5.4}$$

$$H^{\mathrm{OP}} = \{h \in H^1 | h \text{ contains an OP on edge of (un)affected area}\} \tag{5.5}$$

$$R = \{\text{Affected secondary substations}\} \tag{5.6}$$

$$r_0 \in R \text{ is secondary substation closest to 'OP to be placed'} \tag{5.7}$$

$$H^{\mathrm{SWT}} = \{h \in H^1 | h \text{ is switchable}\} \tag{5.8}$$

$$\forall p \in H^{\mathrm{SWT}}, q \in R: \quad \mathbf{Path}_q^p = \{h \in H^{\mathrm{SWT}} | h \text{ is } p \text{ or lies between } p \text{ and } q\} \cup \{h \in H^2 | h \text{ lies between } p \text{ and substation}\} \tag{5.9}$$

$$\mathrm{cap}: H \to \mathbb{R} \text{ is the rest-capacity function} \tag{5.10}$$

$$\mathrm{load}: R \to \mathbb{R} \text{ is the load function} \tag{5.11}$$

$$\alpha : H \times H^{\mathrm{OP}} \times R \text{ defined as } \alpha(h,p,q) = \begin{cases} 1 \text{ if } h \in \mathbf{Path}_q^p \\ 0 \text{ otherwise} \end{cases} \tag{5.12}$$

Being switchable for a section means that an open point on that section can be closed or an open point can be placed on that section. The rest-capacity of a section h is the capacity of h minus the electricity flow through h in the initial situation. With this, the capacity function defined here is different from the one defined in Section 3.1.1. The load function gives the demanded electricity for a certain secondary substation, which is called the load.

Note that the sets and functions of an instance of SPP defined above do not change throughout solving the problem. This means for example that an open point may be closed or created, but H^{OP} remains the same set. The different configurations of the network, such as the initial situation, are not described with above definitions but with other variables. The next section explains this.

5.4.2 Variables

To describe the situation of the network, the following variables are needed:

$$\forall h \in H^{\mathrm{SWT}}\ c_h = \begin{cases} 1 \text{ if } h \text{ closed} \\ 0 \text{ if } h \text{ open} \end{cases} \tag{5.13}$$

$$\forall p \in H^{\mathrm{OP}}, q \in R\ v_q^p = \begin{cases} 1 \text{ if } p \text{ provides } q \\ 0 \text{ otherwise} \end{cases} \tag{5.14}$$

$$\forall h \in H\ S_h \in \mathbb{R} \text{ is amount of electricity on top of initial situation} \tag{5.15}$$

$$\rho \in \mathbb{R} \text{ is rest-capacity of 'OP to be placed'} \tag{5.16}$$

In the above, 'p provides q' means that the secondary substation q is connected to the substation, the power source, again, and that the path over closed sections from the substation towards q includes section p.

5.4.3 Example

An example of how the variables and definitions describe an instance of SPP is given in Figure 5.1.

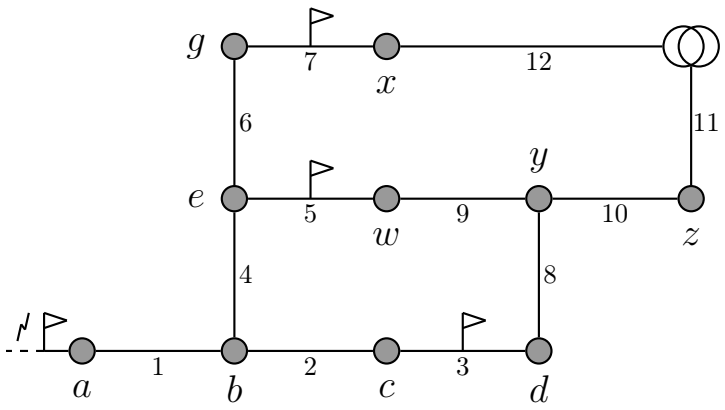


Figure 5.1: Example of an SPP instance

In Figure 5.1, the definitions are as follows:

$$
\begin{aligned}
H &= \{1, \ldots, 12\} \\
H^1 &= \{1, \ldots, 7\} \\
H^2 &= \{8, \ldots, 12\} \\
H^{\mathrm{OP}} &= \{3, 5, 7\} \\
R &= \{a, b, c, e, g\} \\
r_0 &= a \\
H^{\mathrm{SWT}} &= H^1 = \{1, \ldots, 7\} \\
\mathbf{Path}_c^3 &= \{11, 10, 8, 3\} \\
\mathbf{Path}_c^7 &= \{12, 7, 6, 4, 2\}
\end{aligned}
$$

The initial situation is describes as follows:

$$
\begin{aligned}
c_3 = c_5 = c_7 &= 0 \\
c_1 = c_2 = c_4 = c_6 &= 1 \\
\forall p \in H^{\mathrm{OP}}, q \in R,\ v_q^p &= 0 \\
\forall h \in H,\ S_h &= 0
\end{aligned}
$$

A possible situation, by closing section 7 is then described as follows:

$$
\begin{aligned}
c_3 = c_5 &= 0 \\
c_1 = c_2 = c_4 = c_6 = c_7 &= 1 \\
\forall q \in R,\ v_q^7 &= 1 \\
\forall q \in R,\ v_q^3 = v_q^5 &= 0 \\
S_4 &= \mathrm{load}(a) + \mathrm{load}(b) + \mathrm{load}(c) \\
S_3 &= 0 \\
S_{10} &= 0 \\
S_{12} &= \sum_{r \in R} \mathrm{load}(r)
\end{aligned}
$$

5.4.4 Objective function and constraints

Now that everything is defined, the Switch Planner Problem can be stated:

Switch Planner Problem

For a network described by (5.2)-(5.16), minimize:

$$\left(\sum_{h \in H^{\mathrm{SWT}} \cap H^{\mathrm{OP}}} c_h + \sum_{h \in H^{\mathrm{SWT}} \setminus H^{\mathrm{OP}}} -c_h \right) \cdot f - \rho$$

subject to:

$$\begin{aligned}
&\text{A)} \sum_{h \in H^{\mathrm{SWT}}} c_h = |H^{\mathrm{OP}} \cap H^{\mathrm{SWT}}|, \\
&\text{B)} \ \forall p \in H^{\mathrm{OP}}, \forall q \in R: \\
&\quad \text{B1)} \ \forall h \in \mathbf{Path}_q^p \cap H^{\mathrm{SWT}}, \ v_q^p \leq c_h, \\
&\quad \text{B2)} \left(\sum_{h \in \mathbf{Path}_q^p \cap H^{\mathrm{SWT}}} c_h \right) - v_q^p \leq |\mathbf{Path}_q^p \cap H^{\mathrm{SWT}}| - 1, \\
&\text{C)} \ \forall q \in R: \sum_{p \in H^{\mathrm{SWT}}} v_q^p = 1, \\
&\text{D)} \ \forall h \in H: S_h = \sum_{p \in H^{\mathrm{OP}}} \sum_{q \in R} \mathrm{load}(q) \cdot v_q^p \cdot \alpha(h,p,q), \\
&\text{E)} \ \forall h \in H: S_h \leq \mathrm{cap}(h), \\
&\text{F)} \ \forall h \in H: \rho + S_h + f \cdot \sum_{p \text{ s.t. } h \in \mathbf{Path}_{r_0}^p} v_{r_0}^p \leq \mathrm{cap}(h) + f
\end{aligned} \tag{5.17}$$

The objective function states that the number of actions should be minimized with a certain factor f, and that $-\rho$ should be minimized, where the an action is either creating an open point or closing an open point. The factor f depends on the situation of the instance. There are different cases of which two are important for now. The case $f = 0$: this occurs whenever remote controllability can be utilized, since all actions can be taken at once, and taking one additional action does not influence the CML. When $f = 0$, only minimizing $-\rho$ has the priority.

The other case is $f = (\max_{h \in H} \mathrm{cap}(h)) \cdot 1000$ or any other f such that f is larger than the capacity that ρ might have. In this case, taking extra actions does influence the CML caused by the power outage. The priority goes to minimizing the number of actions, and second priority goes to maximizing ρ.

The constraints are defined such that the final solution meet the requirements and definitions of a network. Constraint C) requires the solution to be connected, by demanding every secondary substation $q \in R$ to be connected to the power source through exactly one OP in H^{OP}.

When the solution is connected with C), then A) demands the solution to

be a tree. A) is somewhat similar to the theorem that a connected graph on n vertices without cycles contains exactly $n-1$ edges.

Constraints $B1$) and $B2$) requires the solution to have the v_q^p and c_h to fit with each other with how they have been defined in Section 5.4.2. Constraint D) makes S_h fit with the v_q^p for each $h \in H$.

Constraint E) makes sure that S_h does not exceed the maximum $\text{cap}(h)$ for each $h \in H$.

Finally, constraint F) requires ρ to be the minimum rest-capacity of all sections together and can be seen as the definition of ρ.

With all of the above, SPP is formally stated.

5.5 NP-hardness for SPP

The proof of NP-hardness for the Switch Planner Problem goes by a reduction from the Partition Problem. The Partition Problem is known to be NP-hard, see [2] where it is listed as an NP-complete problem. The Partition Problem is defined as follows.

Partition Problem

For a set S with elements in $\in \mathbb{N}$, find $S_1, S_2 \subset S$ such that:

$$\begin{aligned} &1)\ S_1 \cap S_2 = \emptyset \\ &2)\ S_1 \cup S_2 = S \\ &3)\ \sum_{s \in S_1} s = \sum_{s \in S_2} s \end{aligned}$$

For example, with $S = \{1, 2, 3\}$ a solution is $S_1 = \{1, 2\}$ and $S_2 = \{3\}$ and with $S = \{1, 2\}$ there is no solution.

The Partition Problem can be seen as a decision problem when asking: for an S, do such $S_1, S_2 \subset S$ exist. The decision problem variant SPDP of SPP is worded as follows:

Switch Planner Decision Problem

For a network described by (5.2)-(5.16) and a $N \in \mathbb{N}$, are there $c_h \in \{0, 1\}$ for $h \in H^{\text{SWT}}$ such that:

$$\left(\sum_{h \in H^{\text{SWT}} \cap H^{\text{OP}}} c_h + \sum_{h \in H^{\text{SWT}} \setminus H^{\text{OP}}} -c_h \right) \cdot f - \rho \leq N \tag{5.18}$$

subject to 5.17A)-5.17F)?

Now the proof of NP-hardness of SPDP goes as follows. Given a set $S = \{s_1, \ldots, s_n\}$ with $s_i \in \mathbb{N}$, define an instance of SPDP as in Figure 5.2. Either take $f = 0$ or $f = \left(\sum_{s \in S} s\right) \cdot 1000$, and take $N = \left(\sum_{s \in S} s\right) \cdot |H| \cdot 10^6$.

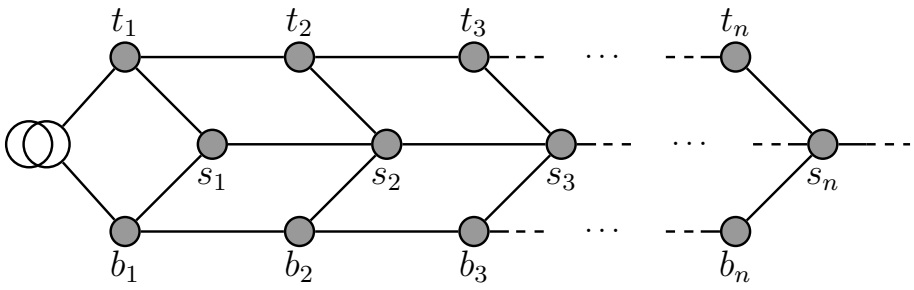


Figure 5.2: SPP instance corresponding to the Partion Problem instance $S = \{s_1, \ldots, s_n\}$

Figure 5.2 does not include the names of the sections. The sections are named as follows. For each $i \in \{1, \ldots, n-1\}$, section $h_{i,1} = (t_i, s_i)$, $h_{i,2} = (b_i, s_i)$, $h_{i,3} = (t_i, t_{i+1})$, $h_{i,4} = (s_i, s_{i+1})$ and $h_{i,5} = (b_i, b_{i+1})$. Furthermore, $h_{0,1}$ and $h_{0,2}$ are the sections from the power source to t_1 and b_1 respectively, $h_{n,1} = (t_n, s_n)$ and $h_{n,2} = (b_n, s_n)$. With this, this instance is further defined by:

$$
\begin{aligned}
H &= \{h_{i,j} | (i,j) \in \{0,n\} \times \{1,2\} \text{ or } (i,j) \in \{1,\ldots,n-1\} \times \{3,\ldots,5\}\} \\
H^1 &= \{h_{i,j} | (i,j) \in \{1,\ldots,n\} \times \{1,2\} \text{ or } (i,j) \in \{1,\ldots,n-1\} \times \{4\}\} \\
H^2 &= \{h_{i,j} | (i,j) \in \{1,\ldots,n-1\} \times \{3,5\} \text{ or } (i,j) \in \{0\} \times \{1,2\}\} \\
H^{\mathrm{OP}} &= \{h_{i,j} | (i,j) \in \{1,2\} \times \{1,\ldots,n\}\} \\
R &= \{s_i | i \in \{1,\ldots,n\}\} \\
r_0 &= s_n \\
H^{\mathrm{SWT}} &= H^1 \\
\mathrm{cap}(h_{0,1}) = \mathrm{cap}(h_{0,2}) &= \tfrac{1}{2} \cdot \sum_{s \in S} s \\
\text{For all other } h \in H,\ &\mathrm{cap}(h) = \infty \\
\forall i \in \{1,\ldots,n\},\ &\mathrm{load}(s_i) = s_i
\end{aligned}
$$

And the initial situation is described as follows:

$$\begin{aligned}
\forall i \in \{1, \ldots, n\},\ c_{h_{i,1}} &= 0 \\
\forall i \in \{1, \ldots, n\},\ c_{h_{i,2}} &= 0 \\
\text{For all other } h \in H^{\text{SWT}},\ c_h &= 1 \\
\forall p \in H^{\text{OP}}, q \in R,\ v_q^p &= 0 \\
\forall h \in H,\ S_h &= 0
\end{aligned}$$

The construction from Figure 5.2 forces a solution to this instances to have certain properties. These properties form the proof to the reduction being a correct one. The following steps prove this:

1. In a feasible solution, each secondary substation s_i in R is connected to the substation through either the top branch (containing $h_{0,1}$) or through the bottom branch (containing $h_{0,2}$). Namely, if s_i is connected to the substation through both the top branch and the bottom branch, then a cycle occurs in the solution, because sections $h_{i,3}$ and $h_{i,5}$ are not switchable for all $i \in \{1, \ldots, n-1\}$, but cycles are not permitted with constraints A) and C).
   For $i < n$, s_i is connected to the top branch by opening section $h_{i,4}$ and closing section $h_{i,1}$, so by setting $c_{h_{i,4}} = 0$ and $c_{h_{i,1}} = 1$. For $i < n$, s_i is connected to the bottom branch by setting $c_{h_{i,4}} = 0$ and $c_{h_{i,2}} = 1$. Finally, s_n is connected to either the top branch or bottom branch by by setting $c_{h_{n,1}} = 1$ or by setting $c_{h_{n,2}} = 1$.

2. In a feasible solution it holds that

   $$S_{h_{0,1}} = S_{h_{0,2}} = \tfrac{1}{2} \sum_{s \in S} s$$

   To start, the previous point forces the s_i to be connected to the substation through either $h_{0,1}$ or $h_{0,2}$, and not both. With constraint D) it holds that

   $$\forall h \in H : S_h = \sum_{p \in H^{\text{OP}}} \sum_{q \in R} \text{load}(q) \cdot v_q^p \cdot \alpha(h, p, q)$$

   meaning here that specifically $S_{h_{0,1}}$ is equal to the sum of the s_i such that $h_{0,1}$ provides s_i with electricity, since all other loads are 0. Then,

   $$S_{h_{0,1}} = \sum_{s_i \in S} \text{load}(s_i) \cdot v_{s_i}^{h_{0,1}} \cdot \alpha(h_{0,1}, h_{0,1}, s_i)$$

   which is

   $$S_{h_{0,1}} = \sum_{s_i \in S} s_i \cdot v_{s_i}^{h_{0,1}} \cdot \alpha(h_{0,1}, h_{0,1}, s_i)$$

   Since the similar holds for $S_{h_{0,2}}$, but for exactly the complement, it holds that

   $$S_{h_{0,2}} = \sum_{s_i \in S} s_i \cdot (1 - v_{s_i}^{h_{0,1}}) \cdot \alpha(h_{0,1}, h_{0,1}, s_i)$$

In particular, this implies $S_{h_{0,1}} + S_{h_{0,2}} = \sum_{s \in S} s$. Now, suppose $S_{h_{0,1}}$ is below the rest-capacity $\text{cap}(h_{0,1})$, so

$$S_{h_{0,1}} < \text{cap}(h_{0,1}) = \tfrac{1}{2} \sum_{s \in S} s$$

Then

$$S_{h_{0,2}} = \left(\sum_{s \in S} s \right) - S_{h_{0,1}} > \left(\sum_{s \in S} s \right) - \left(\tfrac{1}{2} \sum_{s \in S} s \right) = \tfrac{1}{2} \sum_{s \in S} s$$

which is in contradiction with constraint E). So the solution is unfeasible, and it must hold that

$$S_{h_{0,1}} = S_h h_{0,2} = \tfrac{1}{2} \sum_{s \in S} s$$

3. Now suppose that a feasible solution has been found to the instance in Figure 5.2. Then c_h for $h \in H^{\text{SWT}}$ are found such that

$$S_{h_{0,1}} = S_{h_{0,2}} = \tfrac{1}{2} \sum_{s \in S} s$$

Then the solution is a "yes"-answer if Formula 5.18 holds. Here, $\rho = 0$ and definitely

$$\left(\sum_{h \in H^{\text{SWT}} \cap H^{\text{OP}}} c_h + \sum_{h \in H^{\text{SWT}} \setminus H^{\text{OP}}} -c_h \right) < |H|$$

Also, as defined, $N = \left(\sum_{s \in S} s \right) \cdot |H| \cdot 10^6$, and either $f = 0$ or $f = \left(\sum_{s \in S} s \right) \cdot 1000$, resulting in Formula 5.18 to hold. In particular, this means that every feasible solution is a "yes"-answer to this instance of the SPDP.

4. The next step of the proof is the insight that each "yes"-answer (S_1, S_2) to the Partition Problem instance S implies a "yes"-answer to the constructed SPDP instance here, by arranging the c_h exactly such that

$$\forall s_1 \in S_1 : \; v_{s_1}^{h_{0,1}} = 1 \wedge v_{s_1}^{h_{0,2}} = 0$$

$$\forall s_2 \in S_2 : \; v_{s_2}^{h_{0,1}} = 0 \wedge v_{s_2}^{h_{0,2}} = 1$$

and

$$S_{h_{0,1}} = \sum_{s \in S_1} s = \tfrac{1}{2} \sum_{s \in S} = \sum_{s \in S_2} s = S_{h_{0,2}}$$

Also, a "yes"-answer $(c_{h_{0,1}}, \ldots, c_{h_{n,2}})$ to this SPDP instance implies a "yes"-answer to the Partition Problem instance S by defining

$$S_1 = \{ s_i | v_{s_i}^{h_{0,1}} \}$$

$$S_2 = \{s_i | v_{s_i}^{h_{0,2}}\} = S \backslash S_1$$

Hereby it is proven that the reduction from the Partition Problem instance S to the SPDP instance is correct.

5. The last step of the proof is noting that the reduction from the Partition Problem to the SPDP instance is a polynomial time reduction. Namely, the input for the reduction is of size $|S| = n$, and the reduction only uses linear operations, meaning it is in $\mathcal{O}(m \cdot n)$ for some $m \in \mathbb{N}$.

6. Since the Partition Problem is NP-hard, we can conclude that SPDP is NP-hard (see Section 5.2.2).

Now that it is proven that SPDP is NP-hard, it follows that SPP is NP-hard too. Namely, suppose that SPP is not NP-hard. Then it is strictly "easier" to find the minimum of the objective function in Formula 5.17 than to decide whether a solution exists below a certain bound $N \in \mathbb{N}$ (since SDPD is NP-hard), which is a contradiction.

5.6 NP-hardness for OMDP

So far in this chapter the definition for OMDP is given, the definitions for the class NP, for NP-completeness and for NP-hardness are given, a more detailed overview of MSM is given, stating that MSM solves the optimization problem SPP (see Problem 5.17) and wrapping up with a proof that SPP is NP-hard. This section will use all of this to show that OMDP is a decision problem that is not in NP and therefore harder than NP.

With the definitions from Section 5.2 a decision problem Π is in NP if a possible solution S to an instance I of Π can be verified to be a "yes"-answer or not by a polynomial time algorithm. Now assume P≠NP and suppose I is an instance of OMDP with profit boundary Z and S is a solution to I. Then, as part of checking whether $w(S) \geq Z$, the expected CML reduction revenue $r_{\text{CML}}(S)$ of S needs to be determined (see Chapter 3). To do so, the minimization problem 5.17 needs to be solved for each section in the instance I. This problem is NP-hard and therefore there exists no polynomial time algorithm to solve unless P=NP. Since we assumed that P≠NP, we can conclude that there exists no polynomial time algorithm to solve 5.17 to determine $r_{\text{CML}}(S)$. Therefore, there is no polynomial time algorithm that can verify S to be a "yes"-answer or not to the OMDP instance I, and therefore OMDP is not in NP.

For OMP, the same argument holds. For a mix of a certain instance of OMP, the profit of the mix can not be determined by a polynomial time algorithm, since the expected CML reduction revenue needs to be determined, what can only be done by solving the NP-hard problem 5.17 multiple times. Therefore, OMP is also harder than a problem in NP. As for many complexity proofs, this does depend on the assumption that P≠NP, but this is a fair assumption to make.

Now both OMDP and OMP are not in NP, which makes them harder to solve

than problems in NP. Therefore OMDP and OMP trivially are at least as hard as any NP-complete problem, which makes both OMDP and OMP NP-hard. In particular this makes OMP not efficiently solvable.

All in all we can conclude the following from this chapter. With the assumption that P≠NP, we see that problems in P are solvable in polynomial time and NP-hard problems are not. Since both OMDP and OMP are even harder than problems in NP, it makes them both at least NP-hard, and both are not solvable in polynomial time. In other words, the Optimal Mix Problem is not efficiently solvable.

Now that OMP is not efficiently solvable, there is more ground to go and try and to approximate the optimal mix instead of solving OMP completely.

Chapter 6

Heuristic methods for the Optimal Mix Problem

Heuristic methods in mathematics are a class of approximation algorithms that are well-grounded based on expertise and empirical analysis, but there is no guarantee to how well the methods approximate an optimal solution. However, heuristic methods do often achieve a sufficient goal. There are many different heuristic algorithms, each with their own strengths and weaknesses. Understanding what algorithms are applicable and what methods might be good for a certain optimization problem, requires great understanding of that problem. In this chapter it is analyzed how heuristic techniques can be utilized to approximate the solution to the Optimal Mix Problem. The chapter consists of two parts. First, it is shown how heuristics are proposed to be used to change the structure of the current Optimal Mix Model. Secondly, the relevant results of the analysis of the output of the current model for OMP are stated and explained to gain better understanding of the problem. This chapter forms the basis of the algorithms to be defined in the next chapter.

6.1 Proposed heuristic model

The current Optimal Mix Model (see Figure 4.1) contains two flaws that could be solved by applying heuristic methods. The flaws in question are long running time and absence of using generated information. Both are a consequence of the brute force like method that the current model uses during the evaluation of mixes. Except for one selection step based on promising iSS mixes, simply all mixes are evaluated one by one while no information about the finished evaluations is used in a following evaluation.

By using information generated in previous steps, an approximation algorithm could take a shorter running time while still finding a solution of the desired quality. Both flaws are attacked with this, since it aims to shorten running time by using generated information. The concept of using already

generated information is a property often found in heuristics and will form the base of the heuristic algorithms proposed in this paper.

The proposed model (or structure) of these algorithms is in Figure 6.1. This structure uses the same colors as the current model to indicate what is done in these steps of the algorithm. From this it can be seen that only the determination phase is unchanged. In the generation phase the obligatory maximum of three components is dropped, as well as the filters on the iSS (see Section 4.1.2). Though not specifically mentioned in Figure 6.1, the CB-rule is not dropped. It is taken into the standard rules of generating a CB mix at any time that a CB mix is generated. Making use of generated results happens in the loop at the bottom. Mixes are evaluated and then it is checked whether its STOP requirements are met. In case that they are met, the algorithm is finished and outputs its results. Otherwise, mixes are selected from the evaluated mixes based on algorithm specific rules, (new) mixes are generated on basis of the selection of mixes, and afterwards these are evaluated again, and so on.

Note that a specific algorithm can specify its own requirements and rules during a certain phase of the algorithm, which will specify the algorithm. So, for example, an algorithm is allowed to put a maximum on the number of components during the generation phase. The algorithms defined in the next chapter in Section 7.1 all have this model as their structure.

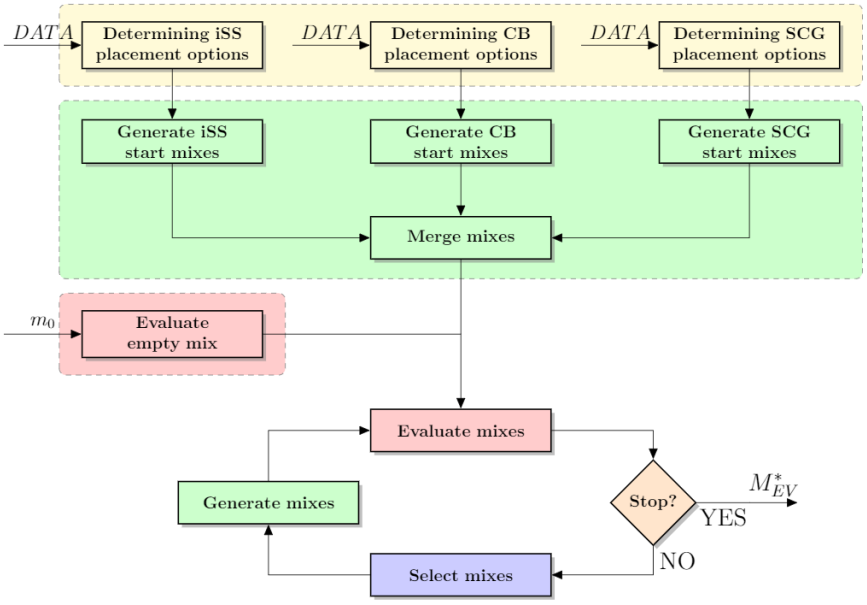


Figure 6.1: Visualization of the proposed new Optimal Mix Model

6.2 Analysis of the Optimal Mix Problem

For this research project the results of the Optimal Mix Model of two substations are taken for analysis, named Huizen (HZN) and Almere (ALR). Validation of the algorithms is done on three substations: Huizen, Almere and Vaassen (VSN) (see Chapter 8). HZN, ALR and VSN are three distinct substations, in the sense that these substations function well as representatives from their type of substation. HZN is a smaller remotely controllable substation, ALR is a large non-remotely controllable substation with sections somewhat equally distributed among its routes, and VSN is a large non-remotely controllable substation with two large routes section-wise and smaller routes for the rest. Together, HZN and ALR form a good couple to analyze, and HZN, ALR and VSN are a good triple to validate the results on.

The main goal of analyzing the results on HZN and ALR is to find relations between components, the topology of a network and all other relevant properties such as the number of clients connected to a secondary substation. These relations help in designing the correct algorithms by giving a direction to attack the problem from, which is needed to make effective use of heuristic methods.

The results that Alliander made available for analysis are tables generated by the current Optimal Mix Model containing per row all the information of an evaluation of a mix on a certain route, meaning the present components, the expected CML, the reduced expected CML, the reduced expected CML revenue, the number of superfluous CFPIs, the superfluous CFPI revenue, the total costs of this mix on this route and the profit. These tables only contain the results from the already selected iSS mixes (see Section 4.1.3).

Some specific instances of mixes are missing from the results provided by Alliander, because they do not match the requirements of the model or are filtered out at some point. Some of these instances are evaluated during this research. This provided some insights too.

First, the definitions of influence are needed:

- A component c influences a section s whenever $\tilde{v}(m_c, s) \neq \tilde{v}(m_0, s)$ where m_c represents the empty mix with only component c added.
- A component c_1 influences another component c_2, when it influences the performance of the other on a certain influenced section s, so when $\tilde{v}(m_{(c_1,c_2)}, s) \neq \tilde{v}(m_{c_2}, s) \neq \tilde{v}(m_0, s)$.
- An influential area for a component c is the set of all sections s that are influenced by component c.

Secondly, the interpretation of *top* is needed. When speaking of a *top (X)* mix, it is meant that the mix is among the highest (X) profitable mixes in its category. So a top iSS mix on a route is an iSS mix that performs better than most other iSS mixes on this route, and a top 5 iSS mix on a route is one of the 5 best iSS mixes on the route (while leaving SCGs and CBs disregarded, since this example only concerns iSSs).

Next follows a list of all relations and insights from the analysis:

1. **iSSs (can) function well when on an open point**
   An iSS notices whether a short circuit has passed, can remotely create open points and can close open points. When an iSS is placed on an open point, it can be used to redirect electricity from one route to another or to redirect electricity inside a route. Especially redirecting electricity from one route to another turns out to be valuable and can make an iSS very profitable. On multiple routes in ALR and HZN it is the case that the separate iSSs on an open point are among the top 5 best possible loose components. Not all iSSs on an open point function well, since an open point can have a low capacity or because the malfunction can not be easily disconnected from the iSS before redirecting the electricity.

2. **iSSs function poorly on non-remotely controllable substations when not on an open point**
   Whenever an iSS is not placed on an open point, and the substation is not remotely controllable, the only functions for the iSS are creating open points, closing created open points and checking if a short circuit has passed. During the initial phase, an iSS can create an open point faster than a mechanic. An iSS needs 10 minutes to do so and a mechanic needs at least 42 minutes to arrive at the scene. After these initial 10 minutes, iSSs can create or close open points immediately at any time. The mechanic still needs 10 minutes however to travel from secondary substation to secondary substation. So, an iSS is way faster at creating these open points. In order to actually reduce the affected area however, electricity needs to be redirected, and the iSS can not do so in this scenario. Therefore, a mechanic still needs to arrive at a certain (secondary) substation to activate the power flow again, to solve the power outage for a part of the affected area. Because this scenario still needs a mechanic to perform actions, it makes it that the iSS only saves around 10 minutes for a part of the affected area, which is way less than 42 minutes. On top of that, iSSs are relatively expensive. Together, this makes it that iSSs function poorly on non-remotely controllable substations when not on an open point.

3. **iSSs behave as CBs in certain situations**
   This behaviour lies in creating open points when the iSS is not placed on an open point. If the malfunction happens behind such an iSS, the only function for the iSS is placing an open point behind the secondary substation it is installed in to reduce the area that is connected to the malfunction. This is exactly the function of a CB. The difference in behaviour is of course that an iSS can also close the open point again. The difference in overall value is that an iSS costs more and takes 10 minutes to create the open point, which makes CBs more efficient in many of these situations.
   Suppose that the iSS is connected to an open point, then the initial be-

haviour stays the same: it can create an open point in similar manner as a CB. In later phases, an iSS may be able to close this open point the redirect the power flow again. In that case, the value of the iSS to the network increases. If the iSS is not able to close the open point because the capacity would be exceeded, then the value of the iSS does not increase.

4. **iSSs (can) function well on a remotely controllable substation when not on an open point**
   When the substation is remotely controllable and a malfunction happens such that the CB next to the substation is triggered, the substation can be seen as an iSS that can always close this open point to redirect electricity, since the capacity of the substation is equal or greater to the capacity needed for the route in question. If this is combined with an iSS not on an open point that also lies in front of the malfunction, then this iSS can create an open point and the substation can close its open point, resulting in a solved power outage for the trajectory between the iSS and the substation in only 10 minutes.
   If a CB would have been placed instead of the iSS, then that CB would have been triggered, and the power outage would have been prevented for the trajectory between that CB and the substation. Often, this CB placement is a more profitable choice. This can not be said in general however, since an iSS may have added value based on the situation.

5. **iSSs can function well when on a crossing**
   A crossing has at least three sections of which the branch can be disconnected from the other two by placing an open point. In general, this gives the iSS a greater influential area, which also raises the profit of the iSS. Whenever the iSS is placed on a crossing of at least three sections that is also connected to an open point of at least a fourth section, then the value of the iSS rises even more.

6. **iSSs duos function can function well when at least one is placed on an open point**
   Placing two iSSs such that one is placed on an open point and the other is placed in the same branch, or the same full trail, as the first iSS, then it can be a well functioning duo. Where the second iSS may be bad as a loose component (because of previous made statements), it can now help in the following way. If a malfunction happens in another branch then the duo, or in front of the iSS closest to the substation, then the trajectory between the two iSSs does not contain a malfuncting cable, and therefore can be disconnected from the malfunctioning area by placing an open point. This placement is often done by the iSSs not on the open point. Now the trajectory is disconnected from the fault in the network, and the iSS on the original open point can close it to redirect electricity and provide the trajectory of power again. This property is seen to be very valuable on HZN and ALR. More then half of the routes containg an iSS in the

optimal mix contained at least two iSSs placed in the manner described here.

7. **SCGs are dominant**
   SCGs are the most common type of component in the optimal mixes on HZN and ALR. Its unique preventive property together with its property to precisely locate fault in its trajectory appeared to be profitable in most cases, even when iSSs and CBs also lie over its trajectory.

8. **Components with negative profit are not included in the optimal mix on HZN and ALR**
   Whenever a single component c evaluates to a negative profit, so $w(c) < 0$, then it will not be included in the best mixes on this substation, unless it is part of a profitable iSS duo. This holds for both HZN and ALR.

9. **Components giving a (high) positive profit relative to their route are (more) likely to be included in the optimal mix**
   This insight goes hand in hand with the previous point. The addition is that components giving a *high* profit are *more* likely to be included. This lies in the following per component type:

   - *Top profitable SCGs mixes translate well to optimal mixes*
     SCGs are dominant, and this also shows in the top mixes. If a loose SCG has a high profit, then it often has a high profit when combined with other SCGs over trajectories that do not overlap. These are still profitable when CBs and iSSs are also placed over these trajectories. For ALR and HZN the top 5 mixes of SCGs of maximum size 3 had a 90%+ presence in the top 10 mixes on route level on more then 90% of the routes.
   - *Top profitable iSS mixes could translate well to optimal mixes*
     It differs per route whether a top profitable iSS mix on route level is included in the optimal mix. For some routes there are one or two standout iSS mixes that have a presence of 100% in the top 100 of mixes on route level, and for some routes the profitable iSS mixes have a presence of less then 10% in the top 500 and 0% in the top 100.
   - *Greatly profitable CB mixes could translate well to optimal mixes*
     Specifically *greatly profitable* is stated here in stead of *top profitable* CB mixes. Reason for this is that it appears in around 30% of the samples on ALR and HZN that many CB mixes on route level with a maximum of 3 CBs perform similar in the sense that the highest profitable CB mixes and the number 100 profitable CB mix differ less then 10%. When this is the case, the top mixes on route level have the same iSSs and SCGs for most of the mixes and only the CB mixes change. In other words, the top 100 mixes on route level contain the same iSSs and SCGs and only differ in CBs, while not differing as much in profit.

When the above is not the case, for example when the empty mix on route level already has many CBs placed and only a small number of new CBs can be placed with the CB-rule, there is a clear top 10 of CB mixes on route level. In that case, the presence of top 10 CB mixes in the top 100 mixes on route level lies around the 80% for the checked samples on ALR and HZN.

10. **Behaviour of a triple of one type can often be explained by the duos of this triple**
From both HZN and ALR it appeared that the behaviour of a triple of a certain type on one route can largely be explained by looking at the duos that form this triple. The idea here is that power outages caused by a malfunction in section s on the mix with the triple placed takes about as long as a power outage caused by a malfunction in section s on the mix with one of the duos included in the triple placed. The following examples illustrate this idea.
Consider Figure 6.2. Here, the CB triple (c_1, c_2, c_3) is placed. When set up like this, c_1 is triggered when a malfunction occurs in section s_1 or s_2, c_2 is triggered when a malfunction occurs in s_3 or s_5 and c_3 is triggered when a malfunction occurs in s_5. The length of the power outages is equal to the length of some power outages in the mixes with only duo (c_1, c_2) or only duo (c_2, c_3) as follows: the CML for failings in s_1 and s_2 is equal for both the triple and duo (c_1, c_2), the CML for failings in s_3 and s_5 is equal for both the triple and duo (c_2, c_3) and the CML for a failing in s_4 is equal for the triple and duo (c_2, c_3). In this sense, the behaviour of the triple lies in the duos. When picking the correct information from the duos, the behaviour of the triple in this example can be predicted without simulating the mix containing all three CBs.[1]

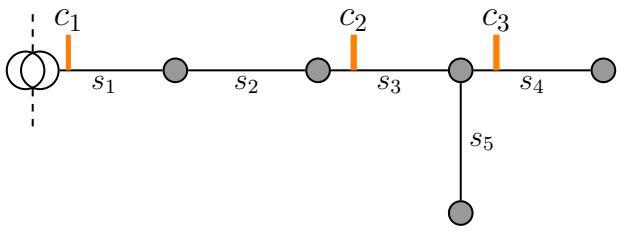


Figure 6.2: First example of three CBs placed on a route

In this second example, the rule does not apply as well. Consider Figure 6.3. In this example, only a part of the CML per section can be precisely predicted beforehand based on the information from the duos. If a malfunction occurs in section s_4 or in section s_5, then the CML for that power

[1] In reality c_1 is a fixed component. The idea of the argument then still holds.

outage is equal for both the mix with the triple as the mix with only duo (c_3, c_4) placed. If a malfunction occurs in section s_1, s_2 or s_3 however, none of the duos can predict the CML for these power outages correctly. In this example the duos can predict the CML with an error of 10 minutes per affected customer however, since only a maximum of 1 action extra (with respect to the mix with the triple) is needed to find the malfunction, namely during phase 3 (see Section 5.3). Of course, if the example looks different, the number of extra actions can grow, and the error in CML can become quite big when a large number of customers is connected to the secondary substations.

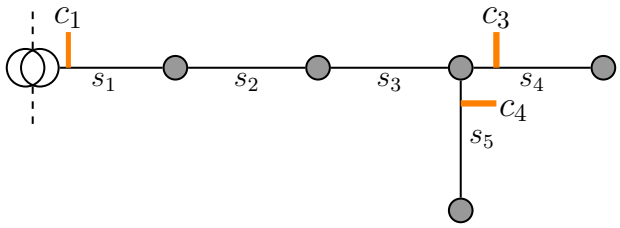
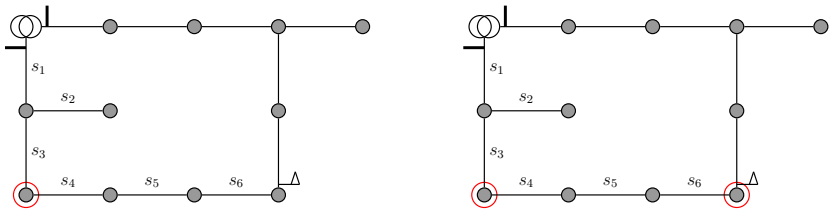


Figure 6.3: Second example of three CBs placed on a route

The second example shows that duos do not always predict the behaviour of a triple correctly. However, the duos still appear to tell us a lot. Namely, if both duos appeared to not be profitable at all, or if the components itself are not profitable, then the triple probably is not profitable either. It hardly happens that a triple is predicted to be non-profitable by the duos forming that triple, while the triple actually is very profitable. In this sense, the duos are still able of telling us whether a triple can be useful to include in a mix. This idea is further explained by the next point in the list.

11. **When components influence each other, they worsen each other, except for two iSSs**
This point is an assumption to be used in many of the algorithms in the next chapter. The assumption was seen to often be true for HZN and ALR. Let c_1 and c_2 be two components, not necessarily of the same type. Then the assumption is that $w(m_{c_1}) + w(m_{c_2}) \geq w\left(m_{(c_1,c_2)}\right)$. The case $w(m_{c_1}) + w(m_{c_2}) = w\left(m_{(c_1,c_2)}\right)$ holds whenever the components do not influence the same sections. For the other cases the value of the separate components c_1 and c_2 is greater then the components as a combination. An intuitive explanation for this is that the components overlap certain CML that they reduce. See for example Figure 6.2. When c_3 is placed separately, it influences section s_4. When c_2 is placed separately, it influences sections s_3, s_4 and s_5. When c_2 and c_3 are placed however, the duration of the power outage caused by a malfunction in section s_4 is com-

pletely determined by the placement of c_3 and CB c_2 does not influence that specific power outage: c_2 has 'lost' its influence over section s_4 in this case, and therefore is off less value. The loss of influence over a section often has an impact such that for two components c_1 and c_2 in ALR and HZN that we have $w(m_{c_1}) + w(m_{c_2}) > w\left(m_{(c_1,c_2)}\right)$.
It needs to be said that duos can show behaviour in specific cases, that can not be explained by separately looking at the components, but only by the components being in a combination. For example, if a route immediately splits into three branches after the first section, then two branches could contain a CB and the third one not. Now, whenever there is a power outage and both CBs in the two branches did not go off, then the malfunction has to have occurred in the third branch. This could not have been known for certain when there was only one CB, and this is in fact additional value to the two CBs as a duo. However, this property does not seem to add enough value to make such a combination proportionally better then the sum of the two separate profits of the components, based on HZN and ALR. It is a property to keep in mind, whenever the assumption that components worsen each other does not work out as desired.
A definite exception to the assumption is shown by the interaction between iSSs. Whenever a substation is not remotely controllable, a single iSS is often not that good, as seen earlier. This changes when the iSS is placed on a specific location in the network and another iSS is placed on an open point behind the first one. The added value of this duo is that it now becomes easier to redirect the electricity whenever a malfunction occurs outside of the region that is now supervised by this iSS duo. See Figure 6.4.



(a) (b)

Figure 6.4: The iSS in (a) helps by taking 10 minutes of a power outage and with locating the bad cable whenever a malfunction occurs in the bottom route: in most cases this does not make the iSS profitable. The iSS duo in (b) helps by locating the bad cable whenever a malfunction occurs in the route, helps by taking 10 minutes of a power outage whenever a malfunction occurs in s_4, s_5 or s_6 and solves the power outage in 10 minutes for people connected to one of the bottom secondary substations whenever a malfunction occurs in s_1, s_2 or s_3 by closing the NOP and creating an OP above the left iSS: this can be very profitable.

When this interaction happens, it is possible to see $w(c_1) + w(c_2) < w((c_1, c_2))$ where c_1 and c_2 now represent two iSSs. Do note however that this does not always happen, since the capacity of sections also needs to be taken into account. The open point in question may not have enough capacity left to provide the relevant parts of the network with electricity again.

It is important to state that the above rules and insights are theoretically easy to counter. An counterexample for "$w(m_{c_1}) + w(m_{c_2}) \geq w\left(m_{(c_1,c_2)}\right)$" can easily be created theoretically, for example by connecting a large amount of customers to specific secondary substations in Figure 6.2. It is important to note that the insights and rules does often hold for HZN and ALR, which are representative networks for the networks that Liander maintains, and therefore it us suspected to often hold on other networks maintained by Liander too.

Chapter 7

Algorithms: definitions and adaptations

The previous chapter provides ground for heuristic algorithms to be based on. This chapter is dedicated to presenting those algorithms and consists of two parts. In the first part, heuristic algorithms are explained and defined specifically for the Optimal Mix Problem based on the list of points in Section 6.2. In the end, not all algorithms have been implemented. The algorithms that are implemented sometimes needed some adaptations in order to make them perform better. The second part of this chapter describes all the adaptations that have been made in the implementation and testing process.

7.1 Algorithms

There are three categories of algorithms to be mentioned in this section: constructive algorithms, an estimator algorithm and evolutionary algorithms. The constructive algorithms, Greedy Algorithm, Beam Search Inspired Algorithm and COPIA, are mainly based on points 9, 10 and 11 from Section 6.2. The estimator algorithm is mainly based on point 9. One evolutionary algorithm, Genetic Algorithm, is based on all of the points from Section 6.2. The other evolutionary algorithm, Particle Swarm Optimization, is purely based on the concept of Particle Swarm Optimization and does not make any special use of mentioned points.

A general remark for the algorithms is that two specifications are not mentioned in the algorithm description, but should be implemented when using the algorithms. First is the feasibility of a mix, which lies in checking the CB-rule. The CB-rule should be checked at all times and a mix should be rejected whenever it does not meet the CB-rule. Second is the addition of duos. Whenever a duo is added to a mix, and one of the components in the duo is already present in the mix, then the addition of the couple is treated as the addition of the single missing component.

Even more specifications to the algorithms are not mentioned in this section below, since they do not influence the methods that the algorithms use. These specifications are mentioned in the next section (Section 7.2).

A final note is that the goal here is to define a variety of heuristic methods to show what is possible. It is not the goal to implement all of these algorithms. Based on testing results it was decided what algorithms to implement. This shall be noted per algorithm.

7.1.1 Greedy Algorithm

A greedy algorithm is an algorithm that makes the locally optimal choice in each step. Its power lies in the speed in which it can produce an already good solution, even though it may not be optimal. The version of the greedy algorithm used here is a constructive algorithm based on first improvement. Consider the first draft for the greedy algorithm, Algorithm 1. Here, the process of building the mix is done by adding component after component, while accepting components that improve the mix and rejecting components that would worsen the mix. This is done by first evaluating all loose components, sorting them in a list based on profit, and then by working through this list from top to bottom. The greedy choice per step is rejecting or accepting the new mix with a component added based on the profit.

Algorithm 1 is only taking loose components into consideration. To make the

Algorithm 1 Greedy Algorithm: first draft

Input: `DATA`
Output: mix m_{current}

1: Determine suitable component locations L_{iSS}, L_{SCG} and L_{CB}
2: Evaluate empty mix m_0
3: $m_{\text{current}} = m_0$
4: Evaluate single components c on locations $L_{\text{iSS}} \cup L_{\text{SCG}} \cup L_{\text{CB}}$
5: Sort components on profit in list L
6: **while** L is not empty **do**
7: Pick best component c_{best} in L
8: $m_{\text{new}} = m_{\text{current}} + c_{\text{best}}$
9: Evaluate m_{new}
10: **if** $w(m_{\text{new}}) > w(m_{\text{current}})$ **then**
11: $m_{\text{current}} = m_{\text{new}}$
12: **end if**
13: $L = L \backslash \{c_{\text{best}}\}$
14: **end while**

algorithm more flexible, inputs $i_1, i_2, i_3 \in \{1, 2\}$ are introduced to represent the maximum combination size for iSS, SCG and CB respectively. A combination here is a single component or two components of one type on one route, therefore $i_1, i_2, i_3 \in \{1, 2\}$. After determining the suitable component locations L_{iSS},

L_{SCG} and L_{CB}, the suitable combinations C_{iSS}, C_{SCG} and C_{CB} per route can be determined. The algorithm now uses the same greedy method as before: first evaluate all loose combinations, sort the combinations in a list on profit, and work through this list while adding or rejecting combinations to the current mix. This gives Algorithm 2.

The Greedy Algorithm is the first algorithm to implement and is expected

Algorithm 2 Greedy Algorithm

Input: `DATA`, iSS combination size i_1, SCG combination size i_2, CB combination size i_3

Output: mix m_{current}

1: Determine suitable component locations L_{iSS}, L_{SCG} and L_{CB}
2: Determine suitable combinations C_{iSS}, C_{SCG} and C_{CB} with maximum combination sizes (i_1, i_2, i_3) respectively
3: Evaluate empty mix m_0
4: $m_{\text{current}} = m_0$
5: Evaluate suitable combinations c in $C_{\text{iSS}} \cup C_{\text{SCG}} \cup C_{\text{CB}}$
6: Sort suitable combinations on profit in list L
7: **while** L is not empty **do**
8: Pick best combination c_{best} from L
9: $m_{\text{new}} = m_{\text{current}} + c_{\text{best}}$
10: Evaluate m_{new}
11: **if** $w(m_{\text{new}}) > w(m_{\text{current}})$ **then**
12: $m_{\text{current}} = m_{\text{new}}$
13: **end if**
14: $L = L \backslash \{c_{\text{best}}\}$
15: **end while**

to generate good solutions, but is also expected to not reach the optimal mix. Points 9 and 10 from Section 6.2 are included in the algorithm by first evaluating all loose combinations and by building the mixes in combinations of maximum size 2. It utilizes point 11, by never allowing a mix that has become worse. If a worsened mix is allowed, then point 11 tells us that the mix would become better by undoing the addition of the worsening combination, since that combination can never start to add (more) profit based on later additions to the mix. The exception to this rule is again the iSS, but this is taken care of by allowing combinations of size 2 for iSSs, such that profitable iSS duos are already recognized and will be considered at some point during the process.

Algorithm 2 is expected to not reach the optimum value, since it does not take locations of the components into account. It simply looks at the highest profitable loose combination that is next on the list, while this combination could very well be located somewhere in the network where another component is already placed. The addition of the highest profitable loose combination can still be profitable however, but this profit can be way lower then the addition of other combinations, because of an overlap of influential areas. The next al-

gorithm aims to attack this problem of the Greedy Algorithm.

Finally, since the Greedy Algorithm is deterministic, in the sense that it outputs the same result each time it terminates, an analysis can be made on the number of iterations, and therefore an analysis on the running time. The exact method of doing this is introduced in Chapter 8 however. For now it is sufficient to state that the Greedy Algorithm is expected to be up to 80 faster than the current model.

7.1.2 Beam Search

The concept of the Greedy Algorithm can be extended by generating more options to choose from. This concept is based on the heuristic Beam Search Algorithm. Beam Search uses breadth-first search to construct a search tree. At each level of the search tree, only a specific number of nodes, say k, are chosen. These k nodes are the k best solutions on this level, which makes Beam Search a greedy algorithm itself. These k nodes are extended again, creating a new level of the search tree. On this level, again only the k best nodes are chosen to continue with. This is done until the bottom level of the search tree is met. The output will be k solutions.

Algorithm 3 is inspired by the Beam Search algorithm. Algorithm 3, from now on called Beam Search Inspired Algorithm (BSIA), uses a similar search technique. The search tree starts with m_{current} equal to the empty mix m_0. Using n as input, the next level of search tree is created by taking n top combinations from the sorted list, and separately adding these to m_{current}. The n new mixes and m_{current} together form the next level of the search tree. This level is evaluated giving us the profit per mix. All mixes that have a profit lower then m_{current} have their corresponding added combination removed from the list. The mix with the highest profit is set to be to the new m_{current}, and its corresponding combination is also removed from the combinations list. The other added combinations are not removed from the list. Now that a new m_{current} is defined, a next level of the tree can be created. The procedure is repeated until the combinations list is empty.

Two remarks are in place here. First, a difference with Beam Search is that BSIA chooses only 1 solution from each level to continue with, while Beam Search chooses k. The reason for this lies in the fact that allowing, for example, 2 solutions to continue with immediately doubles the running time for this part of the algorithm, while it is desired to keep the running time low. When this decision turns out to be bad, the algorithm will be tuned in Section 7.2.

Secondly, the top n combinations are picked a bit differently than described above. The picking starts with checking the current top combination in the combinations list. The route of this combination is taken and the top n combinations on this route are selected, including the one already picked. In this manner the created mixes are compared on their influences on equal routes, which makes a desired comparison. It is not useful to compare two combinations on separate routes, since they do not influence the same part of the network. The only exception is an iSS combination placed on an open point, which can influence

multiple routes at once. Still it is chosen to only consider one route at a time, while keeping this exception in mind. Again, algorithms can be tuned in Section 7.2.

By defining BSIA this way, the problem of the Greedy Algorithm is expected to partially be solved. More options are created to choose from, and this creates a larger chance to compare additions of combinations with each other that lie on different parts in a certain route. The consideration of more solutions per step is expected to make a large impact on the running time, and making it significantly slower than the Greedy Algorithm. BSIA is the second algorithm to be implement.

Algorithm 3 Beam Search Inspired Algorithm (BSIA)

Input: `DATA`, iSS combination size i_1, SCG combination size i_2, CB combination size i_3, beam options n

Output: mix m_{current}

1: Determine suitable component locations L_{iSS}, L_{SCG} and L_{CB}
2: Determine suitable combinations C_{iSS}, C_{SCG} and C_{CB} with maximum combination sizes (i_1, i_2, i_3) respectively
3: Evaluate empty mix m_0
4: $m_{\text{current}} = m_0$
5: Evaluate suitable combinations c in $C_{\text{iSS}} \cup C_{\text{SCG}} \cup C_{\text{CB}}$
6: Sort suitable combinations on profit in list L
7: **while** LIST is not empty **do**
8: `route` is route of best combination c_{best} in L
9: Pick n top combinations, $c_1, \ldots, c_n$ on `route` from L
10: Add $c_1, \ldots, c_n$ to m_{current} resulting in mixes $m_1, \ldots, m_n$
11: Evaluate $m_1, \ldots, m_n$
12: $W = \max_{i \in \{1,\ldots,n\}}(m_i)$
13: **for** i in 1 to n **do**
14: **if** $w(m_i)$ is equal to W and $W > w(m_{\text{current}})$ **then**
15: $m_{\text{current}} = m_i$
16: $L = L \backslash \{c_i\}$
17: **else if** $w(m_i) < w(m_{\text{current}})$ **then**
18: $L = L \backslash \{c_i\}$
19: **end if**
20: **end for**
21: **end while**

7.1.3 COPIA

Recall the interpretation of the Optimal Mix Problem as a Combined Optimization Problem is used (see 3.11 and 7.1), where A is the set of all possible iSS mixes, B the set of all possible SCG mixes and C the set of all possible CB mixes:

The Optimal Mix Combined Problem (OMCP)

Let N be a network. Find m_{opt} in $A_N \times B_N \times C_N$ such that

$$w(m_{\text{opt}}) = \max_{(a,b,c) \in A_N \times B_N \times C_N} w(a,b,c) \tag{7.1}$$

OMCP gave inspiration to an algorithm that optimizes over one component type at the time. For example, start with the empty mix and optimize over SCG to find the optimal SCG mix. Next, optimize over iSS, using the recently found SCG mix as base mix, to find the optimal mix with the same SCGs as before and newly placed iSSs. Then optimize over CB to find the optimal mix with with CBs added. Formally this example looks as follows:

$$\text{Find } b^* \text{ in } B \text{ such that } w(\mathbf{0}, b^*, \mathbf{0}) = \max_{b \in B} w(\mathbf{0}, b, \mathbf{0}) \tag{7.2}$$

$$\text{Next, find } a^* \text{ in } A \text{ such that } w(a^*, b^*, \mathbf{0}) = \max_{a \in A} w(a, b^*, \mathbf{0}) \tag{7.3}$$

$$\text{Next, find } c^* \text{ in } C \text{ such that } w(a^*, b^*, c^*) = \max_{c \in C} w(a^*, b^*, c) \tag{7.4}$$

This form of optimizing in a certain order of objectives is called a hierarchical approach. For OMP, this idea has been brought into algorithmic form in Algorithm 4, named the Combined Optimization Problem Inspired Algorithm (COPIA). Three aspects of this algorithm should first be explained in order to understand the algorithm: `ORDER`, frequency f and *type.method*.

To start, there is the term `ORDER`, which is a list of certain length containing the three component types: $\texttt{ORDER} \in \cup_{l \in \mathbb{N}} \{SCG, iSS, CB\}^l$. `ORDER` represents the order in which the optimization steps happen. If $\texttt{ORDER} = [SCG, CB]$, then the algorithm will first optimize over the type SCG, and then over the type CB with the output of the optimization of SCG as base. Note that it is allowed to optimize over each type as often as preferred.

The frequency f simply is the number of times that `ORDER` is iterated over. f is superfluous in the sense that for $\texttt{ORDER}^* \in \cup_{l \in \mathbb{N}} \{SCG, iSS, CB\}^l$, it is also possible to use $\texttt{ORDER} = [\texttt{ORDER}^*, \texttt{ORDER}^*]$. Still, f is a nice addition when trying out many optimization steps in the same order, to see how the mix develops while optimizing for multiple rounds.

As third, there is the `TYPE.method`. This defines the method that COPIA should use for the optimization per type. So far only a beam search method and a greedy method have been described, and these are both an option for the optimization type. For now, it is hard to specify more options, since it is not clear yet which optimization methods will be implemented and which ones are useful. Therefore, in Algorithm 4 `GREEDY` and `BEAM` are specified as options for the optimization over a type, but this is subject to change based on the performances of other methods.

To finish the algorithm, some more needs to be specified. As all algorithms, the suitable component locations need to be determined and the empty mix needs to be evaluated. Because COPIA can make use of the `GREEDY` and `BEAM` method, similar preparation needs to happen: determine suitable combinations

of sizes (i_1, i_2, i_3) and evaluate these combinations. Also, as input the beam options n should be specified. Next, when optimization over one type commences, m_{current} should be cleared of all present components of that type, since the methods given so far are constructive and can only add new components. An 'empty mix of that type' is the starting point to be used for the constructive algorithms in COPIA. The current mix with components of one type removed is represented by $m_{\text{current}} - \texttt{type}$. Since $m_{\text{current}} - \texttt{type} \neq m_{\text{current}}$ whenever a component of the corresponding type is in m_{current}, $m_{\text{current}} - \texttt{type}$ must be evaluated before continuing.

With all of the above, COPIA can be fully defined. See Algorithm 4 for details. It is the third algorithm that is implemented.

Algorithm 4 Combined Optimization Problem Inspired Algorithm (COPIA)

Input: `DATA`, `ORDER`, FREQUENCY f, `iSS.method` $\in$ {`GREEDY`,`BEAM`}, `SCG.method` $\in$ {`GREEDY`,`BEAM`}, `CB.method` $\in$ {`GREEDY`,`BEAM`}, iSS combination size i_1, SCG combination size i_2, CB combination size i_3, beam options n

Output: mix m_{current}

1: Determine suitable component locations L_{iSS}, L_{SCG} and L_{CB}
2: Determine suitable combinations C_{iSS}, C_{SCG} and C_{CB} with maximum combination sizes (i_1, i_2, i_3) respectively
3: Evaluate empty mix m_0
4: $m_{\text{current}} = m_0$
5: Evaluate suitable combinations c in $C_{\text{iSS}} \cup C_{\text{SCG}} \cup C_{\text{CB}}$
6: Sort suitable combinations on profit in list L
7: **for** i is 1 to f **do**
8: **for** `type`$\in$ `ORDER` **do**
9: $m_{\text{current}} = m_{\text{current}} - \texttt{type}$
10: Evaluate m_{current}
11: Optimize m_{current} over `type.method`
12: Update m_{current}
13: **end for**
14: **end for**

7.1.4 Brute Force Estimator

The Brute Force Estimator is an imitation of the current model using a much faster estimator instead of the Malfunction Simulation Model. Even though the estimator is much faster, some extra preparation is necessary to make this estimator functional. First, all suitable component locations and combinations are determined. Next, MSM is used to evaluate all combinations per type separately. Then the imitation of the current model starts: all iSS mixes are evaluated using the estimator, a selection on these iSS mixes is done, and all

mixes containing one of the selected iSS mixes are evaluated using the estimator again. The output is a list of evaluated mixes.
An important part to the Brute Force Estimator is the functioning of the Estimator. As seen before, the difficult part of the evaluation of a mix is the determining of the expected CML reduction:

$$v_{red}(m) = v(m_0) - v(m) \tag{7.5}$$
$$= \sum_{s \in S} \tilde{v}(m_0, s) \cdot p_s - \sum_{s \in S} \tilde{v}(m, s) \cdot p_s \tag{7.6}$$

Therefore, the goal is to estimate this. Since the fail frequency p_s per section $s \in S$ is given, this boils down to estimating the CML per section for a mix. In other words, find a function $\tilde{v}^{\approx} : M \times S \to \mathbb{R}$ such that $\tilde{v}^{\approx}(m, s)$ estimates $\tilde{v}(m, s)$. The proposed method of doing this is the following:

$$\tilde{v}^{\approx}(m, s) = \min_{c \in C_m} \tilde{v}(c, s) \tag{7.7}$$

such that

$$v^{\approx}(m) = \sum_{s \in S} p_s \cdot \tilde{v}^{\approx}(m, s) \tag{7.8}$$
$$= \sum_{s \in S} p_s \cdot \min_{c \in C_m} \tilde{v}(c, s) \tag{7.9}$$

Here, C_m is the set of all present evaluated combinations c in mix m. In order to obtain C_m, it is necessary to evaluate the list of combinations, similar to previous algorithms in the chapter, and it is necessary to check what combinations are present in the mix that is estimated. This idea requires a number of iterations of the Malfunction Simulation Model to start with, and does not need the Malfunction Simulation Model anymore afterwards. Once the combinations have been evaluated, it is expected to be faster then the Malfunction Simulation Model. The notation for the estimator of the expected CML reduction will then be:

$$v^{\approx}_{red}(m) = v(m_0) - v^{\approx}(m) \tag{7.10}$$
$$= v(m_0) - \sum_{s \in S} p_s \cdot \tilde{v}^{\approx}(m, s) \tag{7.11}$$
$$= v(m_0) - \sum_{s \in S} p_s \cdot \min_{c \in C_m} \tilde{v}(c, s) \tag{7.12}$$

The estimator works accurately when a mix exists of combinations that affect different power outages originating in different sections. For example, when two SCG combinations lie in different routes, then there are no power outages that are affected by both of these combinations. In other words, the combinations

function independently, and therefore taking the minimum is a correct way of determining the expected CML for the mix.

The estimator works less accurately when a mix consists of combinations that do affect equal power outages. For example, when a CB lies on a section s and a SCG contains s in its trajectory, then a malfunction in section s would activate the CB and the SCG immediately tells the headquarters where the malfunction is located exactly. The CML for this power outage is clearly less then taking the minimum of these two components separately: the CB caused a smaller number of people experiencing a power outage and the SCG caused an immediate correct diagnose of where the malfunction occurred. Both actions helped with reducing the number of CML, even when the other component was already present. Taking the minimum over the present combinations therefore is an overestimation of the CML.

Because of the inaccurate behaviour of $\tilde{v}^{\approx}$, an algorithm is designed around this estimator with the following hypothesis: the estimator can distinguish good mixes from bad mixes. This hypothesis is tested by mimicking the current model with the estimator, and afterwards checking the output by evaluating the top 50 mixes by applying the Malfunction Simulation Model again. See Algorithm 5 for details. The Brute Force Estimator is the fourth algorithm to implement.

Algorithm 5 Brute Force Estimator

Input: `DATA`
Output: mix m_{current}

1: Determine suitable component locations L_{iSS}, L_{SCG} and L_{CB}
2: Determine suitable combinations C_{iSS}, C_{SCG} and C_{CB} with maximum combination sizes (i_1, i_2, i_3) respectively
3: Evaluate empty mix m_0
4: $m_{\text{current}} = m_0$
5: Evaluate suitable combinations c in $C_{\text{iSS}} \cup C_{\text{SCG}} \cup C_{\text{CB}}$
6: Generate all suitable mixes $(M^* = A^* \times B^* \times C^*)$
7: Evaluate iSS mixes (A^*) using Formula 7.10 resulting in A^*_{EV}
8: Select iSS mixes with potential from A^*_{EV} resulting in A^*_{SEL}
9: Evaluate mixes m in $M^*_{\text{SEL}} = A^*_{\text{SEL}} \times B^* \times C^*$ using Formula 7.10 resulting in M^*_{EV}
10: Evaluate top 50 from M^*_{EV} using MSM
11: $m_{\text{current}} = m^*$ such that $w(m^*) = \max_{m \in M^*_{(\text{EV},\text{top50})}} w(m)$

7.1.5 Genetic Algorithm

The final two algorithms in this chapter are the evolutionary algorithms. Both are not implemented in the end, because the other algorithms already performed well enough. They are still mentioned however, since they provide more insight in how heuristic methods can be utilized to solve OMP.

The first evolutionary method to consider is the Genetic Algorithm. As the name suggests, this algorithm is based on genetics. It uses four main concepts: population, recombination, mutation and selection. The population is a set of solutions that goes through the process of recombination, mutation and selection. Recombination operators combine two solutions from the population to create a new solution, the "child" solution. Mutation operators "mutate" a solution from the population to create a new solution. Finally, a selection strategy is used to select solutions from the population and the new solutions, and this selection becomes the new population. By applying correct recombination and mutation operators, and by using a good selection strategy, it is hoped to find solutions in the population that approximate the optimal solution well.

For this project, the algorithm will be based on the General Scheme for a Genetic Algorithm from the Master Math course Heuristic Methods in Operations Research taught in the academic year '18/'19 by prof J. Hurink and M. Schutten ([3]):

Things that should be defined to make this scheme applicable to the Opti-

Algorithm 6 General Scheme for Genetic Algorithm

Input: Population size k, new solutions size k', local optimization algorithm A
Output: set S containing final survivors

1: Generate a population of k starting solution $S = \{s_1, \ldots, s_k\}$
2: Apply a given local optimization algorithm A to each $s \in S$
3: Replace each $s \in S$ by its resulting local optimum
4: **while** `!STOP_CRITERIA` **do**
5: Select k' distinct subsets $S_1, \ldots, S'_k$ of S of size 1 or 2
6: **for** S_j with $|S_j| = 1$ **do**
7: Apply a mutation operator to S_j
8: **end for**
9: **for** S_j with $|S_j| = 2$ **do**
10: Apply a recombination operator to S_j
11: **end for**
12: Apply algorithm A to each solution achieved by the previous two steps
13: Let S' be the resulting set of solutions
14: Use a selection strategy to chose k survivors from $S \cup S'$
15: Let S be the set containing the survivors
16: **end while**

mal Mix Problem are the local optimization algorithm A, a strategy to picking subsets, mutations operators, recombination operators, and a survivor strategy. The following describes the specifications for the algorithm as it should be tested initially. Everything is subject to change based on the performance of the algorithm during the implementation and testing phase.

Local Optimization Algorithm A

The method that will function as the local optimization algorithm A will depend on how well the other algorithms function. Some form of the Greedy Algorithm, Beam Search, COPIA, or even the Estimator (see Section 7.1.4) could be used here. It is also an option to leave out the local optimization on solutions, and just work with the solutions. This decision shall be made during the implementation process in the next chapter.

Subset Picking Strategy

The goal here is to pick k' subsets of size 1 or 2. Let k'_1 be the number of subsets of size 1, let k'_2 be the number of subsets of size 2 and $r \in [\frac{1}{2}, \frac{3}{2}]$. Then

$$k'_1 = \tfrac{1}{2} \cdot r \cdot k' \tag{7.13}$$

$$k'_2 = \tfrac{1}{2} \cdot (2 - r) \cdot k' \tag{7.14}$$

$$k'_1 + k'_2 = \tfrac{1}{2} \cdot (r + (2 - r)) \cdot k' \tag{7.15}$$

$$= k' \tag{7.16}$$

The picking strategy takes a random $r \in [\frac{1}{2}, \frac{3}{2}]$ each round. When picking subsets of size 1, take the top $\max(3, k'_1)$ mixes in the current population S and pick the other mixes of sizes 1 randomly from S. When picking subsets of size 2, generate three random numbers $r_1, r_2, r_3 \in \mathbb{N}$ such that $\sum_{i=1}^{3} r_i = k'_2$. Then pick r_1 combinations $(s_1, s_2) \in S \times S$ such that $h(s_1, s_2)$ are the r_1 greatest Hamming distances, where h is the Hamming Distance. Then pick r_2 combinations consisting of two of the $2 \cdot r_2$ solutions with the greatest profits, so if $w(s)$ is in the top $2 \cdot r_2$ values of all solutions, the s could be in one of these r_2 combinations. Pick the final r_3 combinations randomly.

Mutation Operators

Mutation Operators for the Optimal Mix Problem are operators on a single solution, so on a single mix. Below is a list of possible operators. Each operator is described by giving a set, a neighborhood of the solution to mutate. The idea is to pick a random operator for each mutation in the algorithm from this list, and pick one solution from the set that defines the operator. It is suspected that these operators work well on the basis of the insights from Section 6.2.

Note that some of the operators are not always applicable to a certain solution, because a certain action can not be made in that solution, or because certain information, such as the profit per loose component, is not available. For these cases it is assumed that a different operator is picked. Also note that the list of operators can change based on performance of the operators and other algorithms during the implementation and testing process.

1. $\mathcal{N}_{\texttt{iSS-CB-crossing}} = \{\text{Replace an iSS on a crossing for CBs}\}$

2. $\mathcal{N}_{\texttt{CB-iSS-crossing}} = \{\text{Replace one or multiple CBs on one crossing for one iSS}\}$

3. $\mathcal{N}_{\texttt{move CB}}$ = {Move a CB in a certain direction on the full trails it lies on}
4. $\mathcal{N}_{\texttt{random remove}}$ = {Remove a random number of components}
5. $\mathcal{N}_{\texttt{CBiSS-SCG}}$ = {Remove all CBs and iSSs on the trajectory of a specific SCG and add the SCG}
6. $\mathcal{N}_{\texttt{full trail iSS-CB}}$ = {Remove all iSSs over a full trail and add 1 or 2 CBs to this trail}
7. $\mathcal{N}_{\texttt{critical section profit}}$ = {Add the component with greatest profit over the section with the highest currently expected CML}
8. $\mathcal{N}_{\texttt{random section profit}}$ = {Pick a section randomly and add the component with greatest profit over this section}
9. $\mathcal{N}_{\texttt{random route profit}}$ = {Pick a route randomly and add the component with greatest profit over this route}
10. $\mathcal{N}_{\texttt{double}}$ = {Remove a component that influences another component}

Recombination Operators

Recombination Operators for the Optimal Mix Problem are operators on combinations of solutions, so on two mixes. Below is a list of the possible operators, for which the same notions hold as for the Mutation Operators list. So again, based on the insights from Section 6.2 it is suspected that these operators work well.

1. $\mathcal{N}_{\texttt{combine routes}}$ = {Take the best mix per route from the two parent solutions}
2. $\mathcal{N}_{\texttt{random combine strong weak}}$ = {Take the best mix from the two parents and add a random number of components from the other weaker mix}
3. $\mathcal{N}_{\texttt{sections combine strong weak}}$ = {Take the best mix from the two parents and add components from the weaker mix on sections that are influenced by the weaker mix but not by the strong mix}
4. $\mathcal{N}_{\texttt{combine SCG route}}$ = {Take the best mix from the two parents, remove its SCGs and add the SCG mix per route from best performing parent on that route}
5. $\mathcal{N}_{\texttt{combine iSS route}}$ = {Take the best mix from the two parents, remove its iSSs and add the best iSS mix per route from best performing parent on that route}
6. $\mathcal{N}_{\texttt{combine CB route}}$ = {Take the best mix from the two parents, remove its CBs and add the best CB mix per route from best performing parent on that route}

Survivor Strategy

The survivor strategy needs to pick k solutions from $S \cup S'$, where S is the initial population of the start of the round and S' is the set of new solution created by mutation and recombination. Initially, this is done as follows. First, pick the $\frac{3}{4} \cdot k$ mixes from $S \cup S'$ with highest profit. Next, pick the $\frac{1}{4} \cdot k$ mixes with highest profit-component ratio $\frac{w(s)}{h(s)}$. If these mixes overlap, there are fewer than k solutions picked. Fix this by checking the solution that have not been picked yet for expected CML performance on route level for a random route, and pick the solution with the highest CML reduction on this route. Repeat until k solutions have been picked.

7.1.6 Particle Swarm Optimization

The final algorithm to mention in this section Particle Swarm Optimization (PSO), an evolutionary algorithm based on the movement of a flock of birds that want to move optimally in the sky. Similar to Genetic Algorithms, it makes use of a population, recombination and, in the application for PSO, a form of mutation. As said earlier, PSO is not implemented, but it is still mentioned to give an impression of how non-constructive heuristic methods can be utilized for OMP.

Conceptually the PSO method functions as follows. The algorithm starts with a family of particles of size n. The particles will fly through the solution space, searching for the optimal solution. The algorithm is initiated, say at iteration $t = 0$, by randomly placing all particles in the solution space. At $t = 1$ particles start moving towards the best solution of all particles in $t = 0$ with a certain random factor to decide how big this step is. In the next steps, each particle p knows the following: its location p_{current}, its best known location of all steps p_{best}, the best current location of all particles g_{best} and its speed from the previous step v_p. From now on, the speed of each particle for steps $t > 1$ will be a linear combination of v_p, $(p_{\text{best}} - p_{\text{current}})$ and $(g_{\text{best}} - p_{\text{current}})$, where the factors of each piece are partly taken randomly to not get caught in a local optimum too easily:

$$v_p = \omega_1 \cdot r_1 \cdot (p_{\text{best}} - p_{\text{current}}) \oplus \omega_2 \cdot r_2 \cdot (g_{\text{best}} - p_{\text{current}}) \oplus \omega_3 \cdot v_p \qquad (7.17)$$

where $r_1, r_2 \in [0, 1]$ randomly, $\omega_1, \omega_2, \omega_3 \in [0, 1]$ as input factors and $\oplus$ represents the bitwise OR operator. The algorithm terminates after a certain number of iterations.

To apply PSO to the Optimal Mix Problem, specific interpretations of speed and a factor of the speed are needed. The main reason for this is that the solution space for the Optimal Mix Problem is of the form $\{0, 1\}^k$, where $k = |L_{\text{iSS}}| + |L_{\text{SCG}}| + |L_{\text{CB}}|$ is the number of suitable component locations, and where each element $m \in \{0, 1\}^k$ is a mix with each 1 representing the presence and each 0 representing the absence of a specific component (see also sections 3.1.1 and 3.1.2). A continuously and scalably defined speed is not applicable to a discrete solution space. Here, a difference in location of the particles (and

difference of solutions in general) means at least one complete component in difference, since there is no such thing as half a component. Therefore, representing the speed of a particle p in the discrete binary solution space $\{0,1\}^k$ is done by giving a binary vector v_p of length k. If p_{current} is the current solution of particle p, then the solution visited by p in the next step is $p_{\text{current}} + v_p$, where $+$ represents addition in $\mathbb{Z}/2\mathbb{Z}$. In other words, v_p represents the components that should be removed or added in a current solution.
For the above interpretation of speed, the following interpretation of factors is used. Suppose $q \in [0,1]$. Then for a certain speed v_p of a particle p, the following equation holds:

$$h(q \cdot v_p) = \lceil q \cdot h(v_p) \rfloor \tag{7.18}$$

$$(q \cdot v_p)_i = 0 \text{ if } v_p = 0 \text{ for } 1 \leq i \leq k \tag{7.19}$$

$$(v_p)_i = 1 \text{ if } (q \cdot v_p)_i = 1 \text{ for } 1 \leq i \leq k \tag{7.20}$$

where $h : \{0,1\}^k \to \mathbb{N}$ is the Hamming distance. The above means that $q \cdot v_p$ is a selection of components to be changed in speed v_p, and the size of the selection is the natural number closest to the number $q \cdot h(v_p)$. The selection of components is done randomly, as long as components from v_p are chosen.
With the above, the Particle Swarm Optimization algorithm for the Optimal Mix Problem can be defined, see Algorithm 7 on the next page.

A problem of Algorithm 7 is the following: reusing the speed of a previous step implies immediately undoing certain changes made in that step. This holds since $v_p + v_p = \mathbf{0}$ for $v_p \in \{0,1\}^k$. To overcome this problem, but still utilize the particles speed from recent history, the actually added and removed components from the previous step are excluded from the new speed, and the speeds of two steps before are also included in determining the new speed. Let j represent the current iteration, r_1^j, r_2^j the random factors for iteration j, p^j the location in the solution space of particle p in iteration j and v_p^j the speed of particle p in iteration j. Then v_p^j is defined as follows:

$$v_p^{temp} = \sum_{l=j-3}^{j-1} \left(\omega_1 \cdot r_1^l \cdot (p_{\text{best}}^l - p^l) \oplus \omega_2 \cdot r_2^l \cdot (g_{\text{best}}^l - p^l) \oplus \omega_3 \cdot v_p^{l-1}\right) \tag{7.21}$$

$$(v_p^j)_i = \begin{cases} 1 & \text{if } v_p^{temp} = 1,\ v_p^{j-1} \neq 1,\ v_p^{j-2} \neq 1 \text{ and } v_p^{j-3} \neq 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.22}$$

where the sum operator in 7.21 also is the bitwise OR operator $\oplus$. Using this new insight, PSO for OMP can be defined as in Algorithm 8.

Algorithm 7 Particle Swarm Optimization for OMP

Input: `DATA`, particle family size n, factors $\omega_1, \omega_2, \omega_3$, max iterations $j_{\max}$
Output: mix g_{best}

Determine suitable component locations
$k = |\text{suitable component locations}|$
Evaluate m_0
Generate family P of n particles randomly with random position in $\{0,1\}^k$
Evaluate particles $p \in P$
$p_{\text{best}} = p^*$ such that $w(p^*) = \max_{p \in P} w(p)$
for $j \in [1, \ldots, j_{max}]$ **do**
    for particle $p \in P$ **do**
        if $w(p_{\text{current}}) > w(p_{\text{best}})$ **then**
            $p_{\text{best}} = p_{\text{current}}$
        end if
        if $w(p_{\text{current}}) > w(g_{\text{best}})$ **then**
            $g_{\text{best}} = p_{\text{current}}$
        end if
    end for
    for particle $p \in P$ **do**
        $v_p = \omega_1 \cdot r_1 \cdot (p_{\text{best}} - p_{\text{current}}) \oplus \omega_2 \cdot r_2 \cdot (g_{\text{best}} - p_{\text{current}}) \oplus \omega_3 \cdot v_p$
        $p_{\text{current}} = p_{\text{current}} + v_p$
    end for
end for

Algorithm 8 Particle Swarm Optimization for OMP

Input: DATA, particle family size n, factors $\omega_1, \omega_2, \omega_3$, max iterations $j_{\max}$
Output: mix g_{best}

Determine suitable component locations
$k = |\text{suitable component locations}|$
Evaluate m_0
Generate family P of n particles randomly with random position in $\{0,1\}^k$
Evaluate particles $p \in P$
$p^0_{\text{best}} = p^*$ such that $w(p^*) = \max_{p \in P} w(p)$
for $j \in [1, \ldots, j_{\max}]$ **do**
    for particle $p \in P$ **do**
        if $w(p^{j-1}) > w(p^{j-1}_{\text{best}})$ **then**
            $p^j_{\text{best}} = p^{j-1}$
        end if
        if $w(p^{j-1}) > w(g^{j-1}_{\text{best}})$ **then**
            $g^j_{\text{best}} = p^{j-1}$
        end if
    end for
    for particle $p \in P$ **do**
        $v_p^{\text{temp}} = \sum_{l=j-3}^{j-1} \left(\omega_1 \cdot r_1^l \cdot (p^l_{\text{best}} - p^l) \oplus \omega_2 \cdot r_2^l \cdot (g^l_{\text{best}} - p^l) \oplus \omega_3 \cdot v_p^{l-1}\right)$
        for $i \in [1, \ldots, k]$ **do**
            $(v_p^j)_i = \begin{cases} 1 & \text{if } v_p^{\text{temp}} = 1,\ v_p^{j-1} \neq 1,\ v_p^{j-2} \neq 1 \text{ and } v_p^{j-3} \neq 1 \\ 0 & \text{otherwise} \end{cases}$
        end for
        $p^j = p^{j-1} + v_p^j$
    end for
end for

7.2 Adaptations

During the implementation process additions to and adaptations on the algorithms were made, which are based on test results and are aimed at improving the algorithms. This section starts with describing general extras that are implemented for each constructive algorithm. Next, adaptations to the algorithms BSIA, COPIA and BFE are given. The section finishes with introducing a new algorithm, Remove Algorithm, that helps in improving solutions found by Greedy Algorithm, BSIA and COPIA. The Genetic Algorithm and Particle Swarm Optimization are not implemented in the end, because the other algorithms already performed sufficiently quality-wise and because it is suspected that to evolutionary would take a substantially longer time to terminate.

7.2.1 General Extras

Three general additions that aim to lower the running time are to be included in each algorithm, without influencing the functionality of the algorithm:

Expected CML boundary for adding components

The expected CML boundary has the goal to skip evaluations of which the outcome would be a rejected mix, based on the potential of a component for m_{current}, such that the running time is lowered. Each component has a cost. To be profitable when added, a component needs to reduce a minimum amount of expected CML, and this can be calculated. An expected CML reduction of 1 gives a profit of 20. Using this and the costs from Table 2.1, it is easy to determine the minimum expected CML to reduce for a component to be profitable:

$$\text{CB: } \frac{14000}{20} = 700 \tag{7.23}$$

$$\text{SCG: } \frac{30000}{20} = 1500 \tag{7.24}$$

$$\text{iSS: } \frac{72000}{20} = 3600 \tag{7.25}$$

Now, if a component is to be added to m_{current}, the total number of expected CML for m_{current} on routes that are influenced by the component should be higher than the numbers above. If it is not, adding the component will cause a loss. Therefore, there is no need to evaluate m_{current} with the added component any further, since it will be rejected anyway. Of course, this only holds for the algorithms using a greedy method.

In reality, when a route has an expected CML value of 3600, placing 1 iSS will never reduce this to exactly 0 expected CML. The moment a power outage happens, the expected CML for this route is already greater then 0, since an iSS needs at least 10 minutes to be utilized. In general, the expected CML value of a route, or even of a section, is not reducible to 0. Exceptions are sections on a secondary substation at a leaf of the route with no customers connected for example, but these exceptions are scarce.

The expected CML boundary to be used in the algorithms, will be extra input variables. This way, experimenting with the value of the boundary is possible. In total there will be 6 boundaries: two for each type, of which one is for a single component and of which one is for a duo of that type. The basic value for single components are the values above in formulas 7.23, 7.24 and 7.25, and for the duos are the double of those values.

Do note that the profit of superfluous CFPI are not taken into account with this boundary. This is to be kept in mind while testing, but is suspected to be negligible, since CFPI revenue generally only makes up for a small part of the total revenue.

Profit boundary for combinations to optimize

In Section 6.2 it is seen that the optimal mixes for HZN and ALR do not include any components that are not profitable when separately placed. This gave inspiration to a profit boundary. The profit boundary is designed to exclude components that do not meet a certain profit when placed separately. Excluding a component means that it shall not be included in the list L after the initial evaluations. By excluding these components, future evaluations are skipped, and the running time is lowered. It is suspected that when the profit boundary is set to 0, so when a loose component need to make a profit that is greater then 0 to be included, then the quality of the output of the algorithm remains the same for at least HZN and ALR.

The profit boundary is implemented in the algorithms such that a boundary can be given separately per component type for single components. Whenever a combination of size 2 contains a component that separately gives a loss, this combination is also excluded. The exception again is an iSS duo, since iSSs behave differently. Therefore, a fourth boundary option is given exclusively for iSS duos.

The profit boundaries are variables to the algorithm (and not always set to 0), since it can be interesting to experiment with. Raising the boundary to 1000 could lower the final approximation, but may lower the running time even more significantly.

Note that the boundary can only be applied when an algorithm first evaluates loose combinations before starting to build a mix.

Filter on top combinations per route per type

Filtering on top combinations per route per type has the goal to lower the running time, again while skipping evaluations by excluding components without great potential. The idea is to select only the top x, y and z combinations of iSS, CB and SCG respectively, after the initial evaluation of loose combinations. This selection will then form the list L to iterate over in the constructive algorithms.

The results of the top combinations filter are suspected to drop a bit in quality, since a component can be included in an optimal mix, without being in the top combinations of that type. On the other hand, this option allows the possibility to find relatively good mixes in way shorter running time when working with a otherwise large list L.

The option to only include top combinations has x, y and z as input to the algorithm.

Note that the above additions can be turned off by giving $-\infty$ or ∞ as input, depending on the addition. Another note is that testing these options had a lower priority, and due to a lack of time, not all have been tested. In the end, only the basic version of the expected CML boundary and the profit boundary

set to 0 where used in the tests.

7.2.2 Adaptations to Implemented Algorithms

The algorithms that were implemented in the end are the Greedy Algorithm, BSIA, COPIA and BFE. After implementing Greedy Algorithm, BSIA and COPIA, it showed that these performed very well, with results higher than 90% of the optimum. Since Evolutionary Algorithms are known to have a substantially longer running time, the choice was made to leave out these algorithms and focus on improving the implemented algorithms. How well Greedy Algorithm, BSIA, COPIA and BFE perform exactly, is shown in Chapter 8. In the process, the Greedy Algorithm (Algorithm 2) was implemented and not altered. BSIA, COPIA and BFE however gained options that are aimed at improving the functionality of these methods.

BSIA

The adaptation in BSIA allows more combinations to be added, and therefore more mixes to be generated and evaluated, in the phase where the next level of the search tree is created. From the results of the first BSIA runs, it was seen that a level of the search tree often consisted of mixes created by adding combinations of the same type, simply because the top n combinations all were of precisely one type. This resulted in many of one component type added, while not necessarily being a good mix when comparing to the results of the current Optimal Mix Model. To try to overcome this, an option is implemented to force combinations of the missing types into the next level on the search tree, such that the addition of a combination from each type is tested. This can be forced based on previous additions to the mix or not, as follows:

- `force-other-types` is a boolean. When set to `TRUE`, combinations of the two missing types are included in the next level whenever the n initially selected combinations are all of the same type.
- `ignore-previous-combination` is a boolean. When set to `TRUE`, and if `force-other-types`==`TRUE`, combinations of the missing type(s) are included in the next level.

To make the above functional, an `IF`-statement is added to the original BSIA (Algorithm 3) with the condition:

```
force-other-types AND (length(types.of(combinations.added) == 1
                OR ignore-previous-combination)
```

which is exactly `TRUE` when missing types should be added, based on the explanation above. The updates BSIA becomes Algorithm 9.

Algorithm 9 Updated BSIA

Input: `DATA`, iSS combination size i_1, SCG combination size i_2, CB combination size i_3, beam options n, `force-other-types`, `ignore-previous-combination`

Output: mix m_{current}

Determine suitable component locations L_{iSS}, L_{SCG} and L_{CB}
Determine suitable combinations C_{iSS}, C_{SCG} and C_{CB} with maximum combination sizes (i_1, i_2, i_3) respectively
Evaluate empty mix m_0
$m_{\text{current}} = m_0$
Evaluate suitable combinations c in $C_{\text{iSS}} \cup C_{\text{SCG}} \cup C_{\text{CB}}$
Sort suitable combinations on profit in list L
while L is not empty **do**
    `route` is route of best combination c_{best} in L
    Pick n top combinations $c_1, \ldots, c_n$ on `route` from L
    $n' = n$
    Add $c_1, \ldots, c_{n'}$ to m_{current} resulting in mixes $m_1, \ldots, m_{n'}$
    if `force-other-types AND (length(types.of(combinations.added) == 1| ignore-previous-combination` **then**
        Pick n'' top 1 combinations on `route` of missing types ($n'' \in \{0, 1, 2\}$)
        Add n'' combinations to m_{current} resulting in total number of mixes $m_1, \ldots, m_{n'+n''}$
        $n' = n''$
    end if
    Evaluate $m_1, \ldots, m_{n'}$
    $W = \max_{i \in \{1,\ldots,n'\}}(m_i)$
    for i in 1 to n' **do**
        if $w(m_i)$ is equal to W and $W > w(m_{\text{current}})$ **then**
            $m_{\text{current}} = m_i$
            $L = L \backslash \{c_i\}$
        else if $w(m_i) < w(m_{\text{current}})$ **then**
            $L = L \backslash \{c_i\}$
        end if
    end for
end while

COPIA

One addition is done for COPIA specifically. The option is added to evaluate combinations in between instead of at the start, by adding a boolean `evaluate-beforehand` to the input. If `evaluate-beforehand==TRUE`, then combinations are evaluated beforehand, as happens in the Algorithm 9. If `!evaluate-beforehand==TRUE`, then combinations of a certain type are only evaluated when optimization over that type starts. This makes for a more precise list of combinations of that type, but also increases the running time when one type occurs more than once in `ORDER` or when frequency $f > 1$.
The above specifications for COPIA result in Algorithm 10.

Algorithm 10 Updated COPIA

Input: `DATA`, `ORDER`, FREQUENCY f, `iSS.method`$\in$ {`GREEDY`,`BEAM`}, `SCG.method`$\in$ {`GREEDY`,`BEAM`}, `CB.method`$\in$ {`GREEDY`,`BEAM`}, iSS combination size i_1, SCG combination size i_2, CB combination size i_3, beam options n, `evaluate-beforehand`

Output: mix m_{current}

Determine suitable component locations L_{iSS}, L_{SCG} and L_{CB}
Determine suitable combinations C_{iSS}, C_{SCG} and C_{CB} with maximum combination sizes (i_1, i_2, i_3) respectively
Evaluate empty mix m_0
$m_{\text{current}} = m_0$
if `evaluate-beforehand` **then**
    Evaluate suitable combinations c in $C_{\text{iSS}} \cup C_{\text{SCG}} \cup C_{\text{CB}}$
    Sort suitable combinations on profit in list L
end if
for i is 1 to f **do**
    for `type` $\in$ `ORDER` **do**
        $m_{\text{current}} = m_{\text{current}}$ − `type`
        Evaluate m_{current}
        if `!evaluate-beforehand` **then**
            Evaluate suitable combinations of type `type`
            Sort suitable combinations type `type` on profit in list L
        end if
        Optimize m_{current} over `type.method`
        Update m_{current}
    end for
end for

BFE

One simple suggestion is done to improve BFE, and that is to allow the SCG and CB combination sizes to be 1. This is suggested, because Greedy Algorithm and

BSIA performed well with combinations of only size 1, and because the estimator performed well on SCG exclusive and CB exclusive mixes based on combination sizes 1. It is expected that the estimator still performs well on the mixes of all types, when the sizes for SCG and CB becomes 1. Because some restrictions hold for iSS duos and singles, it is still necessary to evaluate all combinations of size 2 for iSS. The updated version becomes Algorithm 11.

Algorithm 11 Updated Brute Force Estimator

Input: `DATA`, SCG combination size i_2, CB combination size i_3
Output: mix m_{current}

1: Determine suitable component locations L_{iSS}, L_{SCG} and L_{CB}
2: Determine suitable combinations C_{iSS}, C_{SCG} and C_{CB} with maximum combination sizes $(2, i_2, i_3)$ respectively
3: Evaluate empty mix m_0
4: $m_{\text{current}} = m_0$
5: Evaluate suitable combinations c in $C_{\text{iSS}} \cup C_{\text{SCG}} \cup C_{\text{CB}}$
6: Generate all suitable mixes ($M^* = A^* \times B^* \times C^*$)
7: Evaluate iSS mixes (A^*) using Formula 7.10 resulting in A^*_{EV}
8: Select iSS mixes with potential from A^*_{EV} resulting in A^*_{SEL}
9: Evaluate mixes m in $M^*_{\text{SEL}} = A^*_{\text{SEL}} \times B^* \times C^*$ using Formula 7.10 resulting in M^*_{EV}
10: Evaluate top 50 from M^*_{EV} using MSM
11: $m_{\text{current}} = m^*$ such that $w(m^*) = \max_{m \in M^*_{(\text{EV}, \top 50)}} w(m)$

7.2.3 Remove Algorithm

Finally, an extra algorithm is suggested based on the performance of the implemented constructive algorithms. The constructive algorithms all use greedy methods, and sometimes this results in too many components that are contained in the final mix. It could be profitable to remove some of these components. Figure 7.1 shows this idea in a simplistic example where removing a component is profitable.

The goal for the Remove Algorithm is to find the optimal submix that is contained in the final mix. The method first removes each component in the mix, noted by $c \in m$, separately and evaluates the resulting mix $m_c = m - c$ per component. All mixes m_c are sorted in a list L based on profit. The removal that brings the greatest profit greater than 0 is chosen to be the new mix. This new mix only differs from the original mix on the routes where the removed component was influential. To update the removal list, re-evaluate the removals of components on the relevant routes, and repeat the process. Do this until no removals are profitable anymore. This is represented in Algorithm 12, that is also implemented and mentioned in the next chapter.

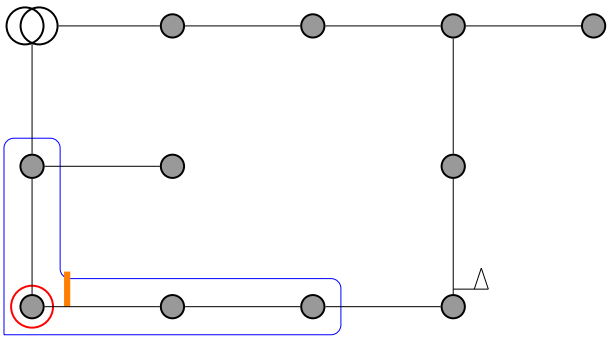


Figure 7.1: Example: suppose $w(m_{\text{iSS}}) = 9000$, $w(m_{\text{SCG}}) = 8000$, $w(m_{\text{CB}}) = 6000$, $w(m_{\text{iSS+SCG+CB}}) = 11000$, $w(m_{\text{iSS+SCG}}) = 10000$ and $w(m_{\text{SCG+CB}}) = 12000$. Components are added in the order iSS, SCG, CB in the Greedy Algorithm because all additions gave a bit more profit ($w(m_{\text{iSS+SCG+CB}}) \geq w(m_{\text{iSS+SCG}}) \geq w(m_{\text{iSS}})$), so $m_{\text{iSS+SCG+CB}}$ is the final mix. But $m_{\text{SCG+CB}}$ is more profitable. Removing the iSS is now profitable.

Algorithm 12 Remove Algorithm

Input: mix m
Output: mix m with components removed

for $c \in m$ **do**
    $m_c = m - c$
    Evaluate m_c
    Add result to list L
end for
while `TRUE` **do**
    $W_{c^*} = \max_{c \in m} w(m_c)$
    if $W_{c^*} > w(m)$ **then**
        $m = m_{c^*}$
    else
        `BREAK`
    end if
    Remove m_{c^*} from L
    `routes` $= c^*$.`routes`
    Re-evaluate m_c for c on `routes`
    Update L
end while

Chapter 8

Results

This chapter is dedicated to presenting the results from the implemented algorithms. First, an explanation is given on what method is used to interpret the results. Secondly, the two test substations Huizen (HZN) and Almere (ALR) are introduced and the results of multiple applications of the algorithms on HZN and ALR are presented. Finally, a third substation Vaassen (VSN) is introduced, and is used for validation of the instances of algorithms that performed well on HZN and ALR.

8.1 Method

Two aspects are needed for evaluating the performance of the heuristic methods: profit of the best mix and running time of the method. Profit is determined per mix and can easily be compared to the profit of the mix found by the current Optimal Mix Model. To compare the running time of the current model with one of the approximation algorithms, more work had to be done. Down below two aspects of camparison of the improvement on the running time are explained: the general comparison of running times and the running time improvement caused by the application of the profit boundary.

8.1.1 Comparing Running Times

The calculations of the current model have been done on 16-core computers where the model used parallel programming. This project mainly makes use of a 4-core computer without parallel programming. Therefore, comparing the results in real time is not sensible. Note that all algorithms are suitable to be programmed parallel, but that even then the comparison can not be made in real time, since the number of cores differs.

To make a sensible estimate, the running time for the current model and for the algorithms is taken by looking at the number of times that the Malfunction

Simulation Model is called upon. In the results this is referred to by 'iterations'. The Malfunction Simulation Model is called upon each time an evaluation needs to be done. The number of iterations increases by one per simulated malfunction. The number of simulated malfunctions is equal to the number of sections that is simulated for a certain evaluation (see also Section 4.2).

The estimate is seen as sensible, because the Malfunction Simulation Model is the time consuming part of both the current Optimal Mix Model and the suggested heuristic algorithms. It is important to note that counting iterations only makes sense for comparing results on the same substation, since the length of one iteration in real time differs per substation.

8.1.2 Profit Boundary

The profit boundary, as explained in Section 7.2.1, is used in determining the speed up factor of the heuristic methods. From now on, the profit boundary is noted as $\mathcal{P}$ and $\mathcal{P} = x$ means that the profit boundary is set to x. The results for $\mathcal{P} > -\infty$ are not results from applying $\mathcal{P} > -\infty$ during the algorithms, but by reasoning what the improvement would be afterwards. This choice has been made because of two reasons.

First, there is a limited amount of time to do the tests for this thesis. Since tests take multiple hours up to days to terminate, it is beneficial to test settings for the algorithms that generate strictly different outcomes. Setting $\mathcal{P}$ higher than $-\infty$ and lower or equal to 0 is suspected to result in giving a submix of the mix where profit boundary equal to $-\infty$ is used, so the result would not be strictly different. This is suspected because the optimal mixes on HZN and ALR do not contain a loss giving component. Secondly, empowering the first argument, it is quite easy to determine a significant part of the effect of $0 \geq \mathcal{P} > -\infty$ on the running time after a run has been completed with $\mathcal{P} = -\infty$. To clarify this, see Figure 8.1 as an example. Here the profit over number of iterations is shown for the run of the Greedy Algorithm on HZN using $i_1 = i_2 = i_3 = 2$ as maximum combinations sizes. This is a typical shape of the progression of the profit of the Greedy Algorithm, BSIA and each optimization step per type for COPIA, and therefore makes a good example. As seen in the figure, the profit does not increase any more from a certain moment. Since all runs keep track of what combination is tested at what point in the algorithm, it can be determined at what point a certain combination with a certain profit is tested. In particular, it can be checked at what iteration a combination is tested that lies closest to and is greater than a certain profit. If that certain profit is $\mathcal{P}$, then all iterations after this combination can be 'cut off'. In Figure 8.1, this means that we simply search the point where the profit boundary lies, and take the profit at that point.

Do note that the above only holds whenever $\mathcal{P} \leq 0$ and when no loss-giving components are added in the constructive algorithms, similar to the optimal mixes on HZN and ALR (see Section 6.2).

It also needs to be said that actually applying $\mathcal{P} > -\infty$ is more effective

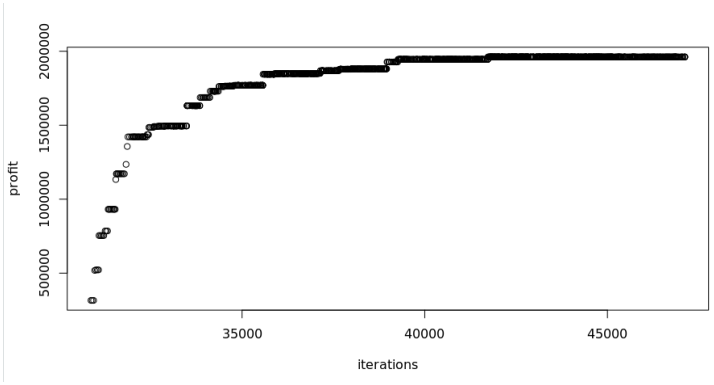


Figure 8.1: Iterations profit graph for Greedy Algorithm on HZN with maximum combinations sizes $i_1 = i_2 = i_3 = 2$

than as described here. As described in Section 7.2.1, the profit boundary also excludes all combinations containing a loss-giving component. This means that a profitable combination of a very profitable component and a loss-giving component is also excluded, but these combinations are not taken into consideration in the approach here. Therefore, the determining of the improvement of running time with applying $0 \geq \mathcal{P} > -\infty$ is an underestimation, and the improvement can even be greater.

Using the above method to compare running times and the method to determine the minimum improvement on the running time when applying $\mathcal{P}$, the performance of the algorithms can be evaluated.

8.2 Testing on ALR and HZN

Almere (ALR) and Huizen (HZN) are the networks used initially to see how well the implemented algorithms perform. Their relevant properties for testing are found in Table 8.1. HZN has a remotely controllable substation ($u_1 \in F_{\text{iSS}}$), and ALR has a non-remotely controllable substation. Furthermore, ALR is of a larger size than HZN. Last important note for these substations is that $F_{\text{SCG}} = \emptyset$, $F_{\text{CB}} \neq \emptyset$ and no iSSs are present on both of the networks, except for the remotely controllable substation u_1 in HZN.

Substation	Profit	(minimum) iterations	$u_1 \in F_{\text{iSS}}$	$\lvert S \rvert$
ALR	2772081	1266942	no	393
HZN	2030316	374164	yes	172

Table 8.1: Properties of ALR and HZN

Minimum iterations are listed in Table 8.1 because the current model does not count iterations, and counting iterations is not as easy as simply counting all mixes and multiplying this with the number of sections in the substation. It is possible however to determine the minimum number of iterations. It is certain that at least this number of iterations has occurred.

On ALR and HZN tests have been done with the Greedy Algorithm, BSIA, COPIA and BFE. Greedy Algorithm, BSIA and COPIA have been tested with different settings of the variables, with either the $\mathcal{P}$ set to $-\infty$ or to 0, and all with the basic CML boundaries in 7.23, 7.24 and 7.25. Results for Greedy Algorithm, BSIA and COPIA on ALR and HZN are in Section 8.2.1, 8.2.2 and 8.2.3 respectively. BFE has been run once on both ALR and HZN and details for that run are found in Section 8.2.4.

8.2.1 Greedy Algorithm

Tables 8.2 contains the results of the Greedy Algorithm on ALR with i_1, i_2 and i_3 the maximum combination size for iSS, SCG and CB respectively. The speed-up factor, denoted by $\mathcal{T}$, is calculated by dividing the number of iterations of the current model by the number of iterations of the run. The basic values for the expected CML boundary are used and the profit boundary is set to $-\infty$.

Greedy (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current model	2772081	100	1266942	1
$(1, 1, 1)$	2582138	93	18286	69
$(2, 2, 2)$	2389471	86	121142	10
$(2, 1, 1)$	2424666	87	62927	20

Table 8.2: Results Greedy Algorithm ALR

Greedy Algorithm exceeds the 90% mark once on ALR. The quality of the results is improved on after applying the Remove Algorithm. Table 8.3 contains the results of applying the Remove Algorithm to the results in Table 8.2. The results in Table 8.3 are further improved on when setting the profit boundary to 0, see Table 8.4.

Greedy + Remove (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current model	2772081	100	1266942	1
$(1, 1, 1)$	2646867	95	20507	62
$(2, 2, 2)$	2731417	99	125477	10
$(2, 1, 1)$	2735613	99	66202	19

Table 8.3: Results Greedy Algorithm + Remove ALR

Greedy + Remove + ($\mathcal{P} = 0$) (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current model	2772081	100	1266942	1
$(1, 1, 1)$	2646867	95	16692	76
$(2, 2, 2)$	2731417	99	104365	12
$(2, 1, 1)$	2735613	99	50106	25

Table 8.4: Results Greedy Algorithm + Remove + ($\mathcal{P} = 0$) ALR

Tables 8.5, 8.6 and 8.7 contain the results to similar Greedy Algorithm instances, but now for HZN. Most notably, Greedy Algorithm on HZN contains a result that exceeds the profit found by the Current Model. This is a re-occurring phenomenon for the results for HZN. It is explained by restrictions that were dropped for iSS placements for HZN, since HZN is a remotely controllable substation. This shall be further explained in the discussion in the next chapter.

Greedy (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
$(1, 1, 1)$	2051915	101	9234	41
$(2, 2, 2)$	1961452	97	47177	8
$(2, 1, 1)$	1836828	90	31483	12

Table 8.5: Results Greedy Algorithm HZN

Greedy + Remove (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
$(1, 1, 1)$	2124430	105	10820	35
$(2, 2, 2)$	2152129	106	49695	8
$(2, 1, 1)$	2088221	103	33953	11

Table 8.6: Results Greedy Algorithm + Remove HZN

Greedy + Remove + ($\mathcal{P} = 0$) (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
$(1, 1, 1)$	2124430	105	9341	40
$(2, 2, 2)$	2152129	106	44020	8
$(2, 1, 1)$	2088221	103	28830	13

Table 8.7: Results Greedy Algorithm + Remove + ($\mathcal{P} = 0$) HZN

8.2.2 BSIA

Tables 8.8 contains the results of BSIA for ALR with i_1, i_2 and i_3 the maximum combination size for iSS, SCG and CB respectively, n the number of options created for the next level in the search tree and `force` and `ignore`, short for `force-other-types` and `ignore-previous-combination`, the booleans for deciding whether more options should be added to the level of the search tree. No expected CML boundaries, profit boundaries or top combination selection is applied here.

BSIA [(i_1, i_2, i_3), n,`force`,`ignore`]	Profit	Profit %	Iterations	$\mathcal{T}$
Current model	2772081	100	1266942	1
[$(1, 1, 1)$, 5,`FALSE`,`FALSE`]	2635772	95	21969	58
[$(2, 2, 2)$, 5,`FALSE`,`FALSE`]	2532044	91	132435	10
[$(2, 1, 1)$, 5,`FALSE`,`FALSE`]	2624726	95	67247	18
[$(1, 1, 1)$, 3,`TRUE`,`FALSE`]	2503663	90	17766	71
[$(1, 1, 1)$, 4,`TRUE`,`FALSE`]	2666242	96	20943	60
[$(1, 1, 1)$, 3,`TRUE`,`TRUE`]	2628882	95	22304	57
[$(1, 1, 1)$, 4,`TRUE`,`TRUE`]	2617730	94	22966	55

Table 8.8: Results BSIA ALR

BSIA exceeds the 90% mark on all tests. Again, the quality of the results is improved on after applying the Remove Algorithm. Table 8.9 contains the results from applying the Remove Algorithm to the results in Table 8.8. The results in Table 8.9 are further improved on when setting the profit boundary to 0, see Table 8.10.

BSIA + Remove $[(i_1, i_2, i_3), n$,`force`,`ignore`]	Profit	Profit %	Iterations	$\mathcal{T}$
Current model	2772081	100	1266942	1
$[(1,1,1), 5$,`FALSE`,`FALSE`]	2635772	95	23502	54
$[(2,2,2), 5$,`FALSE`,`FALSE`]	2755773	99	136085	9
$[(2,1,1), 5$,`FALSE`,`FALSE`]	2776237	100.1	69927	18
$[(1,1,1), 3$,`TRUE`,`FALSE`]	2649661	96	21746	58
$[(1,1,1), 4$,`TRUE`,`FALSE`]	2666242	96	22647	56
$[(1,1,1), 3$,`TRUE`,`TRUE`]	2645396	95	24050	53
$[(1,1,1), 4$,`TRUE`,`TRUE`]	2634245	95	24712	51

Table 8.9: Results BSIA + Remove ALR

BSIA + Remove + $((\mathcal{P} = 0))$ $[(i_1, i_2, i_3), n$,`force`,`ignore`]	Profit	Profit %	Iterations	$\mathcal{T}$
Current model	2772081	100	1266942	1
$[(1,1,1), 5$,`FALSE`,`FALSE`]	2635772	95	19980	63
$[(2,2,2), 5$,`FALSE`,`FALSE`]	2755773	99	119374	11
$[(2,1,1), 5$,`FALSE`,`FALSE`]	2776237	100.1	53263	24
$[(1,1,1), 3$,`TRUE`,`FALSE`]	2649661	96	18952	67
$[(1,1,1), 4$,`TRUE`,`FALSE`]	2666242	96	19401	65
$[(1,1,1), 3$,`TRUE`,`TRUE`]	2645396	95	20032	63
$[(1,1,1), 4$,`TRUE`,`TRUE`]	2634245	95	21098	60

Table 8.10: Results BSIA + Remove + $(\mathcal{P} = 0)$ ALR

Tables 8.11, 8.12 and 8.13 contain the results to similar BSIA instances, but now for HZN. Remarkably, all results in Table 8.13 exceed the 100% mark.

BSIA $[(i_1, i_2, i_3), n$,`force`,`ignore`]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
$[(1,1,1), 5$,`FALSE`,`FALSE`]	2105544	104	11360	33
$[(2,2,2), 5$,`FALSE`,`FALSE`]	2051465	101	48184	8
$[(2,1,1), 5$,`FALSE`,`FALSE`]	1941476	96	33304	11
$[(1,1,1), 3$,`TRUE`,`FALSE`]	2002992	97	7766	48
$[(1,1,1), 4$,`TRUE`,`FALSE`]	2116807	104	11067	33
$[(1,1,1), 3$,`TRUE`,`TRUE`]	2086474	103	11672	32
$[(1,1,1), 4$,`TRUE`,`TRUE`]	2094514	103	12533	30

Table 8.11: Results BSIA HZN

BSIA + Remove $[(i_1, i_2, i_3), n$,`force`,`ignore`]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
$[(1,1,1), 5$,`FALSE`,`FALSE`]	2111642	104	11626	32
$[(2,2,2), 5$,`FALSE`,`FALSE`]	2141288	105	50041	7
$[(2,1,1), 5$,`FALSE`,`FALSE`]	2122318	105	34856	11
$[(1,1,1), 3$,`TRUE`,`FALSE`]	2122318	105	11759	32
$[(1,1,1), 4$,`TRUE`,`FALSE`]	2116807	104	11972	31
$[(1,1,1), 3$,`TRUE`,`TRUE`]	2097530	103	12807	29
$[(1,1,1), 4$,`TRUE`,`TRUE`]	2102136	104	13598	28

Table 8.12: Results BSIA + Remove HZN

BSIA + Remove + $(\mathcal{P} = 0)$ $[(i_1, i_2, i_3), n$,`force`,`ignore`]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
$[(1,1,1), 5$,`FALSE`,`FALSE`]	2111642	104	10924	34
$[(2,2,2), 5$,`FALSE`,`FALSE`]	2141288	105	46344	8
$[(2,1,1), 5$,`FALSE`,`FALSE`]	2122318	105	31402	12
$[(1,1,1), 3$,`TRUE`,`FALSE`]	2122318	105	11198	33
$[(1,1,1), 4$,`TRUE`,`FALSE`]	2116807	104	11320	33
$[(1,1,1), 3$,`TRUE`,`TRUE`]	2097530	103	12351	30
$[(1,1,1), 4$,`TRUE`,`TRUE`]	2102136	104	13022	29

Table 8.13: Results BSIA + Remove + $(\mathcal{P} = 0)$ HZN

8.2.3 COPIA

For COPIA the first goal is to see what `ORDER` may be best to use. This is tested for frequency $f = 1$ and $f = 2$ on HZN with the greedy optimization technique for all types and $i_1, i_2, i_3 = 1$. The results for this are in Table 8.14. The iterations are omitted, since these tests purely concern the quality of the approximation. The results are all quite similar, and therefore it was decided to stick to the initial order [`SCG`,`iSS`,`CB`] for the other tests. To represent this, define $\texttt{ORDER}^* = [\texttt{SCG},\texttt{iSS},\texttt{CB}]$.

COPIA [ORDER, f]	Profit	Profit %
Current Model	2030316	100
[[SCG,iSS,CB], 1]	1953181	96
[[SCG,CB,iSS], 1]	1953181	96
[[iSS,SCG,CB], 1]	1953181	96
[[iSS,CB,SCG], 1]	1947569	96
[[CB,SCG,iSS], 1]	1947569	96
[[CB,iSS,SCG], 1]	1947569	96
[[SCG,iSS,CB], 2]	1971844	97
[[SCG,CB,iSS], 2]	1971844	97
[[iSS,SCG,CB], 2]	1971844	97
[[iSS,CB,SCG], 2]	1997627	98
[[CB,SCG,iSS], 2]	1997627	98
[[CB,iSS,SCG], 2]	1997627	98

Table 8.14: Results ORDER and f for COPIA on HZN

To test the effect of doing multiple rounds over the same ORDER, the (now) standard order ORDER* was taken, and COPIA iterated 5 times over this order. Again, $i_1, i_2, i_3 = 1$. Tables 8.15 and 8.18 contain the results on ALR and HZN respectively for this. Tables 8.16 and 8.19 contain the results of also applying the Remove Algorithm after each iteration over [SCG,iSS,CB]. Tables 8.17 and 8.20 contain the results of additionally applying $\mathcal{P} = 0$.

COPIA [ORDER, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2772081	100	1266942	1
[[ORDER*], 1]	2509923	91	18620	68
[[ORDER*], 2]	2594065	94	27825	46
[[ORDER*], 3]	2604528	94	37030	34
[[ORDER*], 4]	2604528	94	46235	27
[[ORDER*], 5]	2604528	94	55440	23

Table 8.15: Results frequency test COPIA ALR

COPIA + Remove [ORDER, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2772081	100	1266942	1
[[ORDER*], 1]	2644893	95	22256	57
[[ORDER*], 2]	2642473	95	30123	42
[[ORDER*], 3]	2646331	95	39254	32
[[ORDER*], 4]	2646331	95	48459	26
[[ORDER*], 5]	2646331	95	57664	22

Table 8.16: Results frequency test COPIA + Remove ALR

COPIA + Remove + ($\mathcal{P} = 0$) [ORDER, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2772081	100	1266942	1
[[ORDER*], 1]	2644893	95	18423	69
[[ORDER*], 2]	2642473	95	22771	56
[[ORDER*], 3]	2646331	95	28624	44
[[ORDER*], 4]	2646331	95	34551	37
[[ORDER*], 5]	2646331	95	40478	31

Table 8.17: Results frequency test COPIA + Remove + ($\mathcal{P} = 0$) ALR

COPIA [ORDER, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
[[ORDER*], 1]	1928726	95	9646	39
[[ORDER*], 2]	2033775	100.1	14953	25
[[ORDER*], 3]	2025219	99.7	19499	19
[[ORDER*], 4]	2037386	100.3	24225	15
[[ORDER*], 5]	2031090	100.0	29001	13

Table 8.18: Results frequency test COPIA HZN

COPIA + Remove [ORDER, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
[[ORDER*], 1]	2022837	99.6	11172	33
[[ORDER*], 2]	2045353	101	15427	24
[[ORDER*], 3]	2054788	101	20584	18
[[ORDER*], 4]	2049982	101	25217	15
[[ORDER*], 5]	2066858	102	30136	12

Table 8.19: Results frequency test COPIA + Remove HZN

COPIA + Remove + ($\mathcal{P} = 0$) [ORDER, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
[[ORDER*], 1]	2022837	99.6	9460	40
[[ORDER*], 2]	2045353	101	12092	31
[[ORDER*], 3]	2054788	101	15611	24
[[ORDER*], 4]	2049982	101	18632	20
[[ORDER*], 5]	2066858	102	21950	17

Table 8.20: Results frequency test COPIA + Remove + ($\mathcal{P} = 0$) HZN

Until now the runs of COPIA have been done with `evaluate-beforehand=TRUE`. The option to evaluate in between instead of evaluating beforehand, so with `evaluate-beforehand=FALSE`, is tested with ORDER*, frequency 3, $i_1, i_2, i_3 = 1$ and using the greedy method for all types. Tables 8.21 and 8.24 contain these results for ALR and HZN respectively. Here, `before` is short for `evaluate-beforehand`. Tables 8.22 and 8.25 contain the results with both the Remove Algorithm applied afterwards. Tables 8.23 and 8.26 contain the results with both the Remove Algorithm and $\mathcal{P} = 0$ applied afterwards. It appeared that evaluating in between mostly adds to the running time, without improving the results much. Therefore it is not taken along to the next tests.

COPIA [ORDER, before, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2772081	100	1266942	1
[[ORDER*], FALSE, 1]	2530822	91	18191	70
[[ORDER*], FALSE, 2]	2620528	95	37509	34
[[ORDER*], FALSE, 3]	2620528	95	56827	22

Table 8.21: Results COPIA `evaluate-beforehand=FALSE` ALR

COPIA + Remove [ORDER, before, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2772081	100	1266942	1
[[ORDER*], FALSE, 1]	2634249	95	21529	59
[[ORDER*], FALSE, 2]	2637810	95	39454	32
[[ORDER*], FALSE, 3]	2637810	95	58772	22

Table 8.22: Results COPIA evaluate-beforehand=FALSE + Remove ALR

COPIA + Remove + ($\mathcal{P} = 0$) [ORDER, before, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2772081	100	1266942	1
[[ORDER*], FALSE, 1]	2634249	95	18529	68
[[ORDER*], FALSE, 2]	2637810	95	33753	38
[[ORDER*], FALSE, 3]	2637810	95	50370	25

Table 8.23: Results COPIA evaluate-beforehand=FALSE + Remove + ($\mathcal{P} = 0$) ALR

COPIA [ORDER, before, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
[[ORDER*], FALSE, 1]	2018019	99	9188	40
[[ORDER*], FALSE, 2]	2069311	102	18818	20
[[ORDER*], FALSE, 3]	2072857	102	28116	13

Table 8.24: Results COPIA evaluate-beforehand=FALSE HZN

COPIA + Remove [ORDER, before, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
[[ORDER*], FALSE, 1]	2080928	102	10596	35
[[ORDER*], FALSE, 2]	2095156	103	19864	19
[[ORDER*], FALSE, 3]	2098871	103	29128	13

Table 8.25: Results COPIA evaluate-beforehand=FALSE + Remove HZN

COPIA + Remove + ($\mathcal{P} = 0$) [ORDER, before, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
[[ORDER*], FALSE, 1]	2080928	102	9569	39
[[ORDER*], FALSE, 2]	2095156	103	17876	21
[[ORDER*], FALSE, 3]	2098871	103	26220	14

Table 8.26: Results COPIA evaluate-beforehand=FALSE + Remove + ($\mathcal{P} = 0$) HZN

Finally, COPIA was also tested with the optimization method from BSIA for iSS, while still using the greedy method for SCG and CB. It was chosen this way, because COPIA, with the Greedy method in particular, has more trouble with finding better iSS placements, than finding better SCG and CB placements. Two tests were done with the beam method, namely one with $i_1 = 1$ and one with $i_1 = 2$. Tables 8.27 and 8.30 contain the results for these test without the Remove Algorithm for ALR and HZN respectively. Tables 8.28 and 8.31 contain the results with the Remove Algorithm applied afterwards. Tables 8.29 and 8.32 contain the results with both the Remove Algorithm and $\mathcal{P} = 0$ applied afterwards.

COPIA iSS.method=BEAM [ORDER, (i_1, i_2, i_3), f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2772081	100	1266942	1
[[ORDER*], (1, 1, 1), 1]	2548855	92	19651	64
[[ORDER*], (1, 1, 1), 2]	2607516	94	30125	42
[[ORDER*], (2, 1, 1), 1]	2494581	90	68767	18
[[ORDER*], (2, 1, 1), 2]	2667282	96	110166	12

Table 8.27: Results COPIA iSS.method=BEAM ALR

COPIA + Remove iSS.method=BEAM [ORDER, $(i_1, i_2, i_3), f$]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2772081	100	1266942	1
[[SCG,iSS,CB], $(1, 1, 1), 1$]	2648339	96	22758	56
[[SCG,iSS,CB], $(1, 1, 1), 2$]	2644045	95	32127	39
[[SCG,iSS,CB], $(2, 1, 1), 1$]	2703030	98	72549	17
[[SCG,iSS,CB], $(2, 1, 1), 2$]	2726003	98	112308	11

Table 8.28: Results COPIA iSS.method=BEAM + Remove ALR

COPIA + Remove + $(\mathcal{P} = 0)$ iSS.method=BEAM [ORDER, $(i_1, i_2, i_3), f$]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2772081	100	1266942	1
[[SCG,iSS,CB], $(1, 1, 1), 1$]	2648339	96	19565	65
[[SCG,iSS,CB], $(1, 1, 1), 2$]	2644045	95	26319	48
[[SCG,iSS,CB], $(2, 1, 1), 1$]	2703030	98	70435	18
[[SCG,iSS,CB], $(2, 1, 1), 2$]	2726003	98	96622	13

Table 8.29: Results COPIA iSS.method=BEAM + Remove + $(\mathcal{P} = 0)$ ALR

COPIA iSS.method=BEAM [ORDER, $(i_1, i_2, i_3), f$]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
[[ORDER*], $(1, 1, 1), 1$]	1989122	98	9812	38
[[ORDER*], $(1, 1, 1), 2$]	NA	NA	NA	NA
[[ORDER*], $(2, 1, 1), 1$]	2008032	99	34557	11
[[ORDER*], $(2, 1, 1), 2$]	2074813	102	51140	7

Table 8.30: Results COPIA iSS.method=BEAM HZN

COPIA + Remove `iSS.method=BEAM` $[$`ORDER`$, (i_1, i_2, i_3), f]$	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
[[`SCG,iSS,CB`], $(1, 1, 1), 1]$	2080928	102	10596	35
[[`SCG,iSS,CB`], $(1, 1, 1), 2]$	NA	NA	NA	NA
[[`SCG,iSS,CB`], $(2, 1, 1), 1]$	2104304	104	36062	10
[[`SCG,iSS,CB`], $(2, 1, 1), 2]$	2100625	103	52445	7

Table 8.31: Results COPIA `iSS.method=BEAM` + Remove HZN

COPIA + Remove + $(\mathcal{P} = 0)$ `iSS.method=BEAM` $[$`ORDER`$, (i_1, i_2, i_3), f]$	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	2030316	100	374164	1
[[`SCG,iSS,CB`], $(1, 1, 1), 1]$	2080928	102	10024	37
[[`SCG,iSS,CB`], $(1, 1, 1), 2]$	NA	NA	NA	NA
[[`SCG,iSS,CB`], $(2, 1, 1), 1]$	2104304	104	31852	12
[[`SCG,iSS,CB`], $(2, 1, 1), 2]$	2100625	103	44713	8

Table 8.32: Results COPIA `iSS.method=BEAM` + Remove + $(\mathcal{P} = 0)$ HZN

8.2.4 Brute Force Estimator

Although the Brute Force Estimator was implemented, the code contained too many errors. This caused an error in the cost function and an error in the superfluous CFPI function. Algorithm 11 filters on the first evaluation of iSSs. Because costs were incorrect, the wrong mixes ended up being selected, making the final results almost completely incomparable to the results of the current Optimal Mix Model. Therefore, BFE is also not tested on VSN. Nevertheless, a little can be said about the results of the Brute Force Estimator.

Initial tests on ALR with separately estimating SCG mixes showed good results. Also, CB and SCG results CML reduction-wise from the BFE run with error in the code are still useful, since they do not depend on iSS results or costs. These and some other results are picked and stated in the following list:

1. COPIA was run while optimizing once over only SCG with the greedy method and nothing else. The estimator function in 7.10 estimated this mix with an accuracy of $99 + \%$, with only a maximum profit difference of 20 per route. The estimator sometimes overestimated and sometimes underestimated the mix on route level.

2. CB mixes are often over estimated by the estimator function in 7.10. However, when looking purely at the top 10 CB mixes CML reduction-

wise on a route on ALR or HZN, then often at least 5 of them are also in the top 10 in the results of BFE.

3. On multiple routes on ALR and HZN, the top 10 mixes CML reduction-wise containing only CBs and SCGs are completely equal on both BFE as the current model in their orderings of the mixes. The mixes on BFE underestimate the CML reduction a bit, but not by much.

4. Mixes containing all three component types are often underestimated by a greater amount CML reduction-wise by the BFE. For example, the best CML reduction-wise mix on an ALR route was underestimated by 4000.

5. Mixes containing all three component types are often not ordered well by BFE, when compared to the current model. For example, the best mix found by BFE, containing an iSS that was selected by both BFE and the current model, and containing SCGs and CBs, was lower than rank 100 in the current model.

6. As to running time, BFE needs less iterations than all of the tested constructive algorithms, since it only needs to simulate the loose combinations to do the initial evaluation. Table 8.33 contains the number of iterations per setting for BFE. Do note that the actual run would take much longer than stated in these tables, since all mixes need to be estimated piece by piece. Afterwards, some additional simulations need to be done to get an actual evaluated mix instead of an estimation.

Network	Setting	Iterations	$\mathcal{T}$
ALR	Current Model	1266942	1
ALR	$(2, 2, 2)$	74620	17
ALR	$(2, 1, 1)$	27152	47
HZN	Current Model	374164	1
HZN	$(2, 2, 2)$	30865	12
HZN	$(2, 1, 1)$	16130	23

Table 8.33: Iterations for initial evaluation of combinations

8.3 Validation on Substation VSN

The validation of the algorithms is done on Vaassen (VSN), the largest network of the three. Its relevant properties are found in Table 8.34. VSN is a non-remotely controllable substation. Furthermore, for VSN, $F_{\text{SCG}} = F_{\text{iSS}} = \emptyset$ and $F_{\text{CB}} \neq \emptyset$. For VSN certain settings of the tests Greedy Algorithm, BSIA and COPIA are tested for validation. Not all settings are validated since VSN is used to compare the test results on other networks to the results on this bigger network. It is not the goal to find the best settings for VSN specifically. BFE

shall not be validated, as mentioned in Section 8.2.4.

Substation	Profit	(minimum) iterations	$u_1 \in F_{\text{iSS}}$	$\lvert S \rvert$
VSN	1884477	1922669	no	385

Table 8.34: Properties of VSN

8.3.1 Greedy Algorithm

Table 8.35 contains the results for the Greedy Algorithm with the standard expected CML boundary and $\mathcal{P} = -\infty$. Table 8.36 contains the results when the Remove Algorithm is applied to the outcomes in Table 8.35. Lastly, Table 8.37 shows the results with the addition of $\mathcal{P} = 0$.

Greedy (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
$(1, 1, 1)$	1666637	88	17998	107
$(2, 2, 2)$	1681264	89	249619	8
$(2, 1, 1)$	1659897	88	163654	12

Table 8.35: Results Greedy Algorithm VSN

Greedy + Remove (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
$(1, 1, 1)$	1684223	89	20083	96
$(2, 2, 2)$	1798737	95	253007	8
$(2, 1, 1)$	1696407	90	166777	12

Table 8.36: Results Greedy Algorithm + Remove VSN

Greedy + Remove + $(\mathcal{P} = 0)$ (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
$(1, 1, 1)$	1684223	89	17817	108
$(2, 2, 2)$	1798737	95	159649	12
$(2, 1, 1)$	1696407	90	94032	20

Table 8.37: Results Greedy Algorithm + Remove + $(\mathcal{P} = 0)$ VSN

8.3.2 BSIA

Table 8.38 contains the results for BSIA with the standard expected CML boundary and $\mathcal{P} = -\infty$. Table 8.39 contains the results when the Remove Algorithm is applied to the outcomes in Table 8.38. Lastly, Table 8.40 shows the results with the addition of $\mathcal{P} = 0$. Only one BSIA setting was tested on VSN.

BSIA (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
$(1, 1, 1)$	1670158	89	21913	88

Table 8.38: Results BSIA VSN

BSIA + Remove (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
$(1, 1, 1)$	1684260	89	23862	81

Table 8.39: Results BSIA + Remove VSN

BSIA + Remove + $(\mathcal{P} = 0)$ (i_1, i_2, i_3)	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
$(1, 1, 1)$	1684260	89	20754	93

Table 8.40: Results BSIA + Remove + $(\mathcal{P} = 0)$ VSN

8.3.3 COPIA

Table 8.41 contains the results for COPIA with the standard expected CML boundary and $\mathcal{P} = -\infty$, the standard order, `before=TRUE`, uses the greedy method for optimization over each type and frequency 5. Table 8.42 contains the results when the Remove Algorithm is applied to the outcomes in Table 8.41. Lastly, Table 8.43 shows the results with the addition of $\mathcal{P} = 0$.

COPIA [ORDER, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
[[ORDER*], 1]	1620171	86	17053	113
[[ORDER*], 2]	1680426	89	25426	76
[[ORDER*], 3]	1680426	89	33799	57
[[ORDER*], 4]	1680426	89	42172	46
[[ORDER*], 5]	1680426	89	50545	38

Table 8.41: Results frequency test COPIA VSN

COPIA + Remove [ORDER, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
[[ORDER*], 1]	1674161	89	19774	97
[[ORDER*], 2]	1680426	89	27229	71
[[ORDER*], 3]	1680426	89	35602	54
[[ORDER*], 4]	1680426	89	43975	44
[[ORDER*], 5]	1680426	89	52348	37

Table 8.42: Results frequency test COPIA + Remove VSN

COPIA + Remove + ($\mathcal{P} = 0$) [ORDER, f]	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
[[ORDER*], 1]	1674161	89	17644	109
[[ORDER*], 2]	1680426	89	22035	87
[[ORDER*], 3]	1680426	89	28344	68
[[ORDER*], 4]	1680426	89	34653	55
[[ORDER*], 5]	1680426	89	40962	47

Table 8.43: Results frequency test COPIA + Remove + ($\mathcal{P} = 0$) VSN

Table 8.44 contains the results for COPIA with the standard expected CML boundary and $\mathcal{P} = -\infty$, the standard order, `before=TRUE`, frequency 2 and uses method `BEAM` for optimization over iSS. Table 8.45 contains the results when the Remove Algorithm is applied to the outcomes in Table 8.44. Lastly, Table 8.46 shows the results with the addition of $\mathcal{P} = 0$.

COPIA `iSS.method=BEAM` $[\texttt{ORDER}, (i_1, i_2, i_3), f]$	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
$[[\texttt{SCG,iSS,CB}], (1,1,1), 1]$	1620171	86	18125	106
$[[\texttt{SCG,iSS,CB}], (1,1,1), 2]$	1680426	89	27628	70

Table 8.44: Results COPIA `iSS.method=BEAM` VSN

COPIA + Remove `iSS.method=BEAM` $[\texttt{ORDER}, (i_1, i_2, i_3), f]$	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
$[[\texttt{SCG,iSS,CB}], (1,1,1), 1]$	1674161	89	20846	92
$[[\texttt{SCG,iSS,CB}], (1,1,1), 2]$	1680426	89	29431	65

Table 8.45: Results COPIA `iSS.method=BEAM` + Remove VSN

COPIA + Remove $+ (\mathcal{P} = 0)$ `iSS.method=BEAM` $[\texttt{ORDER}, (i_1, i_2, i_3), f]$	Profit	Profit %	Iterations	$\mathcal{T}$
Current Model	1884477	100	1922669	1
$[[\texttt{SCG,iSS,CB}], (1,1,1), 1]$	1674161	89	18194	106
$[[\texttt{SCG,iSS,CB}], (1,1,1), 2]$	1680426	89	24480	79

Table 8.46: Results COPIA `iSS.method=BEAM` + Remove + $(\mathcal{P} = 0)$ VSN

Chapter 9

Discussion

Let us cite the research question to this thesis one more time:

Can the Optimal Mix Model be improved by implementing heuristic optimization techniques in order to substantially lower the running time while closely approximating qualitative solutions?

The short answer to this is "yes, but conditionally". This chapter is dedicated to explaining the answer by reviewing the results: what are the key findings, what is the meaning of the results, what do they implicate and what limitations are shown in the process? The chapter finishes with a recommendation for Alliander.

9.1 Key Findings

The best results both quality wise and running time wise while maintaining an approximation of at least 95% are stated in the following list:

- Best quality run on ALR: 100.1% with $\mathcal{T} = 24$
- Best quality run on HZN: 106% with $\mathcal{T} = 8$
- Best quality run on VSN: 95% with $\mathcal{T} = 12$
- Fastest run on ALR $\geq 95\%$: 95% with $\mathcal{T} = 76$
- Fastest run on HZN $\geq 95\%$: 105% with $\mathcal{T} = 40$
- Fastest run on VSN $\geq 95\%$: 95% with $\mathcal{T} = 12$

From this it can be seen that the algorithms perform better on ALR and HZN than on VSN, but that they nevertheless reach to 95% mark on VSN too. The next section is devoted to explaining the above and other results from Chapter 8 in more detail.

9.2 Interpretations

The interpretation of the results is based on the rate of approximation of the optimum and the speed-up factors, unless more detailed information of the network is needed. The precise details of the networks are omitted when possible, such as the number of clients connected to a secondary substation and such as the capacity of section.

First, the results will be treated generally. Afterwards, the results are interpreted per network. For each network, the most notable results in addition to the general remarks are picked to discuss.

9.2.1 General

Multiple general remarks can be made about the interpretations of the results. Each remark is stated down below with explanation.

Greedy Algorithm and BSIA perform best profit-wise and run-time-wise, COPIA performs a bit worse

The mixes with the highest profit on all networks are found either by the Greedy Algorithm or by BSIA. The setting $[(1, 1, 1), 4, \texttt{TRUE}, \texttt{FALSE}]$ on BSIA gives the highest results on ALR and HZN and the setting $(2, 2, 2)$ on the Greedy Algorithm gives the highest result on VSN, both when not applying the Remove Algorithm. When applying the Remove Algorithm afterwards, the setting $(2, 2, 2)$ on the Greedy Algorithm gives the highest results on VSN and HZN and the setting $(2, 1, 1)$ on BSIA gives the highest result on ALR. Results on COPIA come close to the performances of Greedy Algorithm and BSIA, but are not as high.

Reason for the good performances of the constructive algorithms lies mainly in the ideas that high profit components generally performing well in a mix, and that two components worsen each other (points 9 and 11 in Section 6.2), and partly in that combining the profitable duos can become profitable triples (point 10).

The fastest runs are also done on Greedy Algorithm and BSIA. The simply explanation for this is that COPIA needs to make optimization steps more than once to gain better results, and this also results in a clear higher run-time. BSIA and Greedy Algorithm are capable of finding good mixes in less time.

Algorithms perform better on ALR and HZN than on VSN

Each of the implemented algorithms has multiple settings with which it exceeds the 95% approximation of the Optimal Mix by the current model on ALR and HZN. On VSN however, the best result reaches exactly 95% of the optimum, and this is the slowest run of all on VSN, while other runs only reach 90% at best. There are two ways to explain this.

First, the profit of the optimal mix is lower on VSN than on the other two

networks. That means that missing a good component in the mix has a bigger impact on VSN than on the other two substations. ALR reaches the 95% mark when the profit if 138604 lower at maximum, for HZN that is 101515, and for VSN 94223.

Secondly, VSN contains the three largest routes when taken all routes of ALR, HZN and VSN together, of which one route in particular is substantially larger. The methods that work well on networks such as ALR and HZN may simply perform worse on larger networks, because of the simplicity of the algorithms.

Either way, it is clear that the algorithms work very well for HZN and ALR, and worse for VSN.

The Remove Algorithm efficiently improves outputted mixes

The Remove Algorithm often improves a mix that has been constructed by the Greedy Algorithm, BSIA or COPIA. Reason for this is that the constructive algorithms often add components in an earlier stage that are not beneficial anymore when more components are added. See Section 7.2.3 for further details on this.

Also, the highest profits of mixes are achieved on each network by applying the Remove Algorithm.

In addition, the Remove Algorithm takes a relative low number of iterations to improve a mix, since only included components are checked for removal.

The Profit Boundary $\mathcal{P}$ set to 0 improves running time without influencing results

Setting the profit boundary $\mathcal{P}$ to 0 did not influence the quality of the results, and did shorten the running time. Reason for this is that no loss-giving components are included in the outputted mix in any of the runs. Do note that profitable combinations containing a loss-giving component are still evaluated in the estimated results here, while these combinations are excluded when $\mathcal{P} = 0$ is actually applied during the run. See Section 8.1.2 for further details on this.

BFE performs worse on three component types

Even though there are few results on BFE, it can be seen that it does not perform as preferred. When ordering the output of the BFE on expected CML reduction for mixes containing three component types, there is a large error on the top mixes.

9.2.2 ALR

The results on ALR have already been explained in the general section above, except for the run-time aspect and a result exceeding the 100% mark. On average, the speed-up factor on ALR lies higher than for HZN, and lower than for VSN. This is due to the size of the network. ALR has more sections than HZN,

less sections than VSN, and does not contain significantly large routes as VSN does.

The BSIA solution that exceeds the 100% mark, has many different components than the optimal mix found by the current model. This concerns iSSs specifically. The method used in the constructive algorithms determines the quality of a loose iSS on an open point on both of the routes it influences, while the current model only calculates a loose iSS on an open point on the route the secondary substation lies on. This makes the current model only recognize the value of the iSS when it is part of a duo or when it is on the route itself. It appears that the method used in the constructive algorithms is able to find a better mix now that this rule of the current model is dropped.

9.2.3 HZN

Results on HZN often exceed the 100% mark. There are two explanations for this. The first reason for the high results on HZN is that a specific iSS lies on a secondary substation that is included in two parallel routes. This means that there are two sets of cables in the substation, both connected to different routes and not to each other, not even by an NOP. In reality, an iSS on this secondary substation can only be placed on one of these routes. The current version of the Optimal Mix Model solves this by calculating double the costs for this secondary substation, but the implemented algorithms do not. Therefore, the iSS is evaluated as a highly profitable component, and is included in almost each mix on HZN. This can be compensated by adding the iSS costs to the total costs of the mix, and thereby lowering the profit of the mixes on HZN by about 68000. To gain the correct results, this come down to lowering the results by $\frac{68000}{2030316} \cdot 100\% \approx 3\%$.

When lowering the results by 3%, some results still exceed the 100% mark. The reason for this is that the restrictions on the placement of iSSs are dropped, because HZN has a remotely controllable substation. Both the restriction that iSSs are not allowed to be placed next to each other and the restriction that single iSSs only are allowed to be placed on an NOP, are dropped. As explained in Section 6.2, iSSs can perform well when not placed on an NOP on a remotely controllable substation. For HZN, dropping the restrictions results in mixes with higher profit than the mix found by the current Optimal Mix Model. In appeared that an iSS duo of two iSSs placed next to each other, while both are not lying on an NOP, is a very profitable solution on one of the routes on HZN.

As regards to run-time, HZN is the network with the lowest gain. This lies in the fact that HZN is a small network, making each simulation relatively more impact on the speed-up factor. The actual number of iterations on HZN lies lower than for ALR and VSN, which makes sense, since ALR and VSN both are larger than HZN.

9.2.4 VSN

Results on VSN are lower than the results on HZN and ALR. The reasons for this are already stated in Section 9.2.1. An additional notable result to address for VSN is that the algorithms with setting $(1, 1, 1)$ on combination sizes terminate the fastest from all algorithms, even exceeding the $\mathcal{T} = 100$ mark. Reason for this is that VSN is really large, contains many duos and triples that are evaluated on the current Optimal Mix Model, and wins more run-time here then ALR and HZN. The performance on run-time drops really fast when considering combinations of size 2, since VSN contains some large routes where many combinations are possible. Even though there are less duos on these routes than triples on that are evaluated on the Current Optimal Mix Model, it still makes the algorithms a lot slower.

9.3 Implications

There are three important implications to make based on the results. To start, the results provide the insight that the Optimal Mix Problem can be approximated in shorter running times than the current Optimal Mix Model. Many results exceed the 95% level and they do so while being at least 10 times faster. When lowering the bar towards 90%, the running times become even faster.

Secondly, when VSN is excluded from the results, the quality of the approximations become better on average, and running times become better for the best solutions quality wise too. This may indicate that the algorithms perform very well on certain types of networks, and worse on other types. On itself, that implicates that the results of these algorithms may become unreliable when applied to networks of which is not known if the algorithms perform well on such networks.

As third, in conjunction with the previous point, the algorithms perform very well on networks ALR and HZN. Since these substations are representative for a large part of the areas supervised by Liander, the algorithms make a reliable option for this area.

9.4 Limitations

Some limitations were shown during the research that should be considered when reviewing the results. Let us start with the method used to determine (the improvement of) the running time. Counting the iterations of the Malfunction Simulation Model is a valid way of approximating the running time improvement for this part of the Optimal Mix Model, but it is not fully precise. Other operations, such a determining the suitable combinations, are not taken into account. When the heuristic algorithms are finally implemented on a computer with the same qualities as the computers that run the current Optimal Mix Model, a small deviation in running time from what is stated in this paper is not unimaginable.

Secondly, this research is done on an older version of the Optimal Mix Model. Alliander provided the most up to date model when the research commenced, but since then there have been many developments in the model. These adaptations are mainly focused on making the model more precise, such as taking section fail frequencies per year in stead of assuming the same fail frequency for all 40 years and do not change the structure of the model. The adaptations do influence the outcome of the heuristic methods somewhat, for example by altering the suitable iSS locations somewhat by looking at the secondary substation in more detail. Since the model stayed the same globally, it is suspected that the difference in outcome is not too big.

A final remark on the limitations is that the generalizability of the results is limited by the information that was provided by Alliander, namely the set of substations to analyze and run tests on, ALR, HZN and VSN. Even though ALR, HZN and VSN are assumed to be representative substations for the whole of the medium voltage grid maintained by Alliander, there are still substations that form an exception to the rule. It is uncertain whether the algorithms perform as good on these exceptional substations as they do on ALR, HZN and VSN.

9.5 Recommendations

The goal of the recommendation for Alliander is to present the best options from the algorithms that are tested in this thesis. Recommendations on other implementation methods or improvement on the implemented methods, can be found in Future Work (Chapter 11).

When it comes down to recommendations of algorithms and what settings to utilize, it will depend on the specific wishes of Alliander. All three constructive algorithms, Greedy Algorithm, BSIA and COPIA, perform well, with Greedy Algorithm and BSIA performing slightly better than COPIA. Therefore, Greedy Algorithm and BSIA are recommended. The critical part for these algorithms is the settings that the algorithms should be run with. Two options based on certain preferences are suggested. Before stating these options, first consider the additions to the algorithms.

Based on the results, it is highly recommended to combine the constructive algorithms with both the profit boundary $\mathcal{P} = 0$ and with applying the Remove Algorithm afterwards. The first improved the running time for all tests without lowering results, and the second improved the profit of almost all mixes without lowering the running time significantly long.

Now to consider the options for the algorithms and their settings. The first option is to use $(2, 2, 2)$ on the Greedy Algorithm or $[(2, 2, 2), 5, \texttt{FALSE}, \texttt{FALSE}]$ on BSIA. These settings make the runs significantly slower, but the results become the highest on HZN and ALR, and become significantly better on VSN. This option should be picked when the quality of the mixes to be found on any network has the absolute priority.

The second option is to use $(1, 1, 1)$ on the Greedy Algorithm or either $[(1, 1, 1), 4, \texttt{TRUE}, \texttt{FALSE}]$ or $[(1, 1, 1), 5, \texttt{FALSE}, \texttt{FALSE}]$ on BSIA. These settings give desired results above the 95% mark on networks HZN and ALR, and are expected to give similar results on similar networks. They perform worse on network VSN, and are expected to perform worse on networks similar to VSN. This option should be picked when the quality of mixes to be found on networks with in the same category as HZN or ALR have some priority, and the speed-up factor is seen as a high priority. Even on networks as VSN, the results on profit will not be terrible, but they will be worse. The speed-up factor on networks as VSN will be high.

An additional benefit of the second option is that runs of the different settings can be done subsequently while still being faster than the runs using $(2, 2, 2)$ as the setting for the combination sizes. This gives multiple mixes per network with slight differences per mix. The various results can be analyzed to get a better understanding of what components are more optimal than others.

A third option arising from the first two, is to use setting $(1, 1, 1)$ on the Greedy Algorithm or either $[(1, 1, 1), 4, \texttt{TRUE}, \texttt{FALSE}]$ or $[(1, 1, 1), 5, \texttt{FALSE}, \texttt{FALSE}]$ on BSIA for networks such as ALR and HZN, and to use setting $(2, 2, 2)$ on the Greedy Algorithm or $[(2, 2, 2), 5, \texttt{FALSE}, \texttt{FALSE}]$ on BSIA for networks such as VSN.

In the end it comes down to the preferences of Alliander. When Alliander prioritizes quality over everything, then it is recommended to use the first option. When quality is important, but the speed-up factor is also considered very relevant, then the second option is recommended.

Chapter 10

Conclusion

In this chapter the research as a whole is briefly summarized and the research question and answer for this project are stated.

10.1 Reflection

During this research, the following questions have been answered:

1. How can the Optimal Mix Problem be mathematically described?
2. How does the Optimal Mix Model solve the Optimal Mix Problem?
3. What is the mathematical complexity of the Optimal Mix Problem?
4. How can heuristic optimization techniques be used to improve the running time of the Optimal Mix Problem while closely approximating qualitative solutions?
5. What heuristic optimization techniques can be used to improve the running time of the Optimal Mix Problem while closely approximating qualitative solutions?
6. How well do heuristic optimization techniques improve the running time of the Optimal Mix Problem while closely approximating qualitative solutions?

At the start of this thesis the Optimal Mix Problem was introduced and thoroughly explained. Afterwards, the mathematical formalization of the problem was given. Next, it was showed how the Optimal Mix Model solves the Optimal Mix Problem, where it was also seen that the Optimal Mix Model does not necessarily find the optimal mix, but does at least find a mix that approximates it closely. Afterwards, a proof of the NP-hardness was given, by identifying the

objective function of the Optimal Mix Problem as an NP-hard function. Using this as mathematical justification for the research, specific heuristic methods were designed on the basis of an analysis on the results of the Optimal Mix Model for substations ALR and HZN and on the Optimal Mix Problem in general. The methods were adapted during the implementation process to fit the problem better, and finally results of the methods Greedy Algorithm, BSIA and COPIA for the three substations ALR, HZN and VSN were gathered. The results and methodology were analyzed in the discussion, where an answer for the research question was formulated.

10.2 Research question answered

Can the Optimal Mix Model be improved by implementing heuristic optimization techniques in order to substantially lower the running time while closely approximating qualitative solutions?

Yes, the above is possible. It does depends on what dimensions of the Optimal Mix Model are found more critical to improve on. Examples of well performing algorithms are as follows. The Greedy Algorithm + Remove Algorithm with $(2, 2, 2)$ as input sizes for the combinations finds solutions of at least 95% of the optimum on ALR, HZN and VSN, while doing this at least 8 times faster than the current model. On the other hand, the Greedy Algorithm + Remove algorithm with $(1, 1, 1)$ as input sizes for the combinations finds solutions of at least 95% of the optimum on ALR and HZN, and 89% on VSN, while doing this at least 40 times faster than the current model. Both are reasonable options to consider, and it depends on whether low running time with high profitable solutions is preferred, or higher running time, but still lower than the original, with slightly less profitable solutions. Either way, improvement of the Optimal Mix Model by implementing heuristic optimization techniques to substantially lower the running time while closely approximating qualitative solutions is most certainly possible.

Chapter 11

Future work

While investigating the possibilities of applying heuristic methods to the Optimal Mix Model, other opportunities for research on the subject were found as well. Also, further research on heuristic methods might be beneficial. The final chapter of this thesis treats three classes of future work for the Optimal Mix Problem: improvement of the heuristic algorithms, improvement of running time of the current model and other techniques to research.

11.1 Improvement and Broadening of Heuristic Methods

The three types of heuristic methods as distinguished in Chapter 7 can all be improved in some manner. The suggestions given here could not be executed during this project due to a lack of time.

First, further tuning of the variables of the implemented constructive algorithms is very well possible to optimize running times and quality of the found mixes. To do so, more testing and validation can be done on more networks. This may results in an optimal setting of the variables per type of network.

Secondly, testing of or further research on evolutionary algorithms may be interesting, since these methods function very differently from the implemented algorithms. For this thesis evolutionary algorithms were not implemented in the end, because of the longer running time that stands for these methods and because of the already great results of the constructive algorithms. These evolutionary algorithms could still be interesting however. For example, the implemented algorithms all output one mix, and this output would be the same each time the algorithm is run with the same setting, while evolutionary algorithms make use of a population and are capable of outputting and evaluating multiple mixes at the same time. This broader sense of generating solutions might be very beneficial. Further research and testing could test this.

Finally, the Brute Force Estimator has shown some hopeful results. Initial results implicate that BFE is bad at estimating mixes containing all three types,

but does well when estimating mixes containing less component types. Further research and testing of the BFE can results in application for the well-performing aspects of BFE. For example, an outcome might be that the estimator function 7.10 can be used as one of the optimization techniques in COPIA for one of the types, or that BFE is used in the Genetic Algorithm as one of the neighborhood operators.

11.2 Improvement on Running Time

Four suggestions in total are done to improve the running time of the current model. Two are based on skipping evaluations and can be implemented right away. The other two are suggestions for further research on the matter, that can help in recognizing non-profitable mixes before evaluating.

Skipping evaluations

Two adaptations surrounding skipping evaluations are suggested that could be implemented as of now, without influencing the functionality of the current model. The first is to add a dynamic expected CML boundary on route level for each component type and the number of components, similar to the expected CML boundary introduced in Section 7.2.1. Using such a boundary can lower the running time by skipping evaluations of mixes that would never be profitable. For example: if the expected CML for a route is 5000, then placing two iSSs (not on an open point) will never be profitable, since the costs for two iSSs is at least 144000 while $5000 \cdot 20 = 100000$ is the maximum expected CML reduction revenue. The superfluous CFPI revenue can easily be taken into this calculation, by lowering the expected CML boundary with 500 per superfluous CFPI, since the revenue per CFPI is 10000 and $\frac{10000}{20} = 500$. The boundary can be made even more precise by introducing a boundary level per iSS, since iSS costs differ. Then, the dynamic CML boundary is the sum of the number of SCGs times the SCG CML boundary, of the number of CBs times the CB CML boundary, of the specific iSS CML boundaries minus 500 times the number of superfluous CFPI for this mix. Then, if a certain route has an expected CML lower then the dynamic CML boundary for this mix, the mix certainly will give a loss, and therefore the evaluation can be skipped.

The second method to lower the running time also makes use of skipping certain evaluations. For this the notion of influential area (see Section 6.2) is needed. Suppose a mix m is evaluated and the profit $w(m)$ is known. Then add a component c to gain mix m^c. Now, only malfunctions on the sections in the influential area of c need to be simulated again, since all other sections are not affected by the addition of c. Therefore, simulations on sections outside of the influential area of c, can be skipped. To be able to do this, components should be evaluated once separately to determine the influential area, and for iSSs also duos need to be evaluated separately. Fortunately, all mixes with 1 component and 2 components are evaluated by the current model anyways, so this will not

extend the running time of to the model.

Research

With regards to further research to improve the running time without changing the functionality of the model, another two ideas are suggested. First is a research on the minimum number of CML. As stated before in this thesis, the expected CML for a substation shall never be 0, since not all power outages are avoidable and since power outages take at least 10 minutes to be solved. It could help to gain insight in the relation between the fail frequencies, number of connected clients, topology of a substation, and others and how this is connected to the absolute minimum of expected CML for a substation (on route level). If an absolute minimum can be given, or an approximation from below of the absolute minimum, then the dynamic CML boundaries can become more precise by taking this minimum into account as unreducible expected CML. In the end, this could make it easier to skip and reject more mixes before evaluation.

The second research proposal is done with regards to the Malfunction Simulation Model. The simulations are done using a Mixed Integer Linear Programming module. Everything works as it should, but it could be interesting to research whether this module can become more efficient. This proposal is done because the algorithms and current model spend most of their time solving the MILP module. When this module is improved, it immediately becomes noticeable in the running time. Research can be of great value for the improvement of running time for both the current model as for the heuristic methods.

11.3 Other

Finally, two other approaches to attacking the Optimal Mix Problem are introduced together with a theoretical research proposal to the categorization of medium voltage networks. First, there is techniques from Machine Learning. Many techniques, such as gradient descent, have proven to deliver good results when applied correctly. It has not yet been researched whether machine learning techniques may or may not be applicable to the Optimal Mix Problem, and it is interesting to see whether these new promising methods could be of value.

The second technique depends on research to the NP-hardness proof. If a correct reduction from an NP-hard problem to the Optimal Mix Problem is found while disregarding the objective function, or a reduction from the Optimal Mix Problem to an NP-hard problem, then algorithms for this NP-hard problem can be researched. If there are any well working approximation algorithms for this specific NP-hard problem, then this may introduce new options as to how to solve OMP. This is of course dependent of how easy it is to find such a proof, and whether good approximation algorithms exist for this problem.

Lastly, the theoretical research to the categorization of medium voltage networks helps in distinguishing different networks from each other on clear prop-

erties. Even though earlier in this thesis there is spoken of "similar networks", it is not yet completely clear what "similar" means, except for the size of the network. More properties to consider for example are section capacity, (maximum) length of routes, number of clients and number of open points. Taking more properties into this comparison can give more insight into how networks function and can also help in linking the approximation algorithms to certain types of networks more reliable.

Bibliography

[1] M. van der Meulen, *Gebruikersdocumentatie Optimale Mix Model*, Alliander N.V., Duiven, v0.9, 2018. *(Classified document, only available for employees)*

[2] M. R. Garey and D. S. Johnson, *Computer and Intractiability: A Guide to the Theory of NP-Completeness*, Bell Telephone Laboratories Incorporated, USA, 1979

[3] M. Schutten and J. Hurink, *MasterMath Course: Heuristic Methods in Operations Research: lecture slides 10*, University of Twente, Enschede, 2018-2019. *(Classified document, only available for students and teachers of MasterMath)*