Radboud University Nijmegen

Faculty of Science

Improving the Simulation of Outages in the Power Grid

Removing heuristic choices when restoring power to customers

alliander

Author

Vincent Koppen
s4076028

Supervisors

Radboud University
Dr. W. Bosma

Alliander N.V.
Dr. S. Rieken
F. van Hummel MSc

02 April 2020

Abstract

Alliander maintains the low and medium voltage power grid for a large part of the Netherlands. It uses a simulation model to predict the impact of an outage in the medium voltage network. This model contains a Mixed Integer Linear Program to determine the necessary operations to restore power to customers. This thesis researches how this Mixed Integer Linear Program can be extended to eliminate some of the heuristic choices within the current model. Multiple extensions to the problem have been implemented and tested with promising results.

Acknowledgement

First off I really want to thank my supervisors at Alliander, Fabian van Hummel and Sander Rieken. They were always available to talk and answer any questions I had. I also want to thank Wieb Bosma, for his supervision on behalf of the Radboud University.
Secondly a big thank you to the two teams I participated in, Qirion EC and Systeem Optimalisatie. Because of you I got a clear picture of what it would be like to work at Qirion/Alliander and you showed the business side of things. You were also great company to be with and I really enjoyed my time during the internship.
Thanks to my friends and family for their support. Lastly a big thank you to my partner Myrte. For proofreading this thesis and her support throughout the internship.

Contents

Chapter 1

Introduction

This thesis researches the simulation of outage in the medium voltage network. We will focus on the restoration of power to the customers. We try to improve the current implementation by removing some of the heuristic choices in it.

Alliander

The thesis before you is the result of an internship at Qirion, a subsidiary of Alliander. During this internship I was part of the Energy Consulting team that tackles all kind of data science problems. They mainly work for Liander, a Distribution System Operator that is also part of Alliander. Liander is responsible for a large part of the low and medium voltage network in the Netherlands.
The problem I worked on was part of the System Optimization team of Liander. I worked with this team for 8 months to understand and answer the research question.

Optimal Mix Model

While Alliander can place different types of components into their medium voltage network that help in case of a power outage, placing these is both labor intensive and expensive. Alliander wants to know where it should place the new components, so that we improve the network optimally. This is done using the Optimal Mix Model, which was developed by the System Optimization team.
The model tries millions of different placements of the components and each time rates the newly created network. This rating is based on the time that customers are without power in case of an outage and is determined by the Outage Simulation Module. This means that we want the Outage Simulation to be both as accurate as possible with its rating, while still being fast. Realising both is a challenge.

Research question

The Outage Simulation Module contains the Switch Planner. This is a Mixed Integer Linear Program that tells us how we can restore power to certain parts of the network after an outage has occurred. The module makes some heuristic choices before the Switch Planner is called upon. We want to determine these heuristic choices in an optimal way. The research question that is the subject of this thesis becomes:

Can we extend the Switch Planner so that it incorporates some of the heuristic choices made in the Outage Simulation Module?

If we succeed we have eliminated some of the heuristic choices, since Mixed Integer Linear Programs can be solved optimally. The goal is also to not significantly increase the running time and ideally, to improve on the running time.

Chapter overview

The following two chapters discuss the necessary background information for the rest of the thesis. In **Chapter 2** we discuss the electrical power grid in the Netherlands and the different components in the medium voltage network. Furthermore we define the mathematical representation that will be used throughout the thesis.
Chapter 3 briefly discusses the basics of Mixed Integer Linear Programming, which plays an important role throughout the rest of the thesis.
Chapter 4 explains the current implementation that simulates an outage in the medium voltage network, the Outage Simulation Module. This module determines how power can be restored to all the customers after an outage in one of the cables of the network. Section 4.1 discusses the global working of this module, while Section 4.2 shows all the details of the Switch Planner that is used within this module. The Switch Planner is a Mixed Integer Linear Program that determines what operations are needed to restore power to parts of the network.
In **Chapter 5** we define extensions on the Switch Planner that have been researched and implemented. There are three extensions and one small variation. Section 5.1 discusses an extension that improves on the solution when we can't restore power to every customer. Section 5.2 shows an extension of the Switch Planner that can solve multiple parts of the network together and Section 5.3 defines an improved objective function for the Switch Planners. Section 5.4 discusses a small variation on the Switch Planner discussed in Section 5.2.
In **Chapter 6** we give an answer to our research question and discuss the recommendation for Alliander based on our results. Lastly we discuss possible further research.

Chapter 2

Power grid and its representation

This section describes the power grid in the Netherlands, with a focus on the medium voltage network. It is based on [8] and information within Alliander.
We discuss the structure and properties of the medium voltage network and how to restore power in case of an outage. Here we show the influence of the different secondary components that can be placed in the network. Lastly we also set the mathematical representation that will be used throughout this thesis.

2.1 Transport of electricity

The transport of electricity in the Netherlands happens through different networks, the low-, medium and high voltage networks. This follows from the definition of electric power. Electric power is defined by the product of current and voltage. More specifically

$$P = U \times I,$$

where P is the electric power in Watt (W), U is the voltage in Volt (V) and I is the current in Ampere (A).

During the transport of electric power there will always be losses due to resistance and heat generation. Their impact depends greatly on the current of the electric power. The loss by heat is quadratic with respect to the current, which is not the case for voltage. To minimize power loss we want the current I to be as low as possible because of this. This means that we have to increase the voltage U to provide enough power. High voltage is however not always practical. It can be dangerous to transport and our appliances function at a lower voltage.

The transportation of electricity is therefore divided into different networks that each operate at a certain voltage. The larger distances are done through a network that uses a high voltage. The closer it gets to the customers, the lower the voltage of the network will become. The networks that are used in the Netherlands:

Ultra High Voltage	UHV	220 kV - 340 kV
High Voltage	HV	50kV, 110 kV - 150 kV
Medium Voltage	MV	10 kV - 20 kV
Low Voltage	LV	400 V

The different networks are connected with each other through transformer stations. These reduce the voltage to make the electricity suitable for the new network.

Definition 2.1.1 (Substation)**.** A station that transforms the power from (ultra) high voltage to medium voltage is called an *HV/MV-substation*. Also referred to as *substation*.

Definition 2.1.2 (Secondary substation)**.** Similarly a station that transforms the power from medium voltage to low voltage is called a *MV/LV-substation*. Also referred to as *secondary substation*.

Definition 2.1.3 (Distribution line)**.** The connection between two (secondary) substations is called a *distribution line*.

Typically the usage of the different networks can be thought of as:

- (U)HV-network: Transport the power from the generation site to substations throughout the country.
- MV-network: Transport the power from the substation to secondary substations located in the districts inside a city.
- LV-network: Transport the power from a secondary substation to customers inside the district.

Within the Netherlands these networks are managed by different companies. The (U)HV-network is serviced by Tennet. Liander is one of the dutch Distribution Service Operators (DSO) that maintains the MV and LV-networks in the indicated regions of Figure 2.1.

Figure 2.1: Work area of Liander [1]

Energy transition In the last few years there has been an energy transition. Previously the energy was generated at one location and distributed through the net. Nowadays there are lots of windmills and solar parks that feed directly into the network. As a result the power generation is becoming decentralised.
This puts a lot off stress on the current network, since it wasn't built with this change in mind. The result is that new solar fields and windmills can sometimes not be added to the current network, because of capacity constraint on the cables. This is one of the biggest challenges in the next years for companies like Liander. They constantly need to improve their network to deal with the higher demands.

2.2 Reliability of the network

Liander wants to provide power to their customers with as little outage as possible. Ideally this means that there will never be an outage in the network. However this is not a realistic scenario. Therefore it is important to be able to quickly restore power in case of an outage. The impact of an outage is measured in customer minutes lost.

Definition 2.2.1 (Customer Minute Lost)**.** Let C be the set of all customers in the network. Let O denote an outage in the network. Each customer $c \in C$ will be without power for m_c minutes as a result. The customer minutes lost of O is now defined as:

$$\text{CML}_O = \sum_{c \in C} m_c$$

Example 2.2.1. Suppose that an outage cuts the power to 30 customers. It took 40 minutes for the first 20 customers to get power again and 50 minutes for the last 10 customers. The customer minutes lost of this outage is:

$$\text{CML}_O = \sum_{c \in C} m_c = 40 \cdot 20 + 50 \cdot 10 = 1300$$

Outages can have different causes. The most common reason for an outage is a failure in the connection between two cables. The cables used by Liander are very reliable. But due to the distances between two secondary substations it is often not possible to use one cable to connect them. A distribution line between secondary substations therefore consists of multiple cables. The connection between these cables is the most vulnerable part of the network. This experience is used to determine the expected CML of a whole network.

Definition 2.2.2 (Expected Customer Minutes Lost)**.** Let E be the set of distribution lines in the network, f_e denote the failure frequency for each $e \in E$, O_e the outage that corresponds to a failure in distribution line e.
The *expected customer minutes lost* of network now is:

$$\text{ECML} = \sum_{e \in E} f_e \cdot \text{CML}_{O_e}$$

To determine the expected customer minutes lost of a network we simulate the customer minutes lost of an outage in each of its distribution lines and multiply it with the failure frequency of that line. The failure frequency of each distribution line is determined by a team within Liander.

This thesis focuses on the process to determine the different CML_{O_e}. Therefore w.l.o.g. we will take $f_e = 1 \quad \forall e \in E$, such that $\text{ECML} = \sum_{e \in E} \text{CML}_{O_e}$. Any difference in CML_{O_e} will now be more noticeable when validating our results. Furthermore, any improvement in the method to determine CML_{O_e} will also give a more realistic ECML that does use the correct f_e.

2.3 Representation of the medium voltage network

Outages can occur in both the LV-network and the MV-network. However, the impact of an outage in the MV-network is much larger compared to one in the LV-network. This thesis will focus on the outages in the MV-network. Below we discuss the structure of the MV-network, followed by its mathematical representation.

2.3.1 Medium voltage network

Thus far we have talked about the MV-network as a whole, which might be misleading. In reality we have the substations that are connected to the (U)HV-network and transform the power to medium voltage. Each substation powers its own MV-network. A secondary substation is always connected to one unique substation. All these smaller networks together form the complete MV-network.
We will never consider the complete MV-network as a whole within this thesis. Instead, we will focus on the smaller networks of secondary substations that are powered by the same substation. From now on an MV-network will refer to these smaller networks and not the complete network.

We already mentioned that a secondary substation is connected to one unique substation. We may assume even more. An MV-network is radial.

Definition 2.3.1 (Radial)**.** An MV-network is *radial* when for each secondary substation there is precisely one way to get power from the substation.

So for any secondary substation it is always clear through which distributions lines it is connected with the substation. This gives rise to the definition of a route.

Definition 2.3.2 (Route)**.** A *route* consists of all the secondary substations that can be reached through one distribution line connected to the substation.

So each distribution line departing from the substation defines a route. Each secondary substation in this route gets its power through this distribution line. Within a route there can be multiple traces.

Definition 2.3.3 (Trace)**.** The (secondary) substations and distribution lines on the path from the substation to an endpoint of a route together are called a trace.

2.3.2 Mathematical represention

The MV-network as described above can be translated to a simple graph $G = (V, E)$. Here we have

$$\begin{aligned} V &= \{\text{(secondary) substations of the MV-network}\} \\ E &= \{\text{distribution lines of the MV-network}\} \\ V \ni S &= \text{ the substation of the network.} \end{aligned}$$

This graph captures the topology of the network. The property that the network is radial has a mathematical counterpart.

Theorem 2.3.1. *Let G be the corresponding graph to an MV-network. If the MV-network is radial, then G is acyclic.*

Proof. Suppose that G contains a cycle C. Take $v \in C$, such that $v \in V \backslash \{S\}$. This v exists, since G is simple and therefore a cycle concists of at least three vertices. There are two possibilities for S:

- $S \in C$: Since both are in C, there are two distinct paths from S to v. Contradicting that the network is radial.
- $S \notin C$: C contains at least one more vertex other then v. Let $u \in C$, $u \neq v$. Let p be the unique path from S to v. There are two options, namely $u \in p$ or $u \notin p$.

  $u \in p$ Since u, v in a cycle, there are two paths between them. We could therefore replace the connection between u and v in path p with a different solution, contradicting that p is unique.

  $u \notin p$ There are now two paths to u, since we take p to reach v and then can use the cycle to get to u in two different ways. This again contradicts the fact that N is radial.

Since both situations contradict the fact that the network is radial, this possibility is not possible.

Since both the possibilities lead to a contradiction, we conclude that G is acyclic. □

Since we assumed all MV-networks to be radial, we may now assume the corresponding graphs to be acyclic.

Definition 2.3.4 (Tree)**.** A graph $G = (V, E)$ is called a *tree* when there is a unique path between any two vertices of the graph

Theorem 2.3.2. *The following statements are equivalent for a connected graph $G = (V, E)$*

i) G is a tree

ii) G is acyclic

iii) It holds that $|E| = |V| - 1$

Proof. See Proposition 4.1 and 4.3 of [5] □

From Theorem 2.3.2 it directly follows that a graph G representing a radial network is a tree. The following proposition immediately follows from Theorem 2.3.2 and will be useful later on.

Definition 2.3.5 (Forest)**.** G is a *forest* when any two vertices are connected by at most one path.

Proposition 2.3.1. *Let $G = (V, E)$ be a forest, then the following holds: G has k connected components $\iff |E| = |V| - k$.*

The definitions of a route and trace can also be captured mathematically.

Definition 2.3.6 (Route)**.** Each edge that has S as an endpoint defines a route. For $e' = (S, u)$ its *route* consists of all vertices v and edges e of the unique path p between S and v, for which e' in p.

Intuitively one might think of routes as the connected components that remain after removing S from G.

Definition 2.3.7 (Trace)**.** A *trace* is the unique path from S to a leaf $v \in V$.

Example 2.3.1 (Basic network)**.** Suppose that we have a network N with two routes containing secondary substations $\{A, B, C, D, E, F\}$ and $\{G, H, I, J\}$ respectively and four traces $\{A, B, C, D, E\}$, $\{A, B, C, D, F\}$, $\{G, H, I\}$, $\{G, H, J\}$. The mathematical representation is shown in Figure 2.2.

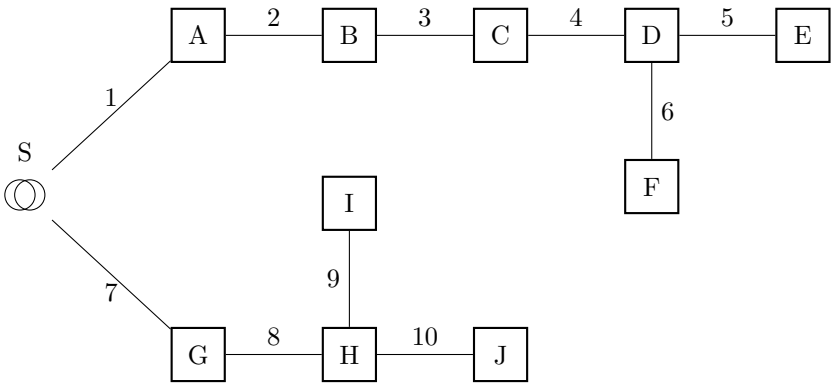


Figure 2.2: Representation of a basic MV-network

Note that we denote the substation by as shown in Figure 2.2.
It's called the basic network, since it doesn't include any secondary components yet, these will be discussed later on.

2.4 Secondary components in the MV-network

Alliander can place secondary components into their network. These help to reduce the customer minutes lost in case of an outage. Below we will discuss the different components that can be placed into a network. But first we need to understand what happens in case of an outage.

2.4.1 Malfunctioning distribution line

A malfunctioning distribution line in the MV-network can be compared to a short-circuit in a home. When an appliance (distribution line) short fuses; the group (route) of the home (network) will be without power. By trial and error we find the appliance (distribution line) that caused the problem. We disconnect it from the power and restore power to all other appliances in the group (route).

Notation We will make use of the mathematical representation described above to discuss the networks. With this it is important to note the (secondary) substations and vertices are equivalent, similarly distribution line and edge are equivalent. So a technician can move to vertex A, indicating that the corresponding secondary substation is visited. Similarly an edge can malfunction, indicating that the corresponding distribution line malfunctioned. The edge that corresponds with the malfunctioning distribution line is denoted by e_m.
The connection between a vertex v and edge e will be denoted by (v, e).

Example 2.4.1 (Restoring power)**.** Figure 2.3 shows the impact when we have $e_m = e_3$. The whole route $\{A, B, C, D, E, F\}$ is without power because of the malfunction.

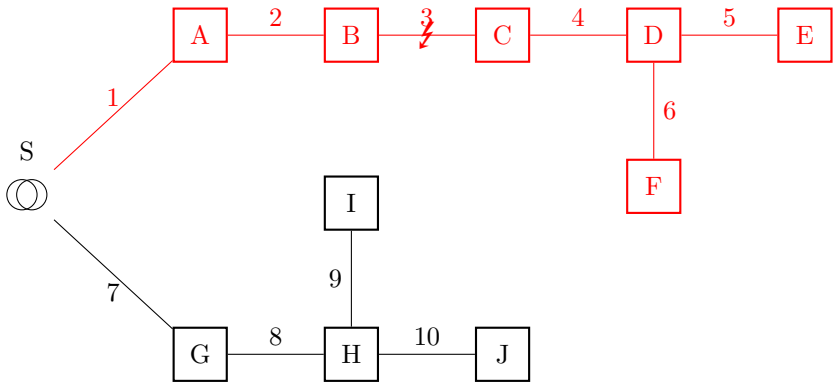


Figure 2.3: Impact when edge e_3 malfunctions

To resolve this outage a technician needs to determine the location of the fault by visiting different vertices. The technician can then determine at which side of the vertices the malfunctioning edge, e_m, is located. This is already quite time consuming, since it involves a lot of relocations for the technician.

When the malfunctioning edge is found it is disconnected from both its endpoints, effectively removing e_m from the graph. This again involves some movement of the technician, namely to both the endpoints of that edge. When this is done, power can be restored from S to vertices A and B. However vertices C, D, E and F cannot gain power from anywhere else, since there is no possible path from S to them. Instead we have to make use of a mobile power generator. This takes a lot of time to set up.

Figure 2.4 shows the result where all the customers are with power again.

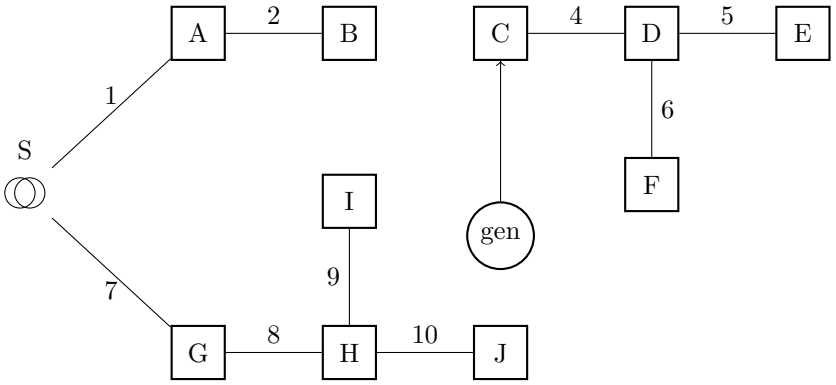


Figure 2.4: Solution to restore power when edge e_3 malfunctions

2.4.2 Secondary components

This section discusses the different secondary components that are placed into the MV-network of Alliander.

Net openings

It is very undesirable to only be able to resolve an outage using a generator. Therefore routes are connected to each other by distribution lines as a redundancy. These distribution lines are however disconnected at one of their endpoints. This way we still have a radial network. The part where we disconnect the edge from the vertex is called a *net opening* and is denoted with the symbol: ⚐. In case of an outage we can now connect the endpoint with this cable to restore power to (part of) the route. This will be referred to as *closing a net opening*. A *switching operation* refers to either the creation or closure of a net opening.
In practice, when a substation S stops delivering power to a route because of an outage a net opening has been made automatically between the substation and the route. So restoring power to the route comes down to closing that net opening again.

Example 2.4.2. We have added e_{11} between vertices C and J. The net opening is located at (C, e_{11}). When we again have $e_m = e_3$, this gives us the graph as shown in Figure 2.5. Note that we have also indicated the net opening at (S, e_1), which wasn't present in Figure 2.3.

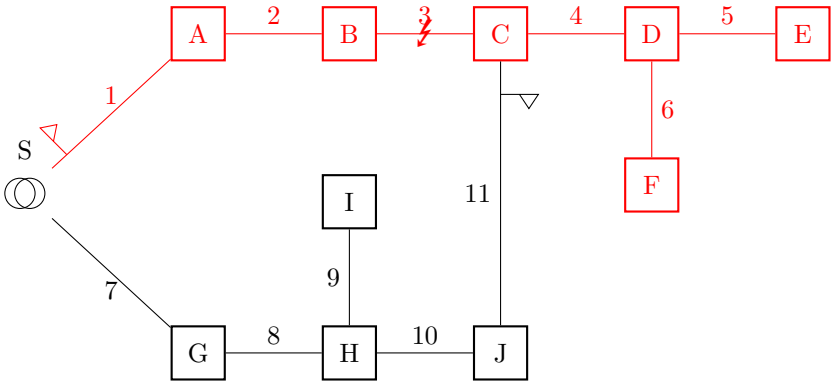


Figure 2.5: Impact when edge e_3 malfunctions

We still have no clue which edge is responsible for the outage. However after the technician has

found the edge and disconnected it from the route, it is now possible to restore the power to all customers without help of a generator. Vertices A, B can gain power by closing the net opening at (S, e_1). Vertices C, D, E and F can be restored by closing the net opening at (C, e_{11}).
Note that "disconnected it from the route" is nothing more than putting net openings at both end points of the edge. The resulting graph now looks like Figure 2.6.

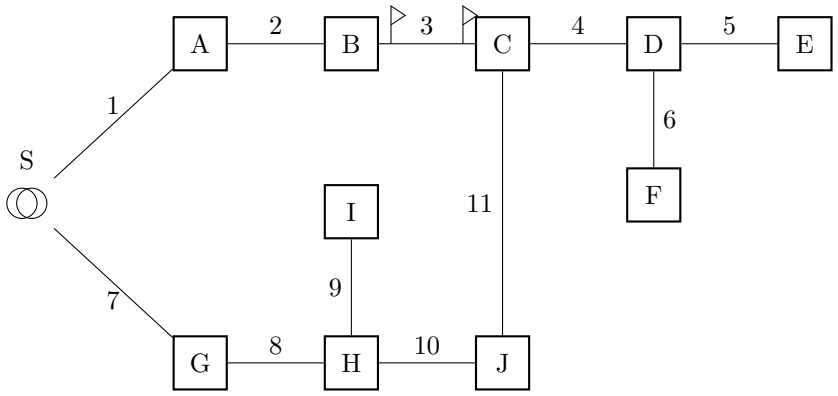


Figure 2.6: Outage resolved after closing two net openings

Fault passage indicator (FPI)

The net opening is useful to reduce the usage of generators. However we also want to decrease the search time for the malfunctioning edge. The first component that is used to improve this is called a *fault passage indicator*. This component is placed at the connection of an edge and vertex, denoted by  . It tells us remotely at which side of the component the malfunctioning edge is located.

Example 2.4.3. We have a similar network to before, but this time we have a fault passage indicator on e_4. This is shown in Figure 2.7.

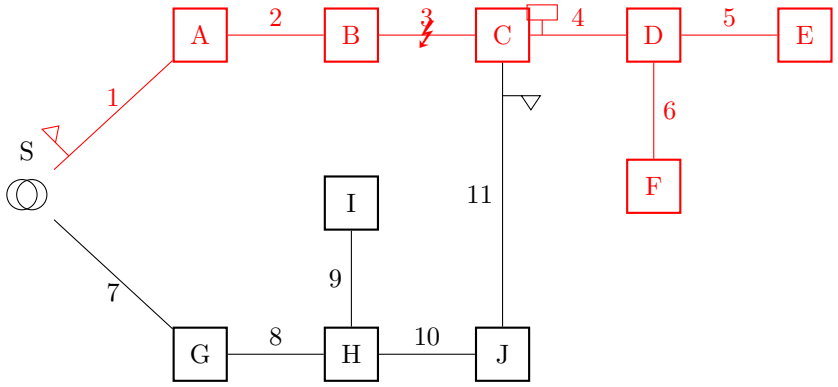


Figure 2.7: Impact when edge e_3 malfunctions

The resulting end configuration of the network is the same as Figure 2.6. However the way that we got there is quite different. A malfunction in e_3 now triggers the following actions:

1. Remotely get an update from the fault passage indicators. This information tells us that the malfunctioning edge must be to the left of C. So $e_m \in \{e_1, e_2, e_3\}$.

2. We now know that edges e_4, e_5 and e_6 are safe to be used. So we can already provide vertices C, D, E and F with power.

To accomplish this we create a net opening at (C, e_3) and close the net opening at (C, e_{11}).

3. Only now we search for the exact malfunctioning edge. Note that the number of edges to be searched is halved in this setup.

4. Create a net opening at (B, e_3).

5. Restore power to vertices A, B through S.

Nowadays fault passage indicators are part of the network, but new ones are not often placed. There is an improved component compared to the fault passage indicator, called a *smart cable guard*.

Smart cable guard (SCG)

A smart cable guard consists of two components that are placed on connections between edges and vertices in the same trace. All the edges that are in the path between the two components are now monitored by the smart cable guard. When a malfunction occurs, there are two options:

- The edge is monitored by the SCG. The smart cable guard now reports which edge malfunctions. No searching on location is needed anymore.
- The edge is not monitored by the SCG. It now has a similar function as a fault passage indicator. It tells us on what side of the monitored path the malfunctioning edge is located.

The components of the smart cable guard are indicated by  .

Example 2.4.4. Figure 2.8 represents the same network as previously, with an SCG added at (A, e_2) and (D, e_4).

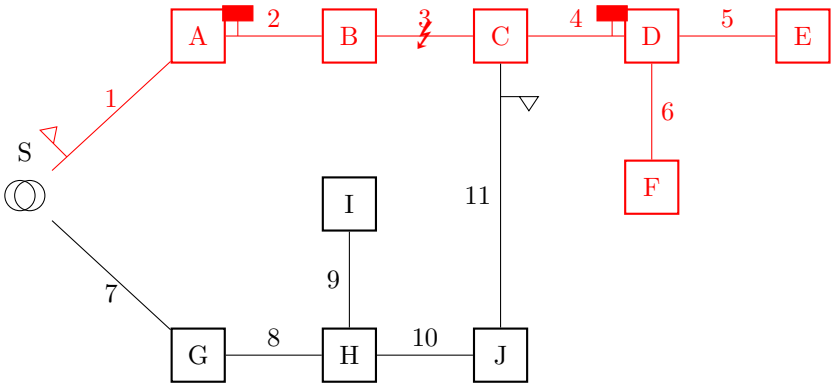


Figure 2.8: Impact when edge e_3 malfunctions

The result is again like Figure 2.6. The steps to get there are however again improved.

1. SCG immediately tells us that the malfunction happened is e_3.

2. The technician doesn't have to determine this location anymore. Instead we can immediately restore power to the two areas again. Which area gets restored first is often dependent on the time it requires to perform the necessary actions and the number of customers in each area.

The SCG has eliminated a lot of movement steps for the technician, reducing the minutes before customers have power again, greatly reducing the expected CML for this network.

Smart secondary substation (sSS)

Each opening or closure of a net opening involves a movement to the corresponding vertex by the technician. This can be reduced by implementing smart secondary substation in the network. These can open/close net openings based on remote commands, thereby reducing the need for the mechanic to move around and reducing the time needed to make these switches. These vertices are indicated by ⬠.
In addition smart secondary substations are equipped with a fault passage indicator on all the connected edges.

Example 2.4.5. Figure 2.9 shows a network in which we have added a smart secondary station at vertex C.

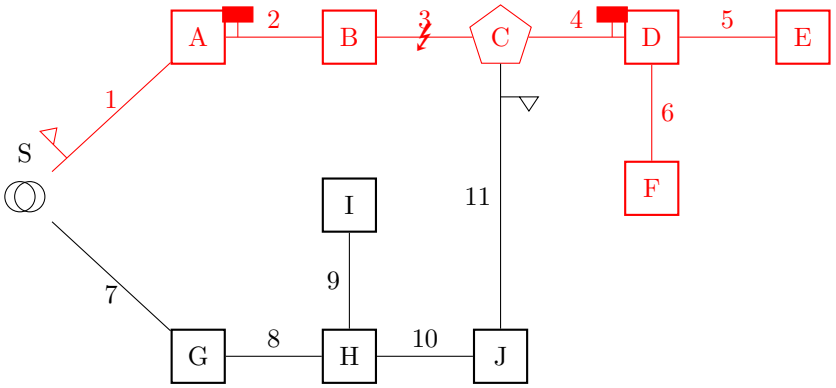


Figure 2.9: Impact when edge e_3 malfunctions

The steps to resolve the outage are analogous to Example 2.4.4. The difference with Example 2.4.4 is the time needed to execute the steps. Since C can be controlled remotely, we can create a net opening at (C, e_3) and close the net opening at (C, e_{11}) without any movement. The technician now only needs to create a net opening at (B, e_3) and close the net opening at the substation.

Circuit breaker

Thus far we have only reduced the restoring time. A circuit breaker is used to minimize the affected area in case of an outage. It acts much like a fuse. Let p be the path from S to the e_m. The circuit breaker in p that is closest to e_m will be triggered and create a net opening on that spot, therefore saving part of the route from any outage at all. We denote a circuit breaker with ■.
Note that a circuit breaker also helps in minimizing the possible area of the malfunctioning edge. Any edge that is still functional can never be the malfunctioning edge.

Example 2.4.6. Figure 2.10 show the situation as we had before, this time with a circuit breaker at (B, e_3).
Figure 2.11 shows the situation after the malfunction happens at e_3. Note that the circuit breaker has become a net opening. Furthermore vertices A and B never experience any outage.
The final configuration is again like Figure 2.6. To resolve this outage the technician doesn't even have to move! The vertices that are hit by the outage can now all be restored remotely by switching at vertex C as previously described.

Overview These are all the different components that can be placed in the MV-network of Liander. The last example is a good example of how the components can work together.

- The circuit breaker minimizes the impact of the malfunction.

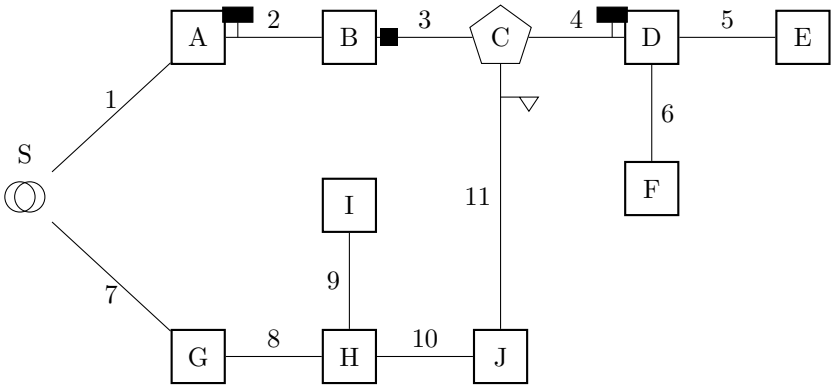


Figure 2.10: Network configuration without an outage

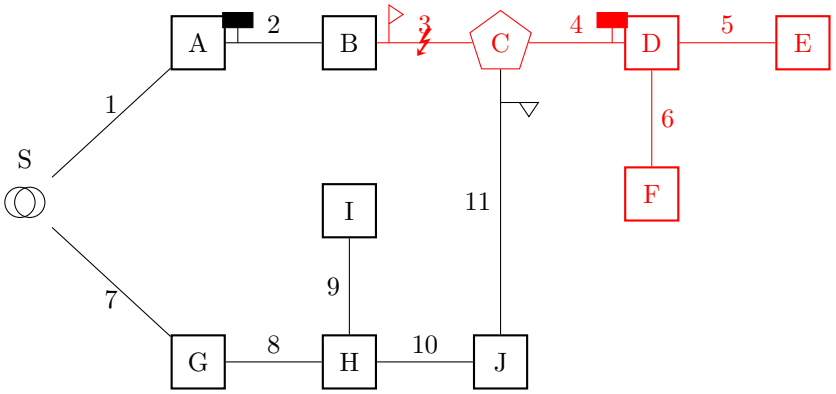


Figure 2.11: Impact when edge e_3 malfunctions

- The smart cable guard tells us which edge malfunctions.
- The net opening give us flexibility in the network to restore power.
- The smart secondary substation minimizes the need to move between different vertices.

An overview of all the different secondary components is shown in Table 2.1.

Component	Function	Notation
net opening	flexibility of the network	
circuit breaker	protect part of the route	
fault passage indicator	fault location detection	
smart cable guard	fault location detection	

Table 2.1

2.4.3 Representation of secondary components

Thus far we have shown the components graphically. But adding them mathematically to our graph representation quickly becomes cumbersome. We constantly need to point towards the connection between vertices and edges.

Instead we can switch to a different graph representation of the network. With this representation it becomes easier to represent all the components in the network. Within Alliander a variant on this notation is used. Below we will briefly discuss this representation. A more detailed description can be found in [2].
We can translate an MV-network into a rooted tree $T = (V, E, s)$ as below

$$
\begin{aligned}
V_v &= \{\text{(secondary) substations in } N\ \} \\
V_e &= \{\text{distribution lines of } N\ \} \\
V &= V_v \cup V_e \\
E &= \{\text{connections between (secondary) substations} \\
&\qquad \text{and distrubution lines}\} \\
V \ni s &= \text{ the substation of } N
\end{aligned}
$$

Intuitively we can think of it as the previous representation, but this time we put an extra vertex at the center of each edge. This way edges represent the connection between distribution lines and secondary substations. Precisely the location where secondary components are often placed.

Example 2.4.7. Figure 2.12 shows the original graph representation of the basic network discussed earlier. The same network represented in the new format is shown in Figure 2.13.

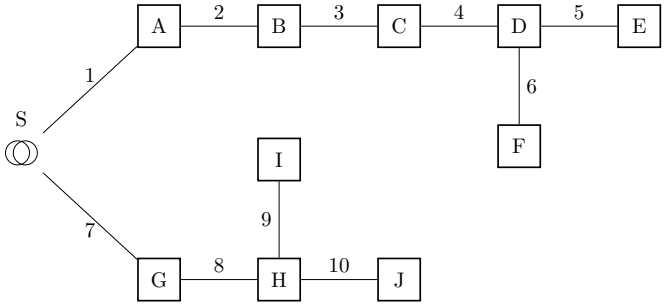


Figure 2.12: Original representation of our example basic network

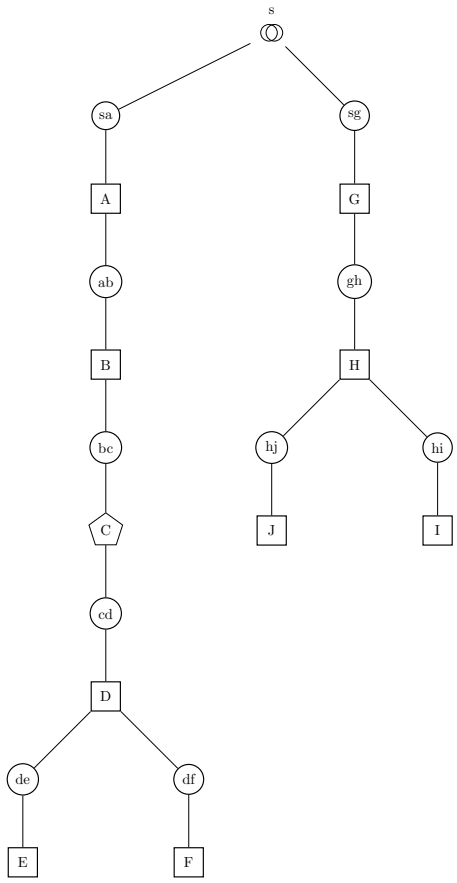


Figure 2.13: New representation of our example basic network

Representing components

The new representation does a better job at capturing all the different components. Almost all the secondary components were placed on the connection between a (secondary) substation and a distribution line. In the new representation, these are just the edges of the graph.
Below we describe for each component how we can indicate a component.

Net openings A net opening comes down to disconnecting a distribution line from a secondary substation. This is equivalent to the removal of that edge in T.

Circuit breaker Attribute that can be placed on the corresponding edge.

Fault passage indicator Similar to the circuit breaker, we just place an attribute on the corresponding edge.

Smart cable guard Represented by a tuple $(e_1, e_2) \in E \times E$, where e_1 is the start location of the SCG and e_2 the end.

sSS This is an attribute on a vertex v , more specifically on the vertices in V_v.

Definition 2.4.1 (Decorated rooted tree)**.** A network and all of its components can now be captured in a *decorated rooted tree* $T = (V, E, s, E_{CB}, E_{FPI}, E_{SCG}, V_{sSS})$. Here we have V, E, s as before and

$$
\begin{aligned}
E_{CB} &= \{\text{set of edges that represent circuit breakers}\} \\
E_{FPI} &= \{\text{set of edges that represent fault passage indicators}\} \\
E_{SCG} &= \{(e_1, e_2) \in E \times E : e_1 \text{start of SCG}, e_2 \text{ endpoint of SCG}\} \\
V_{sSS} &= \{\text{set of vertices that represent smart Secondary Substations}\}
\end{aligned}
$$

Example 2.4.8. The graph representation of the network of Example 2.4.6 is given by Figure 2.14. Note the missing edge between cj and J because of the net opening.

The attributes of the decorated tree are are denoted by:

$$
\begin{aligned}
E_{CB} &= \{(B, bc)\} \\
E_{FPI} &= \emptyset \\
E_{SCG} &= \{((A, ab), (D, cd))\} \\
V_{sSS} &= \{C\}
\end{aligned}
$$

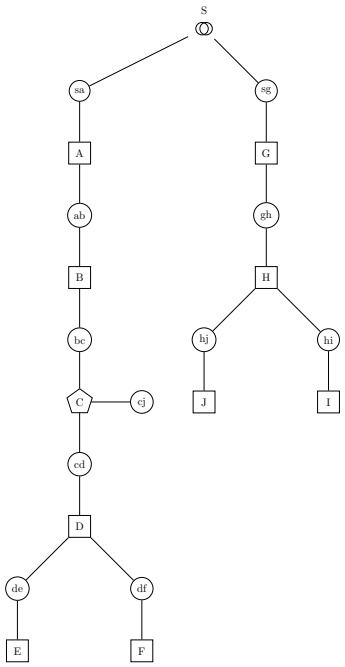


Figure 2.14: Decorated tree

Mostly use first representation Even though the representation mentioned above is more thorough, we will stick with the representation discussed in Section 2.3.2. The secondary components discussed above are important to understand the working of the Outage Simulation Module. However most of this thesis will focus on the Switch Planner inside this module. This Switch Planner only needs to take into account net openings and none of the other components. The current implementation uses the representation of Section 2.3.2 and we have stuck with it. Apart from one small annoyance in Section 5.2, there has been no reason to switch to the more complete representation.

Chapter 3

Linear Programming

Later on we will focus on extending a Mixed Integer Linear Program. This section discusses some general knowledge about (Mixed Integer) Linear Programming.
Linear Programming is a widely used scheme within optimization theory. It can be viewed as a model which fits a lot of problems from different areas. Even if a problem does not fit in it exactly, it can often be *relaxed* to make it fit. Apart from optimization in Mathematics, it is also used in Economics. So much that the founders of Linear Programming, Leonid Kantorovich and Tjalling Koopmans, have been awarded the Nobel Prize of Economics in 1975 for their research in this area.[7]. Below we give a short overview of (Mixed Integer) Linear Programming, based on [3].

Canonical form

Definition 3.0.1 (Linear Program)**.** A Linear Program in canonical form is defined as:

$$\begin{aligned} &\text{min} && c^T \cdot x \\ &\text{subject to} && A \cdot x \leq b \\ & && x \geq 0 \end{aligned}$$

where $c, x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. We call x the *decision variable* of the linear program, $c^T \cdot x$ the *objective function* and we have m *constraints* that tell us that $a_{i,j} \cdot x_j \leq b_i$ for $1 \leq i \leq m$.

It's called a linear problem, since both the objective functions and constraints are linear with respect to the decision variable x.

While the above definition might seem a bit restrictive in some sense, in practice we can also handle cases where e.g., $x < 0$ or $a_{i,j} \cdot x_j \geq b_i$ for some $1 \leq i \leq m$.

Solutions to Linear Programs

There exist a lot of solvers that can find the optimal solution of a Linear Program. These all make use of some of the key properties of linear problems.
Assuming that a linear problem has at least one solution, we have two key properties:

- The objective is a linear function, so it's both concave and convex. Therefore any local optimum is also a global optimum.
- The linear constraints together define a convex polyhedron which contains all the possible feasible solutions.

From this it can be shown that any optimal solution must be located at any of the corners of the convex polytope. It has been proven that it can be solved in polynomial time. A common method is the Simplex Method.

Simplex Globally the Simplex Method works in two phases

Phase 1 Find a starting corner of the polytope

Phase 2 Walk along the edges of the polytope towards a better solution

Theoretically the second phase could end up cycling, making the Simplex Method worst-case exponential. This only happens for specific constructed problems and methods to deal with this exist. In practice it solves problems efficiently.

3.1 Mixed Integer Linear Programming

A Linear Program assumes all the decision variables to be real numbers. However we might want some of them to be integers instead. It turns out this increases the difficulty of the problem immensely.

Definition 3.1.1 (Mixed Integer Linear Program)**.** A Mixed Integer Linear Program in canonical form is defined as:

$$\begin{aligned} &\min && c^T \cdot x \\ &\text{subject to} && A \cdot x \leq b \\ & && x \geq 0 \end{aligned}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Let $I = \{1, \ldots, n\} = I_{\mathbb{Z}} \cup I_{\mathbb{R}}$, with $I_{\mathbb{Z}} \neq \emptyset$ and $I_{\mathbb{Z}} \cap I_{\mathbb{R}} = \emptyset$. Then for x_i, $i \in I$, it holds that $x_i \in \begin{cases} \mathbb{Z} & i \in I_{\mathbb{Z}} \\ \mathbb{R} & i \in I_{\mathbb{R}} \end{cases}$. So some of the x_i are integers, while others might be reals.

The only difference is that we restrict some of the decision variables to be integers. The consequences of this change however are quite big. There no longer exists a polynomial time algorithm to find the solution, if such a solution exists.
The reason that Mixed Integer Linear Programs cannot be solved polynomially is caused by the fact that the feasible solutions no longer form a convex polyhedron. Solutions are therefore not necessarily at the corners of such a polyhedron.

Theorem 3.1.1. *Solving a Mixed Integer Linear Program is NP-hard*

Proof. We make a reduction from the `Partition`-problem. This problem tries to answer whether there exists a set I_{Part} such that for a set $S = \{s_1, \ldots, s_n\}$, $s_i \in \mathbb{Z}$, we have $I_{\text{Part}} \subset I = \{1, \ldots, n\}$ with $\sum_{i \in I_{\text{Part}}} s_i = \frac{1}{2} \sum_{i \in I} s_i$.

Let S be such a set, we define $S_{\frac{1}{2}} = \frac{1}{2} \sum_{i \in I} s_i$. We define the (Mixed Integer) Linear Problem:

$$\begin{aligned} &\max && 0 && \\ &\text{subject to} && s_i \cdot x_i = S_{\frac{1}{2}} && \forall i \in I \\ & && x_i \leq 1 && \forall i \in I \\ & && x \geq 0 && \\ & && x \in \mathbb{Z}^n && \end{aligned}$$

If the problem above finds a solution x^*, we can take $I_{\text{Part}} = \{i : x_i^* = 1\}$ to divide the set as intended. If the problem is infeasible, we know it is not possible to divide the s_i in a correct manner, showing that I_{Part} does not exist. We are now capable of solving the `Partition`-problem.
The translations to and from the Mixed Integer Linear Problem are polynomial. Therefore if we can solve Mixed Integer Linear Problems polynomially, we could also solve the Partition problem in polynomial time, contradicting the known fact that this problem is NP-complete.
This shows that solving a Mixed Integer Linear problem is NP-hard. □

Solving MILP

Algorithm 1 shows a typical way of solving a MILP P, where $P' = \texttt{Relax}(P)$ is the relaxation of P. It's equivalent to P but as a linear program, so now all $x_i \in \mathbb{R}$.

Algorithm 1 Basic branch method to solve an Mixed Integer Linear Program

Input: (Mixed) Integer Linear Program P
Output: Optimal solution x to P

```
1: Queue = {P'}, sol = ∅
2: while items in Queue do
3:     P is first item in Queue
4:     Remove P from Queue
5:     x̂ = solve(Relax(P))
6:     if there exists an x̂_i such that x̂_i ∉ ℤ and i ∈ I_ℤ then
7:         Queue = Queue ∪{P + (x_i ≤ ⌊x̂_i⌋)}
8:         Queue = Queue ∪{P + (x_i ≥ ⌈x̂_i⌉)}
9:     else if x̂ best solution so far then
10:        sol = x̂
11:    end if
12: end while
13: return sol
```

Intuitively we find a solution $\hat{x}$ to the Linear Program P'. Pick an $\hat{x}_i$ that should be an integer but is not. We create two new problems. One where we add the constraint $x_i \leq \lfloor \hat{x}_i \rfloor$ and another where we add $x_i \geq \lceil \hat{x}_i \rceil$. Repeat until we find the solution to the original MILP.
How the solver branches can be of great influence on the total solving time. Picking the right $\hat{x}_i$ to set bounds on in each step is not a trivial choice and could be worth investigating. This is however not the goal of this thesis. We will focus on extensions of Mixed Integer Linear Programs and not the possible tweaks to a solver.

Chapter 4

Current implementation for simulating outages

We now have the prior knowledge to discuss the current implementation for simulating outages. This is done by the Outage Simulation Module. Section 4.1 discusses the workings of this module, together with some examples. The module makes use of the Switch Planner to determine how power can be restored to different parts of the network. The Switch Planner is a Mixed Integer Linear Program that is discussed in detail in Section 4.2. The information within this chapter is based on [6].

4.1 Outage Simulation Module

We want to be able to determine the expected customer minutes lost of a network. We saw in Chapter 2 that we need to determine the customer minutes lost in case of a failure in every distribution line. This is possible using the Outage Simulation Module. The module simulates the process of restoring power to all customers after an outage in case of a failure in one of its distribution lines. This can then be combined to determine the total expected customer minutes lost of the network, as we have seen in Definition 2.2.2.
Before we can explain the inner workings of this module, we define two abstractions that capture the functionality of the discussed secondary components.

Remote observability and remote controllability

The different components in the network need to be taken into account by the module. This is done using two abstractions that together can capture all the components.

Definition 4.1.1 (Remotely observable)**.** An edge in the network is *remotely observable* when in case of an outage the control room remotely gets an update at which side of the edge the outage is located.

All the components discussed in Section 2.3 that help with narrowing the location of the fault can be translated to this new definition.

- A fault passage indicator is the most basic component that adds remote observability to an edge. It reports at which side of the component the outage is located.
- A smart cable guard does more. For edges outside its monitored segment it behaves just like a fault passage indicator. However for edges inside this segment it tells exactly which edge is the cause of the outage. This is the same as assuming remote observability on each edge of the segment.

- A circuit breaker is not remotely observable by itself, but it does induce remote observability on the edge. The control room can remotely determine which customers still have power. This can be used to determine which circuit breaker was activated and at which side of the component the failure is located.

The second abstraction is related to performing switching operations remotely.

Definition 4.1.2 (Remotely controllable)**.** An edge is *remotely controllable* when the control room can perform switching operations remotely on it.

Edges connected to a smart (secondary) substation are precisely the edges that are remotely controllable.

4.1.1 Overview

As mentioned above, the goal of the Outage Simulation Module is to determine the customer minutes lost of an outage. For this we need to determine how to restore power to the affected customers as quickly as possible.
The module creates an ordered list of switching operations that restores power to the customers. This is then combined with default times for each operation. The list is generated using different steps within the module, as shown in Figure 4.1. Some output of the Outage Simulation Module to example networks is given in Section 4.1.2.

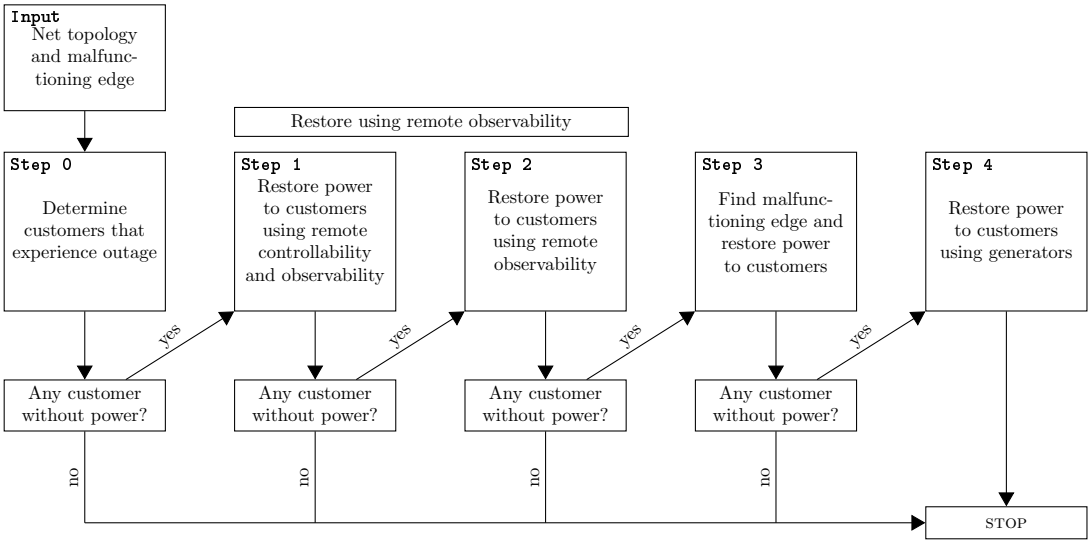


Figure 4.1: Flowchart of the Outage Simulation Module

Input

The input for the module consists of a graph $G = (V, E)$ that contains the net topology and the edge $e_m \in E$ that malfunctions.

Step 0

A circuit breaker inside a route can protect parts of this route in case of failure. When an outage occurs a circuit breaker is automatically triggered and only a part of the route experiences outage. In the worst case the circuit breaker at the substation is triggered and the whole route is without power.

This step determines which circuit breaker would be triggered in the real world scenario. This gives us an overview of all the customers that are without power.

Restore using remote observability

The previous step gives us information on the customers that are without power. Next we can use remote observability to determine the possible location of the fault. When an edge fails, we can expect certain components to be triggered. Based on which components are actually triggered, the control room can limit the possible locations of the malfunctioning edge. This also gives us certain parts of the network that are without power but cannot possibly be the cause of the outage. Each connected component that is certainly not the cause of the outage is called a *subnet*.
Restoring power to these subnets is the focus of Step 1 and Step 2. The subnet first needs to be *isolated* from the malfunctioning edge before we can restore power to the subnet. To do this, we place a net opening between the possible fault location and the vertex that is closest to it. Next we try to restore power to the subnet.

Step 1 First we try to restore power without any movement from the technician. This means that we are only allowed to switch edges that are remote controllable. This includes the net opening that is needed to isolate the subnet.
To determine what switching operations to perform, we create a Mixed Integer Linear Program for the subnet. The details on this translation are described in Section 4.2. If the Mixed Integer Linear Program finds a solution we can restore power to all the customers in the subnet while adhering to the capacities of all the distribution lines.

Step 2 The switches made in `Step 1` are all done remotely, so these will always be completed before the technician arrives at a secondary substation of the network. Therefore we continue with the result of the previous step and determine the remaining subnets.
Finding the switching operations to restore power to the subnet goes analogously to Step 1. The only difference is that we allow the Mixed Integer Linear Program to switch all the edges, instead of just the remotely controllable ones.

Step 3

If this step is used, then the exact location of the malfunctioning edge is not yet known by remote observability. We start with finding out which edge is responsible for the outage. For this we simulate the binary search that would occur in the real world scenario.
Once we have found the malfunction edge, the remaining subnets are restored analogously to before.

Step 4

Unfortunately, it is not always possible to provide all the customers with power again using the method above. The subnets that are without power after Step 3 are restored using estimates. There are two options for each subnet.

- The subnet cannot possibly be connected to a power source, since there is no connected net opening. In this case we need a mobile power generator, as we have seen in Figure 2.4. This is very time consuming, we assume that it takes 180 minutes to restore power to the customers.
- There are possibilities to feed the subnet using net openings, but because of capacity constraints we could not feed this subnet in Step 1, 2 or 3. In this case we assume it takes 4 extra switching operations to restore power to the remaining customers.

Any customers without power?

At the end of each Step the module checks whether we restored power to all the customers. When this is the case, we can stop and determine the customer minutes lost of the outage.
A typical example where we skip Step 3 and 4 is when the malfunctioning section is monitored by a smart cable guard. The exact location is already known, therefore all customers often have power after Step 2.

Stop

Power is restored to all customers in the network.

4.1.2 Determining customer minutes lost of an outage

Given the switching operations, it is now possible to determine the customer minutes lost of the outage. Each movement of the technician takes a certain time. So after each switching operation it is clear which customers have power again and how much time it took. Combining this for all the customers in the affected area, gives us the customer minutes lost.

Example

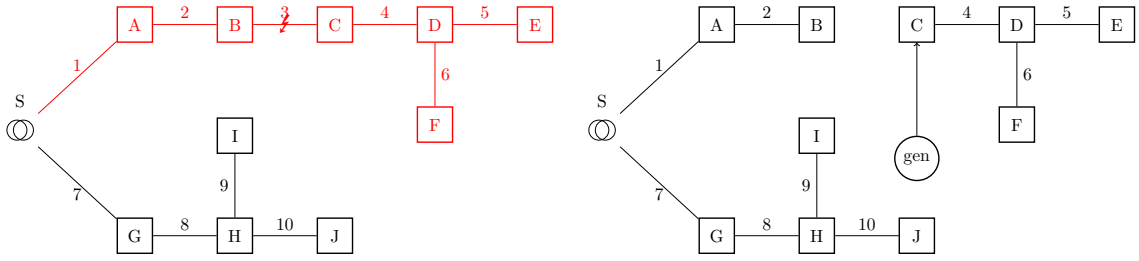
In Section 2.4 we already discussed some solutions to outages. We will repeat some of these again, this time using the Outage Simulation Module. Below we see an overview of the output of the Outage Simulation Module, This can be translated to the customers minutes lost, using the following dummy values.

Action	Time (in minutes)
The switching operation at the first location	40
Other switching operations by technician at a location	10
The very first switching operation is remotely controlled	10
Other remotely controlled switching operations	0
Binary search	20
Provide power using mobile power generator	180

We assume each vertex to have 1 customer.

Notation *close* (v, e) denotes the closure of a net opening at this location. Similarly *open* (v, e) denotes the creation of a net opening.

Basic network



(a) Situation (b) Solution

Figure 4.2: Outage because of edge e_3 in an empty network

Table 4.1 shows the result given by the Outage Simulation Module. Note that we have no remote observability in this network.

Step	Action	Cumulative time	Restored vertices	Vertices without power
0	Circuit breaker (S, e_1)	0		A, B, C, D, E, F
3	Binary search	40 + 20[1]		A, B, C, D, E, F
3	Open (B, e_3)	70		A, B, C, D, E, F
3	Close (S, e_1)	80	A, B	C, D, E, F
4	Open (C, e_3)	90		C, D, E, F
4	Place generator (C,gen)	270	C, D, E, F	

Table 4.1: Output of the outage simulation module for Figure 4.2

The customer minutes lost of the outage now becomes

$$\begin{aligned} \text{CML}_{O_{e_3}} &= 2 \cdot 80 + 4 \cdot 270 \\ &= 1240. \end{aligned}$$

Fault passage indicator and net opening in network

The network in Figure 4.3 adds remote observability by a fault passage indicator. Notice that we now make use of Step 2 of the Outage Simulation Module. Furthermore there is a net opening between C and J. This way, we no longer use Step 4.

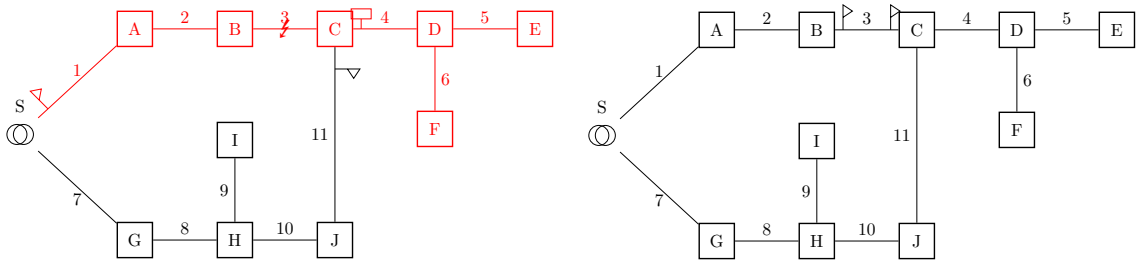


Figure 4.3: Outage because of edge e_3 in an network with a Fault passage indicator

Step	Action	Cumulative time	Restored vertices	Vertices without power
0	Circuit breaker (S, e_1)	0		A, B, C, D, E, F
2	Open (C, e_3)	40		A, B, C, D, E, F
2	Close (C, e_{11})	50	C, D, E, F	A, B
3	Binary search	70		A, B
3	Open (B, e_3)	80		A, B
3	Close (S, e_1)	90	A, B	

Table 4.2: Output of the outage simulation module for Figure 4.3

The customer minutes lost of the outage now becomes

$$\begin{aligned} \text{CML}_{O_{e_3}} &= 4 \cdot 50 + 2 \cdot 90 \\ &= 380. \end{aligned}$$

[1]The technician needs to visit vertices to perform the binary search. Reaching the first vertex takes 40 minutes, from that point onward it takes 20 minutes to perform the actual binary search.

Smart cable guard, smart secondary substation and circuit breaker in network

The network in Figure 4.4 shows a network with a lot of components. We have a circuit breaker, smart cable guard and smart secondary substation. The circuit breaker protects part of the route, while the smart cable guard directly determines the correct e_m. We see that all customers are with power again after Step 1, skipping the other steps of the Outage Simulation Module.

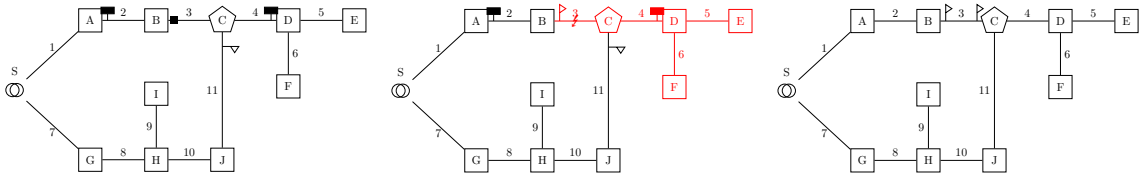


Figure 4.4: Outage because of edge e_3 in well monitored network

Step	Action	Cumulative time	Restored vertices	Vertices without power
0	Circuit breaker (C, e_3)	0		C, D, E, F
1	Open (C, e_3)	10		C, D, E, F
1	Close (C, e_{11})	10	C, D, E, F	

Table 4.3: Output of the outage simulation module for Figure 4.4

The customer minutes lost of the outage now becomes

$$\begin{aligned} \text{CML}_{O_{e_3}} &= 4 \cdot 10 \\ &= 40. \end{aligned}$$

4.2 Switch Planner

In the examples we have seen so far, determining how to restore power was straightforward. There was one option to restore a subnet and we always assumed it could feed the extra vertices. This is not always the case. It is sometimes necessary to split a subnet and use multiple net openings to feed each remaining part because of capacity constraints.
This is where the Switch Planner comes into play. It is a Mixed Integer Linear Program for which the solution tells us how to feed the subnet again. We have already seen that Mixed Integer Linear Problems are NP-hard. We first take a look at the complexity of the problem to justify the use of this method. After that we discuss how the translations works.

Complexity of the problem

The problem of feeding power to the whole subnet should be hard, otherwise using a Mixed Integer Linear Program to solve it might be overkill. We should then instead find other algorithms to solve it. From [9] we see that the problem is actually NP-hard by reduction from the `Partition`-problem. With this knowledge the usage of a Mixed Integer Linear Program fits nicely. Now we only need to worry about the translation of the problem. After that the available solvers will find the optimal solutions.

Translating the problem however is not trivial. A graph contains all kind of correlations that cannot be captured by linear constraints. This can be circumvented by precomputing some data for the constraints.

4.2.1 Data preparation

When we want to feed a subnet N again, We define the following variables that will be useful later on. Here $G = (V, E)$ represents the MV-network. Example values of the defined variables are discussed at the end of this section.

$$\begin{aligned} E_{no} &= \{e \in E : e \text{ a net opening connected to subnet } N\} \\ E_1 &= E_{no} \cup \{e \in E : e \text{ a closed edge in subnet } N\ \} \\ E_2 &= \{e \in E \backslash E_1 : e \text{ part of a path that can feed the subnet}\} \\ E_{op} &= \{e \in E_1 : e \text{ is operable}\} \\ V_N &= \{v \in V : v \text{ in subnet } N\ \} \\ V_0 \ni v_0 &= \text{vertex closest, in number of edges, to the malfunctioning edge} \end{aligned}$$

A net opening is connected to the subnet N when closing the net opening would result in a closed path from the substation to the subnet. Furthermore, any edge in E_1 is *operable* in Step 2 and Step 3 of the Outage Simulation Module, however within Step 1 only the edges connected to a remote controllable vertex are operable.

We also have two functions that give information on the edges and vertices. Using the rescap function we define another variable.

$$\begin{aligned} \text{load} :& V \to \mathbb{R}, & \text{load}(v) =&\ \text{electricity load of vertex } v \\ \text{rescap} :& E \to \mathbb{R}, & \text{rescap}(e) =&\ \text{residual capacity for edge } e \\ f =& \max(1000, 1000 \cdot \max_{e \in E}(\text{rescap}(e))) \end{aligned}$$

The *residual capacity* of an edge e is equal to the maximum capacity of that edge minus the capacity that is already in use in the current configuration. Why we use the definition of f will become clear later on, it is important to remember that $f > 0$ and always larger than the residual capacity at any edge.

Lastly we need to precompute all the paths that can possibly feed a vertex in the subnet.

$$\forall no \in E_{no}, v \in V_N : \quad \text{path}_v^{no} = \begin{matrix} \{e \in E_1 : e \text{ in path from } no \text{ to } v, \text{ including } no\} \cup \\ \{e \in E_2 : e \text{ in path from substation to } no\} \end{matrix}$$

This path is used to define the following function.

$$\alpha : E \times E_{no} \times V_N \to \{0, 1\} \qquad \alpha(e, no, v) = \begin{cases} 1 & \text{if } e \in \text{path}_v^{no} \\ 0 & \text{otherwise} \end{cases}$$

Decision variables

The above variables and sets are all fixed and are used to formulate the constraints. To come to a solution the Mixed Integer Linear Program will have the following decision variables.

$$\forall e \in E_{op}: \quad c_e = \begin{cases} 1 & \text{if edge } e \text{ is closed} \\ 0 & \text{if edge } e \text{ a net opening} \end{cases}$$

$$\forall no \in E_{no}, v \in V_N: \quad p_v^{no} = \begin{cases} 1 & \text{if vertex } v \text{ is powered by a path that contains } no \\ 0 & \text{otherwise} \end{cases}$$

S_e = the extra electrical strain on e in the result

ρ = residual capacity at the net opening that isolates the subnet from e_m

An edge is *closed* when it does not contain a net opening at any of its endpoints. Why we need ρ as a decision variable will become clear by constraint F and Section 4.2.7

How these decision problems capture the whole problem should become clear later on. First we define all the variables for a small example subnet.

Example

We have slightly expanded our example network thusfar, as shown in Figure 4.5. The subnet we want to restore power to is highlighted and consists of vertices C, D, E, F.

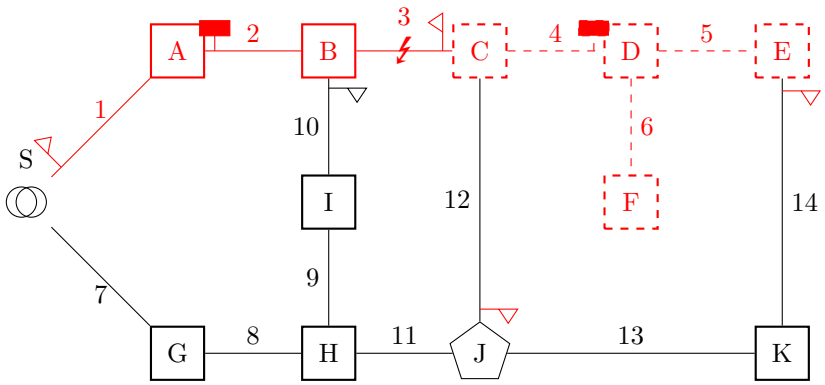


Figure 4.5: Malfunction in edge e_3.

Values

Note that the the Switch Planner is used by three different steps. Most variables stay the same independent of the step, however E_{op} will be different depending on the step.
The variables become

$$E = \{e_1 \dots e_{14}\} \qquad V_N = \{C, D, E, F\}$$
$$E_{no} = \{e_{12}, e_{14}\} \qquad v_0 = C$$
$$E_1 = \{e_4, e_5, e_6, e_{12}, e_{14}\}$$
$$E_2 = \{e_7, e_8, e_{11}, e_{13}\}$$
$$E_{op} = \begin{cases} \{e_{12}\} & \text{if Step} = 1 \\ E_1 & \text{otherwise} \end{cases}$$

The precomputed data looks like

$$\text{path}_D^{e_{12}} = \{e_7, e_8, e_{11}, e_{12}\} \cup \{e_4\}$$
$$\text{path}_D^{e_{14}} = \{e_7, e_8, e_{11}, e_{13}, e_{14}\} \cup \{e_5\}$$

If we took the current situation as the result of the Mixed Integer Linear Program, then the values of the decision variables c_e and p_v^{no} become.

$$c_{e_{12}} = c_{e_{14}} = 0, c_{e_4} = c_{e_5} = c_{e_6} = 1$$
$$\forall no \in E_{no}, v \in V_N : \quad p_v^{no} = 0$$

A possible solution and the corresponding values for the example graph in Figure 4.5 will be discussed in Section 4.2.6.

4.2.2 Mixed Integer Linear Program

The Mixed Integer Linear Program that determines what switching operations to perform is defined below. Recall that we minimize over the possible values of the decision variables as defined in Section 4.2.1.

Objective function (See Section 4.2.3)

$$\min \quad \sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e$$

Constraints (See Section 4.2.4)

Ref	Constraint		Description
A	$\sum_{e \in E_{op}} (1 - c_e) = \lvert E_{no} \rvert - 1$		Number of edges
B	$\forall e \in \text{path}_v^{no} : \quad p_v^{no} \le c_e$ $\sum_{e \in \text{path}_v^{no}} c_e - p_v^{no} \le \lvert \text{path}_v^{no} \rvert - 1$	$\forall v \in V_N$, $no \in E_{no}$	Set p_v^{no} correctly
C	$\sum_{no \in E_{no}} p_v^{no} = 1$	$\forall v \in V_N$	Each vertex gets power by exactly one path
D	$S_e = \sum_{v \in V_N} \sum_{no \in E_{no}} \text{load}(v) \cdot p_v^{no} \cdot \alpha(e, no, v)$	$\forall e \in E_1 \cup E_2$	Set S_e correctly

E	$S_e \leq \text{rescap}(e)$	$\forall e \in E_1 \cup E_2$	Edges stay within their capacity
F	$\rho \leq \text{rescap}(e) - S_e + f \cdot \left(1 - \sum_{no \in E_{no}} \alpha(e, no, v_0) \cdot p_{v_0}^{no}\right)$	$\forall e \in E_1 \cup E_2$	Upper bound on the residual capacity at v_0

4.2.3 Objective function

The Switch Planner wants to know how to provide all the customers in the subnet with power with as little customers minutes lost as possible. This goal needs to be translated into a linear objective. We will see that one of the constraints forces the whole subnet to have power. So the objective has to pick the solution that does this with the minimal customers minutes lost.
As we have already seen, each switching operation at a new location costs time, since the technician has to move. Minimizing the number of switching operations that are needed to restore power therefore seems to be a good measure on the customer minutes lost of a solution.
This is exactly what the objective does

$$\min \quad \sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e,$$

since we have

- For $e \in E_{no}$ there are two options for the resulting value c_e.

  $c_e = 0$ Nothing is changed, since e is still a net opening. It does not affect the value of the objective at all.

  $c_e = 1$ The net opening has been closed. This is a switching operation and adds 1 to the objective value.

  Each net opening that is closed adds 1 to the objective, but zero when not switched. So if any net opening is closed without it being necessary for the constraints, then the objective value can be improved on by not closing that net opening.

- Similarly for all e that are not a net opening, we again have two possibilities.

  $c_e = 1$ The edge is still closed. Therefore the objective value is lowered by one.

  $c_e = 0$ The edge has now become a net opening, resulting in subtracting zero from the objective value.

  We see that when not switched the objective value will improve by 1, whereas switching will not change it. Analogous to above, any unnecessary switch will be removed from a solution

From above we can conclude that the objective function indeed tries to minimize the number of switches on any solution that fulfills the constraints.

4.2.4 Constraints

The constraints of the Mixed Integer Linear Program can be grouped into three different groups.

- Radial network - Constraint A, B, C,
- No overload on distribution lines - Constraints D, E,
- Upper bound residual capacity at isolation - Constraint F.

Radial network

The first constraints make sure that the network connected to the substation is radial and now contains the subnet. Note that the graph $G = (V, E)$ that represents the network configuration after an outage is a forest with 2 trees. One tree contains all the vertices that do not experience any outage. The second tree contains all the vertices that were affected. From Proposition 2.3.1 we know that if the number of vertices, edges and connected components stay the same, then the two resulting connected components are both trees. The number of vertices automatically stays the same. Constraint A makes sure that the number of edges stays the same, while constraint B and C make sure that the number of connected components stays equal.

Constraint A) Number of edges

This constraint makes sure that the result still has the same number of closed edges compared to the original situation. This is the case when the number of net openings in the result, $\sum_{e \in E_{op}} (1 - c_e)$, is the same as the original number of net openings, $|E_{no}|$.
Note that we already isolated the subnet from e_m by placing a net opening in the Outage Simulation Module. This net opening is not included in any of our sets, such as E_{op}. Therefore the number of new net openings must be one less than $|E_{no}|$ to keep the total number edges the same. This results in the constraint:

$$\sum_{e \in E_{op}} (1 - c_e) = |E_{no}| - 1$$

Constraint for the example subnet

$$\sum_{e \in E_{op}} c_e = 2$$

Constraint B) $p_v^{no} = 1$ if and only if the whole path via no to v is closed

This constraint makes sure that we set p_v^{no} correctly. Remember the definition:

$$\forall no \in E_{no}, v \in V_N : \quad p_v^{no} = \begin{cases} 1 & \text{if the path that feeds } v \text{ contains } no \\ 0 & \text{otherwise} \end{cases}$$

This means $p_v^{no} = 1$ iff all the edges in path_v^{no} are closed, since there is no other path possible that feeds v with no in it. But this is the same as stating

$$\forall no \in E_{no}, v \in V_N : \quad p_v^{no} = \min_{e \in \text{path}_v^{no}} c_e,$$

since c_e is binary.
Unfortunately most solvers don't allow min as a constraint, we therefore translate it into a standard linear constraint.

We require p_v^{no} to be at most the value of each c_e. Therefore if any of the $c_e = 0$, then $p_v^{no} = 0$.

$$\forall e \in \text{path}_v^{no} : \quad p_v^{no} \leq c_e$$

Secondly we need that if all the $c_e = 1$, then $p_v^{no} = 1$ as well. This is forced by

$$\sum_{e \in \text{path}_v^{no}} c_e - p_v^{no} \leq |\text{path}_v^{no}| - 1$$

Namely if all $c_e = 1$, then $\sum_{e \in \text{path}_v^{no}} c_e = |\text{path}_v^{no}|$. Forcing $p_v^{no} = 1$ to fulfill the constraint. If however at least one $c_e = 0$, then $|\text{path}_v^{no}| - \sum_{e \in \text{path}_v^{no}} c_e \geq 1$, allowing p_v^{no} to be zero.

Constraint for the example subnet

$$\begin{array}{lll} p_D^{e_{12}} \leq c_{e_7} & p_D^{e_{12}} \leq c_{e_8} & p_D^{e_{12}} \leq c_{e_{11}} \\ p_D^{e_{12}} \leq c_{e_{12}} & p_D^{e_{12}} \leq c_{e_4} & \end{array} \qquad c_{e_4} + c_{e_7} + c_{e_8} + c_{e_{11}} + c_{e_{12}} - p_D^{e_{12}} \leq 4$$

$$\begin{array}{lll} p_D^{e_{14}} \leq c_{e_7} & p_D^{e_{14}} \leq c_{e_8} & p_D^{e_{14}} \leq c_{e_{11}} \\ p_D^{e_{14}} \leq c_{e_{13}} & p_D^{e_{14}} \leq c_{e_{14}} & p_D^{e_{14}} \leq c_{e_5} \end{array} \qquad c_{e_5} + c_{e_7} + c_{e_8} + c_{e_{11}} + c_{e_{13}} + c_{e_{14}} - p_D^{e_{14}} \leq 5$$

Constraint C) Every vertex is powered

Now that all the p_v^{no} are set correctly, we force the number of connected components to stay the same. The component containing the affected customers will shrink, but always stay connected. This constraint therefore focuses on the fact that the remaining vertices should all be connected. The new graph is connected when there is at least one path to each vertex $v \in V_N$. This would give the constraint

$$\forall v \in V_N : \quad \sum_{no \in E_{no}} p_v^{no} \geq 1.$$

However we can be even more precise.

Proposition 4.2.1. *When we take constraints A, B, C into account, it is not possible for any $v \in V_N$ to have $\sum_{no \in E_{no}} p_v^{no} > 1$.*

Proof. The number of edges stays the same because of constraint A, as do the number of vertices. Using constraints B and C we force that there are two connected components. But Proposition 2.3.1 now states that both components must be trees. Therefore contradicting that any vertex is connected to the substation with two different paths. □

So the final constraint becomes:

$$\forall v \in V_N : \quad \sum_{no \in E_{no}} p_v^{no} = 1.$$

Note that in this new form, constraint A has become redundant. There is precisely one path from the substation to each vertex. This is equal to stating that the graph is a tree by Theorem 2.3.1 and Theorem 2.3.2, forcing the number of edges to stay the same.

Constraint for the example subnet

$$p_D^{e_{12}} + p_D^{e_{14}} = 1$$

No overload on distribution lines

The next two constraints make sure that no distribution line gets overloaded.

Constraint D) Determine extra strain on each edge

The extra strain on a edge e is determined by the vertices V' that get their power through edge e. So the load of a vertex $v \in V_N$ should only be added to S_e if e is in the path from the substation to v and that path is actually used to feed v. This becomes

$$\forall e \in E_1 \cup E_2 : \quad S_e = \sum_{v \in V_N} \sum_{no \in E_{no}} \text{load}(v) \cdot p_v^{no} \cdot \alpha(e, no, v).$$

Constraint for the example subnet

$$\begin{aligned} S_4 = \text{load}(D) \cdot p_D^{e_{12}} + \text{load}(E) \cdot p_E^{e_{12}} + \text{load}(F) \cdot p_F^{e_{12}} \\ + \text{load}(C) \cdot p_C^{e_{14}} \end{aligned}$$

Constraint E) Extra strain should be possible

For each edge e the extra strain should never be larger than the residual capacity of that edge. This way we are sure not to overload any of the edges.

$$\forall e \in E_1 \cup E_2 : \quad S_e \leq \text{rescap}(e)$$

Upper bound residual capacity at isolation

Finally we want to determine the residual capacity at v_0. Why this is needed will become clear when we discuss improvements on the objective function of the Mixed Integer Linear Program in Section 4.2.7.

Constraint F) Residual capacity at isolation

Let P be the path that feeds v_0 in the end. Then

$$\rho = \min_{e \in P} (\text{rescap}(e) - S_e)$$

.
However at the start it is not clear which path will feed v_0. To circumvent this problem we put an upper bound on ρ for each $e \in E$. The edges e that are not used to actually feed v_0 get a large constant added to it that is larger than any residual capacity. This way that value doesn't influence the upper bound of ρ, since the ones that do actually feed v_0 do no get this constant added. Therefore these values will always define the actual upper bound on ρ.

The large constant we choose is $f = \max(1000, 1000 \cdot \max_{e \in E}(\text{rescap}(e)))$ and the constraint now becomes

$$\forall e \in E_1 \cup E_2 : \quad \rho \leq \text{rescap}(e) - S_e + f \cdot \left(1 - \sum_{no \in E_{no}} \alpha(e, no, v_0) \cdot p_{v_0}^{no}\right).$$

We only add f to the upper bound if none of the $p_v^{no} = 1$. Which means that the edge is not used to feed v_0. The constraint therefore behaves exactly as described above.

Constraint for the example subnet
Edge e_4 is only used to feed v_0 when power is restored through net opening at e_{14}.

$$\rho \leq \text{rescap}(e_4) - S_{e_4} + f \cdot (1 - p_C^{e_{14}})$$

The above shows that when we use e_{14} to restore power to v_0, then $\rho \leq \text{rescap}(e_4) - S_{e_4}$ since $p_C^{e_{14}} = 1$. However, if power is restored using e_{12}, then we have $\rho \leq \text{rescap}(e_4) - S_{e_4} + f$.

Edge e_{11} will always be part of a path that feeds v_0, since by constraint C we have $p_C^{e_{12}} + p_C^{e_{14}} = 1$.

$$\rho \leq \text{rescap}(e_{11}) - S_{e_{11}} + f \cdot (1 - p_C^{e_{12}} - p_C^{e_{14}})$$

4.2.5 Translating the output of the Switch Planner

The goal of the Switch Planner is to determine what switching operations should be performed to restore power to a subnet. The solution of the Switch Planner sets values for c_e such that all the customers in V_N have power again.

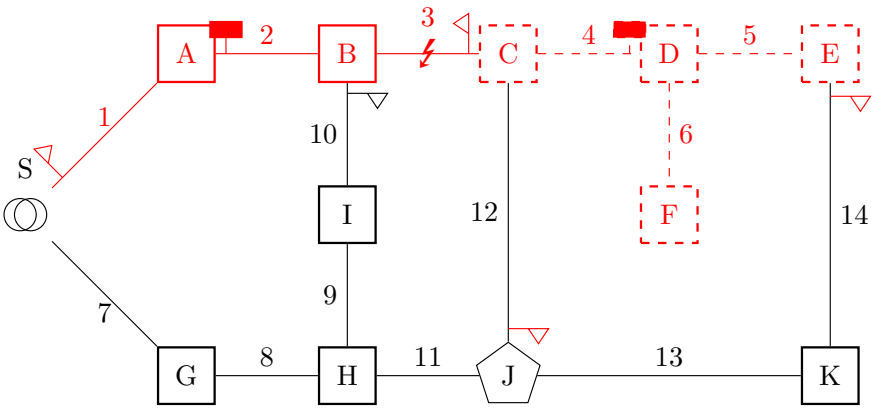
The switching operations can be extracted from the results as follow

$$no_{\text{new}} = \{e \in E_1 \backslash E_{no} : c_e = 0\}$$
$$no_{\text{closed}} = \{e \in E_{no} : c_e = 1\}$$

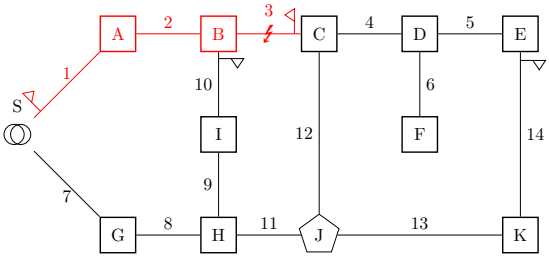
The set no_{new} tells us which edges have become net openings, similarly no_{closed} tells us which net openings should be closed. These are the switching operations that will be performed.
Note that we do not specify at what vertex the new net openings should be positioned, this is determined by the Outage Simulation Module.

4.2.6 Solutions for the example network

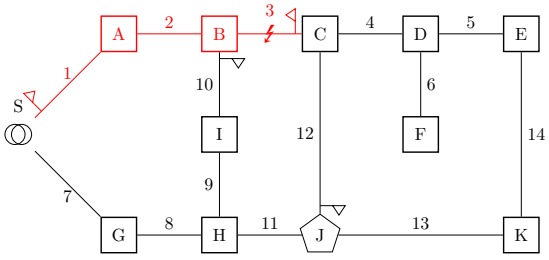
Now that we have defined the Mixed Integer Linear Program we can take a look at the possible solutions of the Switch Planner. In this situation we are in Step 2 of the Outage Simulation Module, so all the edges in E_1 are operable.
Figure 4.6 shows the four different solutions that at least fulfill constraints A, B and C.



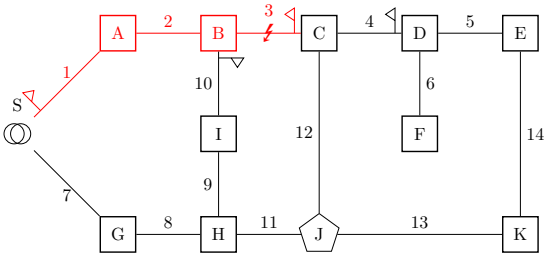
(a) Problem to solve



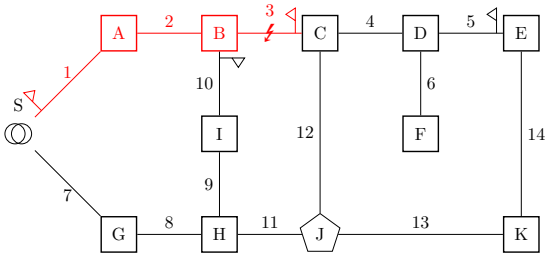
(b) Solution number 1



(c) Solution number 2



(d) Solution number 3



(e) Solution number 4

Figure 4.6: The example problem and all the possible solutions

The objective value of these solution can be computed and comes down to:

The values for Figure 4.6b

E_{no} :

$c_{e_{12}} = 1, c_{e_{14}} = 0$

E_1 :

$c_{e_4} = c_{e_5} = c_{e_6} = 1$

Objective :

$c_{e_{12}} + c_{e_{14}} - c_{e_4} - c_{e_5} - c_{e_6} = -2$

The values for Figure 4.6c

E_{no} :

$c_{e_{12}} = 0, c_{e_{14}} = 1$

E_1 :

$c_{e_4} = c_{e_5} = c_{e_6} = 1$

Objective :

$c_{e_{12}} + c_{e_{14}} - c_{e_4} - c_{e_5} - c_{e_6} = -2$

The values for Figure 4.6d

$$
\begin{aligned}
&E_{no}: \\
&\quad c_{e_{12}} = 1, c_{e_{14}} = 1 \\
&E_1: \\
&\quad c_{e_4} = 0, c_{e_5} = c_{e_6} = 1 \\
&Objective: \\
&\quad c_{e_{12}} + c_{e_{14}} - c_{e_4} - c_{e_5} - c_{e_6} = 0
\end{aligned}
$$

The values for Figure 4.6e

$$
\begin{aligned}
&E_{no}: \\
&\quad c_{e_{12}} = 1, c_{e_{14}} = 1 \\
&E_1: \\
&\quad c_{e_5} = 0, c_{e_4} = c_{e_6} = 1 \\
&Objective: \\
&\quad c_{e_{12}} + c_{e_{14}} - c_{e_4} - c_{e_5} - c_{e_6} = 0
\end{aligned}
$$

We see that there are two optimal solutions, namely Figure 4.6b and Figure 4.6c. These indeed need two switching operations, while the other two solutions need four switching operations.

It is however still possible that the Switch Planner outputs either Figure 4.6d or Figure 4.6e. This can happen because of constraints D, E. If edge e_{12} and either e_{13} or e_{14} have a low residual capacity, then putting the complete subnet over one of these might be too much. But splitting the load over both options could fit.
This is exactly what we want. If it is possible to switch with two switching operations it will give us this option. However if this is not allowed by the constraints, we might get a solution that contains more switching operations than expected at first.

Even though Figure 4.6b and Figure 4.6c have the same number of switching operations, we might prefer one over the other. In this case we would prefer Figure 4.6b, since one of the switching operations can be performed remotely. If we want this distinction by the Switch Planner the objective function needs to be modified. Possible suboptimizations, inlcuding this one, are discussed below.

4.2.7 Improvements on the objective function

There are two improvements built-in to the Switch Planner. One of them we already mentioned: we want to prefer remote switches where possible. The second explains the need of constraint F. We also discuss when each of these optimizations is used in the Outage Simulation Module.

Prefer remote switches

We want to adjust the objective function so that it prefers remote switches where possible. Let $E_{rc} = \{e \in E_{op} : e \text{ remotely controllable }\}$.
Since remote switches do not have any influence on the movement of the technician, these can been seen as free. As a result we do not want to count these c_e as a switch operations in our objective. This is achieved by changing the objective from

$$\min \quad \sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e$$

to

$$\min \quad \sum_{e \in E_{no} \setminus E_{rc}} c_e - \sum_{e \in E_{op} \setminus (E_{no} \cup E_{rc})} c_e.$$

This way, any non-remote switch will always worsen the objective as we discussed before, however a remote switch will not affect the objective at all. Indeed, when we go back to the example we see that the values of the objective now differ and prefer the solution that switches remotely.

The values for Figure 4.6b

$$
\begin{aligned}
&E_{no}: \\
&\quad c_{e_{12}} = 1, c_{e_{14}} = 0 \\
&E_1: \\
&\quad c_{e_4} = c_{e_5} = c_{e_6} = 1 \\
&\textit{Objective}: \\
&\quad c_{e_{14}} - c_{e_4} - c_{e_5} - c_{e_6} = -3
\end{aligned}
$$

The values for Figure 4.6c

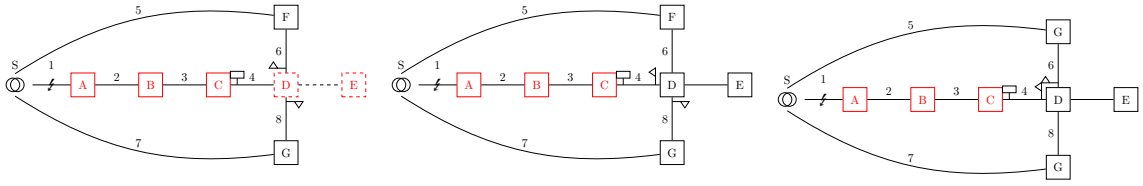
$$
\begin{aligned}
&E_{no}: \\
&\quad c_{e_{12}} = 0, c_{e_{14}} = 1 \\
&E_1: \\
&\quad c_{e_4} = c_{e_5} = c_{e_6} = 1 \\
&\textit{Objective}: \\
&\quad c_{e_{14}} - c_{e_4} - c_{e_5} - c_{e_6} = -2
\end{aligned}
$$

Maximize residual capacity at isolation

We have a constraint that determines the residual capacity at v_0, the vertex from which we isolated the subnet from the outage. This constraint should be used in combination with an adjustment to the objective function as we will see later on. But first we discuss why we want to take the residual capacity into account.

We already discussed that it can happen in Step 1 and Step 2 of the OSM, that e_m is not yet known. Using remote controllability we can still restore power to subnets. When e_m is known in Step 3, it can be necessary to feed some vertices using a subnet we fed in Step 1 or 2. This will always happen through the v_0 of that subnet, which means that we want as much residual capacity at v_0 as possible. This way we are able to restore power to as many customers as possible. Example 4.2.1 is a small example which shows the influence of the residual capacity at v_0.

Example 4.2.1. Figure 4.7 is an example that shows the usefulness of the constraint. When we are in Step 2 of the Outage Simulation Module we know that $e_m \in \{e_1, e_2, e_3\}$. This means that we can restore power to subnet $N = \{D, E\}$. We have two options to restore power, as shown in Figure 4.7b an Figure 4.7c.



(a) Malfunction in e_1, restore power to subnet $N = \{D\}$

(b) Residual capacity at (D, e_4) is 10

(c) Residual capacity at (D, e_4) is 1

Figure 4.7

Thus far the objective value of the Switch Planner would be the same for both situations, we do however prefer Figure 4.7b. When the exact location of e_m is known in Step 3, we want to restore power to subnet $N' = \{A, B, C\}$. With Figure 4.7b we have more residual capacity left and we can probably feed the whole subnet. This is not the case with the solution shown in Figure 4.7c

We already defined the constraint that determines the residual capacity at v_0. All that is left is to adjust the objective value so that it takes ρ into account. We do however only want to take ρ into account between solutions that have the same number of switching operations.

Theorem 4.2.1. *The objective*

$$\min \quad \left\{ f \cdot \left(\sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e \right) - \rho \right\},$$

first optimizes on the number of switching operations and only then on the residual capacity.

Proof. We have already seen that min $\sum_{e\in E_{no}} c_e - \sum_{e\in E_{op}\setminus E_{no}} c_e$ optimizes on the number of switching operations. It is also clear that for two solutions with the same number of operations, we will select the one with the largest ρ. All that is left to show, is that ρ will never cause us to select a solution that uses more switching operations than needed.
Let n be the minimal number of switching operations required for a solution and $m > n$. Let $n = n_{no} + n_{op}$ and $m = m_{no} + m_{op}$. Where n_{no}, m_{no} denote the number of closed net openings and n_{op}, m_{op} the number of new net openings.
In the worst case with n switching operations we have that $\rho_n = 0$. The best case with m switching operations has $\rho = \max_{e\in E}(\text{rescap}(e) - S_e)$. Remember that $f = \max(1000, 1000 \cdot \max_{e\in E}(\text{rescap}(e)))$, and that E_{op}, E_{no} are identical in both situations. This gives us

$$\begin{aligned}
obj_m &= f \cdot \left(\sum_{e\in E_{no}} c_e - \sum_{e\in E_{op}\setminus E_{no}} c_e \right) - \rho_m \\
&= f(m_{no} - (|E_{op}\setminus E_{no}| - m_{op})) - \rho_m \\
&= f(m_{no} + m_{op} - |E_{op}\setminus E_{no}|) - \max_{e\in E}(\text{rescap}(e) - S_e) \\
&> f(m - |E_{op}\setminus E_{no}|) - f \\
&= f((m-1) - |E_{op}\setminus E_{no}|) \\
&\geq f(n - |E_{op}\setminus E_{no}|) \\
&= f(n_{no} - (|E_{op}\setminus E_{no}| - n_{op})) - 0 \\
&= f \cdot \left(\sum_{e\in E_{no}} c_e - \sum_{e\in E_{op}\setminus E_{no}} c_e \right) - \rho_n \\
&= obj_n
\end{aligned}$$

Showing that $obj_m > obj_n$ as needed. □

Example 4.2.2. For the example above we indeed see that ρ now makes us prefer the correct solution.

$$\begin{aligned}
&Figure\ 4.7b : f \cdot (2 - |E_{op}\setminus E_{no}|) - 10 \\
&Figure\ 4.7c : f \cdot (2 - |E_{op}\setminus E_{no}|) - 1
\end{aligned}$$

Usage of the Switch Planner within the Outage Simulation Module

The optimizations above are actively used within the Outage Simulation Module. The optimization over remote controllable edges is used within Step 2 and 3 of the module. It is not used in Step 1, since any of the switching operations will be remote controllable. If these were all set to zero, we would no longer optimize on the number of switching operations.
As discussed the optimization over residual capacity is used by Step 1 and 2 of the Outage Simulation Module. In Step 3 we already know the location, so we no longer need residual capacity to power an extra vertex.

Chapter 5

Extending the Switch Planner

In the following sections we discuss the three main extensions that have been researched during this thesis. The first two replace the current implementation of the Switch Planner and remove heuristic choices within the Outage Simulation Module. The third extension will improve on the currently used objective function by the Switch Planner. Lastly we will discuss a variation on one of the newly defined Switch Planners.

5.1 Customer Switch Planner

This extension of the Switch Planner focuses on letting the Switch Planner decide how to isolate the subnet from the malfunctioning edge and which customers end up with power. In most cases, this will be at the location we originally put a net opening and provide power to all the customers. However, when we can't feed the whole subnet some heuristic choices are made by the Outage Simulation Module.
This section discusses the heuristic choice that will be removed by the extension, followed by the implementation, results and an elaborate example. We will refer to this extension of the Switch Planner as the *Customer Switch Planner*.

5.1.1 Heuristic choice: which customers do not get power?

With the current implementation of the Switch Planner, it is possible that the problem is infeasible. This happens when we can't feed all the vertices in the subnet while staying within the capacities of all the edges.
When this happens, we need to redefine our problem. Since we can't increase the capacity of the edges, we instead choose to not feed all the customers in the subnet. The Outage Simulation Module removes v_0 from the subnet and tries again for the remaining subnet(s). This is repeated for each subnet, until the Switch Planner becomes feasible again.
The removed vertices are not fed in this step of the Outage Simulation Module. With the Customer Switch Planner we want to remove the heuristic choice described above of which vertices can be fed. We want the Switch Planner to decide the optimal switching operations that provide power to as many customers of the subnet as possible. An outage where this choice has a lot of impact is discussed in Example 5.1.2.

5.1.2 Mixed Integer Linear Program

Below we have defined the Customer Switch Planner. Compared to before we see changes to the objective funtion, constraints A and C and the addition of the extra constraint B'. These changes will all be discussed below.

Objective function

$$\min \quad \left\{ -F \cdot \left(\sum_{v \in V_N} \sum_{no \in E_{no}} p_v^{no} \cdot \mathrm{cust}(v) \right) + \sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e \right\}$$

Constraints

Ref	Constraint		Description
A	$\sum_{e \in E_{op}} (1 - c_e) = \lvert E_{no} \rvert$		Number of edges
B	$\forall e \in \mathrm{path}_v^{no}: \quad p_v^{no} \le c_e$ $\sum_{e \in \mathrm{path}_v^{no}} c_e - p_v^{no} \le \lvert \mathrm{path}_v^{no} \rvert - 1$	$\forall v \in V_N$, $no \in E_{no}$	Set p_v^{no} correctly
B'	$\forall e \in \mathrm{mpath}_v: \quad mp_v \le c_e$ $\sum_{e \in \mathrm{mpath}_v} c_e - mp_v \le \lvert \mathrm{mpath}_v \rvert - 1$	$\forall v \in V_N$	Set mp_v correctly
C	$mp_v + \sum_{no \in E_{no}} p_v^{no} \le 1$	$\forall v \in V_N$	Vertex should not have power and be connected to the outage
D	$S_e = \sum_{v \in V_N} \sum_{no \in E_{no}} \mathrm{load}(v) \cdot p_v^{no} \cdot \alpha(e, no, v)$	$\forall e \in E_1 \cup E_2$	Set S_e correctly
E	$S_e \le \mathrm{rescap}(e)$	$\forall e \in E_1 \cup E_2$	Edges stay within their capacity
F	$\rho \le \mathrm{rescap}(e) - S_e + f \cdot \left(1 - \sum_{no \in E_{no}} \alpha(e, no, v_0) \cdot p_{v_0}^{no}\right)$	$\forall e \in E_1 \cup E_2$	Upper bound on the residual capacity at v_0

5.1.3 Data preperation

Before we can explain all the changes to the Switch Planner, we first need to define some extra data. The objective will need to take the number of customers into account. This is possible using the function

$$\text{cust} : V_N \to \mathbb{Z}, \qquad \text{cust}(v) = \text{ number of customers powered by vertex } v.$$

It turns out that we need to precompute some additional paths in the graph, namely the paths from the malfunctioning edge, e_m, to all the vertices of the subnet. Note that this path is unique, since the graph is a tree.

$$\forall v \in V_N : \quad \text{mpath}_v = \{\text{the unique path from } e_m \text{ to } v\}$$

The above path is used to set the value of the new decision variable mp_v correctly

$$\forall v \in V_N : \quad mp_v = \begin{cases} 1 & \text{if there is closed path from } e_m \text{ to } v \\ 0 & \text{otherwise} \end{cases}$$

5.1.4 Objective function

With the original Switch Planner the constraints of the problem ensured that customers would get power again, therefore the objective function aimed to minimize the number of switching operations. With the Customer Switch Planner, we want the MILP to decide how to isolate the subnet and which customers end up with power again, such that this is done in the optimal way. The objective function has to be adapted for this.

Counting the number of customers with power again comes down to checking whether a vertex v is powered, $\sum_{no \in E_{no}} p_v^{no} = 1$, and the number of customers it has, $\text{cust}(v)$. So the objective

$$\min \qquad -1 \cdot \sum_{v \in V_n} \sum_{no \in E_{no}} p_v^{no} \cdot \text{cust}(v)$$

will try to power as many customers as possible, since every customer with power will lower the objective value.
Sometimes multiple solutions can provide power to the same number of customers. We again want to select the solution with the fewest switching operations. We can apply the same method discussed in Section 4.2.7 to follow this order of optimizations. We had to take $f > \rho$ for it to work previously. Similarly we now need an $F > \sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e$. There are at most $|E_{op}|$ switching operations, so we can take $F = 2|E_{op}|$.
The objective function

$$\min \qquad \left\{ -F \cdot \left(\sum_{v \in V_N} \sum_{no \in E_{no}} p_v^{no} \cdot \text{cust}(v) \right) + \sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e \right\}$$

will behave exactly as intended. Any extra customer saved will decrease the objective value by such an amount that a solution with fewer customers saved can never outperform it because it has fewer switching operations.

5.1.5 Constraints

Compared to the original Switch Planner we see that constraints B, D and E have stayed the same. The adjusted constraints are all explained below.

Constraint A)

$$\sum_{e \in E_{op}} (1 - c_e) = |E_{no}| - 1 \qquad \rightarrow \qquad \sum_{e \in E_{op}} (1 - c_e) = |E_{no}|$$

This constraint has only changed slightly, the minus one has disappeared. Recall that previously it was added since we already placed a net opening to isolate the subnet. This is no longer true in the Customer Switch Planner, which determines how to isolate the subnet itself.

Constraint B')

$$\forall v \in V_N : \quad \begin{aligned} &\forall e \in \text{mpath}_v : \quad mp_v \leq c_e \\ &\sum_{e \in \text{mpath}_v} c_e - mp_v \leq |\text{mpath}_v| - 1 \end{aligned}$$

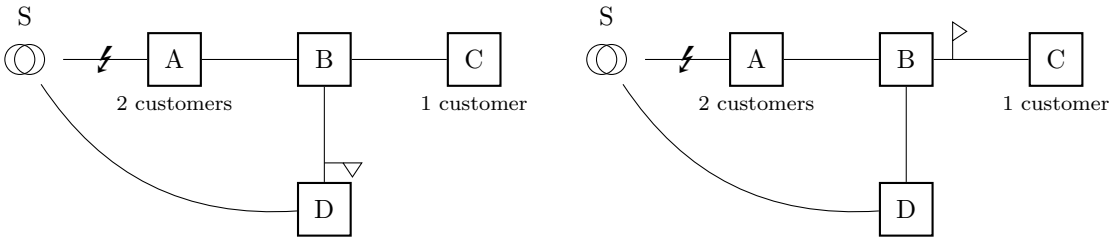
Notice the resemblance between constraints B and B'. We again want to know whether or not a path is closed, this time for all the mpath_v. The method to set this value correctly is completely analogous to constraint B.

Constraint C)

$$\sum_{no \in E_{no}} p_v^{no} = 1 \qquad \rightarrow \qquad mp_v + \sum_{no \in E_{no}} p_v^{no} \leq 1$$

This constraint has changed quite a bit. Remember that previously we started off with $\sum p_v^{no} \geq 1$ to have two connected components, which together with A forced both components to be trees. As a result this also required all the vertices to be powered. This is no longer something we want. To allow some vertices to be without power, we need to change the constraint to $\sum p_v^{no} \leq 1$. This does however allow for incorrect solutions, as shown in the Example 5.1.1.

Example 5.1.1. Assume the situation as shown in Figure 5.1a. We know which edge is malfunctioning and there is not enough capacity left to feed the whole subnet $N = \{A, B, C\}$. With the constraints discussed thusfar, the solution given in Figure 5.1b would be optimal. However this is not something we want to allow, since this connects the substation with the outage again, resulting in another outage.



(a) Problem to solve (b) Proposed solution

Figure 5.1

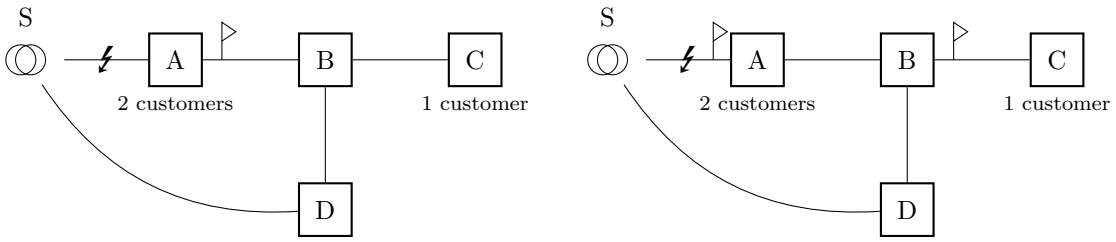
We need to make sure that this situation does not happen. More precisely, we need to make sure that we do isolate (part of) the subnet from e_m. Note that isolating a vertex v from the e_m comes down to checking that mp_v is not equal to 1. Only if this is the case, we are allowed to provide that vertex with power again.
This means that a vertex v must not be both powered, $\sum p_v^{no} = 1$, and connected to e_m, $mp_v = 1$. This results in

$$mp_v + \sum_{no \in E_{no}} p_v^{no} \leq 1,$$

which indeed forces at most one of them to be true.

When we take a look at Example 5.1.1, we see that the solution we discussed before is indeed not possible anymore. For vertices A, B we would have $mp_A = mp_B = 1$ and $\sum p_A^{no} = \sum p_B^{no} = 1$.

Non-optimal solutions Given the constraints, the Customer Switch Planner would find Figure 5.2a as solution to the problem. There is however a solution that performs better for this single case, shown in Figure 5.2b. This solution is however not allowed by constraint A.
While we no longer require A to return a correct solution, we do need A to keep 2 connected components. Without it we would get more connected components, which could result in a lot of small parts of the net that are without power. In this case the estimations discussed in step 4 of the Outage Simulation Module are no longer correct. Since we now might need more than one mobile power generator. We will therefore leave constraint A intact. In Section 5.4 we discuss the impact of removing constraint A.



(a) Solution by Customer Switch Planner

(b) Optimal Solution

Figure 5.2

5.1.6 Handling improved objective function

We discussed two suboptimizations on the objective function in Section 4.2.7. With it we would pick remote switching operations where possible and maximize the residual capacity at v_0. Since these are actually used in the Outage Simulation Module, we take a look at what has changed with the Customer Switch Planner.

The optimization on the remote switching operations is still applicable and can be applied as before. We do however need to take a better look at the optimization on the residual capacity.

Suboptimization on residual capacity

While the constraint for this suboptimization hasn't changed, the way it functions has. Below we discuss the new objective that takes into account the residual capacity and why the impact of constraint F has slightly changed.

Objective function When we want to save as many customers as possible with the fewest possible number of switching operations and with as much residual capacity left at v_0, the objective becomes

$$\min \quad \Bigg\{ -F \cdot f \cdot \left(\sum_{v \in V_N} \sum_{no \in E_{no}} p_v^{no} \cdot \mathrm{cust}(v) \right) + f \cdot \left(\sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e \right) - \rho \Bigg\} \quad .$$

This just applies the method discussed in Theorem 4.2.1 twice. We must make sure that the objective values don't get too big. Otherwise the solver could run into numerical errors. As it turns out, the implementation above still functions as intended using the current values for f, F.

Constraint F While this constraint hasn't changed compared to before, its function has. Previously v_0 was always fed by a path, which meant the upper bound for ρ was set correctly. This is now no longer true. When we can't feed the whole net, v_0 might not be powered. In this case, the constraint changes to

$$\forall e \in E_1 \cup E_2 : \quad \rho \leq \mathrm{rescap}(e) - S_e + f.$$

Here $f = \max(1000, 1000 \cdot \max_{e \in E}(\mathrm{rescap}(e)))$ as before.

This means that ρ is no longer limited by a correct upperbound. This situation only occurs when we cannot feed all customers due to capacity problems. Optimizing on the residual capacity is therefore not necessary any more, since we want to use as much of the residual capacity to feed customers of the current subnet.

We do however need to make sure that our objective function still works correctly, since we no longer have $f > \rho$ which was required for Theorem 4.2.1. Note that we do still have $2 \cdot f > \rho$.
It turns out that this is still enough to keep the order of optimization in the objective function as intended. By constraint A the number of edges must stay the same. Each new net opening removes an edge, so we need to close a net opening to add a new edge and keep the total number of edges equal. This means that switching operations always occur in pairs. Each new net opening is countered by a closure of an net opening. which shows that solutions with more switching operations will always have at least two extra switching operations, influencing the objective value with $2 \cdot f$. Since we have $2 \cdot f > \rho$, we can still use Theorem 4.2.1 to show that the objective function works correctly.
Alternatively we could of course fuse a suitable large number in constraint F other than f. This way we could still fulfill the assumptions needed for Theorem 4.2.1 while correctly setting upper bounds for ρ.

5.1.7 Results

We have tested the Customer Switch Planner on the 3397 routes that we have at our disposal. Below we discuss the results compared to the original Switch Planner that show an overall decrease in expected CML.
It's hard to be really precise on the significance of the results, since there is a lot of variance in the different routes. We have therefore chosen to take a global overview at the results and discuss some of the results found during the validation of the implementation. We will also discuss one particular example that illustrates the usefulness of the Customer Switch Planner.

Global overview

When we compare the Original Switch Planner with the Customer Switch Planner we get the results as shown below Table 5.2, Table 5.3 and Figure 5.3.

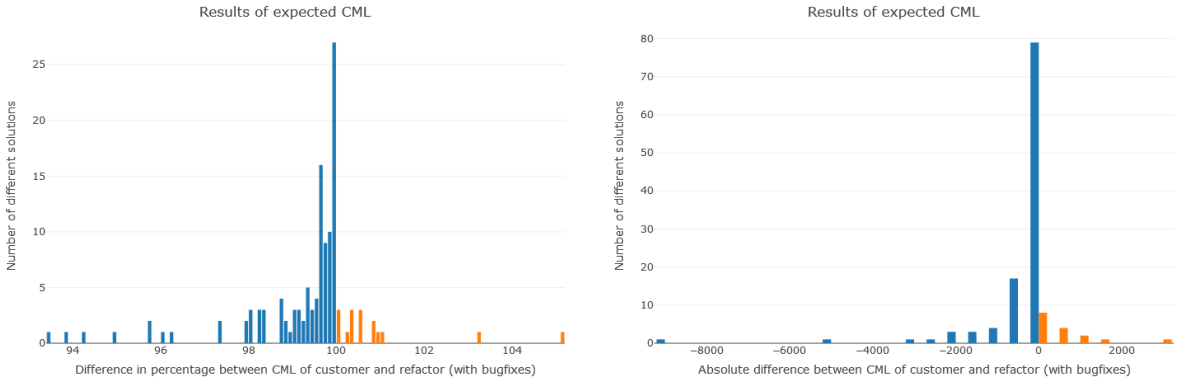
Original Switch Planner versus Customer Switch Planner

Total	
126	
Lower	Higher
110	16

Table 5.2: Routes with different ECML

Extension	Total running time
Original	35869 seconds
Customer	42306 seconds

Table 5.3: Total running time for all routes



(a) Histogram for proportional difference in ECML

(b) Histogram for absolute difference in ECML

Figure 5.3

The histograms in Figure 5.3 only show the routes that have a different outcome compared to before. The blue lines represent the routes with a lower ECML using the Customer Switch Planner, while the orange lines represent the routes with a higher ECML. With Figure 5.3b we see the absolute difference between the ECML of the original Switch Planner and the Customer Switch Planner. This shows a clear picture that the total ECML has decreased. This figure however doesn't really show changes made to routes with a small amount of customers. To check these as well, we have Figure 5.3a. Here we see the proportional difference for each route. This again shows an overall decrease in ECML.
Both the histograms give us confidence that the Customer Switch Planner does generally improve on the original Switch Planner. This has been reassured after validating a lot of different outcomes. Below we discuss the outcome of such a validation.

Lower ECML All most all routes that have a lower ECML indeed had outages that were solved better because of better choices in the removal of vertices. Precisely what the Customer Switch Planner is intended for. The route with the biggest absolute difference will be discussed later on.

Higher ECML Some routes now have a higher ECML. The main reason for this is the ability to distinguish two solutions with the Switch Planner. We saw before that sometimes two solutions might have the same objective value. We might however prefer one over the other. We tried our best to extend the objective function to distinguish more cases, but this doesn't capture all the different situations. It turns out that we now sometimes pick another solution to before, that turns out worse in the end.

Time The total running time to simulate the outages in the routes has increased by roughly 18 percent. While we haven't done a full analysis on why this happens, there are two main suspects. The solution space has grown immensely. Previously a solution had to feed every vertex of the subnet, with the Customer Switch Planner any configuration that is radial and within capacities is a possible solution. Even the starting configuration that restores power to zero customers already is a correct solution. Furthermore the objective function has been changed. It now has to optimize on customers and switch operations, which might also increase the searching time for the solution.

Example network

The route that causes the largest absolute difference in customer minutes lost is a good example on why the Customer Switch Planner makes better decisions. It show that the heuristic choices in the Outage Simulation Module miss out on saving a lot of customers.

Example 5.1.2 (BML 10-1V2.11)**.** Figure 5.4a shows a simplified view of the MV-network in Bemmel, with 2 routes. We know the exact location of e_m and the Customer Switch Planner is called from Step 2 of the Outage Simulation Module. The solution found using the original Switch Planner is shown in Figure 5.4c, while the solution found by the Customer Switch Planner is shown in Figure 5.4d. There is a clear difference in the number of customers saved.

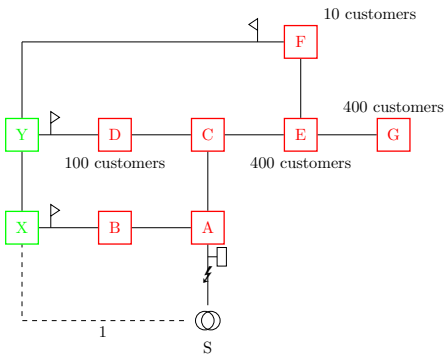
The original Switch Planner doesn't find the optimal solution shown in Figure 5.4d, since it handles the different subnets in the wrong order. The steps it takes are as follows:

1. Try to power the whole subnet $N = \{A, B, C, D, E, F, G\}$; Switch Planner infeasible
   - Remove $v_0 = A$, this results in subnets $N_1 = \{B\}$ and $N_2 = \{C, D, E, F, G\}$
2. Try to power N_1; Switch Planner feasible, power restored as shown in Figure 5.4c
3. Next try to power N_2; Switch Planner is infeasible.
   - Remove $v_0 = C$, this results in subnets $N_3 = \{D\}$, $N_4 = \{E, F, G\}$.
4. Try to power N_3; Switch Planner feasible, power restored as shown in Figure 5.4c
5. Try to power N_4; Switch Planner infeasible.
   - Remove $v_0 = E$, this results in subnets $N_5 = \{F\}$ and $N_6 = \{G\}$
6. Try to power N_5; Switch Planner feasible, power restored as shown in Figure 5.4c
7. Try to power N_6; no connected net opening to power the subnet

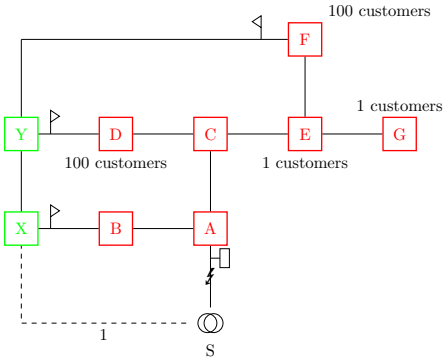
Compare this to the Customer Switch Planner that only needs to be called upon once.

1. Try to restore power to customers of subnet $N = \{A, B, C, D, E, F, G\}$; Switch Planner feasible, result as shown in Figure 5.4d.

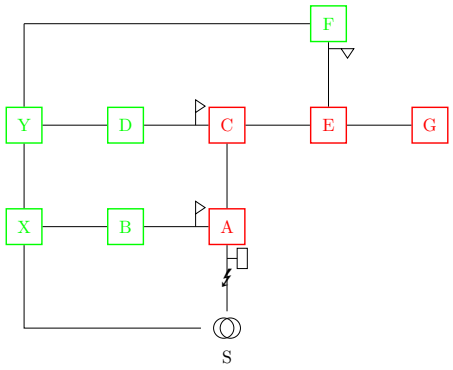
It might look odd that the Customer Switch Planner is able to provide power to subnet N_4, where the original Switch Planner tells us it is infeasible. The reason is that for all paths in Figure 5.4a that feed (part of) N the capacity bottleneck is edge e_1. When the Outage Simulation Module decides to first feed N_3 in step 4 above, it reduces the residual capacity of e_1. Because of this, it no longer has enough residual capacity to feed N_4. Making the Switch Planner infeasible.
Indeed, we see that the Customer Switch Planner does not restore power to vertex D (N_3), but instead restores power to vertices E, F and G (N_4).



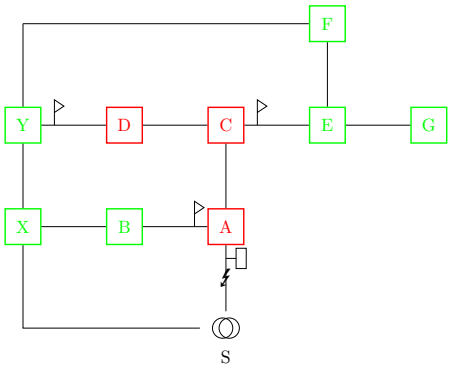
(a) Bemmel - simplified network



(b) Bemmel - alternative customer distribution



(c) Solution by the original Switch Planner to 5.4a



(d) Solution by the Customer Switch Planner to 5.4a

Figure 5.4

If the Outage Simulation Module would first have tried N_4 and then N_3, it would have given the same result as the Customer Switch Planner. But this in turn is not always the optimal order, as shown by Example 5.1.3.

Example 5.1.3. Suppose that the customers are divided as shown in Figure 5.4b and that e_1 has enough residual capacity left to feed 200 extra customers.
If the Outage Simulation Module tries to power N_4 first, it restores power 102 customers. But the Switch Planner for N_3 becomes infeasible, since e_1 cannot feed 202 customers.
If instead we first try N_3, it restores power to N_3 and N_5 as before. Restoring power to 200 customers.

This real world example shows the power of the Customer Switch Planner. Where originally we had to make a heuristic choice, the Customer Switch Planner now tries both the possibilities. This way selecting the optimal solution in both cases.

5.2 Area Switch Planner

The extension discussed in this section continues upon the Customer Switch Planner. Thus far we had to call the Switch Planner for each subnet separately, this extension will handle multiple subnets at once. We show why we want to be able to do this, followed by the implementation and results. We will refer to it as the *Area Switch Planner*.

5.2.1 Heuristic choice: order of subnets

Currently the order used to try and restore power to subnets is based on the number of customers of each subnet. This can give some non-optimal solution however. Example 5.2.1 shows such a situation. The choice we make for the first subnet can be of influence for the feasibility of the Switch Planner for the second subnet.

Example 5.2.1. Figure 5.5 shows a situation where the order in which we solve the subnets matters. Assume that e_3 does not have the residual capacity to feed both A and B, but it can feed each separately.

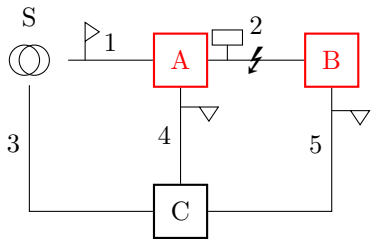


Figure 5.5: Example network where the order of subnets is important

By remote observability we know that e_2 is the cause of the outage. There are two subnets $N_1 = \{A\}$ and $N_2 = \{B\}$. Restoring power to N_1 by closing (C, e_4) would reduce the residual capacity of e_3. The result is that e_3 now does not have enough residual capacity left to feed N_2 as well.
If instead we would close (S, e_1) to feed N_1, the Switch Planner for N_2 would not be infeasible. Now e_3 does have enough residual capacity to feed N_2. Showing the importance of the order and choices we make for each subnet.

To make sure that we return the optimal solution, we now want the Switch Planner to handle all the subnets at once. This way, it notices these dependencies and it can make the best decisions on how to restore power to as many customers as possible.

5.2.2 Mixed Integer Linear Program

The Area Switch Planner is quite similar to the Customer Switch Planner structurally. All the constraints have however been adapted to cope with the bigger input set that exists of multiple subnets. As a result only constraint E stayed the same. Additionally constraint G has been added.

Objective

$$\min \quad \left\{ -F \cdot \left(\sum_{v \in V_N} \sum_{no \in E_{no}} \sum_{i \in I_v^{no}} p_{v,i}^{no} \cdot \text{cust}(v) \right) + \sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e \right\}$$

Constraints

Ref	Constraint		Description
A	$\sum_{e \in E_{op}} (1 - c_e) = \lvert E_{NO} \rvert$		Number of edges
B	$\forall e \in \text{path}_{v,i}^{no}: \quad p_{v,i}^{no} \leq c_e$ $\sum_{e \in \text{path}_{v,i}^{no}} c_e - p_{v,i}^{no} \leq \lvert \text{path}_{v,i}^{no} \rvert - 1$	$\forall v \in V_N$, $no \in E_{no}$	Set p_v^{no} correctly
B'	$\forall e \in \text{mpath}_{v,i}: \quad mp_{v,i}^{no} \leq c_e$ $\sum_{e \in \text{mpath}_{v,i}} c_e - mp_{v,i} \leq \lvert \text{mpath}_{v,i} \rvert - 1$	$\forall v \in V'$	Set mp_v correctly
C	$\sum_{i \in I_v^{mp}} mp_{v,i} + \sum_{no \in E_{no}} \sum_{i \in I_v^{no}} p_{v,i}^{no} \leq 1$	$\forall v \in V_N$	Vertex should not have power and be connected to the outage
D	$S_e = \sum_{v \in V_N} \sum_{no \in E_{no}} \sum_{i \in I_v} \text{load}(v) \cdot p_{v,i}^{no} \cdot \alpha(e, no, v, i)$	$\forall e \in E_1 \cup E_2$	Set S_e correctly
E	$S_e \leq \text{rescap}(e)$	$\forall e \in E_1 \cup E_2$	Edges stay within their capacity
F	$\rho \leq \text{rescap}(e) - S_e + f\left(1 - \sum_{v_0 \in V_0} \sum_{no \in E_{no}} \sum_{i \in I_p(no, v_0)} \alpha(e, no, v_0, i) \cdot p_{v_0,i}^{no}\right)$	$\forall e \in E_1 \cup E_2$	Upper bound on the residual capacity at v_0
G	$c_{NO} \leq \sum_{v \in V'} \sum_{no \in E_{no}} \sum_{i \in I_v^{no}} \alpha(NO, no, v, i) \cdot p_{v,i}^{no}$	$\forall NO \in E_{NO}$	Only allow net opening to be closed, if it's part of a path that feeds a vertex

5.2.3 Data preperation

The data combines the different subnets $N_1, \ldots, N_n$ into one combined network N_c. We will again discuss all the different variables, since a lot has changed. $E_x^{N_i}$, for E_x a set in the data of the orignal Switch Planner, denotes the set corresponding to subnet N_i, i.e. $E_{no}^{N_1} = \{e \in E : e \text{ a net opening connected to subnet } N_1\}$

$$E_{no} = \bigcup_{i=1}^{n} E_{no}^{N_i} = \{e \in E : e \text{ a net opening connected to } \textit{(part of) the combined net } N_C\} \tag{5.1}$$

$$E_1 = E_{no} \cup \{e \in E : e \textit{ an edge} \text{ in the combined net } N_C\} \tag{5.2}$$

$$E_{NO} = \{e \in E_1 : e \text{ is a net opening}\} \tag{5.3}$$

$$E_2 = \bigcup_{i=1}^{n} E_2^{N_i} = \{e \in E \backslash E_1 : e \text{ part of a path that can } \textit{(partly)} \text{ feed } \textit{the combined net } N_C\} \tag{5.4}$$

$$E_{op} = \{e \in E_1 : e \text{ is operable}\} \tag{5.5}$$

$$V' = \bigcup_{i=1}^{n} V_{N_i} \tag{5.6}$$

$$V_0 = \{v_0 \in V_{N_i} : i \in \{1, \ldots, n\}\} \subset V' \tag{5.7}$$

Below we shortly highlight and discuss the most important changes.

(5.2) This now also includes all the net openings in and between the different subnets of N. Previously it only contained the *closed* edges.

(5.3) This is a new set, that contains all the net openings in E_1. Not just the net openings that can feed (part of) the combined net. Notationwise we will refer to a net opening that can feed (part of) the combined network N with no, while NO refers to any of the net openings in E_1.

We still have the same functions as previously.

$$\begin{aligned} \text{load} &: V \to \mathbb{R}, & \text{load}(v) &= \text{ electricity load of vertex } v \\ \text{rescap} &: E \to \mathbb{R}, & \text{rescap}(e) &= \text{ residual capacity for edge } e \\ \text{cust} &: V' \to \mathbb{N}, & \text{cust}(v) &= \text{number of customers powered by secondary substation } v \end{aligned}$$

The paths we precompute have changed and the number of paths has increased a lot. Previously we had one unique path for each combination of no and v. Now we need to know all the different possible paths that exist when we close any of the NO. This is explained in more detail in Example 5.2.2. To calculate these paths, we assume all $NO \in E_{NO}$ to be closed. The network is now no longer radial, but it does contain all the possible paths from the substation to a vertex v. We compute all the simple paths from the substation to v through no and all simple paths from e_m to v.

$$\begin{aligned} \text{paths}_v^{no} &= \{\text{list containing all simple paths from } S \text{ to } v \text{ through } no\} \\ \text{mpaths}_v &= \{\text{list containing all simple paths from } e_m \text{ to } v\} \end{aligned}$$

Note that this number of paths can theoretically grow quickly [4]. In practice it is not that bad, since the number of net openings in E_1 is typically low.
With $\text{paths}_{v,i}^{no}$ and $\text{mpaths}_{v,i}$ we denote the i-th element of paths_v^{no} and mpaths_v respectively.

We have two new functions on paths_v^{no} and mpaths_v, that indicate the number of paths we have for each vertex. Furthermore we have an adapted version of α.

$$I_p : E_{no} \times V' \to \mathbb{N} \qquad I_p(no, v) = |\text{paths}_v^{no}|$$
$$I_{mp} : V' \to \mathbb{N} \qquad I_{mp}(v) = |\text{mpaths}_v|$$
$$\alpha : E \times E_{no} \times V_n \times \mathbb{N} \to \{0, 1\} \qquad \alpha(e, no, v, i) = \begin{cases} 1 & \text{if } e \in \text{path}_{v,i}^{no} \\ 0 & \text{otherwise} \end{cases}$$

Finally we define two variables that will be used in the objective function as we saw before.

$$f = \max(1000, 1000 \cdot \max_{e \in E}(\text{rescap}(e)))$$
$$F = 2 \cdot |E_{op}|$$

Decision variables

The decision variables now become

$$\forall e \in E_{op} : \quad c_e = \begin{cases} 1 & \text{if edge } e \text{ is closed} \\ 0 & \text{if edge } e \text{ a net opening} \end{cases}$$
$$\forall no \in E_{no}, v \in V', i \in I_p(no, v) : \quad p_{v,i}^{no} = \begin{cases} 1 & \text{if vertex } v \text{ is powered by } \text{paths}_{v,i}^{no} \\ 0 & \text{otherwise} \end{cases}$$
$$\forall v \in V', i \in I_{mp}(v) : \quad mp_{v,i} = \begin{cases} 1 & \text{if } \text{mpaths}_{v,i} \text{ is a closed path from } e_m \text{ to } v \\ 0 & \text{otherwise} \end{cases}$$
$$S_e = \text{the extra electrical strain on } e \text{ in the result}$$
$$\rho = \text{residual capacity at the net opening that isolates the subnet}$$

Note that we define a decision variable for each simple path, not for the list of these paths.

Data problem when e_m is known

If the exact location of e_m is known, we might need to place net openings at both sides of e_m. An example of this is shown in Figure 5.6.

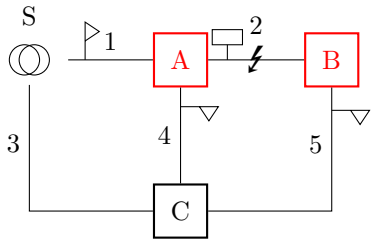


Figure 5.6: To restore power to both subnets, we need two net openings at $e_m = e_2$

The current implementation allows at most one net opening at each edge and doesn't take into account which side it is placed. Therefore failing this case, giving some strange solutions.
To solve this, we make a small adjustment to e_m. We split it in half, so that we have $e_{m,1}$ and $e_{m,2}$. We can now define the mpaths from the correct side. This way we can close e_m twice and take into account the effect that the closure at each side has.

Note that the more extensive graph representation described in Section 2.4.3 already captures this situation, since each edge is already split into two parts.

Example graph with corresponding data

Example 5.2.2. Figure 5.7 shows an example network. By remote controllability we have $e_m \in \{e_1, e_2, e_3\}$. This results in three subnets $N_1 = \{D\}$, $N_2 = \{E, G\}$ and $N_3 = \{F\}$.

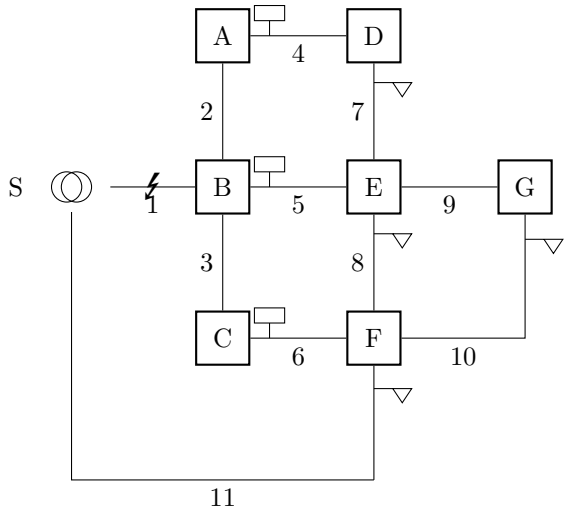


Figure 5.7: Example network with 3 subnets

Values

$E_{no} = \{e_{11}\}$ $\quad \text{paths}_E^{e_{11}} = [\{e_{11}, e_8\}, \{e_{11}, e_{10}, e_9\}]$

$E_1 = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$ $\quad \text{paths}_{E,1}^{e_{11}} = \{e_{11}, e_8\}, \text{paths}_{E,2}^{e_{11}} = \{e_{11}, e_{10}, e_9\}$

$E_{NO} = \{e_7, e_8, e_{10}, e_{11}\}$

$E_2 = \emptyset$

$V' = \{D, E, F, G\}$ $\quad \text{mpaths}_E = [\{e_1, e_2, e_4, e_7\}, \{e_1, e_5\}, \{e_1, e_3, e_6, e_8\}, \{e_1, e_3, e_6, e_{10}, e_9\}]$

$V_0 = \{D, E, F\}$ $\quad \text{mpaths}_{E,3} = \{e_1, e_3, e_6, e_8\}$

Net openings in E_1 and multiple paths

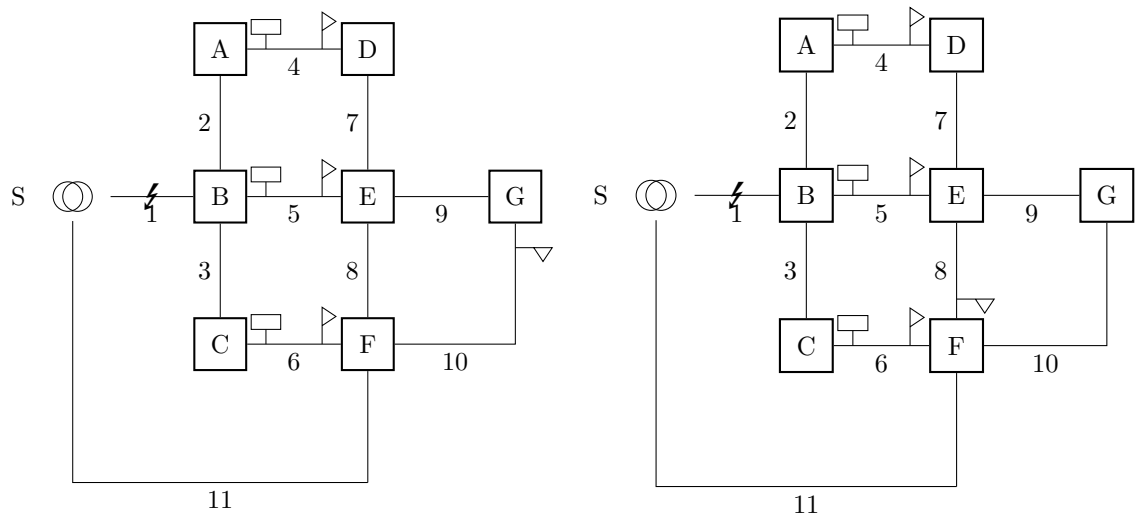
The network in Figure 5.7 shows us why we need to have the net openings between the different subnets. There are 3 subnets in this example network: $N_1 = \{D\}, N_2 = \{E, G\}$ and $N_3 = \{F\}$.

Only one of these has a net opening connected that can feed it, namely N_3. We can however still feed all the subnets by closing the net openings between N_1 and N_2 and between N_2 and N_3. The possible options are shown in Figure 5.8.

Both solutions feed the all the subnets, however the paths they use to feed D, E and G are different. The Switch Planner must be able to take into account all these paths to make sure that a vertex is powered and the capacity constraints are met.

Area vs Customer

The differences between the Customer Switch Planner and Area Switch Planner mostly follow from the different data. It is therefore important to understand the relation between $p_{v,i}^{no}, mp_{v,i}$ in the Area Switch Planner and p_v^{no}, mp_v in the Customer Switch Planner.



(a) Possible solution 1

(b) Possible solution 2

Figure 5.8

Intuitively we can take $p_v^{no} = \sum_i p_{v,i}^{no}$. This follows from the fact that p_v^{no} denoted whether or not a vertex was fed, similarly when one of the $p_{v,i}^{no} = 1$ this means that v is fed. By constraint C we have that this sum is at most one, which was also true in the Customer Switch Planner. Analogous we can take $mp_v = \sum_i mp_{v,i}$.

5.2.4 Objective function

The objective for the Customer Switch Planner was

$$\min \quad \left\{ -F \cdot \left(\sum_{v \in V_N} \sum_{no \in E_{no}} p_v^{no} \cdot \text{cust}(v) \right) + \sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e \right\},$$

this has now become

$$\min \quad \left\{ -F \cdot \left(\sum_{v \in V_N} \sum_{no \in E_{no}} \sum_{i \in I_v^{no}} p_{v,i}^{no} \cdot \text{cust}(v) \right) + \sum_{e \in E_{no}} c_e - \sum_{e \in E_{op} \setminus E_{no}} c_e \right\}.$$

At first we only needed to check whether the unique path was closed, now we have to check all the different possible paths that feed v. Note that by constraint C we have that $\sum p_{v,i}^{no} \leq 1$, so we will never count the customers of a vertex twice.

5.2.5 Constraints

Constraint A)

$$\sum_{e \in E_{op}} (1 - c_e) = |E_{no}| \qquad \rightarrow \qquad \sum_{e \in E_{op}} (1 - c_e) = |E_{NO}|$$

We still want to keep the number of edges equal, but the set containing all the net openings now is E_{NO}.

Constraint B, B')

$$\begin{aligned} \forall e \in \text{path}_v^{no}: \quad p_v^{no} \leq c_e \qquad &\rightarrow \qquad \forall e \in \text{path}_{v,i}^{no}: \quad p_{v,i}^{no} \leq c_e \\ \sum_{e \in \text{path}_v^{no}} c_e - p_v^{no} \leq |\text{path}_v^{no}| - 1 \qquad &\rightarrow \qquad \sum_{e \in \text{path}_{v,i}^{no}} c_e - p_{v,i}^{no} \leq |\text{path}_{v,i}^{no}| - 1 \end{aligned}$$

For each $\text{paths}_{v,i}^{no}$ and $\text{mpaths}_{v,i}$ we want to know whether is it a closed path or not. This is completely analogous to the previous constraint, there are just more paths to check.

Constraint C)

$$mp_v + \sum_{no \in E_{no}} p_v^{no} \leq 1 \qquad \rightarrow \qquad \sum_{i \in I_v^{mp}} mp_{v,i} + \sum_{no \in E_{no}} \sum_{i \in I_v^{no}} p_{v,i}^{no} \leq 1$$

When we use the intuition that $p_v^{no} = \sum_{i \in I_p(no,v)} p_{v,i}^{no}$ and $mp_v = \sum_{i \in I_{mp}(v)} mp_{v,i}$ from Section 5.2.3, we see that constraint C hasn't changed.
This also captures the fact that at most one of the simple paths to v is closed, so the result is radial.

Constraint D)

$$S_e = \sum_{v \in V_N} \sum_{no \in E_{no}} \text{load}(v) \cdot p_v^{no} \cdot \alpha(e, no, v) \quad \rightarrow \quad S_e = \sum_{v \in V_N} \sum_{no \in E_{no}} \sum_{i \in I_v} \text{load}(v) \cdot p_{v,i}^{no} \cdot \alpha(e, no, v, i)$$

S_e is dependent on which path is closed, so we need to check for each $p_{v,i}^{no}$ separately. Again we could take out $\sum p_{v,i}^{no}$ and rewrite it to p_v^{no} to regain the original constraint.

Constraint F)

$$\rho \leq \text{rescap}(e) - S_e + f \cdot \left(1 - \sum_{no \in E_{no}} \alpha(e, no, v_0) \cdot p_{v_0}^{no}\right)$$

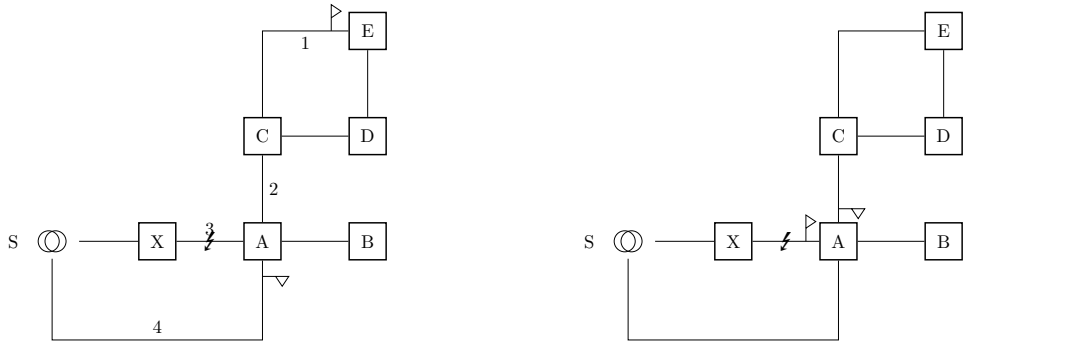
$$\rightarrow$$

$$\rho \leq \text{rescap}(e) - S_e + f \left(1 - \sum_{v_0 \in V_0} \sum_{no \in E_{no}} \sum_{i \in I_p(no, v_0)} \alpha(e, no, v_0, i) \cdot p_{v_0,i}^{no}\right)$$

This constraint also has to take into account all the different paths. However its function is again changed somewhat. This constraint now sets the upper bound of ρ to the residual capacity of all the $v_0 \in V_0$ at once. So the $v_0 \in V_0$ that has the lowest residual capacity left sets the upper bound for ρ. However the residual capacity at $v_0' \in V_0$, $v_0' \neq v_0$, is not necessarily optimized on anymore. Previously, we optimized on the residual capacity of each v_0 of a subnet. The impact of this change in function of the constraint is expected to be small. We have therefore left the constraint as described.

Constraint G) Close net opening if and only if it is used to feed a vertex

Now that we can also close net openings in and between subnets, some strange situation can occur. We already saw that the restriction of constraint A could give a non-optimal solution in the Customer Switch Planner. The Area Switch Planner would sometimes pick these solutions. An example on how this can happen is shown in Figure 5.9.



(a) Outage for subnet $\{A, B, C, D, E\}$ (b) Solution by the Area Switch Planner

Figure 5.9

In this situation it is beneficial to close the two net openings, (E, e_1) and (A, e_4), and create two net openings, (A, e_2) and (A, e_3). This does create a cycle in a non-powered part of the network. The closure of (E, e_1) is only done to fulfill constraint A, but has no purpose. Furthermore the graph now consists of three connected parts instead of two, namely $\{X\}$, $\{A, B\}$ and $\{C, D, E\}$.
We do not want to allow this kind of solutions. A variant that does allow this is discussed in Section 5.4. To prevent this as a possible solution, we force that a net opening can only be closed if it is contained in a path that powers a vertex. This is not the case for (E, e_1) in Figure 5.9.

The constraint below behaves exactly as intended.

$$\forall NO \in E_{NO}: \quad c_{NO} \leq \sum_{v \in V'} \sum_{no \in E_{no}} \sum_{i \in I_v^{no}} \alpha(NO, no, v, i) \cdot p_{v,i}^{no}$$

A net opening NO can now only be closed, $c_{NO} = 1$, if at least one of the paths that contains NO is closed, $p_{v,i}^{no} = 1$. If none of the paths that contain NO are closed, the right-hand side will be 0. Forcing NO to stay open, $c_{NO} = 0$.

5.2.6 Results

We have tested the Area Switch Planner on all the available routes. Since the Area Switch Planner builds upon the Customer Switch Planner we will compare the results of these two Switch Planners below. This way we clearly see the difference created by the new extension.

General overview

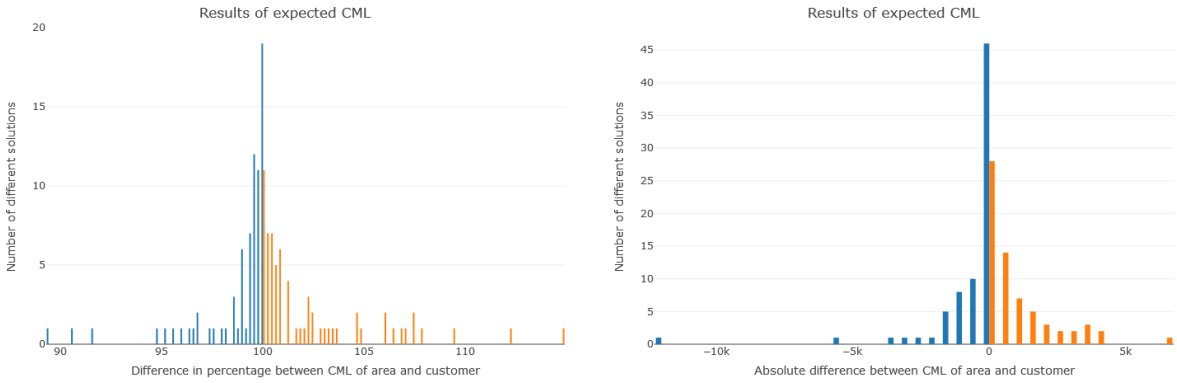
Table 5.5, Table 5.6 and Figure 5.10 show the results as before.
The results are a lot less clear this time. Considering the histograms it's hard to say that the total expected customer minutes lost has improved. Which is not what we expected at first, but after validation of the results the reason has become clear. Both the lower and higher ECML have the same reason:

Total	
142	
Lower	Higher
75	67

Table 5.5: Routes with different ECML

Extension	Total running time
Customer	42306 seconds
Area	45014 seconds

Table 5.6: Total running time for all routes



(a) Histogram for proportional difference in ECML

(b) Histogram for absolute difference in ECML

Figure 5.10

For each step separately the Outage Simulation Module with the Area Switch Planner can save more customers than before.

However depending on how many customers it saves extra, this might result in a lower or higher ECML for the route.

Lower ECML The Area Switch Planner returns a solution with a lower ECML than the Customer Switch Planner when any of the following happens

- The new solution has an equal amount of switching operations as before, but saves more customers.
- The new solution has more switching operations compared to before, but saves way more customers.

This is what we expected to happen.

Higher ECML We see a higher ECML for a route when the following happens

- The new solution has more switching operations compared to before, but saves *only a few extra* customers

When this happens in Step 2 of the Outage Simulation Module, we spend extra time to restore power to some customers, but all the customers that are saved in Step 3 will get power later than before. When the customers in Step 3 outweigh the extra customers saved in Step 2 of the Outage Simulation Module the overall ECML of the outage will increase.

The above validation showed that the Area Switch Planner functioned correctly and raised an interesting question. Should we save a minimal number of customers to make extra switching operations worthwhile?

This is something for Alliander to answer, but for now the Area Switch Planner seems to mimic the real world situation correctly. In every iteration it saves as many customers as possible.

Time There is only a slight increase in the total running time compared to the Customer Switch Planner. The Area Switch Planner is now only called upon once every step of the Outage Simulation Module, whereas this was not the case for any of the previous Switch Planners. The size of the problem has increased however and the solver takes way longer to solve a problem. Where previously this was in the hundredths of seconds, it now takes a couple of seconds. We do however gain time, since we only need to do all the data preparation once.
This is promising, since we haven't optimized the solver yet. We expect that further research in this area could make the Area Switch Planner much quicker.

5.3 Switch Planner - optimize over switching locations

The extension to the Switch Planner in this section can be added to any of the previously defined Switch Planners. We redefine the objective function, such that we optimize over switching locations instead of operations, hopefully improving on the number of relocations by the technician during an outage. Below we discuss the difference in the objective function together with new data and constraints. Lastly we show the results for the Customer Switch Planner with the new objective function.

5.3.1 Switching locations versus operations

The number of relocations of the technician determines the time before customers are with power again. Performing multiple switching operations at the same location might therefore be a good option. With this addition we can determine the number of different vertices that must be visited to perform all the switching operations, instead of only focusing on the number of switching operations.

5.3.2 Data preparation

For each switching operation we need to know at what vertices we are allowed to make the switch. This is denoted by l_e for each $e \in E_{op}$.

$$\forall e \in E_{op}: \quad l_e = \{v \in V : v \text{ endpoint of } e \text{ from which we can switch}\}$$

For most of the edges this will include both the endpoints. However sometimes it will only contain one vertex. The typical example is when e is a net opening. In this case we can only close the net opening at one specific vertex.

Decision variables

As stated we want the Switch Planner to decide at what vertices to switch for us. So we need to introduce some new decision variables for this

$$\forall v \in V: \quad s_v = \begin{cases} 1 & \text{at least one switching operation at vertex } v \\ 0 & \text{no switching operation performed at vertex } v \end{cases}$$

Notice that we do not count the number of switching operations at a location, just whether there is at least on switch operation at that location or not.

5.3.3 Objective function

The objective function that minimizes the number of switching locations becomes

$$\min \qquad \sum_{v \in V} s_v.$$

It just counts the number of switching locations that need to be visited for a solution. By minimizing over it, we only set $s_v = 1$ if necessary.

Example 5.3.1 (Location Customer Switch Planner)**.** The objective function of the Customer Switch Planner that takes into account the number of switching location instead of switching operations now becomes

$$\min \qquad \left\{ -F' \cdot \left(\sum_{v \in V_N} \sum_{no \in E_{no}} p_v^{no} \cdot \text{cust}(v) \right) + \sum_{v \in V} s_v \qquad \right\}.$$

Here we take $F' = 2 \cdot |V|$, since we need $F' > \sum_{v \in V} s_v$ because of Theorem 4.2.1.

5.3.4 Constraints

To correctly set s_v we need to add just one constraint for each e. We have $E_{NO} = \{e \in E_1 : e \text{ a net opening}\}$.

$$\begin{aligned} \forall e \in E_{NO} : \quad & \sum_{v \in l_e} s_v \geq c_e \\ \forall e \in E_1 \backslash E_{NO} : \quad & \sum_{v \in l_e} s_v \geq 1 - c_e. \end{aligned}$$

To understand why this works, we take a look at the two different cases.

$e \in E_{NO}$ Since $e \in E_{no}$ we know that l_e contains just one vertex s_v. There are two options we need to take into account

- $c_e = 0$ The solution hasn't closed the net opening e. So we do not need to visit s_v because of this edge. Indeed, we allow s_v to be zero.
- $c_e = 1$ We closed net opening e, so we should visit s_v to perform this action. Indeed, we have $s_v \geq c_e = 1$ as intended.

$e \in E_1 \backslash E_{NO}$ In this case l_e can contain up to two vertices. We again look at the two different options for c_e.

- $c_e = 1$ The solution still has e closed. So we do not need to visit any of the $s_v \in l_e$ because of this edge. Indeed, we see that $\sum_{v \in V} s_v \geq 1 - c_e = 0$. Allowing both options to be zero.
- $c_e = 0$ We did create a net opening at e. Now we need to visit at least one of it's endpoints. Indeed, we have $\sum_{v \in V} s_v \geq 1 - c_e = 1$. Forcing at least one of the $s_v \in l_e$ to be visited.

This shows us that this constraint behaves as intended.

5.3.5 Handling improved objective function

The suboptimizations that were already included with the original Switch Planner are still possible. The suboptimization over the residual capacity hasn't changed, however we need to take a close look at the suboptimization for smart secondary substations.

Prefer remote switches

We wanted to use remote switches where possible. We may even want to prefer a remote switch over a switching operation at a location that we already were.
Previously we ignored the remote switchable edges in the objective. Similarly we could ignore the vertices that represent a smart secondary substation, denoted by V_{sSS}. This however doesn't work as intended, since there would be no difference for the objective between picking a vertex which already had $s_v = 1$ to solve a solution and picking a smart secondary substation.

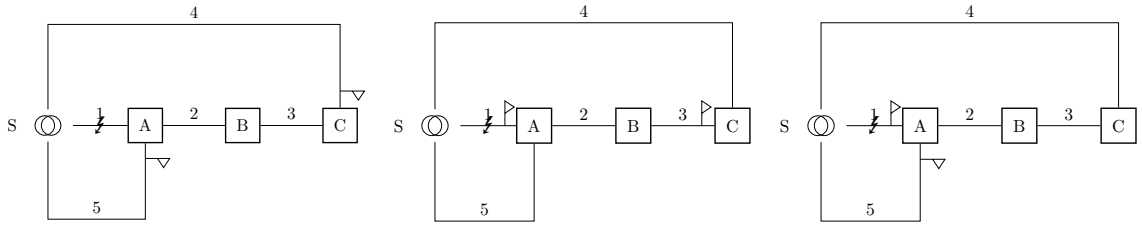
To solve this we let picking a $v \in V_{sSS}$ lower the objective, instead of not influencing at all. This way the objective will always pick these when possible. We do however only want the objective to actually pick them when they are used to perform a switching operation. To force this we need an extra constraint. Here $E_v \subset E_1$ contains all the edges that have v as an endpoint and $\deg(v)$ denotes the degree of vertex v ignoring net openings.

$$\sum_{e \in E_v \setminus E_{NO}} c_e - \sum_{e \in E_v \cap E_{NO}} c_e \leq \deg(v) + s_v$$

Note that in the outage situation we have $\sum_{e \in E_v \setminus E_{NO}} c_e - \sum_{e \in E_v \cap E_{NO}} c_e = \deg(v)$. This shows that s_v can only be set to one if any of the c_e changes value. A closed edge to net opening will lower the the left-side by 1, so will the closure of a net opening.

Number of switching operations

While taking into account the location is nice, we might also want to focus on the number of switching operations. Figure 5.11 shows two possible solutions to an outage.



(a) Example graph (b) Solution 1 (c) Solution 2

Figure 5.11

The two solutions are equal for the Location Customer Switch Planner, since at both we visit A and C to restore the power. We might still prefer the one with the fewest switching operations in this case. To make this distinction, we could add the number of switching operations back into the objective as before. But optimize on the number of location and only then on the number of switching operations.

5.3.6 Results

We have implemented and tested the Location Customer Switch Planner on all the available routes. Since it is based upon the Customer Switch Planner we will compare the results of these two Switch Planners. This way we clearly see the difference as a result of the new objective function.

General overview

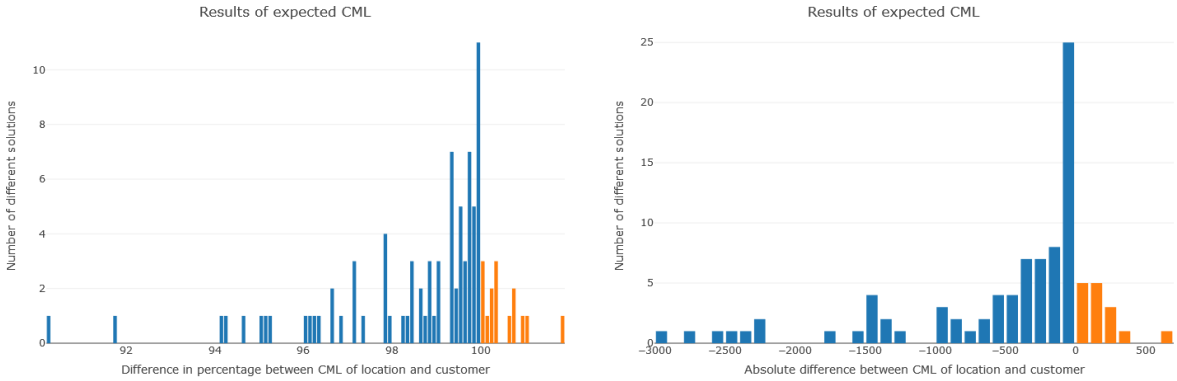
Table 5.7, Table 5.8 and Figure 5.12 show the results as we have seen before.

Total	
94	
Lower	Higher
79	15

Table 5.7: Routes with different ECML

Extension	Total running time
Customer	42306 seconds
Location	49378 seconds

Table 5.8: Total running time for all routes



(a) Histogram for proportional difference in ECML

(b) Histogram for absolute difference in ECML

Figure 5.12

The results clearly show an improvement in the expected customer minutes lost, which is what we expected. From the validation of the results we can conclude the following things.

Lower ECML The new objective function can better predict the number of movement needed by the technician to restore the power. Which is an important factor in the resulting ECML of a route.

Higher ECML While the new objective function better predicts the different locations that should be visited, it does not take the order of these actions into account. It sometimes happens that the technician has to move to another location in between switching operations on the same location. This of course removes all the advantage of having switching operations at the same location.
It would be nice if we were able to take the order into account when determine the objective value, but we have not yet found a way to do so.

Time The total running time of the simulation has increased quite a bit. We have no clear indication why this is. While we have slightly increased the Switch Planner, we did not expect this kind of difference in time.

5.4 Variation - Allowing more than two connected components

As discussed in Section 5.1 and Section 5.2 constraint A is no longer needed to keep the network radial for both the Customer and Area Switch Planner. Instead it is now used to make sure that the graph representing the network is still a forest with 2 trees. Below we briefly discuss why we want it to stay two trees and discuss the results of removing constraint A from the Area Switch Planner together with an example.

Forest with two trees

By keeping the whole graph as a forest with two trees, we force the customers without power to be connected with each other. This way, restoring power can always be done from one point if necessary. This is useful, since otherwise we might limit our remaining options in a later step, see Example 5.4.1. Furthermore when restoration using switching operations isn't possible, we only have to place a mobile power generator at one location to feed the remaining customers. If we remove this constraint, we could end up with lots of smaller parts in the network that each need a mobile power generator, making it more expensive and our estimated customer minutes lost less accurate.

5.4.1 Results

Table 5.9, Table 5.10 and Figure 5.13 show the results compared to the Area Switch Planner. We still use the same heuristics estimations as before in step 4 of the Outage Simulation Module.

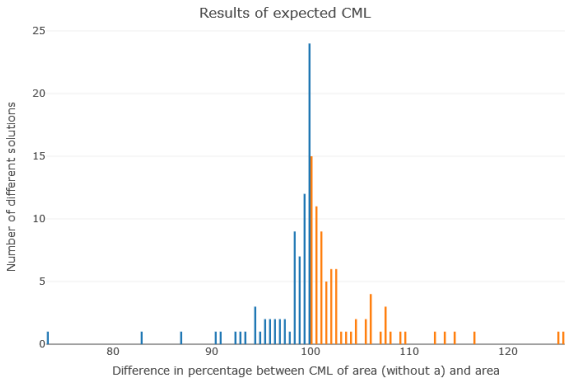
Area Switch Planner (without constraint A) versus Area Switch Planner

Total	
151	
Lower	Higher
76	75

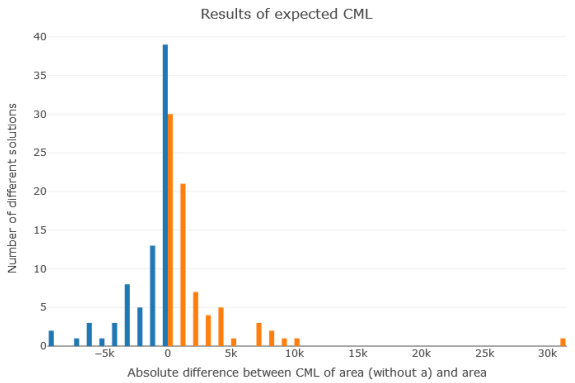
Table 5.9: Routes with different expected customer minutes lost

Extension	Total running time
Area	45014 seconds
Area without A	41528 seconds

Table 5.10: Total running time for all routes



(a) Histogram for proportional difference in expected CML

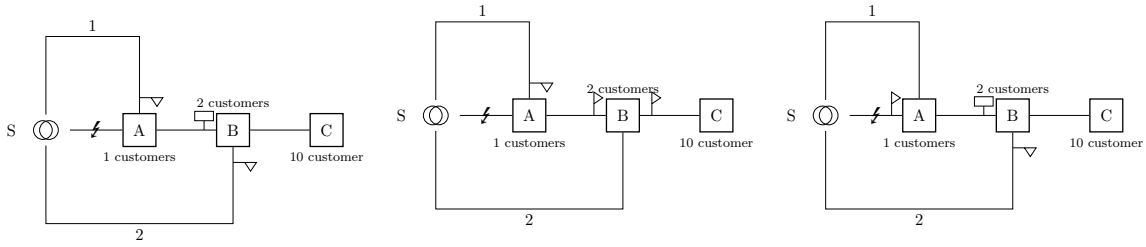


(b) Histogram for absolute difference in expected CML

Figure 5.13

Taking a look the results, there doesn't seem to be any real preference for one over the other purely looking at the customer minutes lost. We already saw why we could get a lower CML in Figure 5.2, Example 5.4.1 describes a simplified situation where the original Area Switch Planner performs better.

Example 5.4.1. Figure 5.14a shows a simplified network. Edge e_1 has enough residual capacity to feed A, B and C, however e_2 can only feed B.
When we follow the Outage Simulation Module for this example, we see that vertices A, B and C are without power because of the outage. There is are no remote controllable edges, so we skip step 1. By remote observability we know that we can restore power to the subnet $N = \{B, C\}$ in step 2.



(a) Example network

(b) Solution to step 2 without constraint A

(c) Solution to Step 3 with constraint A

Figure 5.14

The original Area Switch Planner cannot feed (part of) N in step 2, since it can only set at most one new net opening. Which must be used to isolate the subnet from the outage. This would result in e_2 feeding both B and C, which is impossible because of capacity constraints. So we conclude that we perform no switching operations in step 2, continuing to step 3 of the Outage Simulation Module. Here it solves the outage as shown in Figure 5.14c.
Contrary to above, the Area Switch Planner without cosntraint A does perform switching operations in step 2 of the Outage Simulation Module. This results in Figure 5.14b after Step 2. It's now impossible to restore power to C in step 3. This means that power is restored to C in step 4, which is more time consuming. Increasing the overall CML of the solution, even though that B is restored quicker than before.

The current Area Switch Planner without constraint A doesn't seem to improve on the current implementation. There might possibly exist an extension that could deal with the problems as described in Example 5.4.1, but these haven't been researched thus far.

Chapter 6

Conclusion

This chapter discusses the results of this thesis. We use these to answer the research question, recommend further steps and take a look at future research

6.1 Research question

Recall the research question

Can we extend the Switch Planner so that it incorporates some of the heuristic choices made in the Outage Simulation Module?

The short answer to the question is "yes". We have seen that both the Customer Switch Planner and Area Switch Planner indeed remove heuristic choices made in the Outage Simulation Module. By doing so, we take more possible solutions into account, improving the results of the Outage Simulation Module.

The Customer Switch Planner removed the heuristic choice of what customers not to feed in a subnet when it's impossible to feed all. The Area Switch Planner extended on this, by removing the heuristic choice that determined the order in which subnets were restored. By looking at all subnets at once, we were able to better distribute the capacity of the network to feed as many customers as possible.

Lastly we also discussed an improvement on the objective function of the Switch Planner. With it, we might not so much remove a heuristic choice of the Outage Simulation Module, but we improved the objective function of the Switch Planner, so that it can better take into account the movements of the technician.

Recommendations

This thesis has discussed multiple possible improvements on the Outage Simulation Module. In the end, Alliander has to choose one to actually use. Given the results in this thesis we recommend to implement the Area Switch Planner.
We have seen that this implementation removed the most heuristic choices within the Outage Simulation Module. With this, we are able to restore as many customers per step as possible. This seems to mimic the real world situation the best, since we would then also focus on restoring as many customers at each step.
It also seems to be the most versatile of them all, since we look at everything at once. It turns out that in practice there is often a limit on the number of switching operations they want to perform during each step. This can easily be implemented in the Area Switch Planner, but would be much

harder to solve optimally in any of the other Switch Planners. With the other Switch Planners we solve all the subnets separately, this means that we would have to heuristically determine how to divide the total amount of switching operations.

6.2 Future research

There are still some things left that need further research. These are discussed below.

Mixed Integer Linear Programming solvers During this thesis we have used available solvers for our Mixed Integer Linear Program. We have however not tweaked any of their settings. Given the fact that the solving time of the Mixed Integer Linear Programs has increased a lot with the Area Switch Planner, it could be beneficial to take a look at the available solvers and their settings. Setting a correct branching technique could possibly decrease the solving time noticeably. It would also be interesting to test the influence of the different constraints on the running time of the solvers. Some of them could possibly be tweaked to improve on the solving time.

Customer Minutes Lost as objective for Switch Planner It would be ideal if the Switch Planner could optimize on the actual customer minutes lost of an outage instead of just the number of customers it saves and the switching operations/locations that are needed to do so. We saw that some of the results with a higher ECML than before suffered from the fact that the objective function didn't make a distinction between two solutions, while in practice one was better than the other.
To solve this, the objective function has to take into account the order in which to restore power. It is not yet clear whether it is possible to implement this into the Switch Planner, while keeping it linear.

Questions for Alliander There is still much unclear over the protocol that is followed during an actual outage. This information is necessary since the Outage Simulation Module wants to follow these as closely as possible. We saw that some decisions can have an impact on the expected CML of a network. For instance do we perform two extra switching operations to save only a few extra customers in Step 2 of the Outage Simulation Module? As seen in Section 5.2.
There is talk of implementing software similar to the Outage Simulation Module that would be used to advice switching operations in case of a real outage. This implementation would be more elaborate than the Outage Simulation Module, it would for instance have to solve Step 4 more precisely. It would however make for a perfect benchmark to test results of the Outage Simulation Module.

Bibliography

[1] Alliander. *Mediabank*. Internal images from Alliander.

[2] M. klein Brink. "Minimizing Outage Time in an Electricity Network". MSc Thesis. 2018. URL: https://www.math.ru.nl/~bosma/Students/MyrtekleinBrinkMSc.pdf.

[3] Marco Chiarandini. "Linear and Integer Programming - Lecture Notes". URL: https://imada.sdu.dk/~marco/DM871/Notes/dm545-main.pdf.

[4] The On-Line Encyclopedia of Integer Sequences. *Sequence A000522*. URL: https://oeis.org/A000522.

[5] U.S.R. Murty J.A. Bondy. *Graph Theory*. Springer, 2008. ISBN: 9781846289699.

[6] R. Klumpenaar M. van der Meulen. "Optimale Mix Model". Internal Alliander document. 2019.

[7] Nobel Media. *The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 1975*. URL: https://www.nobelprize.org/prizes/economic-sciences/1975/summary/.

[8] A. Middag. "Elektriciteitsvoorziening". Internal Alliander document. 2013.

[9] J. Zwetsloot. "Approximating optimal solutions for power outages on the electrical grid by applying heuristical methods." MSc Thesis. 2019. URL: https://www.math.ru.nl/~bosma/Students/JelteZwetslootMSc.pdf.