Karlsruhe Institute of Technology

Transmission Network Expansion Planning for Curing Critical Edges

Master Thesis of

Lena Winter

At the Department of Informatics
Institute of Theoretical Informatics

Reviewers:	Prof. Dr. Dorothea Wagner
	Prof. Dr. Peter Sanders
Advisors:	Lukas Barth
	Franziska Wegner
	Matthias Wolf

Time Period: 8th November 2018 – 7th May 2019

Statement of Authorship

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text.

Karlsruhe, May 7, 2019

Abstract

The research field of Transmission Network Expansion Planning (TNEP) focuses on the problem of expanding existing transmission networks. This is done by choosing which edges to build from a candidate network. In this thesis we discuss the problem of expanding transmission networks with the objective to cure critical edges. Critical edges in the transmission network are identified as edges, whose failure lead to black outs. We apply a method proposed by Witthaut et al. [WRZ+16a, WRZ+16b] to identify critical edges. This method relies on a graph theoretical approach using network flows and can be applied independently of the underlying network model. Based on this idea, we discuss the effect of using only the graph theoretical network flows for TNEP for Curing Critical Edges (TNEP-CCE).

First we define the problem TNEP-CCE accurately. We show that the problem restricted to one critical edge is already NP-complete. A major part of this thesis is the development and introduction of different approaches to solve this problem. As a reference model we introduce the formulation as a Mixed-Integer Linear Program (MILP) using the DC approximation. Then we propose heuristics solely relying on the graph theoretical simplification of the transmission network. For the first heuristic we introduce the formulation as a MILP, then we discuss a heuristic based on the general idea of dynamic programming as well as modifications for this heuristic.

We evaluate all methods proposed in this thesis based on optimization time as well as result quality. Generally we observe that the heuristic based on dynamic programming almost always performs worse than the MILP using the network flow. For the optimization times we observe large speed ups for the MILP using the network flows in comparison to the MILP using the DC approximation. Concerning the result quality, we mainly looked at how critical the edges are after the optimization. We call this value the criticality of the network. We observe only very small differences in result quality between both MILPs. On most tested networks they perform equivalently. On the other networks, the model using the DC approximation cured on average one percent more of the initial criticality. This means that the approach of using only the network flow for TNEP-CCE leads to large speed ups while retaining most of the result quality.

Deutsche Zusammenfassung

Forschungsfragen, die sich mit dem Thema der Übertrangungsnetzerweiterung befassen, werden unter dem Begriff „Transmission Network Expansion Planning“ (TNEP) zusammengefasst. Bei der Übertragungsnetzerweiterung geht es in erster Linie darum, Kanten aus einem Kandidatennetz auszuwählen, um diese zur Erweiterung eines existierenden Übertragungsnetzes zu bauen. Diese Masterarbeit behandelt die Fragestellung wie ein Übertragungsnetz erweitert werden kann, um die Auswirkung von kritischen Kanten zu minimieren. In diesem Kontext werden Kanten dann als kritisch bezeichnet, wenn ihr Ausfall zu Stromausfällen führen kann. Zur Identifikation von kritischen Kanten wird in dieser Arbeit eine Methode von Witthaut et al. [WRZ+16a, WRZ+16b] eingesetzt. Diese Methode verwendet graph-theoretische Flüsse auf dem Übertragunsnetz um kritische Kanten zu identifizieren und ist dementsprechend unabhängig vom verwendeten Netzmodel einsetzbar. Basierend auf dieser Idee wird in dieser Arbeit untersucht ob das Problem „TNEP for Curing Critical Edges“ (TNEP-CCE) nur unter Verwendung von graph-theoretischen

Flüssen lösbar ist und wie sich diese Vereinfachung auf eine Lösung des Problems auswirkt.

Im ersten Teil dieser Arbeit wird das Problem TNEP-CCE formal definiert. Anschließend wird die NP-Vollständigkeit des Problems für einfache Instanzen, die nur eine einzelne kritische Kante enthalten, gezeigt. Es werden mehrere Methoden zur Lösung des Problems TNEP-CCE vorgeschlagen. Zunächst wird ein Vergleichsmodell als gemischt-ganzzahliges lineares Modell (MILP) formuliert, dabei wird das DC-Modell zur Berechnung des Lastflusses auf dem Übertragungsnetz verwendet. Für die Problemformulierung TNEP-CCE wird ebenfalls ein MILP eingeführt. Zusätzliche Heuristiken werden basierend auf dynamischer Programmierung entwickelt. Dazu wird zunächst ein Basismodell des Algorithmus zur Lösung von TNEP-CCE vorgestellt. Anschließend werden für dieses Basismodell mehrere Variationen vorgeschlagen um das Modell zu verbessern.

Bei der Evaluation werden sowohl die Laufzeiten der einzelnen Modelle als auch ihre Optimierungsergebnisse verglichen. Dabei wird festgestellt, dass die vorgeschlagene Heuristik basierend auf dynamischer Programmierung in beiden Fällen schlechter abschneidet als das MILP, das TNEP-CCE modelliert. Das Vergleichsmodell MILP mit Wechselstrommodell hat generell deutlich schlechtere Laufzeiten als das MILP für TNEP-CCE. Bei den Optimierungsergebnissen werden hauptsächlich die nach der Optimierung im Netz verbleibenden kritischen Kanten verglichen. Dabei wird der Wert der „Criticality“ verwendet. Dieser Wert gibt an, wie kritisch eine Kante ist. Zwischen beiden MILPs werden fast keine Unterschiede festgestellt. Auf den wenigen Netzen auf denen Unterschiede beobachtet werden, sind die Unterschiede kleiner als ein Prozent der initialen Criticality des zugehörigen Übertragungsnetzes. Für diese Netze hat das MILP für TNEP-CCE eine höhere Criticality.

Zusammengefasst wird in dieser Arbeit gezeigt, dass die Vereinfachung Wechselstromflüsse durch graph-theoretische Flüsse anzunähern für das Problem TNEP-CCE zu deutlichen Beschleunigungen der Laufzeit und dabei nur sehr kleinen Verschlechterungen der Optimierungsergebnisse führt.

Contents

1. Introduction **1**
1.1. Problem Statement . . . 1

2. Preliminaries **3**
2.1. Transmission Network: Description and Notation . . . 3
2.2. Network Model . . . 3
2.3. Finding Critical Edges . . . 5

3. Related Work **9**
3.1. Transmission Network Expansion Planning . . . 9
3.2. Curing Critical Edges . . . 11
3.3. Summary . . . 15

4. Data Preprocessing **17**
4.1. Test Data . . . 17
4.2. Libraries . . . 19
4.2.1. PyPSA . . . 19
4.2.2. Graphtools . . . 20
4.2.3. Gurobi . . . 20
4.3. Candidate Edges . . . 20
4.3.1. Simple Set of Candidate Edges . . . 20
4.3.2. Generate Fixed Number of Candidate Edges per Node . . . 22

5. Modeling to Cure Critical Edges **25**
5.1. Criteria for Criticality . . . 25
5.2. Problem Definition . . . 26
5.3. Mathematical Models . . . 27
5.3.1. Mathematical Model with DC Approximation . . . 27
5.3.2. Mathematical Model with Graph Flow . . . 30
5.4. Problem Complexity . . . 32
5.4.1. Complexity on General Graphs . . . 32
5.4.2. Complexity on Planar Graphs . . . 35
5.5. Algorithm Design . . . 37
5.5.1. Identifying Candidates That Influence a Critical Edge . . . 37
5.5.2. Expected Flow on Added Edge . . . 39
5.5.3. Selecting Edges to Add to the Graph . . . 45
5.5.3.1. A Heuristic for Choosing Candidate Edges . . . 47
5.5.4. Modifications . . . 49
5.5.4.1. Value Function . . . 49

6. Evaluation **51**
6.1. Evaluation Methods . . . 51

6.2. Reference Optimization Method - Mathematical Model with DC Approximation 52
6.2.1. Given Set of Critical Edges versus Calculated Set of Critical Edges . 52
6.2.2. Objective Minimizing the Sum of Criticalities versus Objective Minimizing Number of Critical Edges 61
6.3. Heuristic and Modifications 65
6.3.1. Computation Time 65
6.3.2. Result Quality 67
6.4. Mathematical Model with Graph Flow 72
6.4.1. Computation Times 72
6.4.2. Result Quality 75
6.5. Properties of the Optimized Networks 81

7. Conclusion **87**
7.1. Future Work 88

Bibliography **91**

Appendix **95**
A. Test Data: Topologies of Transmission Networks 95

Glossary

B the expansion planning budget. viii, 24, 25, 27–34, 41, 42, 44–46

E_{cand} set of candidate edges (a, b), $a, b \in N$. viii, 18–20, 24–26, 28–35, 39, 40

(a, b) edge (or link) from a to b, $a, b \in V, (a, b) \in E$. viii, 25, 26

E_{crit} set of critical edges $E_{\text{crit}} \subset E$. viii

$G = (V, E)$ Graph with a set of nodes V and a set of edges E. viii, 3, 10, 34

$P(v)$ demand ($P_v \leq 0$) or generation ($P_v > 0$) of node v. viii, 25

$\Theta(v)$ voltage angle of node v. viii, 3–5, 25

$\text{cap}(a, b)$ the capacity of edge $(a, b) \in E \cup E_{\text{cand}}$. viii, 5–8, 10–13, 24–34, 41, 44

$\text{cost}(a, b)$ investment cost for edge $(a, b) \in E_{\text{cand}}$. viii, 19, 20, 24, 25, 27–33, 41, 42, 44, 45

$\text{flow}(a, b)$ the graph theoretical flow on the edge $(a, b) \in E$. viii, 3, 6, 7

$f(a, b)$ the power flow on the edge $(a, b) \in E$. viii, 3, 5–8, 10–13, 23–29, 32, 34, 35, 38, 40, 44

$f^{\text{add}}(a, b)$ the additional flow needed to cure critical edge $(a, b) \in E_{\text{crit}}$. viii, 24–27, 29, 30, 32, 33, 36, 38–42, 45, 55

$f^{\text{red}}(a, b)$ the redundant capacity of edge $(a, b) \in E$. viii, 6, 7, 10, 11, 13, 23–26, 29, 34, 35, 38, 40

1. Introduction

1.1. Problem Statement

As the investments in renewable energies and therefore the share of those in the power supply increases, new investments in the current power transmission system are unavoidable. Renewable energies—especially wind and solar energy—are highly weather and location dependent and higher loads are expected on certain weather conditions [HVBG+10, WRZ+16a]. Because of this higher load an expansion of the grid capacity may be necessary.

The problems concerning transmission network expansions are categorized as 'Transmission Network Expansion Planning' (TNEP) problems and are more closely discussed in the related work Section 3 of this thesis. They describe the problem where, how and—in case of the dynamic TNEP problem—when to expand a transmission network to meet a given peak generation. Transmission network expansion can be optimized under different aspects. Most commonly the investment costs of new lines should be minimized but there are also aspects like network reliability and stability, consideration of distributed generation and environmental impact to consider [HHK13].

As stated above, future changes in power generation methods will possibly lead to higher loads on the transmission network. For TNEP it can be interesting to know where weak points of the transmission network can be found for higher loads. To show the effects of higher loads on transmission network topologies Witthaut et al. [WRZ+16a] discussed two criteria to identify critical edges in a transmission network. A critical edge describes an edge whose failure will most likely lead to an outage in the investigated system. These critical edges are calculated for a static setting on a network and will vary depending on the load of the system. The first of the two criteria to identify critical edges is based on the topology of the network and works independently from the used power model. It depends on the redundant capacity of an edge that is determined via the Edmond-Karp-Algorithm [EK72]. This criterion is easily calculated and the results presented in the article [EK72] and in the supplementary material [WRZ+16b] imply that this first criterion works quite well. We use the first criterion to identify critical edges in this thesis.

This thesis focuses on the expansion of transmission networks under the objective of curing critical edges, as defined by Witthaut et al. [WRZ+16a] for limited investment costs. To clarify this problem we discuss the network model used in this thesis as well as the criterion we use to identify critical edges in detail in Chapter 2. In Chapter 5 of this thesis we

define TNEP for Curing Critical Edges formally and discuss its complexity. We propose a Mixed-Integer Linear Program using the DC power flow approximation to take the physics of the transmission network into account. We use this MILP as reference model to compare our other optimization strategies to. Inspired by the criterion to identify critical edges using a graph theoretical approach, we also use a graph theoretical approach to propose different heuristics to solve TNEP for Curing Critical Edges. By introducing these heuristics we achieve further insight into the problem. In Chapter 6 we compare the performance of the proposed optimization methods considering computation times and success in curing critical edges.

2. Preliminaries

2.1. Transmission Network: Description and Notation

In this thesis the transmission network is considered to be an undirected graph $G = (V, E)$. In this graph V describes the set of nodes and E describes the set of edges $(a, b) \in E$ with $a, b \in V$. For each node $v \in V$ the active power power injection or withdraw of this node is $P(v)$ with $P : V \to \mathbb{R}$ and the reactive power injection or withdraw is $Q(v)$ with $Q : V \to \mathbb{R}$. A node $v \in V$ of this graph can either be a generator if they insert power in the network meaning $P(v) > 0$ or a consumer if they take power out of the network meaning $P(v) \leq 0$. Each node has a voltage angle denoted by $\Theta(v)$ with $\Theta : V \to \mathbb{R}$. The electrical power lines are modeled by the edges of the graph. Each edge has a given capacity cap: $E \to \mathbb{R}$, resistance $r : E \to \mathbb{R}$, conductance $g : E \to \mathbb{R}$, reactance $x : E \to \mathbb{R}$ and susceptance $b : E \to \mathbb{R}$. There is also the active power flow on each line $f : E \to \mathbb{R}$. The active power flow $f(a, b)$ on an edge $(a, b) \in E$ is calculated depending on the network model. In this thesis we will also use the graph theoretical flow on an edge denoted as flow : $E \to \mathbb{R}$. This graph theoretical flow $\text{flow}(a, b)$ on an edge $(a, b) \in E$ is calculated using a maximum flow algorithm. We define for both flows, the power flow as well as the graph theoretical flow, that $-\text{cap}(a, b) \leq f(a, b) + \text{flow}(a, b) \leq \text{cap}(a, b)$ and that skew symmetry applies meaning $f(a, b) = -f(b, a)$ and $\text{flow}(a, b) = -\text{flow}(b, a)$.

2.2. Network Model

There are different models for the simulation of an AC transmission network. For this thesis the DC power flow approximation of the transmission network will be used to calculate the initial power flow in the benchmark networks.

The most commonly used power flow models for TNEP are the AC power flow model and the DC power flow approximation model [HHK13]. We use the DC power flow approximation as showing the feasibility of a transmission network in the lossless AC power flow model is strongly NP-hard [BV15]. This is the case even though the lossless AC power flow model is already a simplified version of the AC power flow model. It is simplified by assuming transmission on the edges without losses by setting the resistance for each line $(v, u) \in E$ to $r(v, u) = 0$. The DC approximation provides a good approximation of the AC power flow and allows us to model our reference model more efficiently.

For better understanding of the DC power flow approximation, which will be explained later, the terms for active and reactive power in the AC model will be introduced. As

introduced in Section 2.1, the transmission network is modeled as a undirected graph $G = (V, E)$. The active power $P(v)$ of a node $v \in V$ is calculated according to Witthaut et al. [WRZ+16b] by:

$$P(v) = \sum_{u \in V} |U(v)||U(u)|(g(v,u)\cos(\Theta(v) - \Theta(u))) + b(v,u)\sin(\Theta(v) - \Theta(u))) \tag{2.1}$$

The reactive power $Q(v)$ of a node $v \in V$ is calculated according to Witthaut et al [WRZ+16b] by:

$$Q(v) = \sum_{u \in V} |U(v)||U(u)|(g(v,u)\sin(\Theta(v) - \Theta(u)) - b(v,u)\cos(\Theta(v) - \Theta(u))) \tag{2.2}$$

In the Equations 2.1 and 2.2 $U(v)$ denotes the complex voltage of node $v \in V$ and $|U(v)|$ denotes its voltage amplitude. The conductance of an edge (v, u) is $g(v,u) = \frac{r(v,u)}{x(v,u)^2 + r(v,u)^2}$ and the susceptance of the edge is $b(v,u) = \frac{-x(v,u)}{x(v,u)^2 + r(v,u)^2}$.

As mentioned before this thesis uses the DC power flow approximation. The DC power flow approximation simplifies the problem to calculate the AC power flow model by introducing some assumptions [McC12].

The first assumption is that the series resistance $r(v, u)$ is very small in comparison to the reactance $x(v, u)$ and can be approximated as $r(v, u) = 0$ for any edge (v, u). This assumption leads to the conductance being $g(v, u) = 0$ and the susceptance being $b(v,u) = -\frac{1}{x(v,u)}$. Applying the assumption on the power flow equation of the AC model we get for the active power:

$$P(v) = \sum_{u \in V} |U(v)||U(u)|b(v,u)\sin(\Theta(v) - \Theta(u)) \tag{2.3}$$

And the reactive power is calculated as:

$$Q(v) = \sum_{u \in V} -|U(v)||U(u)|b(v,u)\cos(\Theta(v) - \Theta(u)) \tag{2.4}$$

The second assumption used to obtain the DC approximation is that in typical operation conditions of a network the phase angle difference between two nodes is small. Using this assumption leads to the approximations for the trigonometric functions that $\sin(\Theta(v) - \Theta(u)) \approx \Theta(v) - \Theta(u)$, meaning the sinus is nearly linear for small phaseangle differences and $\cos(\Theta(v) - \Theta(u)) \approx 1.0$, meaning the cosinus is nearly constant for small phaseangle differences. Applying the assumption on the power flow equation of the AC model we get for the active power:

$$P(v) = \sum_{u \in V} |U(v)||U(u)|b(v,u)(\Theta(v) - \Theta(u)) \tag{2.5}$$

And the reactive power is calculated as:

$$Q(v) = \sum_{u \in V} -|U(v)||U(u)|b(v,u) \tag{2.6}$$

The equation of the reactive power can be transformed to:

$$Q(v) = -|U(v)|^2 b(v,v) + \sum_{u \in V, u \neq v} |U(v)|b(v,u)(|U(v)| - |U(u)|) \tag{2.7}$$

In most network analysis it is common to use the per-unit-system (p.u.) meaning that all the values of power, voltage, current, impedance, and admittance are given in relation

to chosen base values in this system [EH72]. Most commonly chosen are a voltage base U_{base} and a power base P_{base} value. For the DC approximation we will also use this per-unit-system.

The next assumption is that the amplitudes of the complex voltages are nearly equal in all nodes. Since we can choose the value of U_{base} we will set it so that $|U(v)| = 1.0\ pu$ for each node $v \in V$ if $|U(v)|$ is a multiplicator. Applying the assumption on the power flow equation of the AC model we get for the active power:

$$P(v) = \sum_{u \in V} b(v,u)(\Theta(v) - \Theta(u)) \tag{2.8}$$

And the reactive power is calculated as:

$$Q(v) = -b(v,v) + \sum_{u \in V, u \neq v} b(v,u)(|U(v)| - |U(u)|) \tag{2.9}$$

We assume at last that the difference between the voltage amplitudes $|U(v)| - |U(u)|$ are by a magnitude smaller than the difference between the phase angles $\Theta(v) - \Theta(u)$ and therefore we can neglect the reactive power and set $Q(v) = 0.0$.

Because of the first assumption, of transmission without losses on the edges, this leads to the following equation for calculating the power flow $f(v,u)$ on each edge $(v,u) \in E$ in the transmission network:

$$f(v,u) = b(v,u)(\Theta(v) - \Theta(u)), \quad \forall (v,u) \in E \tag{2.10}$$

The equation for the power flow calculation for a single edge (v,u) is given by Equation 2.10.

2.3. Finding Critical Edges

Witthaut et al. [WRZ+16a, WRZ+16b] propose two criteria to identify critical edges. Critical edge refers to an edge, whose failure could cause an outage. The calculation and definition of critical edges depend on the power flow model used. For the network models mentioned in Section 2.1, the AC power flow model and the DC power flow model, an edge (x,y) in a network $G = (V,E)$ is defined as critical critical, if its failure would lead to an overload $f(a,b)' > \text{cap}(a,b)$. Hereby $f(a,b)'$ is the recalculated flow in the network $G \setminus \{(x,y)\}$ for any edge $(a,b) \in E \setminus \{(x,y)\}$.

In this thesis we use the first criterion to identify critical edges which only depends on topology of the network and not on the used power flow model. Witthaut et al. show in the supplementary material [WRZ+16b] to their article that this criterion provides comparable results regardless of the underlying power flow model. Because of that we give a brief overview of the power flow model they used in comparison to the DC model, used in this thesis, and won't provide greater details of their network model. In the original article [WRZ+16a] they approximate the AC power flow model with the oscillator model. The oscillator model is a dynamic network mode in contrast the DC network model we use in our thesis is a static network model. The oscillator model is like the DC power flow model, described in Section 2.2, a simplification of the AC power flow model. Like in the DC power flow model the edge resistance is $r(a,b) = 0$ for all $(a,b) \in E$ and complex voltages are fixed to $|U_v| = 1.0pu$. The difference to the DC power flow model is that there is no assumption of small voltage angle differences. The second difference is that the nodes are modeled as rotating machines like wind turbines or electric motors [WRZ+16a]. In a stable network

modeled using the oscillator model all nodes are synchronized, meaning their rotating machines have the same frequency. In the oscillator model an edge is called critical if its deletion induce long-term desynchronization of the rotating machines. A desynchronization of the rotating machines leads to an large scale outage in the transmission network.

For a network with fixed generators and consumers, with given power generation or consumption per node, the network can be simulated with the oscillator model to determine critical edges for this load distribution. First the power flow $f(a, b)$ on each edge $(a, b) \in E$ in a transmission network is calculated according to the oscillator model. Then the ground truth, which edge is critical, is determined by deleting single edges from the network and observing whether that induces long-term destabilization in the network. As simulating the network is time consuming and computationally expensive [WRZ+16a] Witthaut et al. propose heuristics to identify critical edges.

Witthaut et al. [WRZ+16a] introduce two criteria for identifying critical edges. The first one is based on the topology of the network and its capacity for rerouting flow, while the second criterion is derived by modeling general changes of capacity of a single edge and their effect on the network. As we base this thesis on first criterion to identify critical edges, the second one is not introduced here.

The first criterion is referred to as redundant capacity predictor in the article. It is calculated in a purely graph theoretical way and does not depend on the underlying network power flow model. We refer to the 'redundant capacity predictor' as 'redundant flow predictor' in this thesis, because in this thesis we refer to capacity in the sense of actual capacity of edges while the capacity meant in the term 'redundant capacity predictor' refers to the capacity of the graph the reroute flow and not the capacity of single edges. Let $(a, b) \in E$ denote an edge, $f(a, b)$ the initial power flow on this edge and $\text{cap}(a, b)$ its capacity. The maximum flow between two nodes $a, b \in V$ after the edge (a, b) is deleted from the network is called the redundant flow $f^{\text{red}}(a, b)$ with $f^{\text{red}} : E \to \mathbb{R}$. The redundant flow $f^{\text{red}}(a, b)$ of an edge $(a, b) \in E$ of the graph $G = (V, E)$ is an indicator for how much flow can be rerouted if the edge (a, b) fails. In other words, the redundant flow $f^{\text{red}}(a, b)$ describes the maximum a,b-flow in the graph $G' = G \setminus \{(a, b)\}$. The redundant flow is calculated using the Edmond-Karp-Algorithm [EK72] for calculating the maximum flows of a single source, single sink network in graph theory. The pseudo code for the calculation of $f^{\text{red}}(a, b)$ from the article [WRZ+16b] can be seen in Algorithm 2.1. The input for the calculation is the graph $G' = (V, E')$, the capacities $\text{cap}(u, v)$ and an initial flow $\text{flow}(u, v) = f(u, v)$ for each edge $(u, v) \in E'$ and the edge $(a, b) \in E$ which should be tested. The input flow result from the initial power flow calculations on the transmission network, done according to the oscillator model. The algorithm has as input the directed equivalent to the undirected graph G because the Edmond-Karp-Algorithm is defined on directed graphs. The first step of the algorithm is to delete the edge (a, b) and its reversed edge (b, a) by setting the capacities $\text{cap}(a, b) = \text{cap}(b, a) = 0$ and initial flow $f(a, b) = f(b, a) = 0$. In Line 3 the residual capacity $\text{cap}'(u, v)$ is calculated for each edge $(u, v) \in E$. The shortest path in the residual graph between nodes a and b is calculated in Line 4. This is done by a subroutine `shortestPath` which could, for instance be an implementation of Dijkstra's Algorithm for Shortest Paths. After pushing the maximum possible flow along the augmenting path, this additional flow is added to $f^{\text{red}}(a, b)$ in Line 8. The algorithm adds flow to $f^{\text{red}}(a, b)$ until no augmenting path is found.

As mentioned the algorithm REDUNDANTFLOW is a version of the Edmond-Karp-Algorithm. The difference between the two algorithms is the deletion of the edge (a, b) and declaring a as source and b as sink, the run time of the algorithm $\text{REDUNDANTFLOW}(G, a, b)$ is in $O(|V||E|^2)$.

Algorithm 2.1: REDUNDANTFLOW

Input: Graph $G = (V, E)$, capacity $\text{cap}(u, v)$ and initial flow $\text{flow}(u, v)$ for each edge $(u, v) \in G$, edge (a, b)
Output: Redundant flow $f^{\text{red}}(a, b)$ of edge (a, b)

$\text{cap}(a, b), \text{cap}(b, a), \text{flow}(a, b), \text{flow}(b, a) \leftarrow 0$
$f^{\text{red}}(a, b) \leftarrow 0$
// Calculate residual capacity for each edge
$\forall (u, v) \in G : \text{cap}'(u, v) \leftarrow \text{cap}(u, v) - \text{flow}(u, v)$
// Calculate shortest path p from a to b in residual graph
$p \leftarrow \text{shortestPath}(\text{cap}', a, b)$
while $p \neq Null$ **do**
    // Calculate the maximum possible flow of path p
    $\text{cap}_{\max} \leftarrow \min_{(i,j) \in p} \text{cap}'(i, j)$
    // Increase flow along the path p
    $\forall (i, j) \in p : \text{flow}(i, j) \leftarrow \text{flow}(i, j) + \text{cap}_{\max}$
    $f^{\text{red}}(a, b) \leftarrow f^{\text{red}}(a, b) + \text{cap}_{\max}$
    // Calculate residual capacity for each edge
    $\forall (u, v) \in G : \text{cap}'(u, v) \leftarrow \text{cap}(u, v) - \text{flow}(u, v)$
    $p \leftarrow \text{shortestPath}(\text{cap}', a, b)$
return $f^{\text{red}}(a, b)$

The criterion for critical edges identifies an edge as critical if $|\frac{f(a,b)}{f^{\text{red}}(a,b)}| > h$ for some threshold h. Considering that the redundant capacity is always a positive value as it is determined using a maximum flow calculation, this can also be written as $|f(a, b)| > h \cdot f^{\text{red}}(a, b)$. For a threshold $h \geq 1.0$ this means that the redundant flow $f^{\text{red}}(a, b)$ the graph G can provide for an edge $(a, b) \in E$, has to be larger than the absolute value of the initial power flow an the edge $f(a, b)$ for the edge to be stable. In the article [WRZ+16a] Witthaut et al. propose a value of $h = 0.614$. This means for the redundant flow of an edge $(a, b) \in E$, that the redundant flow has to fulfill $f^{\text{red}}(a, b) > \frac{1}{0.614} \cdot |f(a, b)| \approx 1.64 \cdot |f(a, b)|$ for the edge (a, b) to be stable.

In the article, the authors test three different topologies of networks with randomly generated distributions of generators and consumers to obtain different load distributions. The three network topologies they used are the topologies of the high voltage transmission grids of Great Britain [RSTW12, SBP+08] and Scandinavia [MHKS14] and the IEEE 118-bus test grid [Chr00]. To obtain a large number of test data they generate test data by randomly selecting 10% of nodes to be generators and the rest to be consumers for a heterogeneous network set up or 50% each for a homogeneous setup. For each test case the total power generation and consumption stays the same and is uniformly distributed to the generators and consumers respectively. Also the edge capacity for each test case stays the same. In the homogeneous case each edge has the same capacity. In the heterogeneous case edges connected to the generators have double the capacity of the other edges. The difference between the test cases is the distribution of generators in the network, which is chosen randomly. In total they test 400 different network settings with 66.000 edges in total.

To quantify the performance of the criteria and show the effects of different thresholds for h they use the *Receiver Operating Characteristics* (ROC) curve. The ROC curve originates in machine learning applications. It depicts the ration of the sensitivity (fraction of correct predictions) to the false positive rate of a decision criterion. There are two measures for how good a ROC curve is. The first one is the minimal distance to the perfect operation

point (0,1), which means that no false positives and all true positives are detected. The second one is the area under the curve, for an optimal prediction result the area is 1.

To use the criterion to identify critical edges $|\frac{f(a,b)}{f^{\text{red}}(a,b)}| > h$, it is important to chose a threshold h that suits the application. A high value of h leads to nearly no false positives in the result but has also relatively low sensitivity compared to a low value for h. For most applications the value of h that leads to the point closest to the perfect operation point, is suitable. For the tested networks the value for h, that leads to the point closest to the perfect operation point, is a value of $h = 0.614$.

Witthaut et al. choose the value $h = 0.614$ to compare the 'redundant flow predictor' to criteria like load $L_{ab} = |\frac{f(a,b)}{\text{cap}(a,b)}|$ and flow $|f(a,b)|$. They find their predictor reduces incorrect predictions by more than 7 times, compared to the simpler criteria load $L_{ab} = |\frac{f(a,b)}{\text{cap}(a,b)}|$ and flow $|f(a,b)|$.

The usage of the redundant flow to formulate a criterion to identify critical edges is quite interesting. By using the criterion the physical properties of the transmission network are not taken into account during the identification of critical edges. However we think that the formulation of the criterion to identify an edge (a,b) as critical if $|\frac{f(a,b)}{f^{\text{red}}(a,b)}| > h$ might be to restricting. Witthaut et al. clearly formulate in their article, that the criterion should represent the ability of the graph to reroute the power flow on the examined edge through the the graph without the examined edge. We know that the skew symmetry property applies for power flows, meaning $f(a,b) = -f(b,a)$ for all edges $(a,b) \in E$ and therefore $|f(a,b)| = |f(b,a)|$. For the redundant flow there is generally no symmetry between $f^{\text{red}}(a,b)$ and $f^{\text{red}}(b,a)$. This means that, if we want an edge (a,b) in the transmission network to be stable, we have to check the criterion for both directions of the edge $|\frac{f(a,b)}{f^{\text{red}}(a,b)}| \leq h$ and $|\frac{f(b,a)}{f^{\text{red}}(b,a)}| = |\frac{f(a,b)}{f^{\text{red}}(b,a)}| \leq h$. Considering the case with an edge $(a,b) \in E$ and $f(a,b) = 50$ units of flow and $h = 1$, this means that we have to have $f^{\text{red}}(a,b) \geq 50$ and $f^{\text{red}}(b,a) \geq 50$.

The mathematical formulation for the redundant flow in both directions to support the absolute value of the power flow of an edge seems to be more restrictive than the original idea for the 'redundant flow predictor' and the way it is implemented in the Algorithm 2.1. The original thought stated in the article is, that the criterion should represent the ability of the graph to reroute the power flow on the examined edge through the the graph without the examined edge. Therefore we think that the formulation for identifying an edge as critical if

$$h < \begin{cases} \frac{f(a,b)}{f^{\text{red}}(a,b)} & f(a,b) \geq 0 \\ \frac{f(a,b)}{f^{\text{red}}(b,a)} & f(a,b) < 0 \end{cases} \tag{2.11}$$

is the intended formulation. We will use this formulation in this thesis.

In this thesis we propose a heuristic to expand transmission networks by minimizing the number of critical edges, identified using the 'redundant flow predictor', just taking the network topology and the initial flow on the network into account. We suspect, that the 'redundant flow predictor' provides these great results because graph theoretical flow full fills the constraint of flow conservation. Flow conservation is one of two constraints modeled in the DC power flow approximation. It is called in this context Kirchhoff's' current law, meaning all current entering a node also leaves the node. However, the graph theoretical flow does not model the second constraint of the DC power flow approximation which describes the power flow on an edge depending on the phase angles of the nodes, which the edge connects. We hope that an heuristic based on the idea of this criterion will propose functional transmission network expansion in a time equal or faster than methods taking the physical properties of an AC transmission network into account.

3. Related Work

This chapter is organized into two parts. As the goal of this thesis mainly concerns the expansion of transmission networks, the first part of this section is focused on state-of-the-art work concerning the 'Transmission Network Expansion Planning' (TNEP).

In Section 3.2 we review an article by Rohden et al. [RWTMO17]. This article is proposed by the same research group like the 'redundant flow predictor' to identify critical edges we use in this thesis. In this article the authors discuss different methods to cure critical edges. Because the authors propose in this article also methods to cure critical edges, which are identified using a slightly different criterion, it is the article which is closest to the thematic of this thesis.

3.1. Transmission Network Expansion Planning

Transmission Network Expansion Planning describes the problem to expand a transmission network considering one or multiple objectives for a given set of constraints with edges from a set of candidate edges. The candidate edges can also supplicate existing network edges, which often symbolizes the capacity expansion of an existing edge. A comprehensive review of this field is provided by Hemmati et al. [HHK13]. The definition of the researched problem can vary a lot between different approaches to this topic, depending on the chosen network model, objectives and constraints.

The most commonly used network models are the AC model and the DC approximation [HHK13]. As we already discussed in Section 2.2 of this thesis, the AC model will provide more accurate power flows than the DC approximation but is more complicated to solve. In the article by Bent et al. [BCGVH14] it is shown that there is a significant difference between solution obtained by using the DC approximation compared to solutions obtained by using the AC model. The short comings when using the DC approximation in TNEP are proven by showing that solutions computed using the DC approximation exhibit constraint violations when converted into the AC model. In the article [BCGVH14] it is proposed to use the LPAC model for power flows to bridge these gaps. The LPAC model is an AC power flow approximation, which captures line losses, the reactive power flow and allows some difference between the voltage angle magnitudes. These properties are not taken into account when using the DC approximation for power flows.

The objectives can also vary for different formulations of TNEP problems. Some commonly used objectives are the minimization of investment costs, operation costs or reliability costs.

The constraints consist of constraints for the chosen network model, as well as additional constraints like for example an investment budget, reliability constraints or environmental impact limits [HHK13].

Moulin et al. [MPS10] show that TNEP is NP-hard. To show the complexity they use the DC approximation as network model and the minimization of investment cost as objective. They use the constraints originating from the use of the DC model without adding additional constraints. They show the NP-hardness by reducing the Steiner-Tree problem to this version of the Transmission Network Expansion Planning problem.

Because of the complexity of TNEP there are different options of optimization methods to use. The optimization methods can be divided into mathematical and meta-heuristic optimization methods.

First we will review some of the mathematical optimization methods. One mathematical method often used, especially when using the DC approximation as network model, is the formulation as Mixed Integer Linear Program (MILP). Alguacil et al. use the formulation as MILP in the article [AAC10]. In this article the Alguacil et al. quantify the effect of an deliberated attack on a transmission network. The goal is to expand the transmission network in an cost-efficient way, while also minimizing the effect of attacks on the network. To achieve this, the authors quantify the effect of attacks on a network to use this quantification in the objective function. As the objective they propose to minimize the sum of their attack quantification and the total expansion cost. The constraints originate in the usage of the DC approximation as network model.

Another popular mathematical approach to formulate the TNEP problem as Mixed Integer Non-Linear Programs and use a fast solving method for example to use Benders decomposition. Benders decomposition is a mathematical programming method to solve large Mixed Integer Non-Linear Programs (MIP). To solve a MIP the Benders decomposition divides the problem into two separated problems, the Master-Problem and the Sub-Problem. Each of this problem has a different part of the decision variables from the original MIP. For a fixed solution for the Master-Problem the Sub-Problem is solved. If the Sub-Problem is infeasible new constrains are added to the Master-Problem, so-called Bender-cuts. This process is repeated until no Bender-cuts are added. The Benders decomposition is for example used by Binato et al. [BPG01]. They use as objective the minimization of investment cost and as network model the DC approximation.

Dusonchet and El-Abiad [DEA73] propose an approach using Dynamic Programming and Discrete Optimizing called Discrete Dynamic Optimizing. In the article the authors consider a dynamic form of TNEP. In the dynamic TNEP planning horizon the problem is considered over a longer time span. This means that changes in load distribution over time need to be considered to answer question when and how to expand the network. We did not further explain dynamic Transmission Network Expansion Planning as this thesis considers the static version of the problem. In the static version of TNEP we propose network expansions by considering snapshots of a network, meaning considering one specific load distribution at a time. None the less this article is quite interesting for the use of the Dynamic Programming. The idea Dusonchet and El-Abiad introduce is, that they initialize an expansion strategy for one planning horizon randomly and than optimize this strategy using Dynamic Programming but only allowing neighboring strategies as alternatives. Therefore they find the local optimum. This procedure is repeated until an heuristic stopping criterion is met. This stopping criterion is, that the expected improvement of another optimization is smaller than some chosen constant.

Some of the meta-heuristic approaches used for solving TNEP problems are Generic Algorithms [CZP10], Tabu Search [DSODOB01], the Frog Leap Algorithm [ESH11] and

artificial Neural Networks [ASEA02]. But since we will not use any meta-heuristic in this thesis we will not further review these articles.

3.2. Curing Critical Edges

Rohden et al. [RWTMO17] propose a heuristic to find edges in the transmission network whose failure would most likely lead to an outage. These edges are called critical edges and the authors also propose three different strategies to cure those critical edges.

As network model the authors use the oscillator model, which is a simplified version of the AC power flow model and is explained in more detail in Section 2.3. The article [WRZ+16a, 2.3], on which this thesis is primarily based on, is written by the same research team as the article described in this section. Therefore the identification of critical edges, which will be explained in the next paragraph, is quite similar.

The transmission network is represented by a graph $G = (V, E)$. The nodes in the graph $v \in V$ represent the buses were power is either injected into the network by generators or withdrawn from the network by consumers. The nodes power output is $P : V \to \mathbb{R}$. A node is called a generator if $P(v) > 0$ or a consumer if $P(v) \leq 0$. The edges $(a, b) \in E$ of the graph correspond to the transmission lines of the transmission network with a capacity of $\text{cap}(a, b)$. The power flow on an edge $(a, b) \in E$ is denoted by $f(a, b)$ and is calculated using the oscillator model. The authors assume that the initial network have a stable operating state, so that the demand of all nodes can be satisfied, no edge is overloaded and the frequencies of generators and consumer are synchronized. The authors also assume that the networks may be operated under a high load so that the failure of one edge could lead to desynchronisation of the network. Edges whose failure lead to desynchronisation are called critical edges.

The authors propose the following heuristic to find critical edges. Let $(a, b) \in E$ denote an edge with power flow $f(a, b)$ and $G' = G \setminus \{(a, b)\}$ the graph, that can be obtained by deleting edge (a, b) from the Graph G. The redundant flow $f^{\text{red}}(a, b)$ is calculated, by calculating the maximum a,b-flow in the graph G'. This is done by running a version of the Edmond-Karp-Algorithm [EK72] with a as source and b as sink in graph G'. An edge is referred to as critical if $f^{\text{red}}(a, b) < f(a, b)$.

The method to obtain the redundant flow $f^{\text{red}}(a, b)$ of an edge $(a, b) \in E$ is the same method as proposed by Witthaut et al. [WRZ+16a, 2.3]. The difference between critical edges in the article by Rohden et al. [RWTMO17] in comparison to the critical edges in the article by Witthaut et al. [WRZ+16a, 2.3] is the relative amount of redundant flow $f^{\text{red}}(a, b)$ needed to declare an edge as not critical or stable. Witthaut et al. propose to use a threshold h to determine an edge to be critical if $|\frac{f(a,b)}{f^{\text{red}}(a,b)}| > h$. Rohden et al. [RWTMO17] use basically a threshold of $h = 1.0$, so that an edge is critical if $|\frac{f(a,b)}{f^{\text{red}}(a,b)}| > 1.0$. The method used by Witthaut et al. [WRZ+16a, 2.3] is more sophisticated, because this method allows to adjust the threshold h accordingly. This is important as explained in [WRZ+16a, 2.3], because the threshold allows adjustments between the false positive rate, which is larger for a small value of h, and the sensitivity of the predictor, which is higher for a small value of h. As already discussed, a value for h which leads to the value closest to the perfect operating point— the point with no false positives but 100% sensitivity—is preferable for most applications. This optimal value for h depends on the topology of the network, but for the test networks used in [WRZ+16a, 2.3] was determined to be $h = 0.614$, which is more sensitive than the value of $h = 1.0$ which is used by Rohden et al. [RWTMO17].

For this thesis we will use a threshold of $h = 0.614$ as proposed by Witthaut et al. [WRZ+16a, 2.3], to identify an edge $(a, b) \in E$ as critical if $\frac{f(a,b)}{f^{\text{red}}(a,b)} < h$. We prefer using a

threshold of $h = 0.614$ in comparison to the threshold of $h = 1.0$ that was proposed by Rohden et al. [RWTMO17], because the first threshold was determined by experiments and reasoning using the Receiver Operating Curve, while the threshold by Rohden et al. seems to be chosen without further explanations.

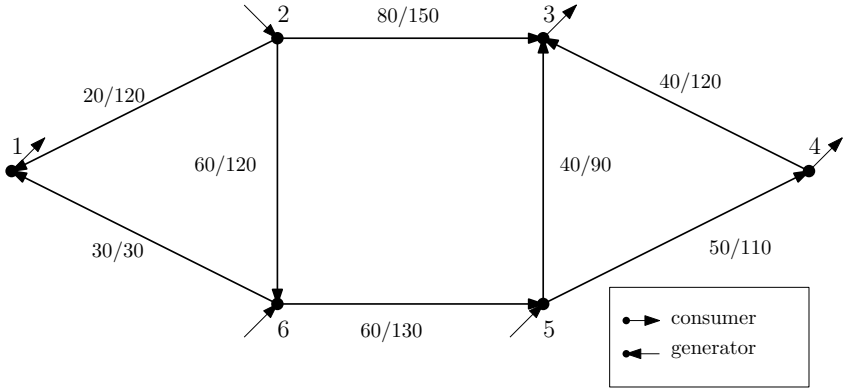


Figure 3.1.: Graph showing the initial example graph $G = (V, E)$. The arrows on the nodes imply whether they are generators (ingoing arrow) or consumer (outgoing arrow). Each edge $(a, b) \in$ is marked with its initial flow and its capacity:$f(a, b)/\text{cap}(a, b)$.

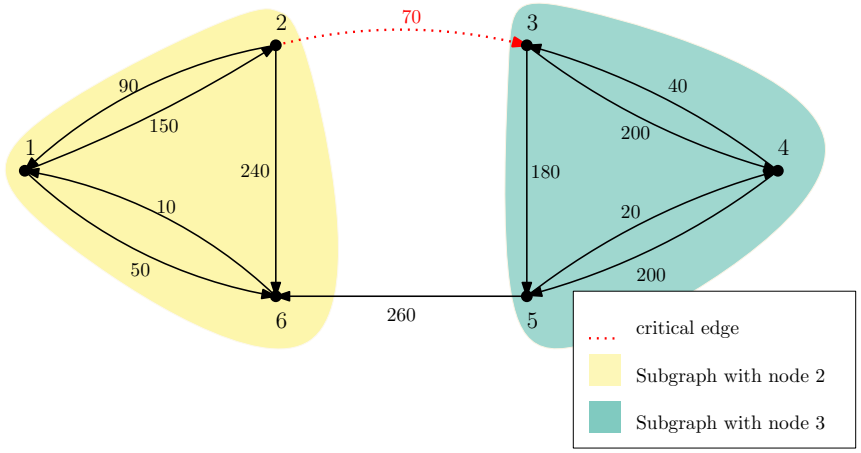


Figure 3.2.: Graph showing the residual graph after calculating the redundant capacity for edge $(2, 3)$ in the Graph $G' = G \backslash \{(2, 3)\}$, which is marked in red. The maximum flow calculated is written on the edge $(2, 3)$. The edge $(2, 3)$ is critical because its redundant capacity is smaller than its flow $f^{\text{red}}(2, 3) = 70 < 80 = f(2, 3)$. The residual graph divides into two parts marked yellow and green.

Figure 3.1 shows an example graph G for better understanding of critical edges. The edges of the graph are marked with their initial flow as well as their capacity. In this example, the flow on the edges was chosen for the sake of the example. In the example we calculate the residual capacity for edge $(2, 3)$. To do this the edge $(2, 3)$ is deleted from the graph resulting in the graph $G' = G \setminus \{(2, 3)\}$. Then the maximum $2, 3$-flow is calculated in G' using the Edmond-Karp-Algorithm. This leads to the residual graph shown in Figure 3.2 the maximum $2, 3$-flow calculated is 70 units. In Figure 3.2 the edge we examine $(2, 3)$ is marked as red dotted edge, marked with the maximum $2, 3$-flow. The residual capacity of edge $(2, 3)$ is therefore $f^{\text{red}}(2, 3) = 70$. The edge $(2, 3)$ of the example graph is a critical edge because its redundant capacity is smaller than the power flow on this edge: $f^{\text{red}}(2, 3) = 70 < 80 = f(2, 3)$. The residual graph divides into two parts marked

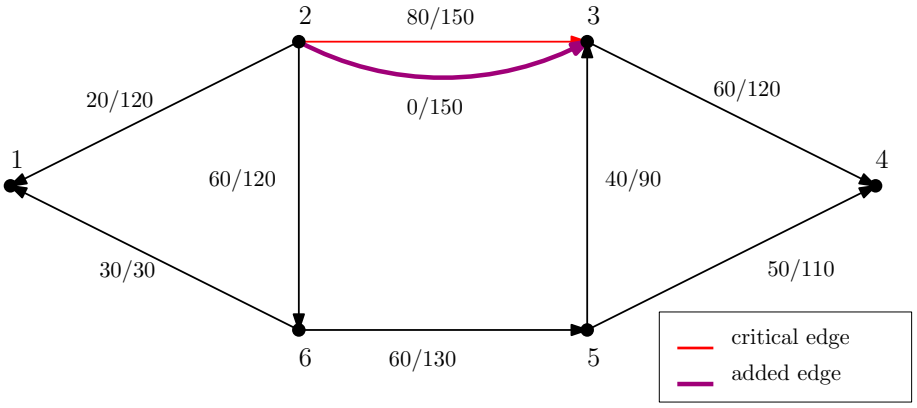
yellow and green, originally connected by the edge $(6, 5)$ in G'. As the capacity of edge $(6, 5)$ is fully used the edge is not included in the residual graph.

The authors propose three strategies for curing a critical edge. The strategies are designed for a single critical edge, which is already known by the time the strategies are applied. The first strategy, referred to as 'strategy A' in the article, is to duplicate the critical edge itself. The second strategy, referred to as 'strategy B' in the article, is a variation of the first strategy. The capacity of the duplicated edge (a, b) is increased incrementally by steps of $0.1K$, where K is the uniform capacity of all edges, until the critical edge is not critical anymore. Since we don't assume uniform capacity we will use K as the average capacity of the network. The third strategy, referred to as 'strategy C' in the article, is to add capacity to the bottleneck of the shortest path p between a and b in G'. The bottleneck is thereby defined as the edge (a, b) with the smallest value $\text{cap}(a, b) - f(a, b)$ for all edges $(a, b) \in p$. The capacity of the bottleneck is then increased to $\text{cap}(a, b)' = 1.1\text{cap}(a, b)$. If the critical edge is not cured the procedure has to be repeated.

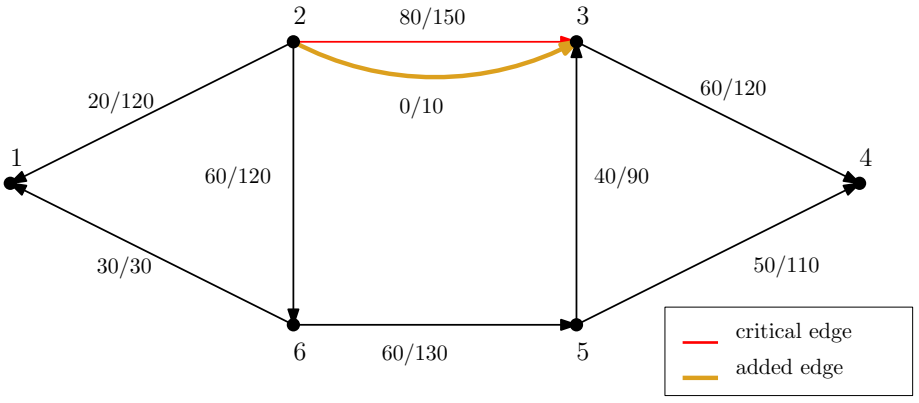
In Figure 3.3 the results are shown if the three curing strategies are applied on the example graph G to cure the critical edge $(2, 3)$. In Figure 3.3a the result of curing strategy A is shown. This strategy is to duplicate the critical edge with the same capacity. For the example graph this means adding another edge $(2, 3)'$ of capacity $\text{cap}(2, 3)' = 150$. In Figure 3.3b the result of applying curing strategy B is shown. Strategy B also duplicates the critical edge $(2, 3)$ but adjusts the capacity of the new edge in steps of $0.1K$. We choose $K = 100$ because in the article the authors assume all edges have uniform capacity K. We won't assume uniform capacity but will take the approximate average capacity of all edges. is the and therefore the critical edge is cured after one step and the resulting capacity $\text{cap}(2, 3)' = 10$. In Figure 3.3c the result of applying curing strategy C is shown. As $\text{cap}(a, b) - f(a, b)$ is smallest for $(5, 3)$ with $\text{cap}(5, 3) - f(5, 3) = 90 - 40 = 50$ the capacity of $(5, 3)$ is increased. Its capacity is increased two times in the beginning, because it is still the bottleneck after the first increase. After those two increases the capacity is $\text{cap}(5, 3)' = 1.1^2\text{cap}(5, 3) = 1.1^2 \cdot 90 = 108.9$. Then $\text{cap}(6, 2) - f(6, 2) = 60$ is smallest and therefore $(6, 2)$ is the bottleneck. The capacity of edge $(6, 2)$ is increased by 12 units to $\text{cap}(6, 2)' = 132$. Then line $(5, 3)$ is the bottleneck again with $\text{cap}(5, 3)' - f(5, 3) = 68.9$ and the capacity is increased to $\text{cap}(5, 3)'') = 119.78$. After this step the edge is still critical. Until this point all increases in capacity had no effect on the redundant flow which is still $f^{\text{red}}(2, 3) = 70$. At least the capacity for the edge $(6, 5)$ is increased by 13 units to $\text{cap}(6, 5)' = 143$. This cures the critical edge because the redundant flow of the graph is increased and therefore $f^{\text{red}}(2, 3) = 83 > 80 = f(2, 3)$.

The performance of these network expansions is measured by how much capacity is needed to cure all critical edges. In the article [RWTMO17] it is stated that strategy B and C outperform strategy A. Also it is stated that the performance of strategy C depends on the network topology. Meaning if the critical edge is caused by one bottleneck in the grid the performance of strategy B and C are nearly identical. If there are more than one bottleneck as shown in the example in Figure 3.3c strategy C performs worse than strategy B. But the authors also state that the time performance of strategy B depends on the used constant K. If K is chosen to large strategy C can outperform strategy B. If K is chosen to small strategy B may need a lot of iterations to cure a critical edge.

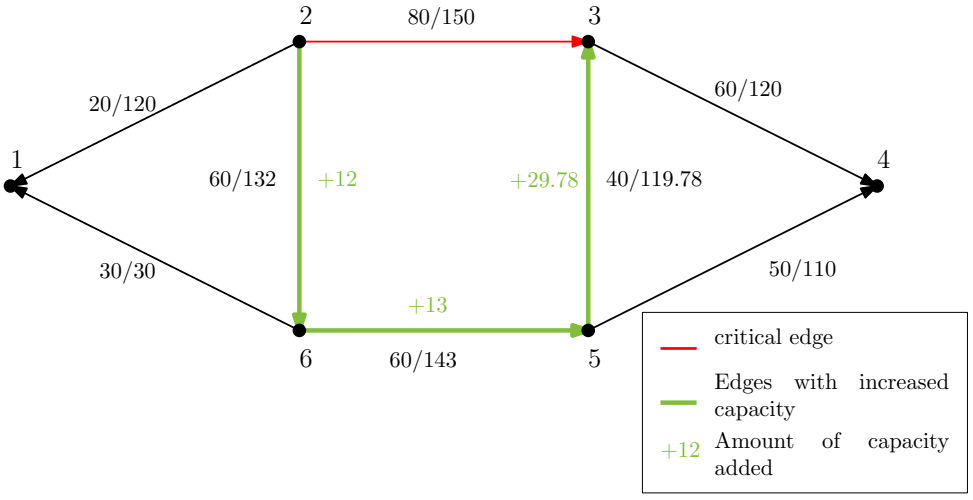
The article [RWTMO17] states that its expansion strategies do not lead to conclusions about the real world grid, as it uses a very simplified model. But the authors also did not test whether the proposed expansions would lead to a feasible network flow or would lead to a network destabilization or capacity reduction.



(a) The curing strategy A duplicates the critical edge with all its properties.



(b) The curing strategy B duplicates the critical edge but adjusts the capacity of the duplicate so that the edge is just not critical anymore by steps of $0.1K$ with $K = 100$ for this example.



(c) The curing strategy C adds capacity to the current bottleneck of the shortest path from 2 to 3 in $G' = G \setminus \{(2,3)\}$ until the critical edge is cured. The green numbers are the added capacities to the edges. They are already added to the capacities written on the lines.

Figure 3.3.: Results of applying the curing strategies A, B and C to the example graph. The critical edge $2, 3$ is marked red.

3.3. Summary

Transmission network expansion planning is a well researched topic. We could not find any formulations of the transmission network expansion planning problem with the objective to cure critical edges, except for the article by Rohden et al. [RWTMO17]. In this thesis a more sophisticated strategy to cure critical edges is developed which is based on nearly the same definition of critical edges as the strategies proposed by Rohden et al. We formulate the problem Transmission Network Expansion Planning Problem for Curing Critical Edges (TNEP-CCE). Like the problem discussed by Rohden et al. [RWTMO17] formulation is based on a graph theoretical approach and does not require any constraints based on power flow. We further research the properties of this problem and show the complexity of a version of TNEP-CCE restriced to only one critical edge. To compare our solving strategies to a model including power flow constraints, we reformulate the problem as Mixed-Integer Linear Program using the DC power flow approximation as network model. We develop different heuristics to solve the problem TNEP-CCE and compare their performances and solutions to the MILP formulation using the DC power flow approximation. Furthermore we test the feasibility of the solutions proposed by our optimization methods by simulating the resulting networks using the DC power flow approximation.

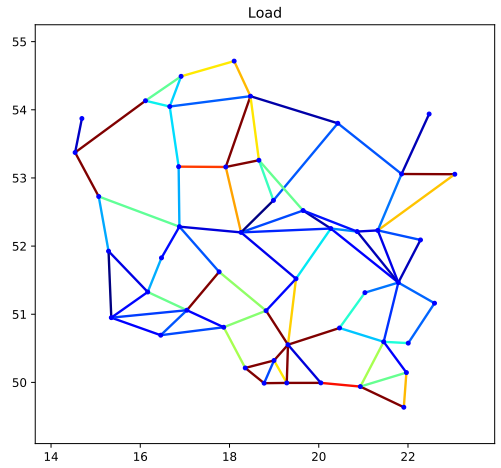
4. Data Preprocessing

This chapter describes the preprocessing done with the test data sets before it can be used in the evaluation for this thesis. First the data sets are described in Section 4.1. Then we describe the libraries used to handle the data in Section 4.2. In the Subsection 4.2.1 we will especially look into the PyPSA library. We use the PyPSA library in this thesis for handling the test data sets and computing the physical properties of our candidate edges, like for example the reactance. Because our test data set do not include candidate edges, we developed methods to generate candidate edge sets for the test data set which are suited for our application. We describe these methods in Section 4.3.

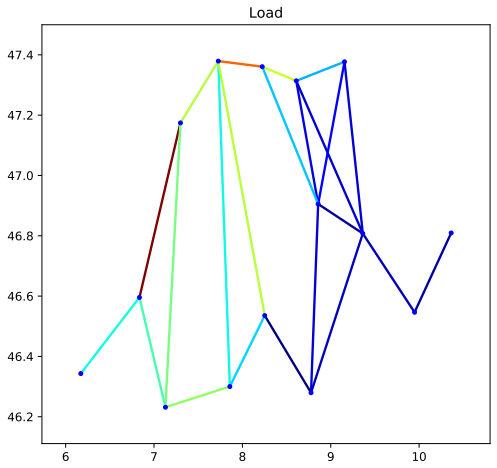
4.1. Test Data

To evaluate the resulting algorithm of this thesis we will use subsets of the European transmission network. The data set provides data for 17 transmission networks from different countries. In the following table the networks available in the data set are listed with their node count as well as the number of edges in the network:

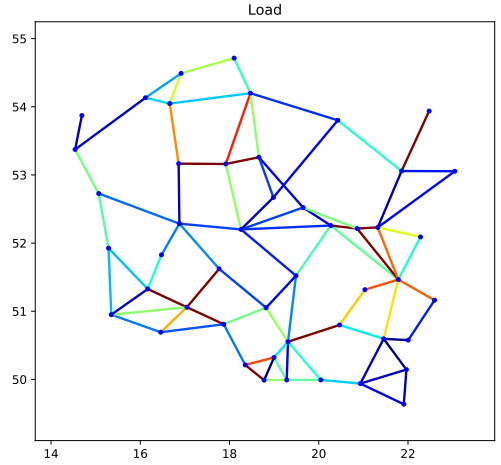
country	number of nodes	number of edges
Austria	23	29
Belgium	28	32
Bulgaria	12	17
Croatia	6	6
Czech Republic	21	35
Denmark	11	11
Hungary	13	21
Ireland	8	12
Netherlands	33	40
Norway	42	65
Poland	48	84
Portugal	16	22
Romania	17	27
Slovakia	9	13
Slovenia	4	4
Sweden	46	72
Switzerland	15	23



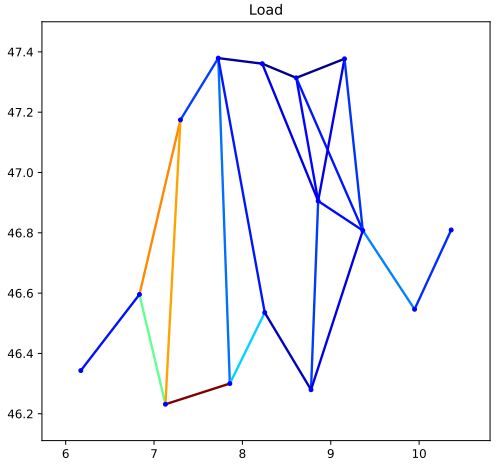
(a) Load on the Polish transmission network on the 1st February 2013 at 16:00.



(b) Load on the Swiss transmission network on the 1st February 2013 at 16:00.



(c) Load on the Polish transmission network on the 25th June 2013 at 02:00.



(d) Load on the Swiss transmission network on the 25th June 2013 at 02:00.

Figure 4.1.: Load on the Polish and Swiss transmission networks. The unit of the x-axis is given in degrees longitude and the y-axis in degrees latitude in geocoordinates. Blue symbolizes low loads, while red symbolizes high loads on an edge.

For each of these networks the data set provides extensive data, all data entries provided by this model are described in detail in the PyPSA documentation. Most important for this thesis is the data for the edges and nodes in the network. For the edges of the data set we will mainly use the data entries for bus0 and bus1, meaning the nodes the edge is connecting, s_nom which describes the capacity of the edge in MVA and the type describing the line type from which the line standard type impedance parameters are used to compute x the series reactance and r the resistance of the edge. For the nodes we use their geocoordinates to compute the length of the candidate edges as well as the data of generators and loads associated with each node. The loads are in the format of a timeline containing data for the year 2013 in one hour snapshots. Additional load or generation is provided by timeline data of the storage units in the transmission network, which can either dispatch or consume power. The generators are divided into different types of generators, influencing power cost the ability to expand the power output of the generator and its efficiency. This data is not primarily important for this thesis, but it is used to compute a linear optimal power flow on the networks for a chosen snapshot.

In Figure 4.1a and Figure 4.1c the result of a power flow analysis using the DC approximation of the Polish transmission network can be seen on two different days and times. In Figure 4.1a the power flow from the 1st February 2013 at 16:00 can be seen and in Figure 4.1c the power flow on the 25th June 2013 at 02:00:00 can be seen. In Figure 4.1b and 4.1d the power flow on the Swiss transmission network can be seen for the same snapshots. In the figures high loads are marked with red colors while low loads on the edges are marked in blue colors. The unit of the x-axis is given in longitude and the y-axis in latitude in geocoordinates. In can be seen that the data provides different network topologies and various power demands for each network. In total the data includes 8760 different snapshots for each network.

Figures for the other transmission networks in the test data set displaying the power flow from the 1st February 2013 at 16:00 can be found in the appendix in Section A.

4.2. Libraries

For the implementation of the heuristic proposed in Section 5.5.3.1 we use python 3.7. We use python as it makes data handling very comfortable, especially since the test data sets introduced in Section 4.1 are formatted to be used with the python library PyPSA.

4.2.1. PyPSA

PyPSA [BHS18] stands for 'Python for Power System Analysis' and is an open source python library. It provides a lot of features for simulating and optimizing modern power systems. We will use PyPSA to load the data sets introduced in Section 4.1 from files and perform an linear optimal power flow analysis on the data sets to get an initial flow on the network.

The linear optimal power flow (lopf) function of the PyPSA library optimizes the dispatch of generation and storage to meet a given load for every snapshot examined. As objective function for the lopf the total costs for the power system are minimized. Where the total costs consist of the capital cost for transmission assets — transmission lines, transformer,... — generation and storage facilities as well as marginal cost for generation and storage facilities. As additionally it is possible to let the optimizer expand the capacities for transmission, generation and storage. In the lopf the AC power flow is approximated by using the DC approximation as introduced in Section 2.2. More details on the used constrains in the lopf can be found in the PyPSA documentation.

We are interested to expand transmission network based on static analysis, therefore we will handle one snapshot at a time. Since we want to expand the transmission capacity of the network ourselves we don't use this feature for the optimization. Also we don't use the possibility to expand storage capacity. But since it is preferable for our problem to work with highly loaded transmission networks we allow expansion of the generation capacity. By allowing this expansion of generation capacity we get higher loads as the capacity of generators with low marginal costs is expanded to load the energy storage units. We use the results from the lopf as input for our problem. Mainly we are interested in the power flow on each edge of the network, as well as the resulting active power at each node. The active power at a node is the aggregated active power from all consumer and generators directly connected to this node. Since this detailed view on each node is not necessary for our problem we will still refer to a node as consumer if the active power is negative or as generator if the active power is positive.

4.2.2. Graphtools

The python library graph-tool [dPP27] is an open source library, which provides functionality for efficient network analysis. The library itself is mostly implemented in C++ which allows performance that is comparable to that of a pure C/C++ library. This is beneficial for the computation times of our algorithms and therefore we use this library for all graph based computations.

The main feature of this library we use is the computation of maximum s-t-flows. Generally the library implements three algorithms to compute the maximum s-t-flow in a graph. The three algorithms are the Edmond-Karp-Algorithm, the Push-Relabel-Algorithm and the Boykov-Kolmogorov-Algorithm. We use the Boykov-Kolmogorov-Algorithm in this thesis to achieve better computation times. But generally we could use any of these algorithms.

For more information on the graph-tool library consult the documentation.

4.2.3. Gurobi

Gurobi [GO18] is a state-of-the-art solver for mathematical programming.

We use Gurobi to solve the linear optimal power flow internally in the PyPSA library 4.2.1, as well as the mathematical models for our problem described in Section 5.3.

For the evaluation in Section 6 we will compare the computation time as well as the results of the optimizations of our mathematical models done by Gurobi to our own algorithm.

4.3. Candidate Edges

The algorithms presented in this thesis has a set of candidate edges E_{cand} as input. Each of these candidate edges has a cost, which symbolizes the cost to build this line, and a capacity. The test data sets introduced in Section 4.1 do not provide suitable candidate edges. The only similar functionality they provide are expansion costs for edges in E, but since we want to limit the number of candidate edges and are especially interested in building new edges, we restrict the choice of candidate edges by excluding duplicates of already existing edges from E_{cand}, meaning E_{cand} and E are disjoint.

Since the overall performance of our algorithms also depends on the candidate edges they have as input, we describe the methods used for creating a set of candidate edges E_{cand} for a given transmission network G in this section.

There are a lot of different types of transmission lines. The type of line is important as is influences the physical properties, like the resistance, reactance and capacitance, of the candidate edges. These physical properties are important for calculating the power flow on the expanded network. To provide a feasible line type we will choose the line type for the candidate edges which is most commonly used in the transmission network we examine. In the test data sets we used, which where introduced in Section 4.1, all edges are of the line type 'Al/St 240/40 4-bundle 380.0' defined in [OO11]. Therefore we will also use this line type for the sets of candidate edges generated for the test data sets.

4.3.1. Simple Set of Candidate Edges

The first method we used to create a set of candidate edges E_{cand} for a given transmission network is the least sophisticated one we used. The method uses as input a threshold $d_{\max}$, a set capacity cap_{cand} and the transmission network represented by the graph $G = (V, E)$. Since we use the threshold $d_{\max}$ to limit the length of the candidate edges, we refer to the length of an edge $e \in (V \times V)$ as $\text{len}(e)$. The output is a set of candidate edges

$E_{\text{cand}} = \{e \mid \text{len}(e) \leq d_{\max} \cap e \in (V \times V) \setminus E\}$ with capacity cap_{cand}. Meaning E_{cand} consist of all edges of the fully connected graph with nodes V, which are not in E and are equal in length or shorter than $d_{\max}$.

The position of the nodes in the test data sets introduced in Section 4.1 are given in geocoordinates. Because we want to calculate the edge length along the surface of the earth we use the formula for the geographical distance to calculate the length of the edges. We assume we want to measure the edge length between to nodes $u = (\text{lon}_1, \text{lat}_1)$ and $v = (\text{lon}_2, \text{lat}_2)$, $u, v \in V$. The earth radius is given by $R = 6371km$. The geographical distance $d(u, v)$ between the two nodes u and v is calculated by:

$$
\begin{aligned}
d(u,v) &= R \cdot 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}) \\
a &= \sin(\frac{\delta_{\text{lat}}}{2})^2 + \cos(\text{lat}_1)\cos(\text{lat}_2)\sin(\frac{\delta_{\text{lon}}}{2})^2 \\
\delta_{\text{lat}} &= \text{lat}_2 - \text{lat}_1 \\
\delta_{\text{lon}} &= \text{lon}_2 - \text{lon}_1
\end{aligned}
$$

We set the costs for each candidate edge $e \in E_{\text{cand}}$ to be $\text{cost}(a, b) = \frac{len(e)}{10}$.

As our goal is to minimize the extend of the critical edges in the network, we want our candidate edges to be as equally distributed over the network as possible. It is especially important that nodes connected by only one edge in the original graph are also connected by a candidate edge, since the edges connecting such nodes are critical for sure, as disconnecting them leads to the network dividing into two separated parts.

The proposed method works well for transmission networks with nearly evenly spaced nodes. Meaning the candidate edges are quite evenly spaced for a suited choice of $d_{\max}$. For example in Figure 4.2 the Swiss transmission network can be seen with the generated set of candidatelines with $d_{\max} = 100$ and $\text{cap}_{\text{cand}} = 1000$.

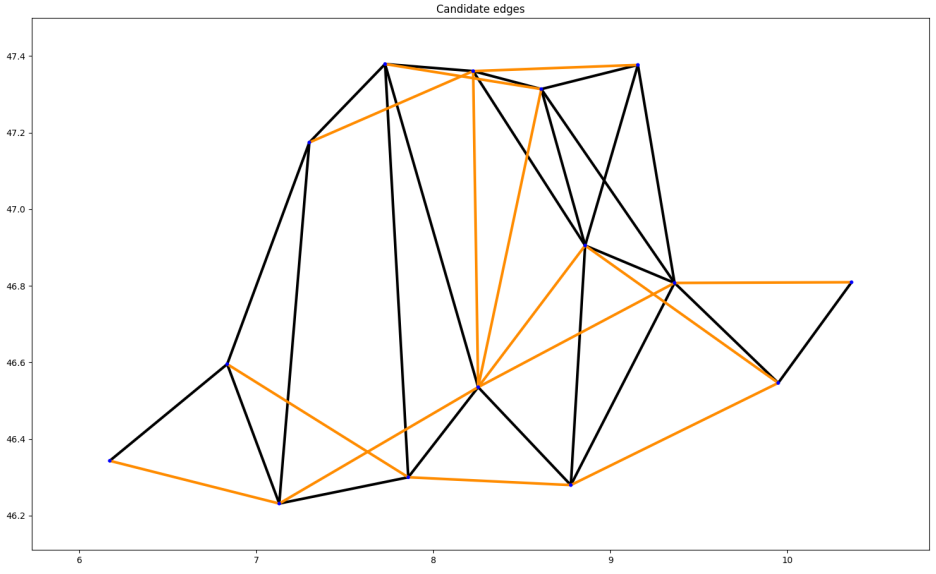


Figure 4.2.: Swiss transmission network with the original edges in black and a generated set of candidate edges with $d_{\max} = 100$ and $\text{cap}_{\text{cand}} = 1000$ in orange. The set of candidate edges was generated using the method of a simple set of candidate edges. The unit of the x-axis is given in degrees longitude and the y-axis in degrees latitude in geocoordinates.

If a transmission network is very heterogeneous considering the space between nodes this method does not produce a very good set of candidate edges. This can be seen for example in Figure 4.3. The figure shows the generated set of candidate edges with $d_{\max} = 100$ and $\text{cap}_{\text{cand}} = 1000$ for the Norwegian transmission network. This set of candidate edges is not suited for our application. Because of the topology of the network all edges in the path leading to the node in the right top corner are critical because they split the network into two parts by disconnecting them. But the candidate edges do not connect this node in the right top corner to any other node and therefore those critical edges can't be cured.

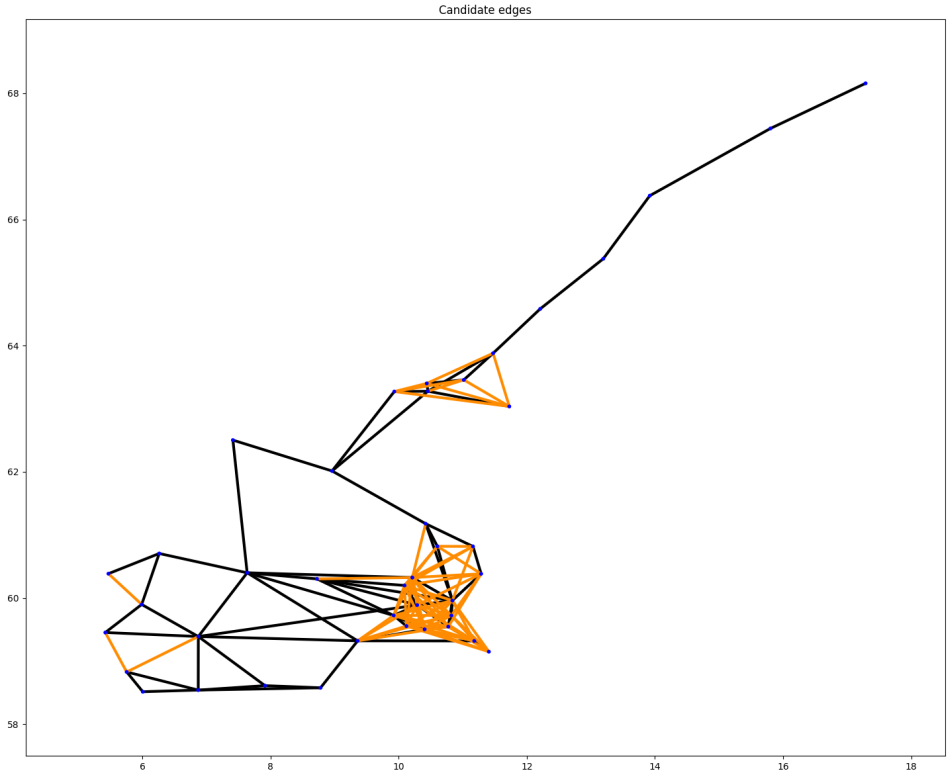


Figure 4.3.: Norwegian transmission network in black with a generated set of candidate edges with $d_{\max} = 100$ and $\text{cap}_{\text{cand}} = 1000$ in orange. The set of candidate edges was generated using the method of a simple set of candidate edges. The unit of the x-axis is given in degrees longitude and the y-axis in degrees latitude in geocoordinates.

Another problem is that the values for $d_{\max}$ and cap_{cand} have to be manually tuned to generate the best possible result for each transmission network.

4.3.2. Generate Fixed Number of Candidate Edges per Node

The first method we use to generate a set of candidate edges, has the problem, that for heterogeneous networks considering the spacing between the nodes, it does not produce good results as discussed in Section 4.3.1. To improve our set of candidate edges we implemented this second method to generate a set of candidate edges for a given network. This method uses as input a parameter k and the transmission network represented by the graph $G = (V, E)$. It will generate at least k candidate edges for each node $v \in V$. To create these candidate edges we use the distance function introduced in Section 4.3.1 to calculate the distance between two nodes. For a given node v we calculate the distances $len(v, u)$ for all nodes in V if $(v, u) \notin E$. Then we add candidate edges for the k couples of

nodes with the smallest distances calculated. In other words we add edges from v to its k nearest neighbors, that are not already connected to v. We add each candidate edge (v, u) just once but count it for the k candidate edges for the node u as well as for the node v.

The capacity of the candidate edges is set to be the average capacity of the original edges in the network. We set the costs for each candidate edge $e \in E_{\text{cand}}$ to be $\text{cost}(e) = \frac{len(e)}{10}$.

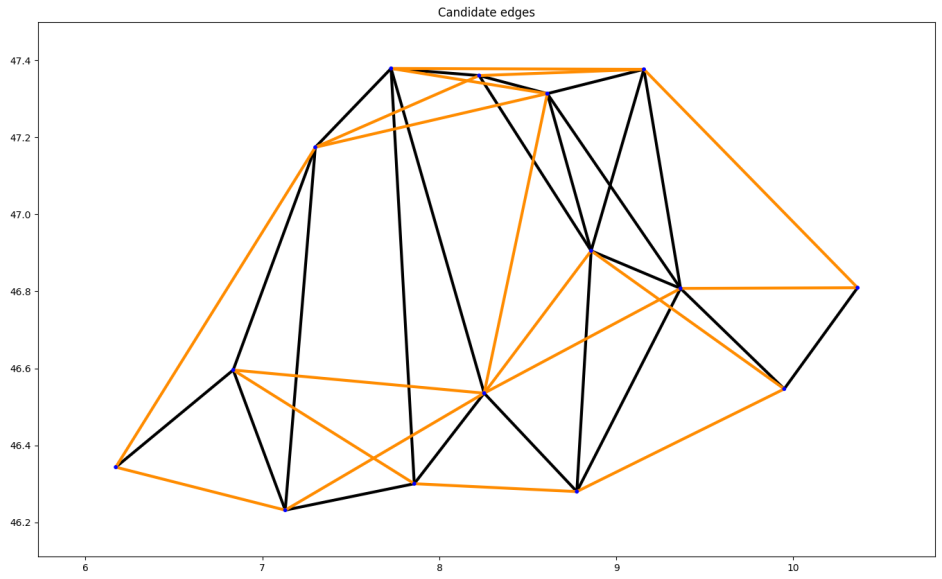


Figure 4.4.: Swiss transmission network with the original edges in black and a generated set of candidate edges with $k = 2$ in orange. The set of candidate edges was generated using the method generating a fixed number of candidate edges for each node. The unit of the x-axis is given in degrees longitude and the y-axis in degrees latitude in geocoordinates.

The result of this method for the Swiss transmission network with $k = 2$ can be seen in Figure 4.4. Comparing the results for the Swiss transmission network of both methods in Figure 4.4 for this method and in Figure 4.2 for the simple method from Section 4.3.1. We can see that there are some small differences of the sets of candidate edges but overall the sets are quite similar for the Swiss transmission network. The results of this method for the Norwegian transmission network with k=2 can be seen in Figure 4.5. Comparing the results of the previous method in Figure 4.3 and of this method in Figure 4.5, we see that the candidate edges generated by this method are more evenly distributed over the entire network than the candidate edges generated by the simple generation method. This indicated that this method is more suitable for generating candidates edges in cases of heterogeneous networks considering the spacing between the nodes.

This method generates sets of candidate edges, which are better or equally good compared to the sets of candidate edges provided by the method explained in section 4.3.1. By using the average capacity over all original network edges we avoid the need to set a suitable capacity for the candidate edges. But this method still needs the parameter k to be tuned manually.

Because of the benefits this method provides we will mainly use this method to create candidate networks for our evaluation. We will refer to this method as the knn-method for creating candidate edges.

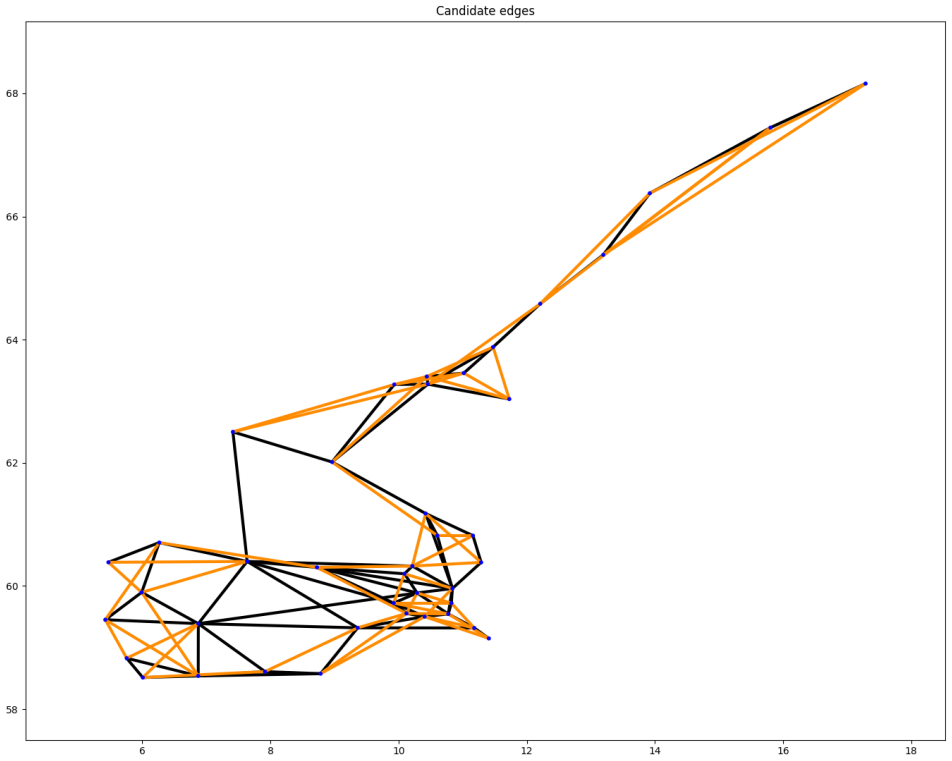


Figure 4.5.: Norwegian transmission network with the original edges in black and a generated set of candidate edges with $k = 2$ and $\text{cap}_{\text{cand}} = 1000$ in orange. The set of candidate edges was generated using the method generating a fixed number of candidate edges for each node. The unit of the x-axis is given in degrees longitude and the y-axis in degrees latitude in geocoordinates.

5. Modeling to Cure Critical Edges

From this point onwards we will only use the definition of Witthaut et al. [WRZ+16a, WRZ+16b] for critical edges with the modification we proposed at the end of Section 2.3. Meaning we identify an edge $(a, b) \in E$ with a threshold h as critical if:

$$h < \begin{cases} \frac{f(a,b)}{f^{\text{red}}(a,b)} & f(a,b) \geq 0 \\ \frac{f(a,b)}{f^{\text{red}}(b,a)} & f(a,b) < 0 \end{cases} \tag{5.1}$$

For our implementation as well as the evaluation we use a threshold of $h = 0.614$ since this value was proposed by Witthaut et al. [WRZ+16a, WRZ+16b].

Our goal is to expand the given transmission network G by adding edges. The redundant flow $f^{\text{red}}(a, b)$ of an critical edge depends on the graph the redundant flow is calculated on. Until this point the graph G on which the redundant flow was calculated did not change. For more clarity we will from now on specify the graph the redundant flow is calculated on. Therefore we will redefine the redundant flow as $f^{\text{red}} : G \times E \to \mathbb{R}$. So that $f^{\text{red}}(G, (a, b))$ is the redundant flow on graph G for the edge $(a, b) \in E$. For short we will write $f^{\text{red}}(G, (a, b))$ as $f^{\text{red}}(G, a, b)$.

Since we want to modify the graph $G = (V, E)$ by adding one or more edges E_{build} to it, we will write $G' = G \cup E_{\text{build}}$ to shorten the expression $G' = (V, E \cup E_{\text{build}})$. As we will also talk about removing edges from Graphs in some proofs we will analogously write $G' = G \setminus E_{\text{build}}$ instead of $G' = (V, E \setminus E_{\text{build}})$.

In this chapter we will introduce a criteria to determine how critical an edge is in Section 5.1. Furthermore we will define the problem, which is subject of this thesis, formally in Section 5.2. This criteria is important for the mathematical models to solve this problem, which are introduced in Section 5.3. In next Section 5.4 we will take a look at the complexity of the problem. In the last section, Section 5.5, of this chapter we will propose a heuristic algorithm after discussing the challenges of the algorithm design.

5.1. Criteria for Criticality

Until this point we identified an edge either as critical or not as critical by using the method we proposed in 2.3 which is strongly based on a method by Witthaut et al. [WRZ+16a, WRZ+16b]. This method identifies an edge $(a, b) \in E$ as critical if the initial

flow $f(a,b)$ on (a,b) divided by the redundant flow $f^{\text{red}}(G,a,b)$ or $f^{\text{red}}(G,b,a)$ the graph can support is larger than a threshold h by considering the direction of the initial power flow:

$$h < \begin{cases} \frac{f(a,b)}{f^{\text{red}}(a,b)} & f(a,b) \geq 0 \\ \frac{f(a,b)}{f^{\text{red}}(b,a)} & f(a,b) < 0 \end{cases} \tag{5.2}$$

Since the transmission network is undirected we will from now on assume without loss of generality that $f(a,b) \geq 0$.

For the following sections it is not only important that an edge is critical, but also how much flow needs to be added to $f^{\text{red}}(G,a,b)$, so that the critical edge (a,b) is not critical as defined in Equation 5.2. We will refer to adding enough redundant flow to $f^{\text{red}}(G,a,b)$ for a critical $(a,b) \in E$ so that the critical edge (a,b) is not critical anymore as curing the critical edge (a,b). To cure a critical edge in a given graph G you need to add at least a certain amount of additional redundant flow. We refer to this minimally required additional redundant flow as the criticality of a critical edge (a,b) or as $f^{\text{add}}(a,b)$. The criticality of (a,b) can be calculated by setting $\frac{f(a,b)}{(f^{\text{red}}(G,a,b)+f^{\text{add}}(G,a,b))} = h$. The redundant flow $f^{\text{red}}(G,a,b) + f^{\text{add}}(G,a,b)$, that solves this equation is the smallest redundant flow necessary in the graph G, for which (a,b) is not critical any longer. By transforming this equation you get $f^{\text{add}}(G,a,b) = \frac{f(a,b)}{h} - f^{\text{red}}(G,a,b)$ as minimum additional redundant flow to add, to cure the critical edge (a,b). As we want to cure the critical edge (a,b) by adding edges to the graph G and therefore modifying the graph, we will define f^{add} as:

$$\begin{aligned} f^{\text{add}}: \quad & G \times E \to \mathbb{R} \\ & f^{\text{add}}(G,(a,b)) = \frac{f(a,b)}{h} - f^{\text{red}}(G,a,b) \end{aligned}$$

Where $f^{\text{red}}(G,a,b)$ is the redundant flow of Graph G for edge (a,b). We will write $f^{\text{add}}(G,a,b)$ for short. We will also refer to $f^{\text{add}}(G,a,b)$ as criticality of the critical edge (a,b). Be aware that $f^{\text{add}}(G,a,b) > 0$ if edge (a,b) is critical in graph G and $f^{\text{add}}(G,a,b) \leq 0$ otherwise.

5.2. Problem Definition

As briefly described in the previous chapters, the goal of this thesis is the development of an algorithm to expand the transmission network with the objective to minimize the extend of the critical edges. Generally we see two approaches to this problem.

The first approach is to optimize the transmission network considering the physical properties of the power flow. This approach considers the fact, that the power flow on the transmission network will change after adding even one edge.

The second approach is to simplify the transmission network as graph with a given initial flow, as it is already done to identify the critical edges. We remain on this simplification level and formulate the problem as graph theoretical problem. We have seen this approach already in the article by Rohden et al. [RWTMO17]. We already discussed in Section 2.3 that the graph theoretical flow used for the 'redundant flow predictor' considers flow conservation, which is in case of the DC approximation equal to Kirchhoff's' current law. Therefore the second approach will at least full fill one of the two constraints of a feasible power flow as modeled in the DC approximation. This second approach has the advantage of being less complex but but also less accurate. Because of the lower complexity we discuss mainly the second approach of the problem in this thesis. The formal definition of the problem for the second approach is given in Definition 5.1.

Definition 5.1 (Transmission Network Expansion Planning for Curing Critical Edges (TNEP-CCE))**.**
Instance: *A graph $G = (V, E)$, with a set of nodes V, a set of edges E and capacities* $\text{cap}(u, v)$ *for each edge $(u, v) \in E$. For each edge $(u, v) \in E$ the initial flow on this edge is given by $f(u, v)$. Additionally, a set of candidate edges E_{cand} for graph G is given with expected building costs* $\text{cost}(x, y)$ *for each candidate edge $(x, y) \in E_{\text{cand}}$. And a set of critical edges E_{crit} as well as an expansion planning budget $B \in \mathbb{R}_{\geq 0}$ is given.*
Objective: *Finding a subset $E_{build} \subset E_{\text{cand}}$ of the candidate edges expanding G to $G' = (V, E \cup E_{build})$ in such way, that:*

- *the criticality of critical edges $\sum_{(a,b)\in E_{crit}} f^{\text{add}}(G', a, b)$ in G' is minimal*
- *the cost of the edges in E_{build} is at most equal to the budget: $\sum_{(x,y)\in E_{build}} \text{cost}(x, y) \leq B$*

The decision problem for TNEP-CCE is the question whether all critical edges can be cured $\sum_{(a,b)\in E_{\text{crit}}} f^{\text{add}}(G', a, b) = 0$.

We use a given set of critical edges E_{crit} for this problem definition because we do not recalculate the power flow for this graph theoretical approach. If the power flow is not recalculated, there will be no additional critical edges. This is because by not recalculating the power flow the only way an edge (a, b) can become critical is by reducing the maximum a,b-flow. But since we are only adding edges to the graph the maximum a,b-flow can only increase if it changes. We also know, that there is no initial power flow on newly added edges and hence they cannot be critical. Therefore we can use a given list of critical edges for this problem without losing accuracy.

5.3. Mathematical Models

In this section we will discuss mathematical models for solving the transmission network expansion problem in this thesis. In Subsection 5.3.1 we introduce a mathematical model based on the DC Approximation of the transmission network. This model does not simplify the transmission network as drastically as proposed in the Section 5.2 instead uses the DC approximation to apply some physical properties of the transmission network in the optimization process. We will use this model in the evaluation 6 to compare to our other solving strategies, which work with the problem definition TNEP-CCE. In Subsection 5.3.2 we introduce a mathematical model for solving TNEP-CCE, meaning without concerning the physical properties of the transmission network but instead using the network flow to approximate the physically correct behavior. This mathematical model has the same level of abstraction as used in the heuristic proposed in Section 5.5.3.1 of this thesis.

5.3.1. Mathematical Model with DC Approximation

In this section we formulate the mathematical model for the problem to expand transmission networks to cure critical edges including the physical properties of the transmission network.

As input for this problem we have the original transmission network $G = (V, E)$. For this network we have for each edge $(a, b) \in E$ a given capacity $\text{cap}(a, b)$ and susceptance $b(a, b)$. For each node $v \in V$ we are given the power consumption or generation $P(v)$. Additionally to the network we are given a list of critical edges $E_{\text{crit}} \subset E$, which are identified by the method described in Section 2.3. Also part of the input are a set of candidate edges E_{cand}. Each candidate edge $(a, b) \in E_{\text{cand}}$ has a capacity $\text{cap}(a, b)$, a susceptance $b(a, b)$ and building costs $\text{cost}(a, b)$. The last part of the input is the expansion planning budget B.

In the formulation of the mathematical model we use $G' = (V, E \cup E_{\text{cand}})$.

The problem can be formulated as Mixed-Integer Linear Program with the following constraints:

$$s : E_{\text{cand}} \to \{0, 1\} \tag{5.3}$$

In Equation 5.3 the function s models whether a candidate edge $(a, b) \in E_{\text{cand}}$ is build, $s(a, b) = 1$, or not, $s(a, b) = 0$.

$$\sum_{(a,b)\in E_{\text{cand}}} \text{cost}(a, b)s(a, b) \leq B \tag{5.4}$$

Equation 5.4 is the constraint that the cost of the added edges must not exceed the given expansion budget. The constant B stands for the total expansion planning budget. The cost of a candidate edge (a, b) is denoted by $\text{cost}(a, b)$. The function $s(a, b)$ indicates if the candidate edge (a, b) is build or not.

$$P(v) - \sum_{(x,v)\in E\cup E_{\text{build}}} f(x, v) + \sum_{(v,x)\in E\cup E_{\text{build}}} f(v, x) = 0, \quad \forall v \in V \tag{5.5}$$

The second constraint, Equation 5.5 implements Kirchhoff's current law. The meaning of this equation is, that all current entering a node of the transmission network must also leave that node. The constant $P(v)$ stands for the generation ($P(v) > 0$) or consumption ($P(v) \leq 0$) of the node. The directed power flow from node a to node b is denoted by $f(a, b)$.

$$f(a, b) = b(a, b)(\Theta(a) - \Theta(b)), \quad \forall(a, b) \in E \tag{5.6}$$

$$f(a, b) = s(a, b)b(a, b)(\Theta(a) - \Theta(b)), \quad \forall(a, b) \in E_{\text{cand}} \tag{5.7}$$

Equations 5.6 and 5.7 model how the power flow $f(a, b)$ on an edge (a, b) is calculated according to the DC model. The constant $b(a, b)$ refers to the susceptance of the line $(a, b) \in E$. The variable $\Theta(a)$ denote the voltage angle for each node $a \in V$. When formulating the Constraint 5.7 like described above we obtain a quadratic constraint. Thus this problem formulation becomes a Mixed Integer Quadratic Program which is harder to solve than a Mixed Integer Linear Program. To avoid unnecessarily long computation times we therefore reformulate the Constraint 5.7 as indicator constraints in Equations 5.8 and 5.9.

$$s(a, b) = 1 \implies f(a, b) = b(a, b)(\Theta(a) - \Theta(b)), \quad \forall(a, b) \in E_{\text{cand}} \tag{5.8}$$

$$s(a, b) = 0 \implies f(a, b) = 0, \quad \forall(a, b) \in E_{\text{cand}} \tag{5.9}$$

$$f(a, b) \geq 0 \implies f^{\text{add}}(G', a, b) \geq \frac{f(a, b)}{h} - f^{\text{red}}(G', a, b), \quad (a, b) \in E_{\text{crit}} \tag{5.10}$$

$$f(a, b) \leq 0 \implies f^{\text{add}}(G', a, b) \geq \frac{f(a, b)}{h} - f^{\text{red}}(G', b, a), \quad (a, b) \in E_{\text{crit}} \tag{5.11}$$

$$f^{\text{add}}(G', a, b) \geq 0, \quad (a, b) \in E_{\text{crit}} \tag{5.12}$$

The criticality of each critical edge $f^{\text{add}}(G', a, b)$, which is equivalent to amount of additional redundant flow needed cure the critical edge in graph G', is modeled in Equation 5.10 and 5.11 for each case of power flow direction on the edge. The Equation 5.12 models, that we avoid benefits from adding redundant flow for an already cured critical edge meaning getting a value $f^{\text{add}}(G', a, b) < 0$.

$$f^{\text{red}}(G', a, b) = \sum_{(a,x)\in G} \text{flow}_{(a,b)}(a, x) - \sum_{(x,a)\in G} \text{flow}_{(a,b)}(x, a), \quad \forall(a, b) \in E_{\text{crit}} \tag{5.13}$$

The redundant flow $f^{\text{red}}(G', a, b)$ of an edge (a, b) in the graph G' is defined in Equation 5.13. The variable $\text{flow}_{(a,b)}(u, v)$ is the graph theoretical flow on edge (u, v) resulting from a maximum a,b-flow calculation. We previously defined G' as undirected graph therefore we have for each pair of edges in the directed equivalent of G (u, v) and (v, u) one edge either (u, v) or (v, u) in G. We decided to model the flow so that $\text{flow}_{(a,b)}(u, v) > 0$ if the flow is directed from u to v and $\text{flow}_{(a,b)}(u, v) < 0$ otherwise.

The graph theoretical flow $\text{flow}_{(a,b)}(x, y)$ on each edge $(x, y) \in E \cup E_{\text{cand}}$ is modeled by the Constraints 5.14 and 5.15 for the capacity limits and the Constraint 5.16 describing that for each node not being the source or sink the ingoing flow has to be equal to the outgoing flow. In the Equations 5.14 and 5.15 the power flow on an edge is taken into account to model the capacity limits on each edge. In Equation 5.17 is modeled, that the graph theoretical flow of an candidate edge is zero if it is not built $s(x, y) = 0$. In Equation 5.18 we model, that the graph theoretical flow on an critical edge is zero when examining the maximum flow on the edge which is equivalent to deleting the edge from the graph.

$$-\text{cap}(x, y) \leq \text{flow}_{(a,b)}(x, y) + f(x, y), \quad \forall (x, y) \in E \cup E_{\text{cand}}, (a, b) \in E_{\text{crit}} \tag{5.14}$$

$$\text{flow}_{(a,b)}(x, y) + f(x, y) \leq \text{cap}(x, y), \quad \forall (x, y) \in E \cup E_{\text{cand}}, (a, b) \in E_{\text{crit}} \tag{5.15}$$

$$\sum_{(u,x)\in G'} \text{flow}_{(a,b)}(u, x) - \sum_{(x,u)\in G'} \text{flow}_{(a,b)}(x, u) = 0, \quad \forall u \in V \setminus \{a, b\}, (a, b) \in E_{\text{crit}} \tag{5.16}$$

$$s(x, y) = 0 \Longrightarrow \text{flow}_{(a,b)}(x, y) = 0, \quad \forall (x, y) \in E_{\text{cand}}, (a, b) \in E_{\text{crit}} \tag{5.17}$$

$$\text{flow}_{(a,b)}(a, b) = 0, \quad \forall (a, b) \in E_{\text{crit}} \tag{5.18}$$

For the problem as discussed in Section 5.2 we will use the following objective:

$$\text{minimize} \qquad \sum_{(a,b)\in E_{\text{crit}}} f^{\text{add}}(G', a, b) \tag{5.19}$$

The objective in Equation 5.19 is to minimize the sum of criticalities of the critical edges. The graph G' is equal to $G'' = (V, E \cup E_{\text{build}})$, concerning the criticality of the critical edges. The graph G'' describes the network resulting from building a subset $E_{\text{build}} \subset E_{\text{cand}}$ of the candidate edges. The set E_{build} consists of those candidate edges (a, b) for which the binary variable $s(a, b) = 1$. The criticality of the critical edge (a, b) is denoted by $f^{\text{add}}(G', a, b)$, and is equivalent to the amount of additional redundant flow needed to cure the critical edge in graph G'. Both graphs are equal concerning the criticality, because the model defines that all edges in $E_{\text{cand}} \setminus E_{\text{build}}$ have no graph theoretical flow in equation 5.17.

In the following sections we refer to this mathematical model as MMDC_SET as abbreviation for the name 'Mathematical Model with DC Approximation and a Given Set of Critical Edges'.

Since we are interested in expanding the transmission network in a way to minimize the effect of the critical edges, we will also evaluate the effects of changing the objective to:

$$\text{minimize} \qquad |E'_{\text{crit}}|; \quad E'_{\text{crit}} = \{e \mid e \in E_{\text{crit}}, f^{\text{add}}(G', e) > 0\} \tag{5.20}$$

The Objective 5.20 minimizes the number of critical edges in the expanded graph. In the following sections we refer to the model using Objective 5.20 as MMDC_NUM. We use MMDC_NUM as abbreviation for the name 'Mathematical Model with DC Approximation with the Objective to Minimize the Number of Critical Edges'.

The two Objectives 5.19 and 5.20 seem to be quite similar on first sight, but in some cases can lead to very different results. For example, if $\sum_{(a,b)\in E_{\text{crit}}} f^{\text{add}}(G', a, b)$ is very small but not zero for some assignment of $s(u, v)$ for all $(u, v) \in E_{\text{cand}}$ this is a good result for Objective 5.19. But it can be that for the same assignment of $s(a, b)$ no critical edges are cured but instead have some small $f^{\text{add}}(G', a, b)$ for all critical edges $(a, b) \in E_{\text{crit}}$ and therefore $|E'_{\text{crit}}| = |E_{\text{crit}}|$. This means that measured for the objective value for the Objective 5.20, we have made no improvements by choosing those candidate edges.

Considering this, we want to evaluate the results for both different models MMDC_SET and MMDC_NUM, resulting from the Objectives 5.19 and 5.20, in Section 6.

In the beginning of the subsection we remarked that we use a set of critical edges as input for the models MMDC_SET and MMDC_NUM. We do this because for the mathematical model with network flow introduced in the next Subsection 5.3.2 as well as for the algorithms proposed in this thesis other edges will not become critical during the optimization process. The reason for this is, that we do not consider power flow changes in those models. Because we consider these power flow changes in this model the critical edges cannot only be cured during the addition of candidate edges but because of shifts in the power flow it is also possible, that new critical edges occur during the optimization.

To consider this behavior we modify the model MMDC_SET. We do this by choosing the set of critical edges as $E_{\text{crit}} = E \cup E_{\text{cand}}$. We do not need to add any additional constraints for the candidate edges since, if an candidate edge is not build its power flow is zero and therefore it cannot be critical. Also all edges $(a, b) \in E$ which are not critical in G' have a negative $f^{\text{add}}(G', a, b)$ and therefore we set it in Equation 5.12 to zero to avoid benefits for adding additional redundant flow to not critical edges. We refer to this model, which recalculates the criticality for all edges, as MMDC_RECALC in the following sections of this thesis.

5.3.2. Mathematical Model with Graph Flow

In the previous section we modeled the problem to expand the network with the objective to cure critical edges as Mixed-Integer Linear Program using the DC approximation. As discussed in Section 5.2 we are interested in solving the problem TNEP-CCE without using power flow constraints. In this section we formulate a model of the problem TNEP-CCE as Mixed-Integer Linear Program using the graph approximation. The input for the MILP is the transmission network as undirected graph $G = (V, E)$. The list of edges E is given with capacity $\text{cap}(a, b)$ and initial flow $f(a, b)$ for each edge $(a, b) \in E$. Additionally we have as input a set of candidate edges $E_{\text{cand}}$with capacity $\text{cap}(a, b)$ and costs $\text{cost}(a, b)$ for each candidate edge $(a, b) \in E_{\text{cand}}$, the expansion planning budget B and a set of critical edges $E_{\text{crit}} \subset E$.

The objective of the MILP is to minimize the sum criticalities for all critical edge:

$$\text{minimize} \sum_{(a,b)\in E_{\text{crit}}} f^{\text{add}}(G', a, b) \tag{5.21}$$

In the Objective 5.21 the variable G' is the graph representing the network $G' = (V, E \cup E_{\text{cand}})$. The nodes of the graph G' are fixed as are the original edges E. The objective is influenced by the choices of candidate edges $E_{\text{cand}}$to build. For each candidate edges $(a, b) \in E_{\text{cand}}$ we define in Equation 5.22 a binary variable $s(a, b)$. These binary variables indicate for each candidate edge $(a, b) \in E_{\text{cand}}$ whether the edge is build , $s(a, b) = 1$, or not, $s(a, b) = 0$.

$$s : E_{\text{cand}} \to \{0, 1\} \tag{5.22}$$

The next equation models the criticality $f^{\text{add}}(G', a, b)$, which describes the additional redundant flow needed to cure the critical edge (a, b) in graph G'.

$$f^{\text{add}}(G', a, b) \geq \frac{f(a, b)}{h} - f^{\text{red}}(G', a, b), \quad \forall (a, b) \in E_{\text{crit}} \tag{5.23}$$

$$f^{\text{add}}(G', a, b) \geq 0, \quad \forall (a, b) \in E_{\text{crit}} \tag{5.24}$$

In the Equation 5.23 the constant $f(a, b)$ is the initial power flow on the critical edge (a, b), since we do not recalculate the power flow but have it as part of our input we assume $f(a, b) \geq 0$ for all edges $(a, b) \in E$ without loss of generality. The constant h is a threshold for determining critical edges. The variable $f^{\text{red}}(G', a, b)$ stands for the maximum a,b-flow in the Graph $G' \setminus \{(a, b)\}$. The Equation 5.24 is necessary to formulate that $f^{\text{add}}(G', a, b) = \max(0, \frac{f(a,b)}{h} - f^{\text{red}}(G', a, b))$. The maximum is used to avoid benefits for expanding the network for an already cured critical edge.

$$f^{\text{red}}(G', a, b) = \sum_{(a,x) \in G'} \text{flow}_{(a,b)}(a, x) - \sum_{(x,a) \in G'} \text{flow}_{(a,b)}(x, a), \quad \forall (a, b) \in E_{\text{crit}} \tag{5.25}$$

In Equation 5.25 the redundant flow of a critical edge $(a, b) \in E_{\text{crit}}$ is described. The redundant flow of an critical edge $(a, b) \in E_{\text{crit}}$ is equivalent to the maximum a,b-flow in the Graph $G' \setminus \{(a, b)\}$. The variable $\text{flow}_{(a,b)}(a, x)$ stands for the flow on edge (a, x) as calculated in the maximum a,b-flow calculation. We previously defined G' as undirected graph therefore we have for each pair of edges (u, v) and (v, u) in the directed equivalent of the graph one edge either (u, v) or (v, u) in G'. We decided to model the flow so that $\text{flow}_{(a,b)}(u, v) > 0$ if the flow is directed from u to v and $\text{flow}_{(a,b)}(u, v) < 0$ otherwise. The flow on each edge is modeled by the Constraint 5.26 for the capacity limits and the Constraint 5.27 describing, that for each node not being the source or sink the ingoing flow has to be equal to the outgoing flow. In the Equation 5.26 the original flow on an edge is taken into account. Additionally the flow on the candidate edges is modeled in Equation 5.28, expressing that the flow of an candidate edge (x, y) is zero if it is not built $s(x, y) = 0$.

$$-\text{cap}(x, y) \leq \text{flow}_{(a,b)}(x, y) + f(x, y) \leq \text{cap}(x, y), \quad \forall (x, y) \in E \tag{5.26}$$

$$\sum_{(u,x) \in G} \text{flow}_{(a,b)}(u, x) - \sum_{(x,u) \in G} \text{flow}_{(a,b)}(x, u) = 0, \quad \forall u \in V \setminus \{a, b\} \tag{5.27}$$

$$s(x, y) = 0 \Longrightarrow \text{flow}_{(a,b)}(x, y) = 0, \quad \forall (x, y) \in E_{\text{cand}}, (a, b) \in E_{\text{crit}} \tag{5.28}$$

Equation 5.29 models that the costs of all candidatelines built must not exceed the expansion planning budget.

$$\sum_{(x,y) \in E_{\text{cand}}} s(x, y)\text{cost}(x, y) \leq B \tag{5.29}$$

Solving this MILP will result in an optimal solution for the TNEP-CCE problem. We refer to this model in the following sections as MM_GRAPH as abbreviation for Mathematical Model with Graph Flow.

5.4. Problem Complexity

In this section it is shown that TNEP-CSCE (see Definition 5.2)is NP-complete. The problem is formulated for only one critical edge for sake of the proof of NP-completeness. For this section we also assume $f(x, y) \geq 0$ for all edges $(x, y) \in E$.

Definition 5.2 (Transmission Network Expansion Planning for Curing a Single Critical Edge (TNEP-CSCE))**.**
Instance: *A graph $G = (V, E)$, with a set of nodes V and a set of edges E and capacities* $\text{cap}(u, v)$*, for each edge $(u, v) \in E$. For each edge $(u, v) \in E$ the initial flow on this edge is given by $f(u, v)$. Additionally a set of candidate edges E_{cand} for graph G is given, with building costs* $\text{cost}(x, y)$ *and capacity* $\text{cap}(x, y)$*, for all candidate edges $(x, y) \in E_{\text{cand}}$ and one critical edge $E_{crit} = \{(a, b)\}$ as well as a expansion planning budget B.*
Objective: *Decide whether there is a subset $E_{build} \subset E_{\text{cand}}$ of the candidate edges so that:*

- *The critical edge (a, b) is cured.*
- *The cost of the edges in E_{build} is at most equal to the budget:* $\sum_{(x,y)\in E_{build}} \text{cost}(x, y) \leq B$.

5.4.1. Complexity on General Graphs

To prove the NP-completeness of the Problem 5.2 on general graphs, the 'Bin Packing Problem' in Definition 5.3 is introduced. Bin Packing was proven to be strongly NP-complete by Garey and Johnson [GJ78].

Definition 5.3 (Bin Packing Problem)**.**
Instance: *The instance of the bin packing problem is $I = (X, A, b, \ell)$ with X a set of l bins $X_1, \ldots, X_\ell$ of size b and A a set with elements $1 \leq a_i \leq b$.*
Objective: *Decide whether there is a partition of A into the bins $X_1, \ldots, X_\ell$, so that* $\sum_{a_j \in X_i} a_j \leq b,\ \forall i \in \{1, \ldots, \ell\}$.

Lemma 5.4. *The problem 'Transmission Network Expansion Planning for Curing a Single Critical Edge' from Definition 5.2 is strongly* NP*-complete.*

Proof: To show that the problem of TNEP-CSCE is strongly NP-complete, we will show that TNEP-CSCE is in NP, then we will show that there is a polynomial-time algorithm that transforms an instance of the Bin Packing Problem to an instance of TNEP-CSCE and at last we will show the correctness of the polynomial-time algorithm.

TNEP-CSCE is in NP*:* Given is an instance of the TNEP-CSCE problem as described in Definition 5.2 and a subset of candidate edges $E_{\text{build}} \subset E_{\text{cand}}$ as possible solution. The following steps check the validity of the solution:

1. Check if the total costs of selected candidate edges E_{build} complies with the budget B: $\sum_{(x,y)\in E_{\text{build}}} \text{cost}(x, y) \leq B$
2. Construct a graph G' by adding the selected candidate edges E_{build} to the graph G: $G' = G \cup E_{\text{build}}$
3. Check if critical edge (a, b) is critical in G' by using the algorithm REDUNDANTFLOW(G', a, b) defined in Definition 2.1 meaning: $f(a, b)/f^{\text{red}}(G', a, b) > h$

The first two steps are in $O(|E_{\text{build}}|)$ time and we know $|E_{\text{build}}| \leq |E_{\text{cand}}|$. The third step uses the algorithm REDUNDANTCAPACITY as proposed by Witthaut et al. [WRZ+16a, WRZ+16b]. In Section 2.3, where the algorithm is introduced, we argue that the running time for REDUNDANTFLOW(G, a, b) is in $O(|V||E|^2)$ time. Because of that we know

that REDUNDANTFLOW(G', a, b) runs in $O(|V||E + E_{\text{build}}|^2)$ time. Therefore you get an polynomial-time algorithm checking the correctness of a proposed solution for an instance of TNEP-CSCE by executing the three steps in the given order. This means that TNEP-CSCE is in NP.

Construct an instance of TNEP-CSCE from an instance of the Bin Packing Problem: An instance of the Bin Packing Problem is given by $I = (X, A, b, \ell)$ as described in Definition 5.3. To recollect, X describes the set of bins, A describes the set of elements to partition into the bins, b describes the bin size and ℓ the number of bins.

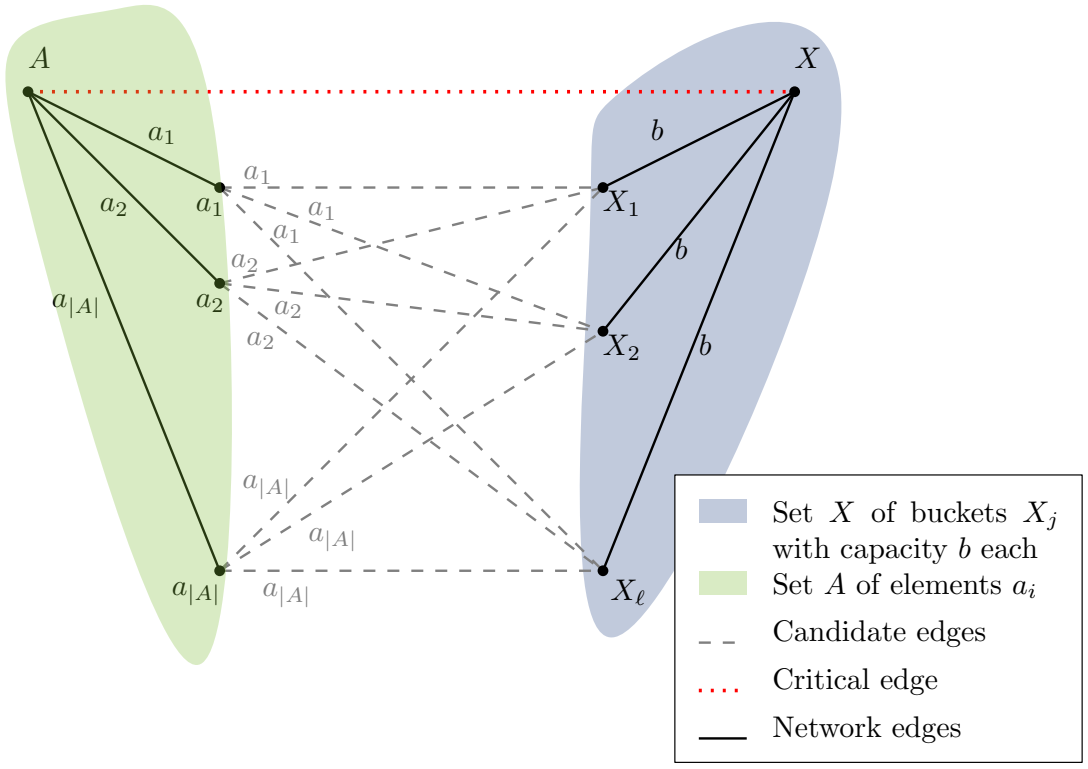


Figure 5.1.: Sketch of the graph created to reduce Bin Packing to TNEP-CSCE. Dashed lines represent candidate edges. Candidate edges $(a_i, X_j) \in E_{\text{cand}}$ have capacity $\text{cap}(a_i, X_j) = a_i$ and costs of $\text{cost}(a_i, X_j) = 1$. The red dotted line (A, X) represents the critical edge. The critical edge needs $f^{\text{add}}(G, A, X) = \sum_{a_i \in A} a_i$ units of additional redundant flow to be cured. The capacity of an edge is noted on the edge.

In the following paragraph we describe the construction of the resulting TNEP-CSCE instance, which can be seen in Figure 5.1. To construct the graph G from I we map each element in X and each element in A to a node. Additionally, we add a node for X and A respectively. The edges E of the graph connect each node representing an element $a_i \in A$ to the node representing A and all nodes representing a bin $X_i \in X$ to the node representing X. The initial flow for each edge $(u, v) \in E$ is $f(u, v) = 0$. The capacities for the edges are $\text{cap}(X, X_j) = b$ for all bins $X_j \in X$ and $\text{cap}(A, a_j) = a_j$ for all elements $a_j \in A$. As critical edge we add an edge (A, X), which we chose to be critical with $f^{\text{add}}(G, A, X) = \sum_{a_i \in A} a_i$ units of additional redundant flow missing to cure (A, X). The critical edge is marked as red dotted line in Figure 5.1 For the candidate edges E_{cand} we add for each element $a_i \in A$ edges to each element $X_j \in X$. The resulting candidate network represents a complete bipartite graph. In Figure 5.1 the candidate edges are depicted by the dashed lines. For the candidate edges $(a_i, X_j) \in E_{\text{cand}}$ we choose unit line costs $\text{cost}(a_i, X_j) = 1$ and capacity $\text{cap}(a_i, X_j) = a_i$. The expansion planning budget is $B = |A|$.

The construction of the graph $G = (V, E)$ is in $O(\ell+|A|)$ time as we have $|V| = |X|+|A|+2 = \ell + |A| + 2$ nodes as well as $|E| = |X| + |A| + 1 = \ell + |A| + 1$ edges to add, including the critical edge. The construction of the candidate edges is in $O(\ell|A|)$ time as we have $|E_{\text{cand}}| = \ell|A|$ candidate edges. Therefore the mapping algorithm from a Bin Packing instance to an TNEP-CSCE instance is a polynomial-time algorithm with a running time in $O(\ell|A|)$ time.

For the proof of strong NP-completeness it is necessary to show that all numerical parameters used by our TNEP-CSCE instance are bounded by a polynomial in the length of the input size, meaning the input of the Bin Packing instance. The constant numerical parameters we use for the cost and capacity of the candidate edges as well as for the initial flow obviously fulfill this requirement. A numerical parameter that is not constant is the expansion planning budget with $B = |A|$. Because A is part of the input, $|A|$ is bounded by the length of the input. Also not constant are the edge capacities $\text{cap}(A, a_j) = a_j$ for all $a_j \in A$ and $\text{cap}(X_j, X) = b$ for all $X_j \in X$. But because the numerical values of $a_j \in A$ and b are also used in the Bin Packing Problem, which is known to be strongly NP-complete, we know that they have to be bounded by a polynomial in the length of the input. The last parameter which is not constant is the criticality of the critical edge (A, X) which is $f^{\text{add}}(G, A, X) = \sum_{a_i \in A} a_i$. In the definition of the Bin Packing Problem each element of A is bounded by $1 \leq a_i \leq b$ for all $a_i \in A$, this means for the sum $\sum_{a_i \in A} a_i \leq b|A|$. We established that $|A|$ as well as b are bounded by a polynomial of the length of the input, therefore this is also true for their product. Therefore all used numerical parameters fulfill this requirement.

Proving equivalence: First we will show that if there is a solution of the constructed TNEP-CSCE instance, this implies the existence of a solution of the underlying Bin Packing instance. A solution of the TNEP-CSCE instance is a subset $E_{\text{build}} \subset E_{\text{cand}}$, for investment cost less or equal the expansion planning budget $\sum_{(a,x) \in E_{\text{build}}} \text{cost}(a, x) \leq B$. The expansion planning budget is $B = |A|$ and the costs for each candidate edge is $\text{cost}(a_i, X_j) = 1,\ (a_i, X_j) \in E_{\text{cand}}$ for each candidate edge. Therefore at most $|A|$ candidate edges can be build, $|E_{\text{build}}| \leq |A|$. An assignment for elements from A to bins in X can be constructed from a solution of the TNEP-CSCE instance by adding $a_i \in A$ to bin $X_j \in X$ if and only if $(a_i, X_j) \in E_{\text{build}}$. To show that this assignment for elements from A to bins in X is a valid solution for the Bin Packing Problem, we have to show that each element $a_i \in A$ is associated with exactly one bin $X_j \in X$ and that each bin $X_j \in X$ contains at most b units $\sum_{a_i \in X_j} a_i \leq b$.

First we will show that for each node representing an element a_i with $a_i \in A$ there is exactly one built edge $(a_i, X_j) \in E_{\text{build}}$. Assuming there is an element a_i with $a_i \in A$ so that there is no edge built, meaning $(a_i, X_j) \notin E_{\text{build}}$ for all $X_j \in X$. However in this case the critical edge (A, X) can not be cured. This is because just $|A| - 1$ elements of A would have a path to X. This means adding a_j units of additional redundant flow for each $a_j \in A$ which is connected to X, resulting in a maximum of $\sum_{a_j \in A} a_j - a_i$ units of additional redundant flow in total. This leads to the conclusion, that a valid solution for the TNEP-CSCE instance E_{build} contains at least one edge for each element $a_i \in A$, connecting the node representing the element to a bin. Because of the expansion planning planning budget $B = |A|$ and the cost for each candidate edge $\text{cost}(a_i, X_j) = 1$ for all $(a_i, X_j) \in E_{\text{cand}}$, we can only build $|A|$ edges. Therefore we know that a valid solution for the TNEP-CSCE instance E_{build} contains exactly one edge for each element $a_i \in A$, connecting the node representing the element to a bin.

The second part is showing, that each bin X_j contains at most b units $\sum_{a_i \in X_j} a_i \leq b$, for the assignment of the a_i as implied by E_{build}. We assume that there is a node X_j so that $\sum_{(a_i, X_j)} a_i > b$ with $(a_i, X_j) \in E_{\text{build}}$. The edge (X_j, X) has a capacity of $\text{cap}(X_j, X) = b$.

This means that all paths containing X_j together can at most add b units of additional redundant flow to cure the critical edge. Because $\sum_{(a_i,X_j)} a_i > b$ with $(a_i, X_j) \in E_{\text{build}}$, this means that at least one edge $(a_i, X_j) \in E_{\text{build}}$ will not be saturated. But because there is only one edge in E_{build} for each element $a_i \in A$, all edges in E_{build} have to be saturated to provide the $\sum_{a_i \in A} a_i$ units of additional redundant flow to cure the critical edge (A, X). Therefore there cannot be a node X_j so that $\sum_{(a_i,X_j)} a_i > b$ with$(a_i, X_j) \in E_{\text{build}}$. This means that if we have a solution for the TNEP-CSCE instance we also have a solution for the Bin Packing instance.

Next we need to show that if there is a solution for Bin Packing, we will also find a valid solution for the corresponding TNEP-CSCE instance. We know that in the solution each element a_i is contained in one bin X_j and each bin X_j contains at most b units $\sum_{a_i \in X_j} a_i \leq b$. If we have a Bin Packing solution we can create a TNEP-CSCE instance solution by choosing the edges $E_{\text{build}} = \{(a_i, X_j) \mid a_i \in X_j\}$. Since each element $a_i \in A$ is placed in exactly one bin $X_j \in X$, there is exactly one edge $(a_i, X_j) \in E_{\text{build}}$. We also know that $\sum_{a_i \in X_j} a_i \leq b$ for every bin X_j, therefore for a bin X_j the edges $(a_i, X_j) \in E_{\text{build}}$ will add $\sum_{a_i \in X_j} a_i$ units of flow to cure the critical edge. This means in total $\sum_{a_i \in A} a_i$ units of flow are added. This is sufficient additional redundant flow to cure the critical edge (A, X). Since there is one edge added for each $a_i \in A$ this means in total $|A|$ edges are added for costs of one unit each. This means that $\sum_{(a_i,X_j) \in E_{\text{build}}} \text{cost}(a_i, X_j) = |A| = B$. Thus, it is true that if there is a solution for Bin Packing, there is also a solution for the corresponding TNEP-CSCE instance.

We have shown that TNEP-CSCE is in NP and that there is a polynomial-time algorithm to reduce an instance of Bin Packing to an equivalent instance of TNEP-CSCE. Additionally we have shown that all numerical parameters used by our TNEP-CSCE instance are bounded by a polynomial in the length of the input. Because we know that Bin Packing is strongly NP-complete TNEP-CSCE is also strongly NP-complete. □

5.4.2. Complexity on Planar Graphs

To prove that TNEP-CSCE is NP-complete even if the graph $G' = (V, E \cup E_{\text{cand}})$ is series parallel, the Subset Sum Problem 5.5 is reduced to an instance of TNEP-CSCE.

Definition 5.5 (Subset Sum Problem)**.**
Instance: *The instance of the Subset Sum Problem is $I = (X, m)$, with a set of integers X and an integer m*
Objective: *Decide if there is a subset of X^*, $X^* \subset X$ so that $\sum_{x \in X^*} x = m$*

It is known that the Subset Sum Problem 5.5 is NP-complete [GJ79].

Lemma 5.6. *The problem 'Transmission Network Expansion for Curing a Single Critical Edge' from Definition 5.2 is NP-complete even if the instance $G' = (V, E \cup E_{\text{cand}})$ is series parallel.*

Proof: To show that TNEP-CSCE is NP-complete for series parallel graphs, we will show that TNEP-CSCE is in NP, find a polynomial time algorithm that reduces the Subset Sum Problem to an series parallel instance of TNEP-CSCE and prove that this algorithm is correct.

TNEP-CSCE is in NP: We have already shown this in the proof for Lemma 5.4.

Construct series parallel instance of TNEP-CSCE from an instance of the Subset Sum Problem: The instance of the Subset Sum Problem is $I = (X, m)$, with a set of integers X and an integer m, as described in Definition 5.5. The resulting TNEP-CSCE instance

can be seen in Figure 5.2. As critical edge we choose an edge $(s,t) \in E$ with a criticality of $f^{\text{add}}(G,s,t) = m$ units of additional redundant flow to be cured. The critical edge is marked as red, dotted edge in Figure 5.2. For each element $x_i \in X$ we add two nodes x_i' and x_i'' with edges $(s,x_i'),(x_i'',t) \in E$. As set of candidate edges we choose the edges $(x_i',x_i'') \in E_{\text{cand}}$ for $x_i \in X$. All candidate edges $(x_i',x_i'') \in E_{\text{cand}}$ have the construction cost $\text{cost}(x_i',x_i'') = x_i$. The candidate edges are depicted as dashed edges in Figure 5.2. The capacities of all edges corresponding to an element $x_i \in X$ have capacity $\text{cap}(s,x_i'') = \text{cap}(x_i',x_i'') = \text{cap}(x_i'',t) = x_i$. The initial flow $f(u,v) = 0$ for each edge $(u,v) \in E$. The expansion planning budget is $B = m$.

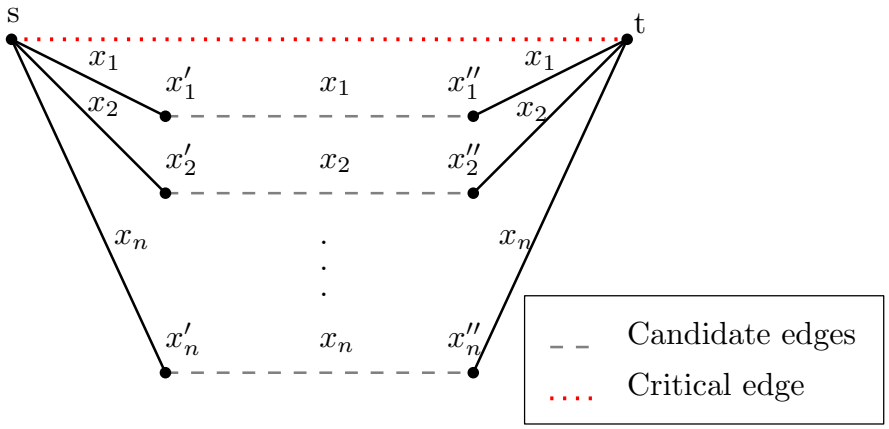


Figure 5.2.: Sketch of the graph created to reduce the Subset Sum Problem to TNEP-CSCE. Dashed lines represent candidate edges E_{cand}. The red dotted line represents the critical edge. In this figure we assume that the set X has size n. The capacity and cost for each line associated with an element x_i is the value of x_i.

Proving equivalence: It is obvious that the resulting graph is series parallel for all instances of the Subset Sum Problem.

A solution of the TNEP-CSCE instance is a subset $E_{\text{build}} \subset E_{\text{cand}}$, for investment cost less or equal the expansion planning budget $\sum_{(u,v)\in E_{\text{build}}} \text{cost}(u,v) \leq B$. The solution of the TNEP-CSCE instance $(x_i',x_i'') \in E_{build}$ induces a subset $\overline{X} \subset X$, $\overline{X} = \{x_i \mid (x_i',x_i'') \in E_{build}\}$. For all candidate edges $(x_i',x_i'') \in E_{\text{cand}}$ the construction costs are $\text{cost}(x_i',x_i'') = x_i$ and the expansion planning budget B is m. Since the set of built edges E_{build} is a solution of the TNEP-CSCE instance, the induced subset $\overline{X}$ has total costs within the budget of B. Because of the cost $\text{cost}(x_i',x_i'') = x_i$ for each candidate edge $(x_i',x_i'') \in E_{\text{cand}}$, this leads to the equation $\sum_{x\in\overline{X}} x \leq m$. The capacity of a candidate edge (x_i',x_i'') is also chosen as $\text{cap}(x_i',x_i'') = x_i$ and the critical edge needs at least $f^{\text{add}}(G,s,t) = m$ units of additional redundant flow to be cured. Since E_{build} is a solution for the TNEP-CSCE instance this means that for $G'' = (V, E \cup E_{\text{build}})$ that the critical edge (s,t) is cured. Thus at least m units of redundant flow have to be added by E_{build}. This means for the set $\overline{X}$, that $\sum_{x\in\overline{X}} x \geq m$. Taking the restrictions from the budget into account, we get $\sum_{x\in\overline{X}} x = m$. In conclusion, this means if E_{build} is a solution for the TNEP-CSCE instance, the subset $\overline{X} = \{x_i | (x_i',x_i'') \in E_{build}\}$ is a solution of the given Subset Sum Problem.

If the given subset sum instance has a solution, this means that there is a subset $X^* \subset X$, so that $\sum_{x\in X^*} x = m$. According to the construction of the corresponding TNEP-CSCE instance, there is a solution for the TNEP-CSCE instance by setting $E_{build} = \{(x_i',x_i'') \mid x_i \in X^*\}$. This is because the edges in E_{build} will add exactly m units of flow to the graph, which is enough to cure the critical edge $(s,t) \in E$ and their combined cost is m, which is within the expansion planning budget.

We know that the Subset Sum Problem is NP-complete and we can reduce any instance of Subset Sum to a series parallel instance of TNEP-CSCE. We already showed that TNEP-CSCE is in NPand therefore TNEP-CSCE is NP-complete for series parallel graphs. □

5.5. Algorithm Design

In this section we discuss the design of an algorithm for solving the problem TNEP-CCE of minimizing the extend of the critical edges in the network by adding edges from the set of candidate edges to the network. In the first Subsection 5.5.1 we discuss which candidate edges will assist in curing a specific critical edge. In the following Subsection 5.5.2 we discuss how much redundant flow an candidate edge will provide to cure an critical edge. In the third Subsection 5.5.3 we will look into different optimization algorithms and their applicability on the TNEP-CCE problem. At last we will introduce a heuristic for choosing candidate edges in Subsection 5.5.3.1, which is based on the findings discussed in the first Subsections.

Like in the last section we can also assume here without loss of generality that the initial power flow $f(x, y) \geq 0$ for all edges $(x, y) \in E$, since we do not recalculate the power flow.

Some of the challenges, that arise during the algorithm design are explained on the example graph G_{exp} in Figure 5.3. The figure shows a graph with nodes representing the generators and consumers of the transmission network and edges representing the transmission lines. The consumers are marked with an outgoing arrow and the generators are marked with an ingoing arrow. The candidate edges of the network—marked as dashed edges—are the edges than can be added to cure critical edges. On each edge of the network you can see the capacity and the current flow on this edges, this is represented by the tuple 'flow/capacity'. The candidate edges are only marked with their capacity.

The goal of the algorithm is to select the candidate edges, that will reduce as much of the criticality of the existing critical edges as possible. Note, that it is necessary to quantify the benefit a candidate edge contributes to the solution to make a qualitative statement. For the quantification it is interesting how many critical edges will be influenced by adding a candidate edge or formulated differently which candidate edges influence the curing of a specific critical edge. This topic is discussed in Section 5.5.1. In Section 5.5.2 the question what quantity a candidate edge can maximally contribute to the curing of one critical edge is discussed.

5.5.1. Identifying Candidates That Influence a Critical Edge

As discussed, the goal of this thesis is to find an algorithm to expand the transmission network in a way that minimizes the criticality of the critical edges. The research question we discuss in this section is which candidate edges are beneficial to add to the network. In this section we just discuss the question if it is beneficial to add an candidate edge and not how beneficial it is to add the edge. By discussing this question we hope to provide a better understanding of this problem as well as a way to speed up the resulting algorithm.

We assume we have a graph $G = (V, E)$ with given capacities $\text{cap}(a, b)$ and initial flow $f(a, b)$ for all edges $(a, b) \in E$. Additionally we have a set candidate edges E_{cand} with given capacities and an expansion planning budget B. Normally, we assume a set of critical edges E_{crit} to be in the input, but since this section is based on the algorithm used to identify critical edges we review the identification of critical edges as described in Section 2.3 at first.

As the identification of critical edges is described in detail in Section 2.3, we will just provide a short summary of the identification of critical edges at this point as this section

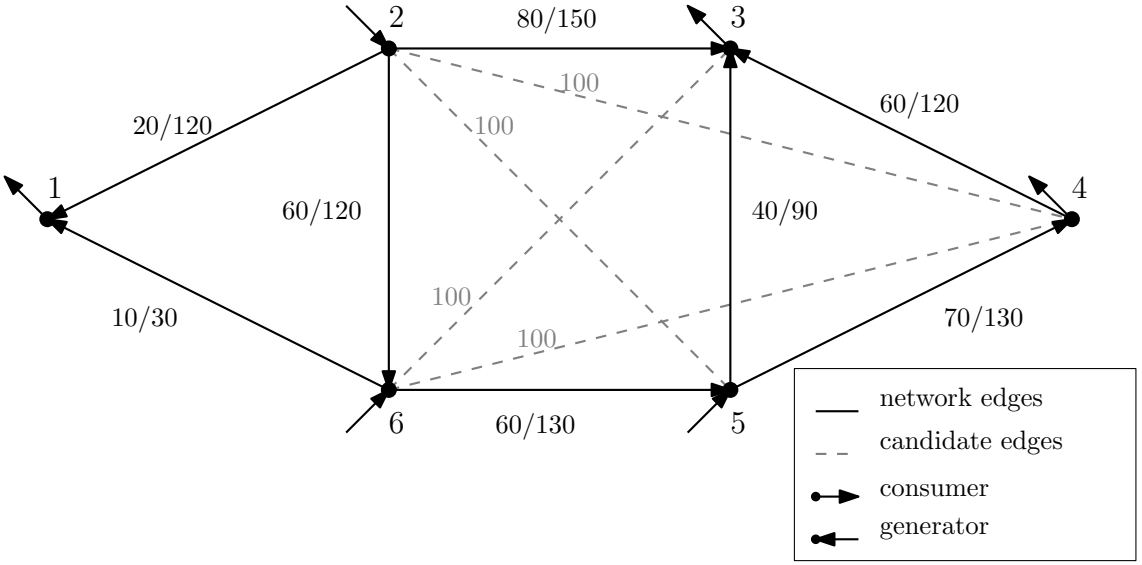


Figure 5.3.: Example graph G_{exp} used in this section to explain some steps of the algorithm design. Nodes symbolize generators and consumer. Consumer have an outgoing arrow and generators an in going arrow connected to the node. The edges symbolize the edges of the transmission network. On each edge of the network you can see the capacity and the current flow on this edges, this is represented by the tuple 'flow/capacity'. Direction of the edges depict the direction of the flow on the edge. Candidate edges are drawn as dashed edges and are marked with their capacity.

strongly depends on it. Given a graph $G = (V, E)$ representing a transmission network, each edge $(a, b) \in E$ has to be tested to determine whether it is critical. To test an edge (a, b) the edge is removed from G. Furthermore, if the flow on the edge (a, b) was directed in the direction from node a to node b, node a is selected as source and node b as sink. The redundant flow $f^{\text{red}}(G, a, b)$ is calculated by determining the maximum flow between source a and sink b in the network without the edge (a, b). The edge (a, b) is critical if $f(a, b)/f^{\text{red}}(G, a, b) > h$, for a given threshold h.

Candidate edges which influence a critical edge (a, b) have to increase the maximum a,b-flow in $G \setminus \{(a, b)\}$. To determine whether the maximum flow is increased by adding a candidate edge, we make use of the residual graph of the maximum flow calculation to determine $f^{\text{red}}(G, a, b)$. We know that as the maximum flow is calculated to determine $f^{\text{red}}(G, a, b)$ the residual graph decomposes into at least two disconnected subgraphs. One of these subgraphs contains the source a, while an other one contains the sink b. We will refer to the set of nodes in the subgraph containing a as $S_a \subset V$ and the set of nodes in the subgraph containing b as $T_b \subset V$. The two subgraphs are induced by the minimum cuts of the graph. The Min-Cut-Max-Flow Theorem says, that the value of the maximum flow is equal to the sum of capacities of edges in a minimum cut [FF56]. At this point we will ignore the nodes in $V \setminus (S_a \cup T_b)$. The redundant flow $f^{\text{red}}(G, a, b)$, being the maximum flow between a and b, can only be increased by increasing the minimum cut between the subgraphs S_a and T_b. This can only be done by adding edges between those two partitions, because adding an edge to the residual graph between the nodes of the two sets will result in at least one augmenting path between a and b in the residual graph. Thus adding edges connecting the two subgraphs will increase the maximum flow and therefore the redundant flow $f^{\text{red}}(G, a, b)$ of the critical edge (a, b).

The minimum cuts can change after adding an edge to the graph. Therefore it is possible, that the edges influencing the critical edge (a, b) in G may differ from the edges influencing the critical edge in $G \cup \{e\}$, for an $e \in E_{\text{cand}}$.

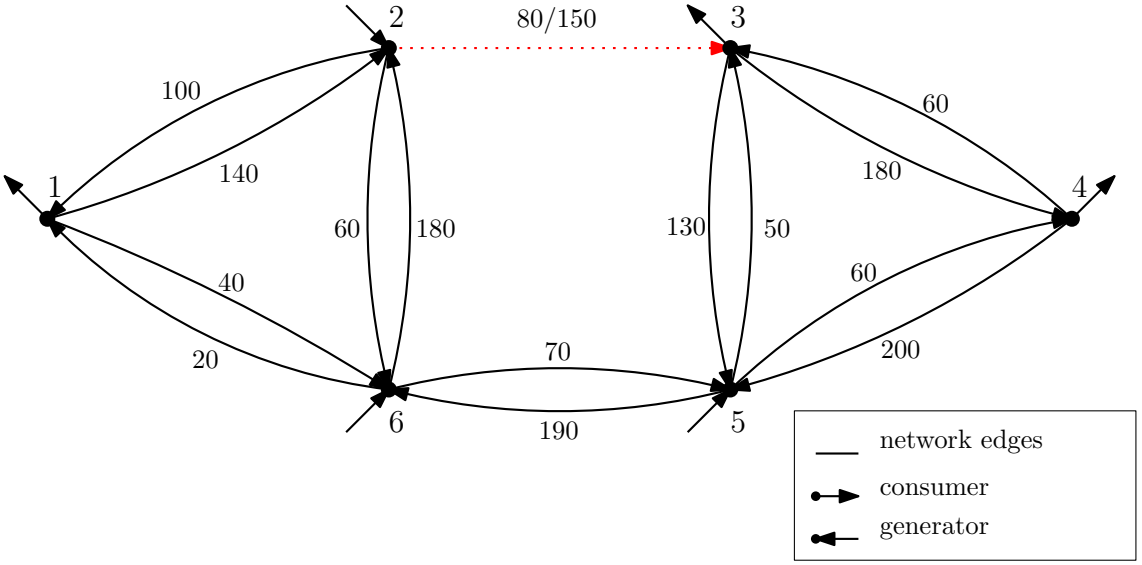


Figure 5.4.: The residual graph belonging to the example graph, as shown in Figure 5.3, after deleting edge $(2, 3)$, represented as a red dashed edge. The edges of the residual graph are represented by black edges and are marked with their capacities. The edge $(2, 3)$ is not part of the residual graph but is marked with the tuple of the original flow and capacity on this edge.

For better understanding we will test whether the edge $(2, 3)$ of the example graph G_{exp}, shown in Figure 5.3, is critical. To do that, edge $(2, 3)$ is removed from the example graph. The residual graph of the example graph G_{exp} after removing edge $(2, 3)$ can be seen in Figure 5.4. The next step is to choose the node 2 as source and 3 as sink and calculate the maximum 2,3-flow. The maximum flow between nodes 2 and 3 is calculated to be 70 units of flow. In Figure 5.5 the final residual graph resulting from the calculation of the redundant flow $f^{\text{red}}(G_{\text{exp}}, 2, 3)$ of edge $(2, 3)$ in the example graph is shown. The result for the redundant flow of edge $(2, 3)$ is the value of the calculated maximum flow and therefore $f^{\text{red}}(G_{\text{exp}}, 2, 3) = 70$. The edge $(2, 3)$ is considered as critical, because $f(2, 3)/f^{\text{red}}(G_{\text{exp}}, 2, 3) = 80/70 = 1.143 > 0.614 = h$. It can be seen that, the residual graph decomposes into two divisions. The nodes in the subgraph containing $s = 2$ are in the set S_2, but for easier understanding this is symbolized by a violet area. The nodes in the second subgraph containing $t = 3$ are in the set T_3 symbolized by an orange area. Be aware that the subgraph containing $s = 2$ is defined as containing all nodes $v \in V$ for which a path $2 \to v$ exists. The subgraph containing $t = 3$ is defined as all nodes $v \in V$ for which a path $v \to 3$ exists. The minimum cut of the example graph contains only the edge $(6, 5)$.

As all candidate edges of the example graph G_{exp}, shown in Figure 5.3, connect the two subgraphs. All candidate edges will effect the critical edge $(2, 3)$ if added to the example graph.

5.5.2. Expected Flow on Added Edge

When adding a set of candidate edges to the network, we can observe, that the additional redundant flow they provide, is not equal to the sum of capacities of the added candidate edges. Instead the additional redundant flow describes the additional flow the network can

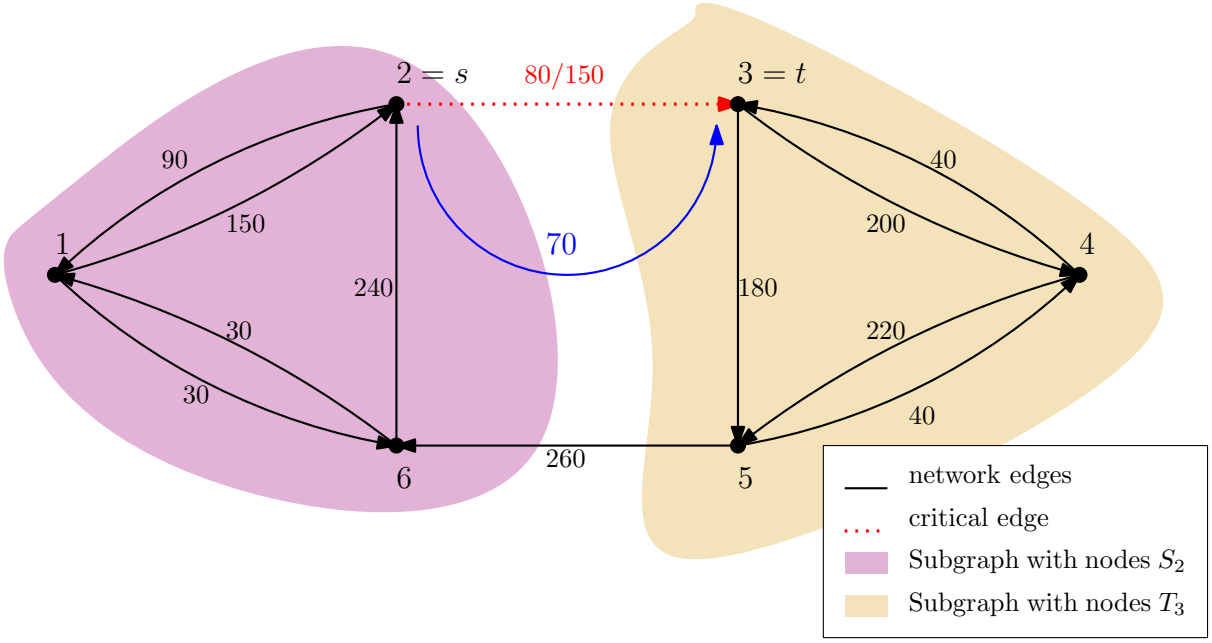
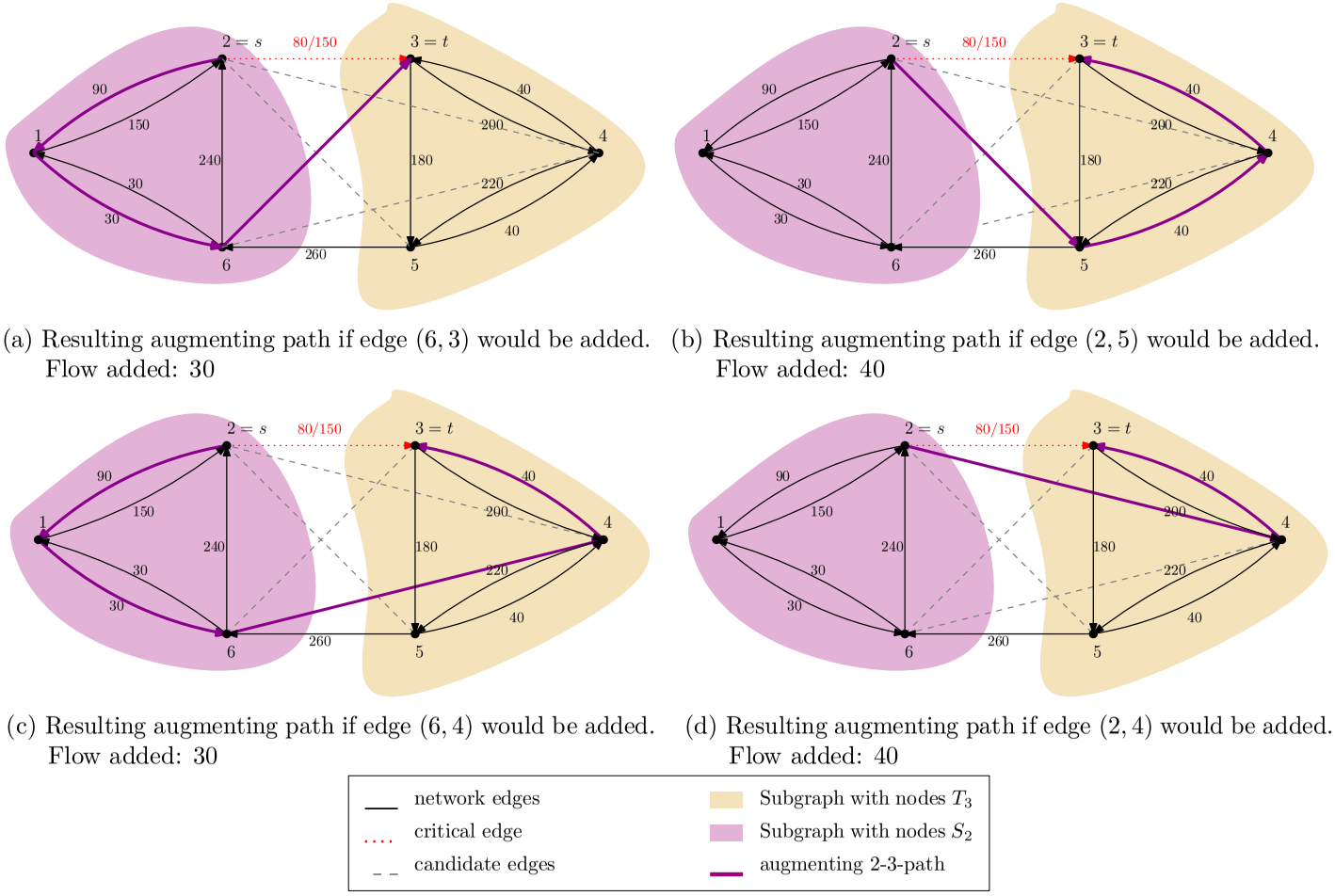


Figure 5.5.: The residual graph resulting from the check if edge $(2, 3)$—represented as a red dashed edge—is critical in G_{exp}. The edges of the residual graph are represented by black edges and are marked with their capacities. The edge $(2, 3)$ is not part of the residual graph. The minimum s-t-cut consists of edge $(5, 6)$. The minimum cut decomposes the graph into two subgraphs marked in violet and orange. The value of the calculated max flow is marked in blue. Since the roles of the nodes as generator and consumer are irrelevant for this step we did not marked them as such to simplify the visualization.

support by adding those edges. This means, if we have a critical edge (a, b), that adding an edge between two nodes $x \in S_a$ and $y \in T_b$ with a capacity c will not necessarily add c units of additional redundant flow to $f^{\text{add}}(G, a, b)$. This is the case because to actually add c units of flow we need to be able to have an a,x-flow of c units in the subgraph S_a as well as a y,b-flow of c units in the subgraph T_b. If for example the maximum a,x-flow can just support $t < c$ units of flow, the added edge will also just add t units of additional redundant flow to $f^{\text{add}}(G, a, b)$.

In this section we want to figure out a value for each edge in the set of candidate edges. From the previous observation we know that the additional redundant flow a candidate edge can provide is a good measure for its value.



(a) Resulting augmenting path if edge $(6, 3)$ would be added. Flow added: 30

(b) Resulting augmenting path if edge $(2, 5)$ would be added. Flow added: 40

(c) Resulting augmenting path if edge $(6, 4)$ would be added. Flow added: 30

(d) Resulting augmenting path if edge $(2, 4)$ would be added. Flow added: 40

Figure 5.6.: Because of the how the example graph G_{exp} was chosen there is just one augmenting path for each candidate edge. Thus the maximum flow each edge would add can be easily calculated as the minimal free capacity on the augmenting path including the candidate edge in question. Each subfigure shows the augmenting path for one of the candidate edges in dark violet. The two subgraphs the example graph decomposes into are marked as violet and orange areas. The edges not contained in the augmenting path are drawn in black. All edges are marked with their capacity. The critical edge is marked as a red dotted line and is not part of the residual graph. The critical edge is marked with its original flow and capacity.

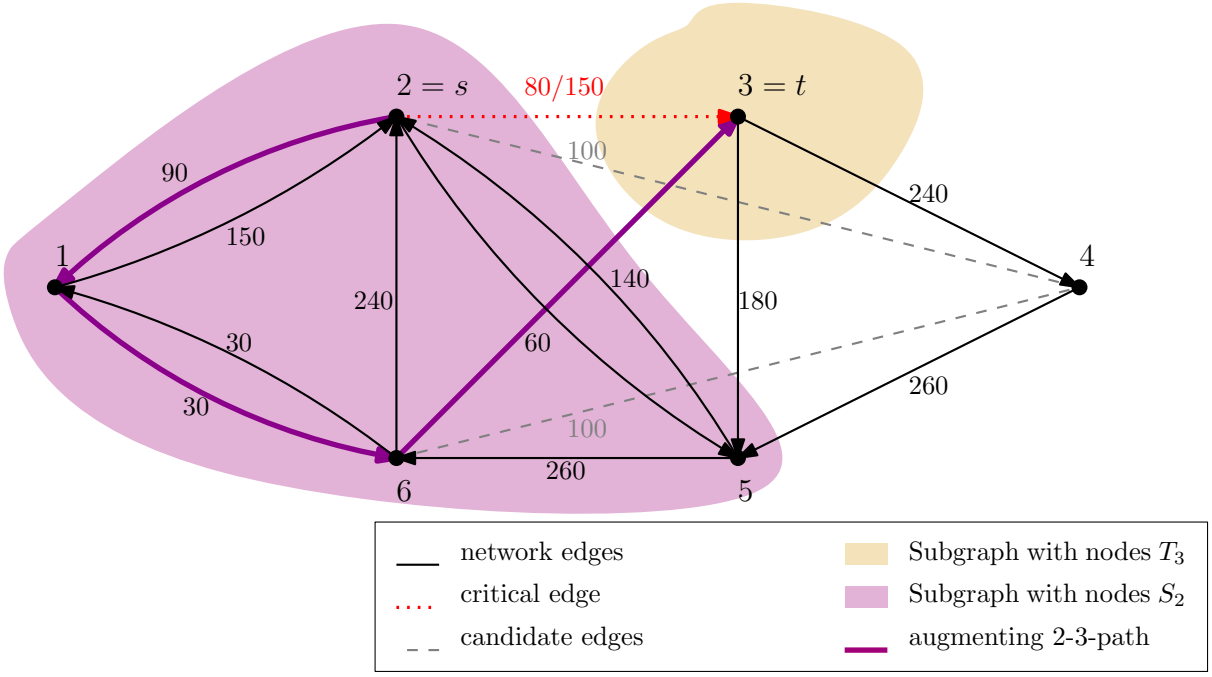


Figure 5.7.: Just the candidate line $(6, 3)$ has an augmenting path. For the other two remaining candidate no augmenting path exists. Flow added: 30

For better understanding we will look at the candidate edges in the example graph and their values. For the example graph the minimal amount of additional redundant flow you need to add to cure the critical edge $(2, 3)$ is $f^{\text{add}}(G_{\text{exp}}, 2, 3) = \frac{f(2,3)}{h} - f^{\text{red}}(G_{\text{exp}}, 2, 3) = \frac{80}{0.614} - 70 \approx 60.3$. There are four candidate edges to cure the critical edge $(2, 3)$. In Figure 5.6 each candidate edge is considered to be added to the graph. We calculated for each candidate edge, how much additional redundant flow this edge would provide to cure the critical edge $(2, 3)$. As no edge would add more than 40 units of flow, adding one edge would not be sufficient to cure the critical edge as 60.3 units of flow would be needed to do so. In the Figure 5.6 we can see that the augmenting path of the different candidate edges overlap even for such a small example. Adding candidate edges with overlapping augmenting paths, can lead to the effect that the total flow added will just increase slightly or not at all when adding the second edge. Therefore after adding one edge to the graph it is necessary to recalculate the flow a candidate edge would add. In Figure 5.7 the recalculation is shown after candidate edge $(2, 5)$ was added to the example graph. We can see that the subgraphs induced by the minimum cut changed by adding the edge $(2, 5)$. Only the candidate edge $(6, 3)$ is shown in Figure 5.7 because there is no augmenting path for the other two candidate edges for the maximum flow calculation. Therefore adding the edges $(6, 4)$ and $(2, 4)$ would not be beneficial to cure the critical edge. However, adding the edge $(6, 3)$ to the graph cures the critical edge, because 70 flow units can be added to cure the critical edge by doing so. As the other two candidate edges would not improve the redundant flow in the second iteration, this emphasizes the importance to recalculate the value of the candidate edges after adding one edge to the graph.

Therefore it is necessary to have at least a heuristic to estimate the (s, x)-flow for each $x \in S_a$ that can be supported in the subgraph containing a or the (y, t)-flow for $y \in T_b$ that can be routed throw the subgraph containing b respectively.

Additionally the max-flow to a node $x \in S_a$ or $y \in T_b$ will most likely change after a edge was added.

For the time being we won't use heuristics for the flow an candidate edge would add to cure a critical edge, but rather use the exact value. We will define this value over all edges,

not just critical edges. For edges, which are not critical adding a candidate edge should have no value. To achieve this goal we define the value of an candidate edge concerning one critical edge as $\text{val}_1 : G \times E \times E_{\text{cand}} \to \mathbb{R}_{\geq 0}$. This leads to the equation for a graph G, possibly critical edge $\text{crit} \in E$ and candidate edge $e \in E_{\text{cand}}$:

$$\text{val}_1(G, \text{crit}, e) = \begin{cases} 0, & f^{\text{add}}(G, \text{crit}) \leq 0 \\ \min(f^{\text{add}}(G, \text{crit}) - f^{\text{add}}(G \cup \{e\}, \text{crit}), f^{\text{add}}(G, \text{crit})), & \text{otherwise} \end{cases} \tag{5.30}$$

The Equation 5.30 describes the amount the critical edge crit is cured by adding candidate edge $e \in E_{\text{cand}}$ to Graph G. The term $f^{\text{add}}(G, \text{crit}) - f^{\text{add}}(G \cup \{e\}, \text{crit})$ calculates the amount the critical edge was cured. Because a critical edge can not be cured more than it was critical, we need the minimum expression. The distinction for the cases is necessary because if the edge crit is not critical, there should be no value in adding an edge.

The total value $val : G \times E_{\text{cand}} \to \mathbb{R}_{\geq 0}$ of an candidate edge $e \in E_{\text{cand}}$ is given by:

$$\text{val}(G, e) = \sum_{\text{crit} \in E_{\text{crit}}} \text{val}_1(G, \text{crit}, e) \tag{5.31}$$

The total value of an candidate edge $e \in E_{\text{cand}}$, as expressed in Equation 5.31 adds up all values $\text{val}_1(G, \text{crit}, e)$ for all critical edges $\text{crit} \in E_{\text{crit}}$.

The value of an set of edges $E_{\text{build}} = \{e_1, e_2, \dots e_n\}$ in a graph G is derived in Equation 5.32.

$$\text{val}(G, E_{\text{build}}) = \sum_{i \in 1, \dots |E_{\text{build}}|} \text{val}(G \cup \{e_j \mid j \in 0, \dots i-1\}, e_i) \tag{5.32}$$

The total value of an edge set are the summed up values for each edge in the set if one edge after another is added into the graph. Therefore written without the sum sign $\text{val}(G, e_1) + \text{val}(G \cup \{e_1\}, e2) + \dots$.

The formulation for an edge set is preferable, since the value of an set of edges for a given graph is always the same no matter in which order the edges were added to the graph. This property can be explained with the properties of the additional redundant flow $f^{\text{add}}(G, x, y)$ needed to cure a critical edge $(x, y) \in E$ in a graph $G = (V, E)$. The criticality $f^{\text{add}}(G, x, y)$ for a graph G and a critical edge $(x, y) \in E$ is positive as long as (x, y) is critical. We assume we add two edges $e_1, e_2 \in E_{\text{build}}$ to the graph. We know that $f^{\text{add}}(G, x, y)$ depends on the calculation of the maximum flow therefore the criticality has to be constant for the same graph and the same critical edge (x, y). This means the equation $f^{\text{add}}((G \cup \{e_1\}) \cup \{e_2\}, x, y) = f^{\text{add}}((G \cup \{e_2\}) \cup \{e_1\}, x, y) = f^{\text{add}}((G \cup \{e_1, e_2\}, x, y)$ is true and therefore, that the order in which the edges in E_{build} are added has no influence on the criticality $f^{\text{add}}(G \cup E_{\text{build}}, x, y)$, which is the amount of additional redundant flow needed to cure the critical edge (x, y). Therefore the value of an edge set is also independent from the order the edges are inserted into the graph.

We want to show that the Equation 5.32 can be expressed as:

$$\text{val}(G, E_{\text{build}}) = \sum_{\text{crit} \in E_{\text{crit}}} \text{val}_2(G, \text{crit}, E_{\text{build}}) \tag{5.33}$$

with:

$$\text{val}_2(G, \text{crit}, E_{\text{build}}) = \begin{cases} 0, & f^{\text{add}}(G, \text{crit}) \leq 0 \\ \min(f^{\text{add}}(G, \text{crit}) - f^{\text{add}}(G \cup E_{\text{build}}, \text{crit}), f^{\text{add}}(G, \text{crit})), & \text{otherwise} \end{cases} \tag{5.34}$$

To show this we will look at an set of critical edges $E_{\text{crit}} = \{\text{crit}\}$ containing only a single critical edge. We will show it only for one critical edge, since the calculation for each critical edge are independent from one another and therefore it is sufficient to show the equivalence for one critical edge. We will divide the problem in three cases:

case 1: The edge crit was never critical

If the edge crit was never critical then we now for the criticality, that $f^{\text{add}}(G, \text{crit}) \leq 0$. We know that $f^{\text{add}}(G, \text{crit})$ is calculated using the redundant flow of the Graph $f^{\text{red}}(G, \text{crit})$, and since $f^{\text{red}}(G, \text{crit})$ is calculated using a maximum flow, we know that $f^{\text{red}}(G \cup \{e_1\}, \text{crit}) \geq f^{\text{red}}(G, \text{crit})$ for any added edge $e_1 \in E_{\text{cand}}$. Since $f^{\text{add}}(G, \text{crit}) = \frac{f(\text{crit})}{h} - f^{\text{red}}(G, \text{crit})$ and $f(\text{crit})$ is the initial flow on the edge which is static for our problem, we know that $f^{\text{add}}(G \cup \{e_1\}, \text{crit}) \leq f^{\text{add}}(G, \text{crit})$. Meaning in the worst case that adding an edge e_1 to the Graph G does nothing to cure the a critical edge crit. Therefore we know that $f^{\text{add}}(G \cup \overline{E_{\text{build}}}, \text{crit}) \leq f^{\text{add}}(G, \text{crit}) \leq 0$ for any subset $\overline{E_{\text{build}}} \subset E_{\text{build}}$. If $f^{\text{add}}(G \cup \overline{E_{\text{build}}}, \text{crit}) \leq 0$ for any subset $\overline{E_{\text{build}}} \subset E_{\text{build}}$, we also know that:

$$\begin{aligned}
\sum_{\text{crit} \in E_{\text{crit}}} \text{val}_2(G, \text{crit}, \overline{E_{\text{build}}}) &= \text{val}_2(G, \text{crit}, \overline{E_{\text{build}}}) \\
&= 0 \\
&= \text{val}_1(G, \text{crit}, e_1) + \cdots + \text{val}_1(G \cup \{E_{\text{build}} \setminus \{e_n\}\}, \text{crit}, e_n) \\
&= \sum_{i \in 1, \ldots |E_{\text{build}}|} \text{val}(G \cup \{e_j \mid j \in 0, \ldots i-1\}, e_i) \\
&= \text{val}(G, \overline{E_{\text{build}}})
\end{aligned}$$

Therefore we know that our claim that the Equations 5.32 and 5.33 are equal is true in this case.

case 2: The critical edge crit is not cured by adding all edges from E_{build}

If the edge crit is never cured we know that $f^{\text{add}}(G \cup \overline{E_{\text{build}}}, \text{crit}) \geq 0$ for any subset $\overline{E_{\text{build}}} \subset E_{\text{build}}$. From this follows that $\text{val}_2(G, \text{crit}, \overline{E_{\text{build}}}) = \min(f^{\text{add}}(G, \text{crit}) - f^{\text{add}}(G \cup \overline{E_{\text{build}}}, \text{crit}), f^{\text{add}}(G, \text{crit}))$. We also know that $f^{\text{add}}(G \cup \{e_1\}, \text{crit}) \leq f^{\text{add}}(G, \text{crit})$ for any edge $e_1 \in E_{\text{cand}}$. Considering this and that the edge crit is not cured by adding all edges in E_{build} we know that $f^{\text{add}}(G, \text{crit}) \geq f^{\text{add}}(G, \text{crit}) - f^{\text{add}}(G \cup \overline{E_{\text{build}}}, \text{crit}) > 0$. Therefore we can ignore the minimum term and get $\text{val}_2(G, \text{crit}, \overline{E_{\text{build}}}) = f^{\text{add}}(G, \text{crit}) - f^{\text{add}}(G \cup \overline{E_{\text{build}}}, \text{crit})$. Now we can formulate for E_{build}:

$$\begin{aligned}
\sum_{\text{crit} \in E_{\text{crit}}} \text{val}_2(G, \text{crit}, E_{\text{build}}) &= \text{val}_2(G, \text{crit}, E_{\text{build}}) \\
&= f^{\text{add}}(G, \text{crit}) - f^{\text{add}}(G \cup E_{\text{build}}, \text{crit}) \\
&= f^{\text{add}}(G, \text{crit}) + (-f^{\text{add}}(G \cup \{e_1\}, \text{crit}) + f^{\text{add}}(G \cup \{e_1\}, \text{crit})) \\
&\quad + \cdots - f^{\text{add}}(G \cup E_{\text{build}}, \text{crit}) \\
&= (f^{\text{add}}(G, \text{crit}) - f^{\text{add}}(G \cup \{e_1\}, \text{crit})) + f^{\text{add}}(G \cup \{e_1\}, \text{crit})) \\
&\quad + \cdots - f^{\text{add}}(G \cup E_{\text{build}}, \text{crit}) \\
&= \text{val}_1(G, \text{crit}, e_1) + \text{val}_1(G \cup \{e_1\}, \text{crit}, e_2) + \ldots \\
&\quad + \text{val}_1(G \cup \{E_{\text{build}} \setminus \{e_n\}\}, \text{crit}, e_n) \\
&= \text{val}(G, E_{\text{build}})
\end{aligned}$$

Therefore we know that our claim that the Equations 5.32 and 5.33 are equal is also true in this case.

case 3: The critical edge crit is cured by adding a subset $\overline{E_{\text{build}}}$ edges from E_{build}
Without loss of generality we assume an index i, such that if the set of edges $E_{<i} = \{e_j \mid e_j \in E_{\text{build}}, j < i\}$ is added the critical edge crit is not cured and if any of the edges from $E_{\geq i} = \{e_j \mid e_j \in E_{\text{build}}, j \geq i\}$ is additionally added the critical edge is cured. For the first subset $E_{<i}$ we can use the explanation from case 2. After adding the first subset $E_{<i}$ and one edge e_x from $E_{\geq i}$ we can use the explanation from case 1 for the remaining edges from $E_{\geq i} \setminus \{e_x\}$. And for the single edge e_x we know that $\text{val}_2(G, \text{crit}, \{e_x\}) = \text{val}_1(G, \text{crit}, e_x)$.

So we can show that the expressions in Equation 5.32 and 5.33 are equivalent if E_{crit} consist of one critical edge. Since this can be any critical edge and the calculations for the critical edges are independent, we can generalize it from one critical edge to a set of critical edges.

5.5.3. Selecting Edges to Add to the Graph

In Sections 5.5.1 and 5.5.2 we discussed the aspects influencing the value of an candidate edge. Now we want to select candidate edges based on this value and the cost of each candidate edge i, which is given as input to the problem.

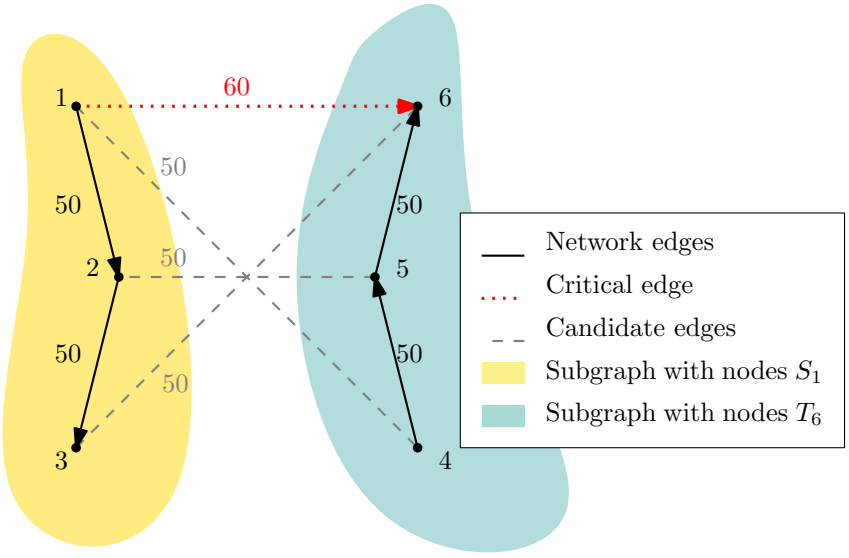


Figure 5.8.: Small example of a residual graph G_{greed} for which a greedy algorithm would result in a suboptimal solution. The critical edge is marked as red dotted edge. Candidate edges are marked as dashed edges. Cost for the candidate edges are $\text{cost}(1,4) = \text{cost}(3,6) = 2$ and $\text{cost}(2,5) = 1$. Each candidate edge has a capacity of 50 units. The criticality of the critical edge is $f^{\text{add}}(G_{\text{greed}}, 1, 6) = 60$.

The first idea might be to use an greedy algorithm to select edges. This can be done by ordering edges by their value-to-cost-ratio and adding edges until the critical edges are cured. But it can easily be shown that a greedy algorithm does not necessarily achieve the optimal solution even for very small instances of the problem TNEP-CCE. To demonstrate this, Figure 5.8 shows a small example graph G_{greed} consisting only of one critical edge and three candidate edges. The total budget for adding candidate edges is $B = 4$. The capacity of each candidate edge (a, b) is $\text{cap}(a, b) = 50$. The value of each candidate edge is also $val(a, b) = 50$, because there is just one critical edge all candidate edges influence it and can support 50 units of flow. The criticality of the critical edge is $f^{\text{add}}(G_{\text{greed}}, 1, 6) = 60$ units of additional redundant flow. Because of the candidate edges capacities it is necessary to add at least two candidate edges to cure the critical edge. The costs of the candidate edges are $\text{cost}(1,4) = \text{cost}(3,6) = 2$ and $\text{cost}(2,5) = 1$. The ratios of value to cost are 25 for edges $(1,4)$ and $(3,6)$ and 50 for line $(2,5)$. Therefore a greedy algorithm would

choose edge $(2, 5)$ as first candidate edge to add. This would result in the critical edge being not curable as none of the other edges would have any value in the second step. The optimal solution would be to add the candidate edges $(1, 4)$ and $(3, 6)$, order being irrelevant. Therefore using a greedy algorithm would not lead to the optimal solution for this example.

So instead of using a greedy algorithm the next idea might be to use an dynamic programming approach to chose from the candidate edges. Dynamic programming is based on Bellman's Principle of Optimality [Bel57]. The Principle of Optimality states that there are problems for which the optimal solution of a problem can be composed from the optimal solutions of its subproblems. Problems that fulfill this requirement are also called problems with optimal substructures. Meaning that a problem can only be solved by dynamic programming if Bellman's Principle of Optimality applies to it or in other words if it has an optimal substructure.

For this proof we have to define the subproblems for a TNEP-CCE instance. Let $I = \{G = (V, E), E_{\text{cand}}, E_{\text{crit}}, B\}$ be a TNEP-CCE instance then we define I' as instance of the subproblems as $I' = \{G = (V, E), E'_{\text{cand}} \subset E_{\text{cand}}, E_{\text{crit}}, B' \leq B\}$.

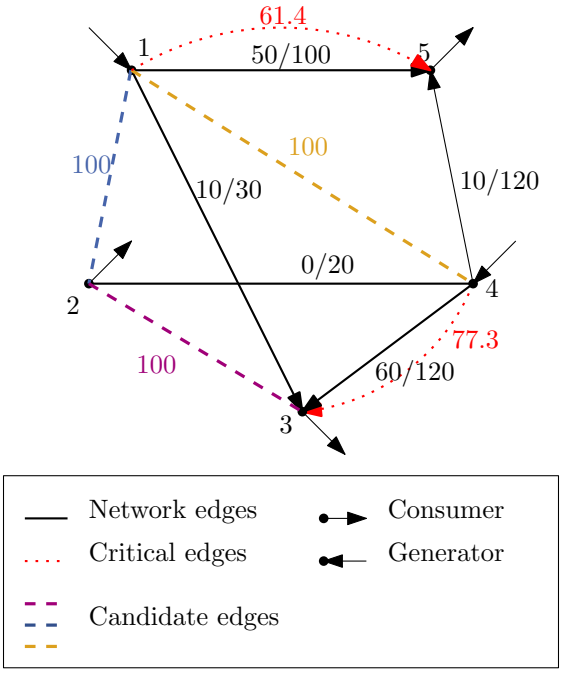
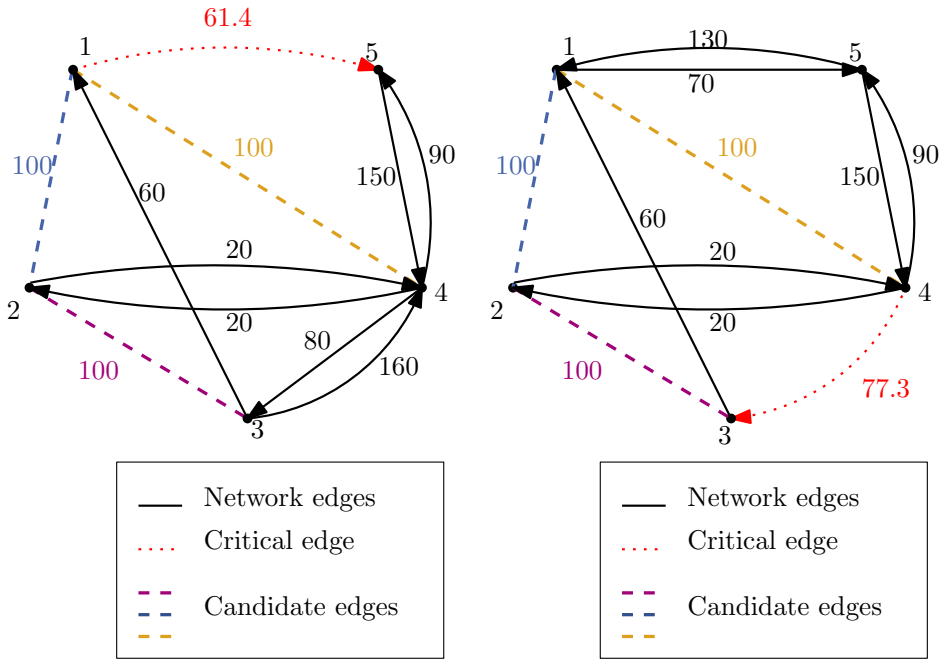


Figure 5.9.: Small example of a graph G for which a dynamic programming algorithm would result in a suboptimal solution. The graph has two critical edges $(1, 5)$ with $f^{\text{add}}(G, 1, 5) = 61.4$ and $(4, 3)$ with $f^{\text{add}}(G, 4, 3) = 77.3$. Candidate edges are marked as dashed edges. Cost for the candidate edges are $\text{cost}(1, 4) = \text{cost}(2, 3) = 2$ and $\text{cost}(1, 2) = 3$. The capacity for each candidate edge has is marked on the edge. The direction of the edges marks the direction of the flow on the edges.

We can show that Bellman's Principle of Optimality does not apply to TNEP-CCE problem. To show this we will use the counterexample provided by the graph in Figure 5.9. The graph has two critical edges $(1, 5)$ with $f^{\text{add}}(G, 1, 5) = 61.4$ and $(4, 3)$ with $f^{\text{add}}(G, 4, 3) = 77.3$ capacity needed to add. For better understanding the residual graphs resulting from calculating the critical edges are displayed in Figure 5.10a and Figure 5.10b. We assume an expansion planning budget of $B = 5$ units for this Problem. The optimal solution for this graph would be to add the candidate edges $(1, 2)$ and $(2, 3)$, which would cure both critical



(a) Residual graph resulting from check if edge $(1, 5)$ is critical.

(b) Residual graph resulting from check if edge $(4, 3)$ is critical.

Figure 5.10.: Residuals graphs of the graph introduced in Figure 5.9. The edges of the residual graph are marked black. The candidate edges are marked as dashed lines. The capacity of each edge is written next to it. The edges checked for being critical are marked as red dotted lines and the value of criticality of the edge is written next to it.

edges within the budget. If TNEP-CCE would have an optimal substructure this optimal solution should consist of optimal solutions of the subproblems. Therefore looking at the subproblem excluding the candidate edge $(2, 3)$ for a budget of $B = 5 - \text{cost}(2, 3) = 3$ the optimal solution should be adding edge $(1, 2)$. But adding candidate edge $(1, 2)$ would not cure any of the critical edges while adding candidate edge $(1, 4)$ would cure critical edge $(1, 5)$ completely. This means that adding candidate edge $(1, 4)$ would be the optimal solution for the given subproblem. Therefore Bellman's Principle of Optimality does not apply to the TNEP-CCE problem as it has no optimal substructure and an dynamic programming algorithm would not be able to solve the problem optimally.

5.5.3.1. A Heuristic for Choosing Candidate Edges

We already established that neither dynamic programming nor greedy algorithms can be used to solve the TNEP-CCE problem optimally. Until now the most efficient way we found to solve TNEP-CCE optimally is the formulation as MILP and solving it. At this point we will start to look into heuristics to choose from the candidate edges to solve TNEP-CCE. We hope that by using a heuristic approach for solving TNEP-CCE we will find a more intuitive algorithm to solve this problem even if the solution is not optimal.

The general idea behind the heuristic is, that our problem has some parallels to the 0-1 Knapsack problem. A definition for the 0-1 Knapsack problem is given in 5.7.

Definition 5.7 (0-1 Knapsack)**.**
Instance: *A set of objects O with a given weight function $w : O \to \mathbb{R}_{\geq 0}$ and a given value function $v : O \to \mathbb{R}_{\geq 0}$. Also a budget $B \in \mathbb{R}_{\geq 0}$ is given.*
Objective: *Find an assignment for $x_i \in \{0, 1\}$ for each $i \in O$ so that:*

- *the sum of values is maximal:* $\max \sum_{i \in O} x_i v(i)$
- *and the weights are with in the budget:* $\sum_{i \in O} x_i w(i) \leq B$

The parallels of the two problems can be seen if you compare the candidate edges from the TNEP-CCE problem to the objects of the 0-1 Knapsack problem. In both problems you have to choose a subset from these sets, so that the costs/weights of those subsets are within the budget. Furthermore in both cases there are values associated with the items in the candidate edges/objects. There are two differences the first one being that TNEP-CCE is a minimization problem while Knapsack is a maximization problem. The second is the value function.
It is easy to reformulate TNEP-CCE to be a maximization problem by using the objective $\max_{E_{\text{build}} \subset E_{\text{cand}}} \text{val}(G, E_{\text{build}})$. Because of the definition of the value for a subset of candidate edges $E_{\text{build}} \subset E_{\text{cand}}$ in Equation 5.33, the formulation as maximization problem is equivalent to the formulation as minimization problem.
We know that TNEP-CCE has no optimal substructure. This fact is reflected in the value function in Equation 5.33. While the value of an object $v(a)$, $a \in O$ in the 0-1 Knapsack problem is independent from previous object selections and therefore the equation $v(O_{\text{add}}) = \sum_{o \in O_{\text{add}}} v(o)$ is true, this is not the case for the value function in the TNEP-CCE problem as shown in Equation 5.32.

The idea for the heuristic approach is, that the instances, in which dynamic programming does not work, might not occur often in real transmission network topologies. Therefore we assume that a solution given by a dynamic programming approach will be good in most cases. In more detail this means that we base our edge selection on the edge selections for the subproblem of the current problem, which has a equal or smaller budget and less candidate edges to chose from. Because of the similarities between 0-1 Knapsack and TNEP-CCE we use a dynamic programming algorithm for 0-1 Knapsack to base the selection algorithm for TNEP-CCE on. The pseudo code for the resulting algorithm ExpandNetwork can be seen in the Algorithm 5.1.

The Algorithm ExpandNetwork is not a dynamic programming algorithm but a heuristic based on the idea of dynamic programming. This is because we do not reuse the calculated value of the subproblems but recalculate the value of the edge selection in each step.

For the algorithm we assume, that the set of candidatelines is ordered in the way that $E_{\text{cand}}[i] = e_i$ with $e_i \in E_{\text{cand}}$. We also assume, that array indices start with 0. The array K we use has dimensions of $(B+1) \times (\texttt{length}(E_{\text{cand}}) + 1)$. We iterate the array in the outer for-loop over the candidate edges $E_{\text{cand}}$and in the inner for-loop over the budget B. Each field of the array $K[j][i]$ describes the probably best solution for the subproblem with the set of candidate edges $E'_{\text{cand}} = \{e_k | k \leq i, e_k \in E_{\text{cand}}\}$ and the budget $B' = j$. Each entry $K[j][i]$ is a 3-tuple (value, cost, E'_{build}), with value being value = $\text{val}(G, E'_{\text{build}})$, cost being cost = $\sum_{e_i \in E'_{\text{build}}} \text{cost}(e_i)$ and $E'_{\text{build}} \subset E'_{\text{cand}}$ being the optimal solution found for the subproblem described by this array entry.

The runtime of ExpandNetwork is in $O(B|E_{\text{cand}}| * \text{runtime}(\text{Val}))$ time. Therefore the runtime of ExpandNetwork strongly depends on the runtime of the algorithm which determines the value of an candidate subset Val. The pseudo code of the Val algorithm can be found in 5.2. The runtime of the Val is the same as $|E_{\text{crit}}|$ times the runtime of the Algorithm redundantFlow 2.1 as this algorithm is executed in the function $f^{\text{add}}(G, a, b)$. Therefore the runtime of Val is in $O(E_{\text{crit}} \cdot |V| \cdot (|E| + |E_{\text{build}}|)^2)$ if we use the redundantFlow 2.1 algorithm.

5.5.4. Modifications

In the following section we will propose some modification to the algorithm introduced before to improve the performance of the algorithm. We argued before that the runtime of the Algorithm 5.1 strongly depends on the runtime of the subroutine to calculate the value of a subset of candidate edges. Thus we will first look into improvements concerning the value function. The effects of these improvements are evaluated in the Section 6. In the Section 6 we refer to the originalExpandNetwork Algorithm as HEU for short.

5.5.4.1. Value Function

Lookup table

The first modification is the idea to introduce a lookup table for already calculated values. This will not have any affect of the worst-case runtime of the algorithm, but we assume that there are cases in which the values for some edge combinations are calculated multiple times. Therefore a look up table may improve the runtime of the algorithm.

We implemented the lookup table so that we calculate and add a value to it if the value for an subset of candidate edges is not already in the table when requested by the Algorithm ExpandNetwork. The lookup table will not influence the quality of the result of the Algorithm ExpandNetwork as we use the same method to calulcate the values. In the Section 6 we will refer to the algorithm resulting from this modification as HEU_LU.

Value approximation

In section 5.5.1 we discussed the necessity to recalculate the value for an edge combination $E_{\text{com}} \cup \{e\}$,$E_{\text{com}} \subset E_{\text{cand}}$ and $e \in E_{\text{cand}}$ even if we already know the values for E_{com} and e. The idea of this modification is to ignore this fact and approximate $\text{val}(E_{\text{com}} \cup \{e\})$ as $\text{val}(E_{\text{com}} \cup \{e\}) = \text{val}(\{e\}) + \text{val}(E_{\text{com}})$. This means we only have to calculate the value for each candidate edge once which adds up to $|E_{\text{crit}}| \cdot |E_{\text{cand}}|$ maximum flow calculations instead of $|E_{\text{crit}}| \cdot B \cdot |E_{\text{cand}}|$ maximum flow calculations. This approximation also means, that we have an optimal subset of our problem. This means we can reuse the previously computed values for the subsets of candidate edges and therefore have a real dynamic programming algorithm. But since we already showed the flaws of this approximation we can expect a drop of quality for the result of the algorithm using this value approximation.

In the Section 6 we will refer to the algorithm resulting from this modification as HEU_APPROX.

Algorithm 5.1: ExpandNetwork

Input: Graph $G = (V, E)$,capacity $\text{cap}(u, v)$, initial flow $f(u, v)$ for each $(u, v) \in E$, set of candidate edges E_{cand} with capacity $\text{cap}(u, v)$ and cost $\text{cost}(u, v)$ for each $(u, v) \in E_{\text{cand}}$, set of critical edges E_{crit}, expansion planning budget B

Data: K Array of size $(B + 1) \times (\text{LENGTH}(E_{\text{cand}}) + 1)$ initialized with tuple $(0, 0, \emptyset)$

Output: List of selected edges E_{build}

```
for i = 1 to |E_cand| + 1 do
    for j = 1 to B + 1 do
        e_i = E_cand[i]
        // If there is enough money to add candidate line i check if
           adding it is beneficial
        if j ≥ cost(e_i) then
            old_value, old_cost, old_edge_set = K[j − cost(e_i)][i − 1]
            // compute new values
            new_edge_set = old_edge_set ∪ {e_i}
            new_value = VAL(G, new_edge_set, E_crit)
            new_cost = old_cost + cost(e_i)
            if new_value > K[j][i − 1][0] then
                // If new_value is better take new
                K[j][i] = (new_value, new_cost, new_edge_set)
            else if new_value == K[j][i − 1][0] then
                // If values are the same take the one with smaller
                   costs
                if new_cost ≤ K[j][i − 1][1] then
                    K[j][i] = (new_value, new_cost, new_edge_set)
                else
                    K[j][i] = K[j][i − 1]
            else
                // Take old entries as they have better values
                K[j][i] = K[j][i − 1]
        else
            // Not enough budget to build line, take old values
            K[j][i] = K[j][i − 1]
return K[B][|E_cand|]
```

Algorithm 5.2: Val

Input: Graph $G = (V, E)$,set of selected edges E_{build}, set of critical edges E_{crit}

Data: float value

Output: Value for selected edges

```
value = 0.0
forall (a, b) ∈ E_crit do
    value+ = MAX(0, f^add(G, a, b) − f^add(G ∪ E_build, x, y)
return value
```

6. Evaluation

In this section we evaluate the methods we developed to solve the Problem TNEP-CCE as introduced in Section 5.2.

6.1. Evaluation Methods

To obtain reliable data for our test data set we test the optimization methods on all transmission networks we introduced in Section 4.1. For each network we test 48 different load distributions on the network by taking the hourly snapshots for the 1st of January 2013 as well for the 25th June 2013. We know that the computation time and the amount of criticality our optimization methods can cure depend on the provided candidate edges as well as on the given budget. To test the effects of varying budget and varying number of candidate edges we run two test series.

For the first test series we use a fixed set of candidate edges and modify the budget. For each snapshot of a network we test 11 different budgets between 0% and 50% of the total costs of all candidate edges. We choose these percentages as we can cure all criticalities for tested networks within this range. We start with 0% of the total cost for the budget and increase the budget in 5% steps until we reach 50%. Thus we run the optimization for each network 528 times which should provide meaningful data. The candidate edges for this test series are generated using the knn-method, introduced in Section 4.3.2. For creating the candidate network we use $k = 2$, meaning we create at least two candidate edges for each node. In the following sections, we refer to this test series as the `Budget` test series.

For the second test series we choose a fixed budget of 30% of the total costs for all candidate edges with a varying amount of candidate edges. We decide to use 30% for the budget as it is not possible to cure all critical edges with this budget in most test networks and we expect to see more variations between the evaluated algorithms. For each network we generate a set of candidate edges using the knn-method with $k = 2$, which is introduced in Section 4.3.2. From this set of candidate edges we generate random subsets consisting of 0% to 100% of the candidate edges in 10% steps. This is achieved by using the python package `random` which allows us to generate a random subset of the candidate edges given a specified size. For more information on the python package `random`, consult the documentation of the Python Standard Library [Fou19b]. We use each of those subsets as input for all of our optimization methods for 48 different load distributions of the tested network. Therefore we run each optimization method on each network 528 times for this test series. In the following sections we refer to this test series as the `Candidate` test series.

Generally, we are most interested in the computation times of the methods as well as the quality of their results. For measuring the computation times, we decide to measure the time for generating the mathematical model as well as the time for optimizing the model. For the result quality, we are interested in the prediction error of each optimization method as well as their result in comparison to the other methods. The *prediction error* we define for each optimization method as the absolute difference of the optimal objective value, which is the amount of criticality predicted for the network after building the proposed edges, and the amount of criticality which is actually cured by adding the proposed candidate edges to the network. To obtain the latter value, we add the proposed edges to the original network and perform a DC power flow approximation on this new network. Depending on this recalculated power flow we then calculate for each edge in the network its value of criticality.

All optimization methods are tested on a laptop with a Intel Core i7-6700HQ processor with 6 MB cache, 2.6 GHz, 8 cores and 16 GB RAM. For the evaluation the operating system used is Arch Linux with the Linux kernel version 4.20.11. The optimization methods are implemented using Python 3.7.2 [Fou19a] and Gurobi version 8.1 [GO18]. The used python packages are graph-tool version 2.27 [dPP27] and PyPSA version 0.13.2 [BHS19, BHS18].

6.2. Reference Optimization Method - Mathematical Model with DC Approximation

The mathematical model with DC power flow approximation is the reference model to which we compare our results. This model, first introduced in Section 5.3.1, calculates the best possible subset of candidate edges while considering the power flow changes caused by adding candidate edges to the network. In total we discuss three variants of this model in Section 5.3.1.

The first variant is the model with the objective to minimize the sum of the criticalities for a given set of critical edges, which are part of the input. We refer to this model as MMDC_SET, an abbreviation for mathematical model using the DC power flow approximation with a given set of critical edges.

The second variant has the same objective but minimizes the sum of criticalities for all edges, to avoid creating new critical edges caused by power flow changes when adding edges to the network. Because of the recalculation of all criticalities we refer to this model as MMDC_RECALC.

We decide in Section 5.2 to use the objective of minimizing the sum of criticalities as main objective of the problem TNEP-CCE for this thesis. Nonetheless we are interested in the effect of choosing the different but similar objective of minimizing the number of critical edges. Therefore we choose as third variant the objective to minimize the number of critical edges for a given set of critical edges. We refer to this version as MMDC_NUM. Since we already decided in Section 5.2 that we use the objective of minimizing the sum of criticalities, we refrain from evaluating more than one model using a different objective.

First, we compare the versions MMDC_SET and MMDC_RECALC with the same objectives to see which effect a given set of critical edges has versus checking all edges for criticality in Subsection 6.2.1.

6.2.1. Given Set of Critical Edges versus Calculated Set of Critical Edges

In this subsection we compare the model MMDC_SET, with a given set of critical edges, to the model MMDC_RECALC, which models the criticality for all edges.

We have some hypotheses, what we expect to observe during the evaluation, that we introduce in the following paragraph. For the model MMDC_RECALC we do not introduce any additional simplifications, which leads to Hypothesis 1.

Hypothesis 1. *We expect to see no prediction error for the model MMDC_RECALC.*

For the model MMDC_SET we use a fixed set of critical edges. Because of this simplification criticalities of previously not critical edges are not taken into account in the optimization process, meaning that the criticality of the network predicted by MMDC_SET might be too low. This leads to Hypothesis 2 and 3.

Hypothesis 2. *For the model MMDC_SET we expect to see some kind of prediction error.*

Hypothesis 3. *We expect the model MMDC_SET to predict a lower value of criticality than the actual value of criticality.*

Because MMDC_RECALC models the criticality for all edges, the model has a higher number of constraints and variables than MMDC_SET for the same network. This results in Hypothesis 4

Hypothesis 4. *We also expect a higher optimization and model generation time for the model MMDC_RECALC in comparison to the model MMDC_SET.*

The candidate edges are an important part of the mathematical models, with multiple variables and constraints for each candidate edge. Because an increasing number of candidate edges increases the model complexity accordingly, this leads to Hypothesis 5

Hypothesis 5. *We expect for both models, that optimization and model generation time become longer for increasing number of candidate edges.*

In contrast to the candidate edges we only have one constraint for the budget. This means that a higher budget will not lead to more constraints or variables. But we know that the budgets limits the investment cost for the candidate edges, meaning if we have a higher budget we can add more edges and therefore more edge combinations can be feasible. As long as the budget limits the choice of candidate edges the computation time should therefore increase. These thoughts lead to Hypothesis 6

Hypothesis 6. *We expect for both models, that optimization time will increase for increasing budget as long as not all candidate edges can be added. The model generation time should not be effected by the budget.*

We already discussed that we can add more candidate edges with an increasing budget. For most critical edges and sensible chosen sets of candidate edges we expect that having the possibility to add more candidate edges will make it possible to cure more criticalities. We expect that is is possible to cure more criticalities with increasing budget until all criticalities are cured or until we added all candidate edges which effect the criticalities. This leads to Hypothesis 7.

Hypothesis 7. *We expect for both models, that the sum of criticalities decreases with increasing budget.*

For the `Candidate` test series we assume that if we have a really small set of candidate edges it is not possible to make a lot of choices. If the amount of candidate edges is increased, it is possible that there are more suitable candidate edges or better edge combinations possible when choosing from the larger set of candidate edges. This leads to Hypothesis 8.

Hypothesis 8. *We expect for both models, that the sum of criticalities decreases with increasing number of candidate edges.*

First we discuss the results for the `Budget` test series, having a fixed set of candidate edges and a variable budget.

For all networks we observe a decline in criticality with increasing budget for both models, just as we predicted in Hypothesis 7. This can be seen for example in Figure 6.1, in which the value of criticality is depicted for both models in solid lines for the Slovak transmission network. We observe for all networks that the value of criticality of both models is equal or nearly the same. This means both models cure approximately the same amount of criticality.

As predicted in the Hypothesis 1 we do not observe a prediction error for the MMDC_RECALC model. Interestingly, for the MMDC_SET model we only observe a prediction error for three networks. The first network we see the error for the MMDC_SET model is the Slovak transmission network with 9 nodes and 13 edges, and the second one being the Bulgarian transmission network with 12 nodes and 17 edges and the third one is the Romanian transmission network with 17 nodes and 27 edges. Also those prediction errors only occur for the Bulgarian transmission network for the budget being around 20% of the total cost of candidate edges and for the Romanian only for the budget being around 15% of the total cost of candidate edges. For all other networks both models MMDC_SET and MMDC_RECALC do not have any prediction error. This observation is contrary to what we expected in Hypothesis 2, because the prediction error of MMDC_SET just occurs only on few networks, while we expected it to occur on almost all networks. This means that the model MMDC_SET using a given set of critical edges performs optimal for the utmost part of our test data. Figure 6.1 shows the error of the model MMDC_SET in relation to the budget in percentage of the total cost of all candidate edges. In the same figure we also depict the real value of criticality for MMDC_SET drawn as solid blue line and for model MMDC_RECALC as solid orange line. We can also see the predicted value of criticality for MMDC_SET as dotted blue line. The prediction error of the model MMDC_SET is drawn as dashed blue line in the figure. The prediction error is equal to the absolute difference of the predicted value and real value of criticality. This can also be observed in the figure. We already established that our evaluation results do not support Hypothesis 2 for almost all networks. But for all occurrences of the prediction error for model MMDC_SET, it is relatively small. For example the largest prediction error occurred on the Slovak transmission network and is around 161 MW, but this is only 6% of the initial value of criticality. The average error for all snapshots shown in Figure 6.1 for the Slovak transmission network for the same budget is only around 35 MW. This is less than 2% of the initial value of criticality, which is around 2600 MW and can be seen in the figure for a budget of 0% when looking at the real values of criticality for both models. The value of the error depends on the topology of the network as well as on the set of candidate edges. So we have no further explanation why the error for MMDC_SET depicted in Figure 6.1 has a local minimum for a budget of 40%.

In Figure 6.1 we can also observe, that the predicted value of criticality is always lower than the real value of criticality. This result agrees with Hypothesis 3. This is the case for all networks in which we observe a prediction error.

We did not test the model MMDC_RECALC for all transmission networks for all snapshots. The reason for this is, that the optimization time of the model MMDC_RECALC is way higher than for the the model MMDC_SET. The optimization and model generation times for the Bulgarian transmission network can be seen in Figure 6.2 and the times for the Swiss transmission network can be seen in Figure 6.3 for both models. In the figures the

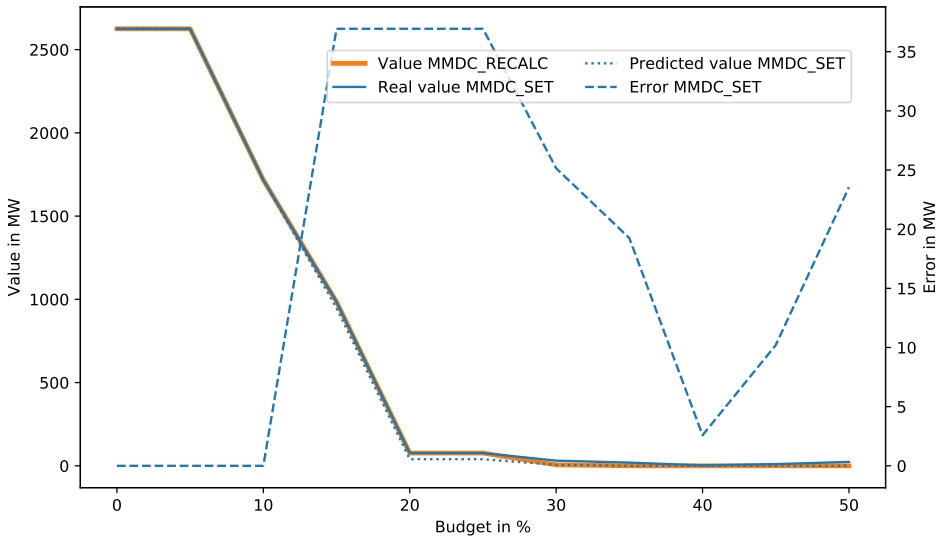


Figure 6.1.: This figure shows the results for the models MMDC_SET and MMDC_RECALC for the Slovak transmission network for the `Budget` test series. In this figure the real values of criticality for the model MMDC_SET is drawn as solid blue line and the real value of criticality for MMDC_RECALC is drawn as solid orange line. For the model MMDC_SET the predicted value of criticality is drawn as dotted blue line. All values of criticality are marked on the left side of the figure. The prediction error for the model MMDC_SET is drawn as dashed blue line. The values for the prediction error are marked on the right side of the figure.

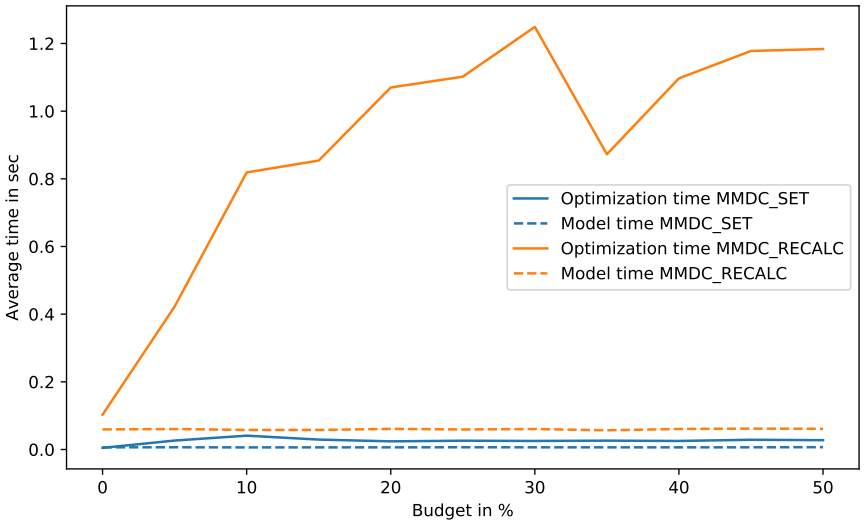


Figure 6.2.: Optimization and model generating time of the two model versions MMDC_SET and MMDC_RECALC for the Bulgarian transmission network for the `Budget` test series. Times for the MMDC_SET are depicted in blue and times for MMDC_RECALC are depicted in orange. The model generation time is depicted using dashed lines. The solid lines are used for depicting the values for the optimization times.

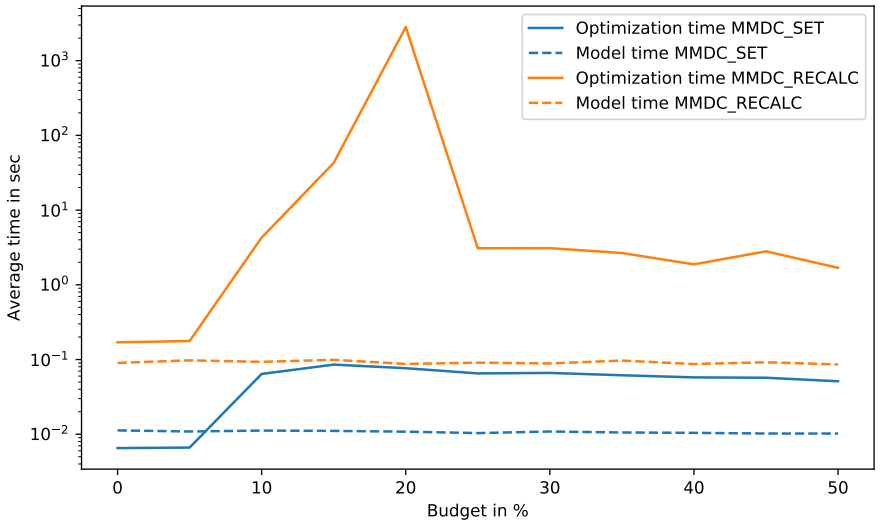


Figure 6.3.: Optimization and model generating time of the two model versions MMDC_SET and MMDC_RECALC for the Swiss transmission network in the `Budget` test series. Times for the MMDC_SET are depicted in blue and times for MMDC_RECALC are depicted in orange. The model generation time is depicted using dashed lines. The solid lines are used for depicting the values for the optimization times.

optimization times are depicted as solid lines while the model generating times are depicted as dashed lines. The model MMDC_SET is represented by using blue lines and the model MMDC_RECALC is represented using orange lines in both figures. Both figures show that the times for the model MMDC_RECALC, depicted by the orange lines, both for the model generation and optimization times, are higher than their blue counterparts. Especially in Figure 6.3 where the times for the Swiss transmission network are shown, we can see large differences between the both models. We observe this behavior on all tested networks. This means that the model MMDC_RECALC has really high computation times in comparison with the the model MMDC_SET as discussed in Hypothesis 4. The higher optimization time and model generation time for the model MMDC_RECALC are as expected, because this model has a lot more constraints and variables to consider criticalities for all edges. We can observe for all networks that the budget does not effect the model generation time. For example in Figure 6.2 we can see for the Bulgarian transmission network, that neither one of the model generation time, depicted with dashed lines, increase for increasing budgets. This supports one part of the Hypothesis 6. For both models we observe an all networks increasing optimization times with increasing budget, as long as the budget is between 0% and 15% of the total costs of candidate edges. This is also according to Hypothesis 6. However we observe on most networks for larger budgets than 15% similar behavior as shown for the Swiss transmission network in Figure 6.3. Meaning the average optimization time is longest for a budget of around 15% to 25% depending on the network. For budgets larger than 25% we observe decreasing or nearly constant optimization times on most transmission networks. This contradicts the Hypothesis 6, in which we expected the optimization time to generally increase with increasing budget. We think this behavior correlates with the value of criticality that can be cured for this budget. For all test networks the utmost amount of criticality can be cured with a budget of around 20% of the total candidate costs. This can be seen for example in Figure 6.1 for the Slovak transmission network.

To get a better overview on the optimization process by Gurobi, Figures 6.4 and 6.5 show the objective value and the gap for the optimization process on the Austrian transmission network. The normalized gap of the of the optimization process is calculated as $\text{gap}(t) = 1 - \frac{\text{lb}(t)}{\text{obj}(t)}$, with $\text{lb}(t)$ as lower bound at time t and $\text{obj}(t)$ as the objective value at time t. We normalized the objective value by dividing the objective value by the initial criticality for this snapshot. The objective value is drawn as solid line while the gap is drawn as dashed line. The results for the model MMDC_SET are marked blue and the results for the model MMDC_RECALC are marked orange. The normalized objective value for zero seconds is different for both models and not 1 as both models start the optimization with a random initialization of their variables. We display both figures as we normalized the optimization time in Figure 6.5 to focus on the objective value in comparison for both models. In Figure 6.4 we do not normalize the optimization time, to focus on the differences in optimization time of both models. The figures show the optimization process of the snapshot on the 1st January 2013 at 17:00 of the Austrian transmission network with a budget of 20%. We decided to use the Austrian transmission network because the differences between snapshots of this network are relatively small. Also the Austrian transmission network is a good example for all tested networks as the optimization times on this network are within the average of all networks for both models.

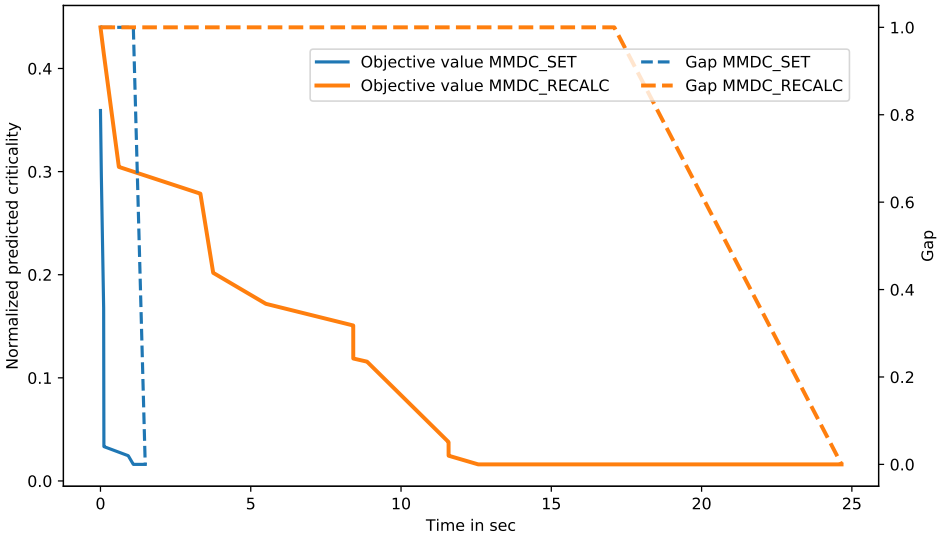


Figure 6.4.: Gurobi optimization process for the Austrian transmission network. The optimization process is for the 1st January 2013 at 17:00 with a budget of 20%. The solid lines represent the results for the normalized objective value and the dashed lines represent the normalized gaps relative to the optimization time. The results for MMDC_SET are marked blue and the results for MMDC_RECALC are marked orange. The optimization for model MMDC_SET finished after 1.6 sec, therefore the blue line ends at this time. The objective value of both models start at different points as the models are initialized randomly.

Figure 6.4 shows the optimization process for the 1st January 2013 at 17:00 of the Austrian transmission network. The figure shows the normalized objective value relative to the optimization time. We can see that the blue lines, representing the results for MMDC_SET, end at 1.6 sec. This is because the model MMDC_SET successfully finishes the optimization at this time. This can also be seen by the observation, that the gap becomes zero at 1.6 sec. The resulting objective value is ca. 0.04 as it is not possible to cure all criticalities for this snapshot with a budget of 20%. The optimization time for MMDC_RECALC

is 27.7 sec for this snapshot, as can be seen by observing the orange lines. The model MMDC_RECALC also finishes the optimization successfully with a gap of zero and a objective value of ca. 0.04.

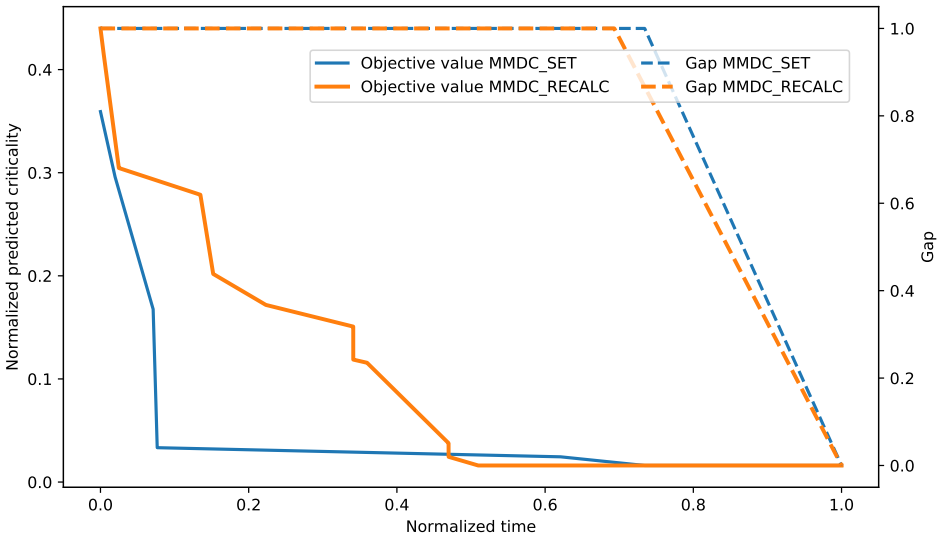


Figure 6.5.: Gurobi optimization process with normalized time for the Austrian transmission network. The optimization process is for the 1st January 2013 at 17:00 with budget of 20%. The solid lines represent the results for the normalized objective value and the dashed lines represent the normalized gaps relative to the normalized optimization time. The results belonging to MMDC_SET are marked blue and the results belonging to MMDC_RECALC are marked orange. The time for each model is normalized by dividing the measured time for each step by the total optimization time of the model.

If we normalize the optimization times, shown in Figure 6.4, by dividing the current optimization time by the total optimization time for each model, we obtain the results shown in Figure 6.5. In the Figure 6.5 we do not observe, a large difference between both models MMDC_SET and MMDC_RECALC, especially for the gaps, drawn as dashed lines. When comparing both objective values, drawn as solid lines, we observe that the optimization values are the same at the end of each optimization. On average over all tested snapshots both objective values decreased equivalently fast relative to the normalized time.

The optimization of the Austrian network shown in Figures 6.4 and 6.5 represents the average of the optimizations we observe on all networks we tested. For the Swiss transmission network however we observe extremely long optimization times for the model MMDC_RECALC for some snapshots for a budget of 20%. Figure 6.6 shows the optimization process of one of the snapshots of the Swiss transmission network with extensive computation times. We chose the snapshot of the Swiss transmission network on the 1st January 2013 at 10:00. The figure shows the results relative to the normalized time because the optimization time for the model MMDC_RECALC is 8857 sec while the model MMDC_SET optimization time is 0.1 sec on the same snapshot. The results for MMDC_SET are drawn as blue lines and the results of MMDC_RECALC are drawn as orange lines. The lines showing the gap are dashed, while the lines showing the normalized objective value are solid. We observe in Figure 6.6 that the optimal objective value, especially for MMDC_RECALC, is reached early on in the optimization process and most of the optimization time is spend to adjust the minimum objective bound. we only observe these extensive optimization times for MMDC_RECALC in comparison to the optimization times for MMDC_SET we on the Swiss transmission network and only for a budget around

20% for the Swiss network. It seems to be caused by a specific combination of network topology, including load distribution, budget and candidate network. We can neither observe the same behavior in this magnitude for all snapshots of the Swiss transmission network nor for different budgets for the same snapshots.

This concludes our observations for the models MMDC_SET and MMDC_RECALC for the `Budget` test series. In the following paragraph we discuss the hypotheses on the `Budget` test series, which are 8 and 5. As well as the hypotheses we expect to be true for both test series. These Hypotheses are 1, 2 3 and 4.

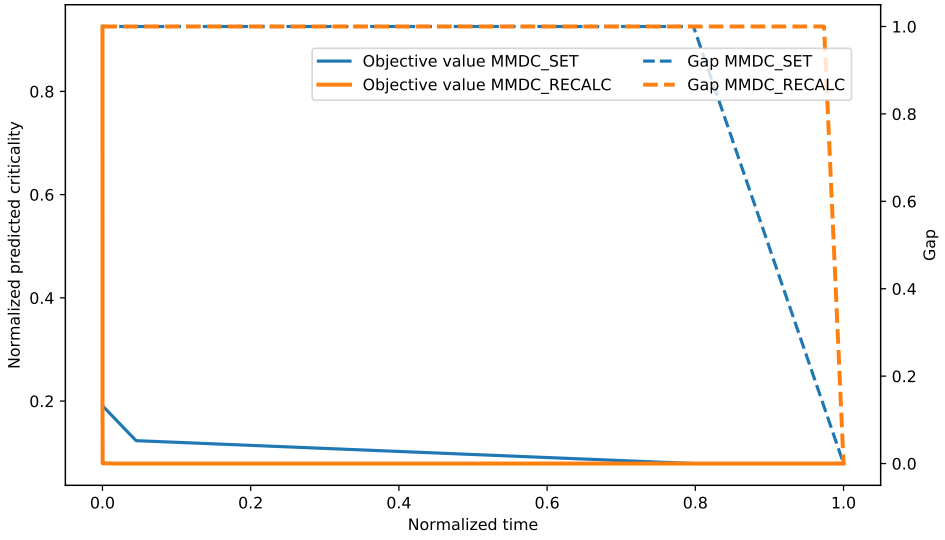


Figure 6.6.: Gurobi optimization process with normalized time for the Swiss transmission network. The optimization process is for the 1st January 2013 at 10 o'clock with a budget of 20%. The solid lines represent the normalize objective value and the dashed lines represent the normalized gap. The lines belonging to MMDC_SET are blue and the lines for MMDC_RECALC are orange. The final normalized objective value for both methods is 0.07 as not all criticalities can be cured.

We observe no prediction error of MMDC_RECALC on all tested networks in the `Candidate` test series. This supports Hypothesis 1.

For the `Candidate` test series we observe decreasing values of criticality with increasing number of candidate edges for both models and all networks. This supports Hypothesis 8. This behavior can be observed for example in Figure 6.7. In this figure we observe the value of criticality as blue solid line for MMDC_SET and as orange solid line for MMDC_RECALC. We observe for both objective values, that they decrease with increasing number of candidate edges.

We also observe increasing optimization and model generation times for increasing the number of candidate edges for both models, which supports Hypothesis 5. This behavior can be seen for all tested networks. One example for the increasing optimization time is depicted in Figure 6.8, by showing results for the Austrian transmission network. The optimization times are drawn as solid lines and the model generation times are drawn as dashed lines. The lines belonging to model MMDC_SET are blue while the lines for MMDC_RECALC are orange. For the optimization and model generation times we observe for all networks, that the time for the model MMDC_RECALC is higher than for the the model MMDC_SET, which supports Hypothesis 4.

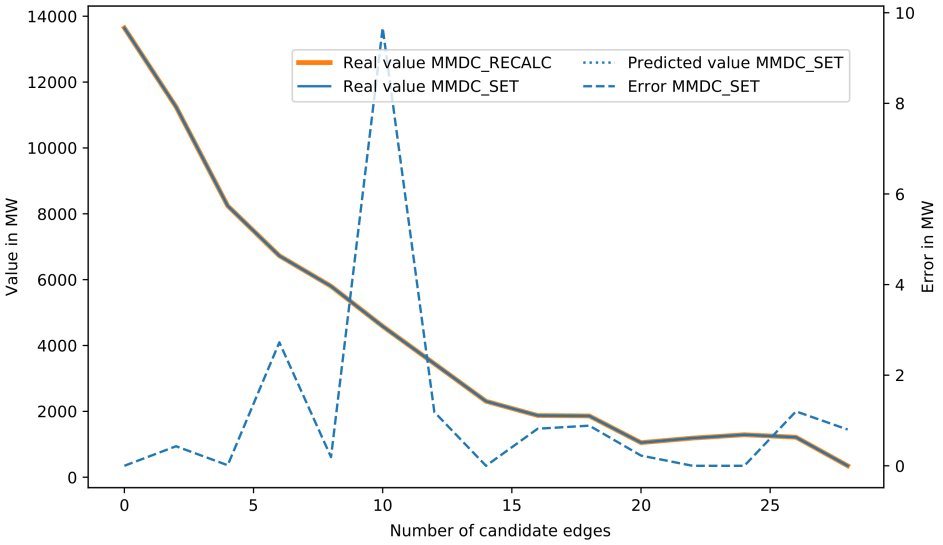


Figure 6.7.: Average real value of criticality for the models MMDC_SET and MMDC_RECALC for the Austrian transmission network as well as the prediction error and predicted value of criticality for MMDC_SET. The figure shows the results for the `Candidate` test series. The real values of criticality are drawn as solid lines. The prediction error is drawn as dashed line and the predicted value of criticality is drawn as dotted line. All lines concerning the model MMDC_SET are blue and all lines concerning MMDC_RECALC are orange.

For the `Candidate` test series with fixed budget and variable number of candidate edges we observe in total 8 networks where the first model version had some kind of prediction error. But for all those networks except the Slovak, Bulgarian and Romanian, for which we already observe a prediction error in the `Budget` test series, the prediction errors are smaller than 10 MW and are most likely caused by the integer feasibility tolerance of Gurobi. This tolerance is per default set to $1e^{-5}$ meaning Gurobi can insert an edge for 99.99% or 100.01% even though we defined the insertion as binary variable. Considering the amount of criticality for those networks, for example for the Dutch network with a initial criticality of 4531 MW we observe the maximum prediction error for MMDC_SET to be 2.5 MW, this is about 0.06% of the original criticality and can therefore be neglected. Thus, we conclude that the test data of the `Candidate` test series also contradicts Hypothesis 2. Except for the prediction error the values of criticality are almost the same for all network for both models. An example for this behavior is shown in the Figure 6.7 in which the result for the Austrian transmission network are depicted. The real values of criticality are drawn as solid lines and the prediction error is drawn as dashed line. The results for MMDC_SET are drawn in blue and for MMDCRECALC in orange. We see in this figure the maximal average prediction error for ten candidate edges, which is 9.8 MW. The initial value of criticality for the Austrian transmission network can be seen for the real value for zero candidate edges, it is approximately 13,986 MW. Therefore the maximal average error on the Austrian transmission network is less than 0.08%.

To provide an overview over our result for this subsection, we summarize the results of all networks in the test data set in Table 6.1. In this table we have one entry for the combination of each hypothesis and each network in which we enter whether the hypothesis is correct for this network (✓) or not (**x**). Since we did not test the model MMDC_RECALC on all test networks, because of extensive computation times, we marked networks on which we did not test an hypothesis with **-**. For general hypothesis not specifying the test series we

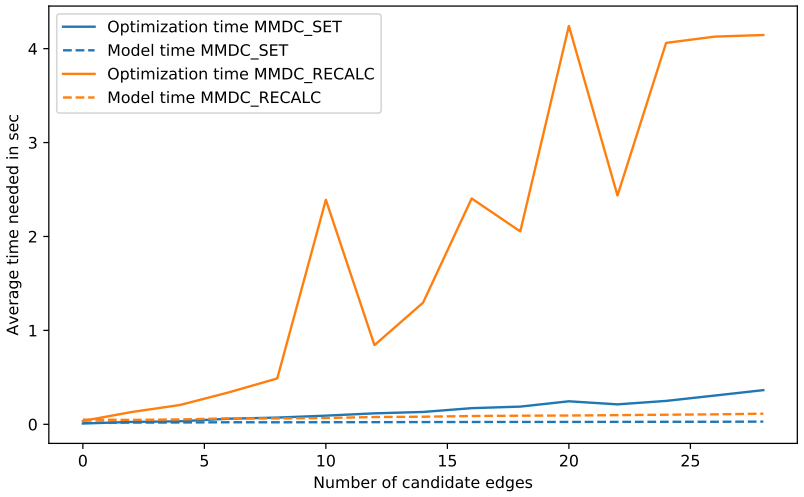


Figure 6.8.: Average computation times for the Austrian transmission network and the models MMDC_SET and MMDC_RECALC for the `Candidate` test series. The optimization times are drawn as solid lines and the model generation times are drawn as dashed lines. The lines for MMDC_SET are blue and the lines for MMDC_RECALC are orange.

write the results as (`Budget`,`Candidate`) to differ between both test series. When looking at the results we can see that Hypothesis 2 is not supported by our test data for the `Budget` test series. This is because we could not observe any prediction error for MMDC_SET on most networks for the `Budget` test series. For Hypothesis 3 we do not have a lot of data. The Hypothesis 3 was that we expect an lower predicted value of criticality for the model MMDC_SET than the actual value. For most of our test data in the `Budget` test series we could not observe a difference between two values, therefore we can not draw any conclusion concerning the hypothesis.

The most significant conclusions of this comparison are, that the model MMDC_RECALC is more precise than the model MMDC_SET. But the model MMDC_RECALC has also higher optimization and model building times. For application where time is not an essential factor the model MMDC_RECALC can therefore be used. For interactive or time critical applications we would recommend using the model MMDC_SET. The model MMDC_SET has significantly lower optimization and model building times, while providing good results and having a low prediction error.

We use the computational more efficient model MMDC_SET as the reference optimization model for the other optimization methods, because it solves the problem TNEP-CCE as modeled in Section 5.2.

6.2.2. Objective Minimizing the Sum of Criticalities versus Objective Minimizing Number of Critical Edges

In this subsection we compare the model MMDC_SET with the model MMDC_NUM. The model MMDC_SET has the objective to the minimize the sum of criticalities. The model MMDC_NUM has the objective to minimize the number of critical edges. Both variants, we compare in this subsection, have a fixed set of critical edges as input.

We evaluate the networks and discuss the following hypothesis we have for the comparison of the model MMDC_SET and MMDC_NUM. MMDC_SET and MMDC_NUM have

Table 6.1.: This table summarizes in which networks which Hypothesis from Subsection 6.2.1 are correct for the different test series. Since we did not test the model MMDC_RECALC on all networks we use the symbol - to indicate, that we did not test the model. The entries for the networks on which we did not test for the MMDC_RECALC model are for the model MMDC_SET. If a hypothesis is supported by the test data we w use the symbol ✓. If the test data opposes a hypothesis we use the symbol **x**. If we cannot draw any conclusion concerning the hypothesis from the test data we use the symbol ∼. We do not include the Irish and Slovenian transmission network as both have no critical edges for all testes snapshots. For general hypothesis not specifying the test series we write the results as (`Budget`,`Candidate`) to differ between both test series.

country	H. 1	H. 2	H.3	H. 4	H. 5	H. 6	H. 7	H. 8
Austria	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)	✓	x	✓	✓
Belgium	(✓|✓)	(x|✓)	(∼|✓)	(✓|✓)	✓	x	✓	✓
Bulgaria	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)	✓	✓	✓	✓
Croatia	(✓|✓)	(x|x)	(∼|∼)	(✓|✓)	✓	x	✓	✓
Czech Republic	-	(x|✓)	(∼|✓)	-	✓	x	✓	✓
Denmark	(✓|✓)	(x|x)	(∼|∼)	(✓|✓)	✓	x	✓	✓
Hungary	-	(x|✓)	(∼|✓)	-	✓	✓	✓	✓
Netherlands	-	(x|✓)	(∼|✓)	-	✓	x	✓	✓
Norway	-	(x|x)	(∼|∼)	-	✓	x	✓	✓
Poland	-	(x|✓)	(∼|✓)	-	✓	x	✓	✓
Portugal	(✓|✓)	(x|x)	(∼|∼)	(✓|✓)	✓	x	✓	✓
Romania	-	(x|✓)	(∼|✓)	-	✓	✓	✓	✓
Slovakia	(✓|✓)	(x|✓)	(∼|✓)	(✓|✓)	✓	✓	✓	✓
Sweden	-	(x|x)	(∼|∼)	-	✓	x	✓	✓
Switzerland	(✓|✓)	(x|✓)	(∼|✓)	(✓|✓)	✓	x	✓	✓

different objectives. MMDC_SET optimizes the sum of criticalities this leads to Hypothesis 9.

Hypothesis 9. *We expect when optimizing the network using the model MMDC_SET to have an overall smaller sum of criticalities after the optimization than when using the model MMDC_NUM.*

Hypothesis 10. *Analogously MMDC_NUM optimizes the number of critical edges, which lead to Hypothesis 10. We expect that the number of critical edges is smaller when using the model MMDC_NUM.*

Because both mathematical models are identical except for the objective, we expect Hypothesis 11.

Hypothesis 11. *Considering the model generating times we do not expect to see large differences.*

Because for MMDC_SET adding an candidate edge as more often an immediate effect on the objective value than for the model MMDC_NUM, we think Hypothesis 12 is correct.

Hypothesis 12. *We expect the model MMDC_SET to have a better optimization time because adding an candidate edge as more often an immediate effect on the objective value than for the model MMDC_NUM.*

Because more edges can be added for increasing budgets and increasing number of candidate edges provide more possibilities to choose from, we formulate Hypothesis 13.

Hypothesis 13. *We expect decreasing value of criticality with increasing number of critical edges or increasing budget for both models.*

Because adding candidate edges will increase the model size and increasing the budget will allow more combination of edges, we formulate Hypothesis 14.

Hypothesis 14. *We expect increasing optimization times with increasing number of critical edges or increasing budget for both models.*

We observe for both test series `Candidate` and `Budget` and both models the behavior same behavior of both models. The only exception is Hypothesis 14. Like in the previous section we observe for the `Budget` test series on most networks an increase in optimization time to a budget of 15% to 25% and a decrease in optimization time when further increasing the budget. For the `Candidate` test series Hypothesis 14 is supported by the results for all transmission network. Because all other hypotheses behave the same for both test series we do not distinguish between the result of both test series in this Section.

We observe for all networks a decrease in criticality for increasing budget and increasing number of candidate edges. This observation supports Hypothesis 13.

When comparing the results of both models we observe that the value of criticality for the model MMDC_SET is nearly always lower than the real value of criticality for MMDC_NUM. Therefore most of the test data supports Hypothesis 9. Interestingly we observe for the Slovak transmission network, that the value of criticality is lower for the model MMDC_NUM, optimizing the number of critical edges. The Slovak transmission network is the only network we observe a lower real value of criticality for MMDC_NUM. As discussed in Subsection 6.2.1, we observe for the Slovak transmission network that previously not critical edges become critical in the optimization process for MMDC_SET. Apparently this is not the case for the model MMDC_NUM even though we have the same list of critical edges as input and do not recalculate the list during the optimization. This is most likely caused by the different objectives and the resulting choices in candidate edges. We also observe that the number of critical edges is the same for both models or lower for the model MMDC_NUM. This supports the Hypothesis 10. The results for optimizing the Austrian transmission network can be seen in Figure 6.9. In this Figure we display the number of critical edges for both models as well as the value of criticality for the `Budget` test series on the Austrian transmission network. The real value of criticalities are drawn as solid lines while the number of critical edges is drawn as dotted lines. The results for MMDC_SET are drawn in blue and the results for MMDC_NUM are drawn in orange. For this example we can see, that the number of critical edges is clearly lower for the model MMDC_NUM. Also we observe that the real value of criticality is lower for the model MMDC_SET.

We also observe that the model MMDC_NUM using the objective to minimize the number of critical edges has on average higher computation times. This supports Hypothesis 12. In fact for the Swiss transmission network we observe for one snapshot with a budget of 15% of the total costs of candidate edges an optimization time over ten hours. The model MMDC_SET had an optimization time under two seconds for the same snapshot. We do not observe such dramatic differences in optimization time for all networks or even all snapshots of the same network. But we see for all networks in our test data set an increase in computation time for the model MMDC_NUM in comparison to the model MMDC_SET. A good example for the increase in optimization time is shown in Figure

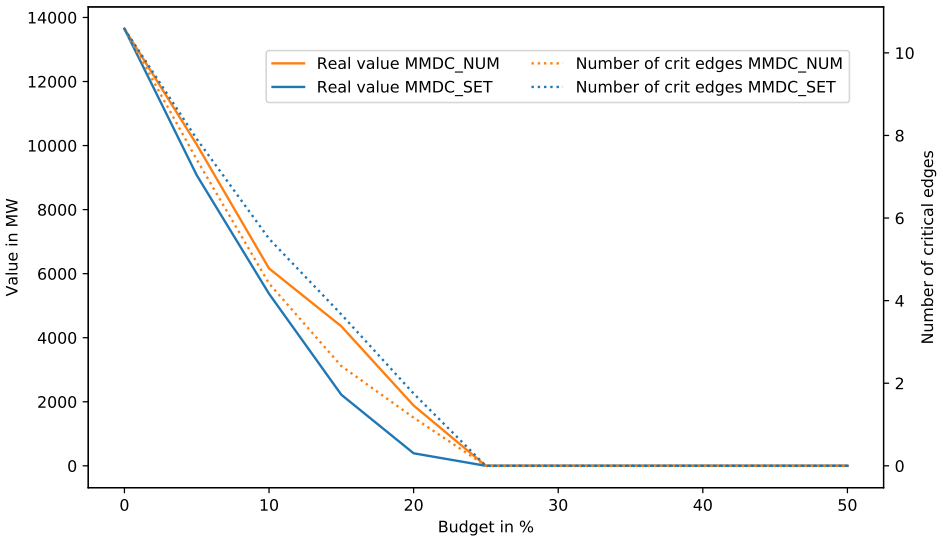


Figure 6.9.: Average value of criticality and average number of critical edges for the Austrian transmission network and the models MMDC_SET and MMDC_NUM for the `Budget` test series.. The real value of criticalities are drawn as solid lines while the number of critical edges is drawn as dotted lines. The results for MMDC_SET are drawn in blue and the results for MMDC_NUM are drawn in orange.

6.10. In Figure 6.10 the optimization and model generation times for both models for the Austrian transmission network are depicted. In this figure we display the optimization times as solid lines and the model generation times as dashed lines. The times for model MMDC_SET are drawn in blue and the results for MMDC_NUM are drawn in orange. In the figure we observe for all budgets a longer optimization and model generation time for MMDC_NUM. It is quite interesting that the model generation time is also higher for the model MMDC_NUM even though the models are nearly the same. We observe the higher model generation time for MMDC_NUM in comparison to the model generation time for MMDC_SET not only for this example but for all tested networks. This observation contradict Hypothesis 11.

To provide an overview over our result for this subsection, we summarize the results of all networks in the test data set in Table 6.2. In this table we have one entry for the combination of each hypothesis and each network in which we enter whether the hypothesis is correct for this network (✓) or not (**x**). Since we did not test the model MMDC_NUM on all test networks, because of extensive computation times, we marked networks on which we did not test an hypothesis with **-**. For general hypothesis not specifying the test series we write the results as (`Budget`,`Candidate`) to differ between both test series. We can see in the table that most of our hypotheses were correct. Only Hypotheses 11 and 14 for the `Budget` test series are not supported by our test data.

When excluding the Hypothesis 11, where we suspect the same model generation times, and the first part of Hypothesis 14, in which we suspect increasing optimization times for increasing budgets, the results of this evaluation are as expected. While the model MMDC_SET minimizing the sum of criticalities showed over all a lower sum of criticalities we observe a lower number of critical edges for the model MMDC_NUM. Because of the sometimes significantly higher computation times of the model MMDC_NUM we prefer for this thesis the version with the objective to minimize the sum of criticalities.

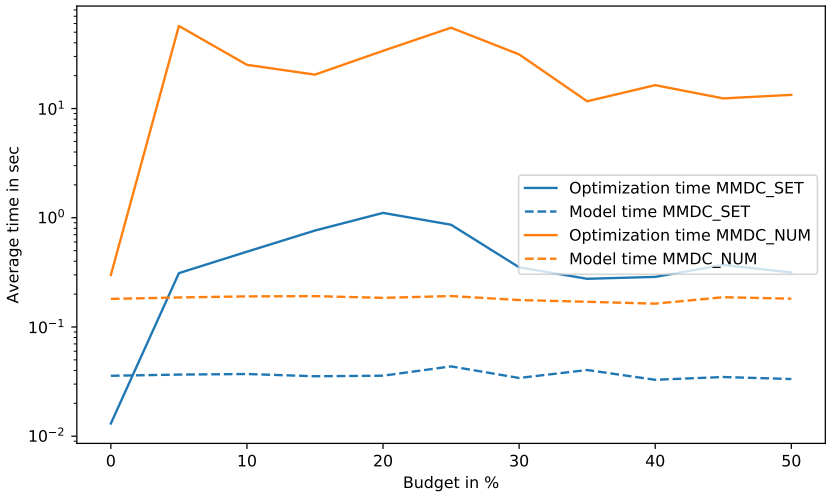


Figure 6.10.: Average computation times for the Austrian transmission network and the models MMDC_SET and MMDC_NUM for the `Budget` test series. In this figure we display the optimization times as solid lines and the model generation times as dashed lines. The times for model MMDC_SET are drawn in blue and the results for MMDC_NUM are drawn in orange.

6.3. Heuristic and Modifications

In this section we review the original heuristic proposed in Section 5.5.3.1 in comparison the modifications proposed in Section 5.5.4. As modifications we propose in Section 5.5.4 the introduction of a lookup table to avoid multiple computations of the same value and the approximation of the value by using a fixed value for each candidate edge. For short we refer to the original heuristic as HEU. To the modification of the heuristic using a lookup table we refer to as HEU_LU and to the version with value approximation we refer to as HEU_APPROX. First we compare the computation times of these three variations and then we compare the quality of their results.

6.3.1. Computation Time

Like in the previous section we also measured the model generation time as well as the optimization time of the model. For all versions of the heuristic the model generation is exactly the same. During the model generation, we generate a graph object of the network using the python library graph-tools [dPP27]. Because we do not compute anything on the graph and the largest of our networks just has 48 nodes and 84 edges we expect to have really small model generation times for all networks.

For the optimization time we have the following hypothesis. We designed the modifications especially to speed up HEU, thus we expect to see results supporting Hypothesis 15.

Hypothesis 15. *We expect for the computation times that the original heuristic HEU has the longest optimization time. We expect a speed up by using the look up tables in the version HEU_LU and an even larger speed up when using the value approximation in HEU_APPROX.*

Because increasing budgets and increasing number of candidate lines make the problem more complex we expect to see results supporting the Hypotheses 16 and 17.

Table 6.2.: This table summarizes in which networks which Hypothesis from Subsection 6.2.2 are correct for the different test series. Since we did not test the model MMDC_NUM on all networks we use the symbol - to indicate, that we did not test the model. The entries for the networks on which we did not test for the MMDC_NUM model are for the model MMDC_SET. If a hypothesis is supported by the test data we use the symbol ✓. If the test data opposes a hypothesis we use the symbol **x**. If we cannot draw any conclusion concerning the hypothesis from the test data we use the symbol ∼. We do not include the Irish and Slovenian transmission network as both have no critical edges for all testes snapshots. For general hypothesis not specifying the test series we write the results as (`Budget`,`Candidate`) to differ between both test series.

country	H. 9	H. 10	H.11	H. 12	H. 13	H. 14
Austria	(✓|✓)	(✓|✓)	x	✓	(✓|✓)	(x|✓)
Belgium	(✓|✓)	(✓|✓)	x	✓	(✓|✓)	(x|✓)
Bulgaria	(✓|✓)	(✓|✓)	x	✓	(✓|✓)	(✓|✓)
Croatia	(✓|✓)	(✓|✓)	x	✓	(✓|✓)	(x|✓)
Czech Republic	-	-	-	-	(✓|✓)	(x|✓)
Denmark	(✓|✓)	(✓|✓)	x	✓	(✓|✓)	(x|✓)
Hungary	-	-	-	-	(✓|✓)	(✓|✓)
Netherlands	-	-	-	-	(✓|✓)	(x|✓)
Norway	-	-	-	-	(✓|✓)	(x|✓)
Poland	-	-	-	-	(✓|✓)	(x|✓)
Portugal	(✓|✓)	(✓|✓)	x	✓	(✓|✓)	(x|✓)
Romania	-	-	-	-	(✓|✓)	(✓|✓)
Slovakia	(✓|✓)	(✓|✓)	x	✓	(✓|✓)	(✓|✓)
Sweden	-	-	-	-	(✓|✓)	(x|✓)
Switzerland	(✓|-)	(✓|-)	x	✓	(✓|✓)	(x|✓)

Hypothesis 16. *We expect for the `Budget` test series to see increasing computation times with increasing budget for all models but especially for the heuristic HEU. We expect this because the number of maximum s,t-flow computation is for this model proportional to the budget.*

Hypothesis 17. *We expect for the `Candidate` test series to see increasing computation times with increasing number of candidate edges for all models.*

We measure the model generation only for the original heuristic, because the model generation is exactly the same for the modifications HEU_LU and HEU_APPROX. We measure model generation times between 0.009 and 0.02 sec for all our networks. These model generation times are quite small and can be neglected in comparison to the optimization times for all versions of the heuristic. As the model generation for the heuristic is the generation of an graph object including the original edges and nodes, neither the budget nor the candidate edges are relevant for the model generation. Thus it is reasonable that we observe that both variables have no impact on the model generation times.

The results for the optimization time for the Swiss transmission network for the `Budget` test series and the `Candidate` test series can be seen in Figure 6.11. These results for the network are representative for all networks in the test data set. In the figures we can see the optimization times for the original Heuristic HEU in light green. The optimization times for HEU_APPROX are depicted in a darker green and the optimization times for HEU_APPROX are depicted in orange in both figures. As expected in the Hypothesis 15

we can observe significant speed ups for both modifications HEU_LU and HEU_APPROX. We also observe that the modification HEU_APPROX is always significantly faster than the modification HEU_LU for both test series. For all networks we observe an increase in optimization time for increasing budget and increasing number of candidate edges. These observation support our Hypothesis 16 and 17.

The speed up for the modification HEU_APPROX can be easily explained. The majority of the optimization time in the algorithm is caused by the computation of maximum s,t-flows. In each optimization step in the original heuristic HEU there are $|E_{\text{crit}}|$ maximum s,t-flows computed, to calculate the criticalities of the critical edges E_{crit} if a specific set of candidate edges is added to the network. For both the original heuristic HEU and the modification with look up HEU_LU table the number of optimization steps increases proportionally to the budget as well as to the number of candidate edges. This is because the heuristics are based on the idea of dynamic programming and we use an array of size budget $\times$ $|E_{\text{cand}}|$. For each optimization step we fill one array entry by computing $|E_{\text{crit}}|$ maximum s,t-flows for the heuristic HEU and also for HEU_LU if the value is not in the look up table. This also implies that the number of computed maximum s,t-flows increases proportionally to both the budget and the number of candidate edges. The modification HEU_APPROX with value approximation just computes one value for each candidate edge and is not depending on the budget considering the number of computed maximum s,t-flows. We also do not see an dramatic increase in optimization time while increasing the number of candidate edges, as for each candidate edge there are just $|E_{\text{crit}}|$ maximum s,t-flows computed in the modification HEU_APPROX. For the other version of the heuristic we compute the $|E_{\text{crit}}|$ maximum s,t-flows for edge combinations in each optimization step and by adding one candidate edge we add |budget| optimization steps.

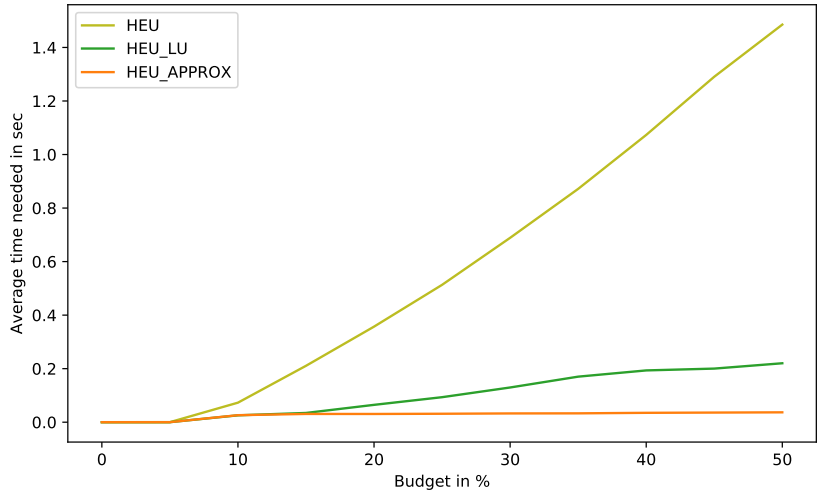
The speed up for the modification HEU_LU depends on the used test data. For all of our tested networks and generated candidate networks we can see a significant speed up. Especially for large budgets and candidate networks we observe speed ups larger than 90%. Looking at the numbers of look ups versus value computations we can see for example that for the Swiss transmission network for the snapshot at 0:00 at the 1st January 2013 with a budget of 50 % of the total cost for the set of candidate edges we have in total 4936 calls of the value function, from which 4604 can be handle through look ups and the remaining 332 calls are value computations. Meaning only 7% of the value calls have to be computed for this example in comparison to the original algorithm. For the same example the computation times were 2.90 sec for the original heuristic and 0.23 sec for the heuristic with look up table.

6.3.2. Result Quality

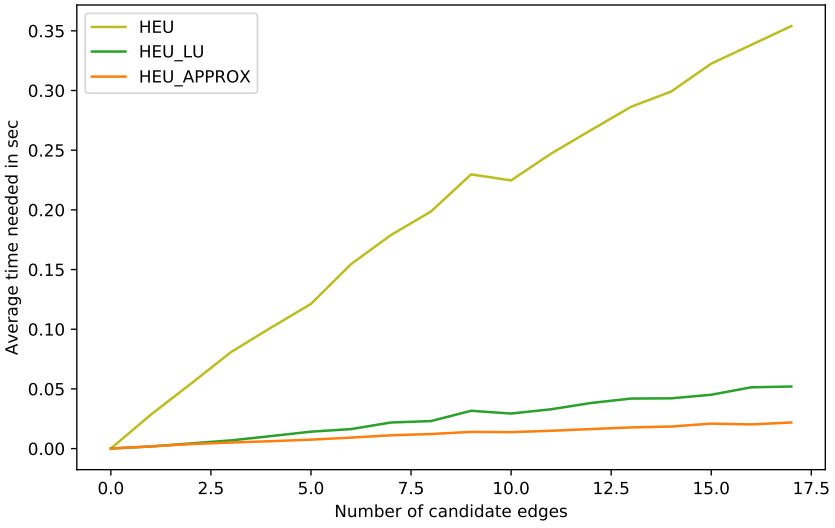
As already pointed out in Section 5.5.4 the modification using the lookup table HEU_LU has the same results as the original algorithm HEU. The reason for this is, that the lookup table does not change the value of an edge combination. Therefore we only compare the values for the proposed subset of candidate edges for the original heuristic versus the modification with value approximation.

We look into two different aspects of the values. One the one hand we are interested in the real value of criticality after optimizing the network with each heuristic, especially in comparison with the reference model MMDC_SET. The other aspect we want to discuss is the prediction error of each heuristic.

HEU_APPROX uses a value approximation to optimize the network. Because of this approximation we expect the results of HEU_APPROX to be worse than for HEU_LU this leads to the Hypotheses 18 and 19.



(a) Average computation time for the three algorithm versions proposed for the Swiss transmission network. For the `Budget` test series. The optimization time for HEU is depicted in light green. The optimization time for HEU_LU is depicted in a darker green and the optimization time for HEU_APPROX is depicted in orange.



(b) Average computation time for the three algorithm versions proposed for the Swiss transmission network. For the `Candidate` test series. The optimization time for HEU is depicted in light green. The optimization time for HEU_LU is depicted in a darker green and the optimization time for HEU_APPROX is depicted in orange.

Figure 6.11.: Average computation time for the three algorithm versions for two different networks

Hypothesis 18. *We expect the heuristic HEU_LU to cure less criticality than the heuristic HEU_APPROX.*

Hypothesis 19. *We expect a higher prediction error for the heuristic HEU_APPROX than for the Heuristic HEU_APPROX.*

Because both models HEU_LU and HEU_APPROX are heuristics and do not recalculate the power flow we expect Hypothesis 20.

Hypothesis 20. *We expect both heuristics to perform worse in both aspects than the reference model MMDC_SET.*

We do not observe any remarkable difference for the values of criticality or the prediction error between the test series `Budget` and `Candidate`.

For all networks we observe the network optimized with the reference model MMDC_SET to have a lower sum of criticality than the networks optimized using any heuristic. This supports the Hypothesis 20.

Interestingly we observe for some evaluated networks, that the values of criticality for the heuristic HEU_APPROX and the heuristic HEU_LU are nearly identical. This can be seen for example in Figure 6.12. This Figure shows the real values of criticality and the prediction error as the result of optimizing the Slovak transmission network with the heuristics HEU_APPROX and HEU_LU as well as with the reference model MMDC_SET. The real values of criticality are drawn as solid lines in the figure. The prediction errors are drawn as dashed lines. The results for the the reference model MMDC_SET are drawn in blue. The results for HEU_LU are drawn in green and the results for HEU_APPROX are drawn in orange. In this figure it can be seen that the values of both heuristics diverge just for the last budget of 50% for all other budgets the values were the same. We observe nearly identical values of criticalities for 7 of the 17 networks of the test data set. As nearly identical we hereby identify differences of both values of less than 0.5% of the initial value of criticality. This observation is contrary to what we expected in Hypotheses 18 and 19.

For other of the evaluated networks we can see divergence between the two heuristics HEU_LU and HEU_APPROX. As an example the results for the Austrian transmission network are shown in Figure 6.13 for the `Candidate` and in Figure 6.13 for the `Budget` test series. In both plots we display the real values of criticality as well as the prediction errors for both heuristics HEU_LU and HEU_APPROX and the reference model MMDC_SET. The real values of criticality are drawn as solid lines in both figures. The prediction errors are drawn as dashed lines. The results for the the reference model MMDC_SET are drawn in blue. The results for HEU_LU are drawn in green and the results for HEU_APPROX are drawn in orange. We can see a difference between the real values of criticality as well as the error of the heuristic HEU_LU and the heuristic with value approximation HEU_APPROX. We can see that the value of criticality is always smaller for the heuristic HEU_LU than for the heuristic with value approximation HEU_APPROX, which is the behavior we expected originally in Hypothesis 18.

Interestingly the networks we observe the values for both heuristics HEU_LU and HEU_APPROX to be identical are mostly the smaller networks in our test data set with the largest one being the Czech transmission network with 21 nodes and 35 edges. We exclude the Irish and Slovenian network here because both networks have no critical edges in all tested snapshots and of course all values for all optimization methods are the same for both networks. The smallest network the effect of diverging values between both heuristics occurs is the Danish network with 11 nodes and 11 edges.

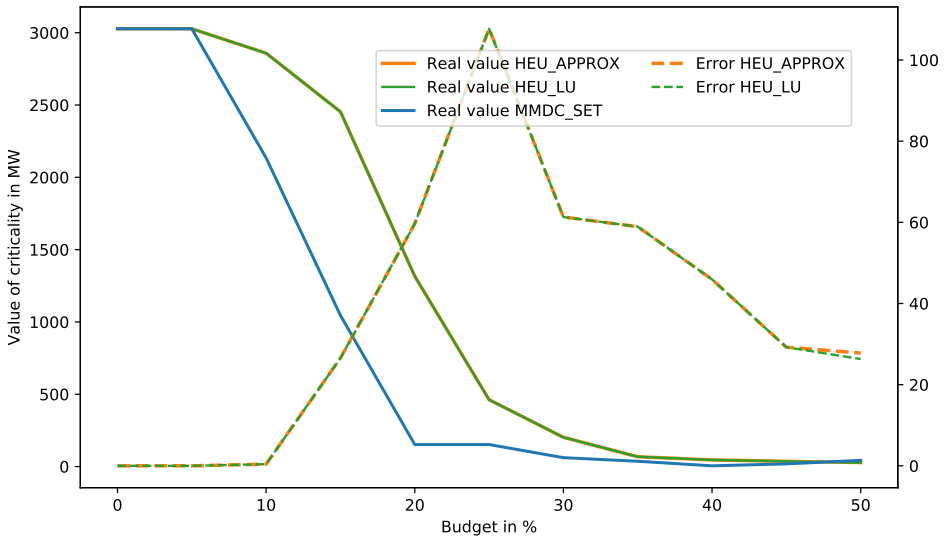


Figure 6.12.: Average value of criticality for the heuristics HEU_APPROX and HEU_LU in comparison with reference model MMDC_SET for optimizing the Slovak transmission network. In this figure the actual value of criticality as well as the prediction error are displayed for the `Budget` test series. The real values of criticality are drawn as solid lines in the figure. The prediction errors are drawn as dashed lines. The results for the the reference model MMDC_SET are drawn in blue. The results for HEU_LU are drawn in green and the results for HEU_APPROX are drawn in orange.

The effect of the both heuristics HEU_LU and HEU_APPROX performing the same depends strongly on the topology of the network and also on the number of candidate edges chosen by the algorithm. As explained in Section 5.5, in reality the values of candidate edges change when adding edges to the graph. Depending on the topology of the graph this can happen after adding the first candidate edge as shown in Section 5.5. But we assume it is more likely to happen if there are more critical edges and a lot of candidate edges are added to the network, as is the case for our larger test networks.

The prediction error of the heuristics HEU_LU and HEU_APPROX is the difference between the value of criticality the predict for the optimized network and the actual value of criticality of the optimized network. Interestingly we observe, that for all tested networks the predicted value of criticality was higher than the actual value of criticality. This means that both heuristics predict to perform worse than they actually do. We assume this is most likely caused by the redistribution of power flow when adding edges. Because we test our networks for a given power generation and consumption we have on average a lower power flow on all edges when adding edges to a network. Therefore it can occur that the power flow on a critical edge decreases after adding a candidate edge so that the criticality of the critical edge decreases.

On the networks the heuristics HEU_LU and HEU_APPROX perform identical we have of course the same error. In the other cases we can always observe a larger prediction error for the heuristic HEU_APPROX as we originally expected in Hypothesis 19. This can for example be seen in Figure 6.14 for the `Budget` and in Figure 6.13 for the `Candidate` test series. In these figures the prediction error for both heuristics are displayed for the Austrian transmission network as dashed lines. The lines for the Heuristic HEU_APPROX are orange and for HEU_LU the lines are green. In Figure 6.13 we can see that the prediction error is zero if the set of candidate edges is empty. For all other amounts

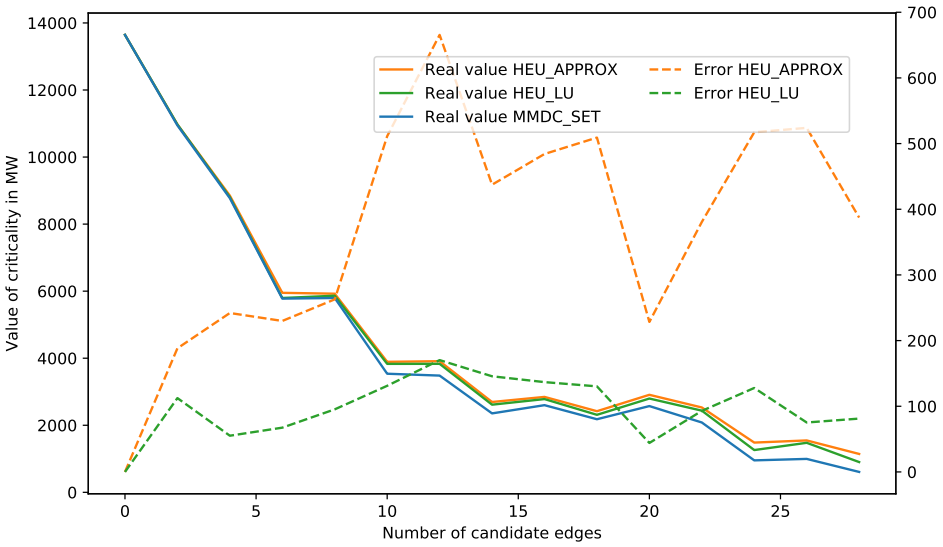


Figure 6.13.: Average value of criticality for the heuristics HEU_APPROX and HEU_LU in comparison with reference model MMDC_SET for optimizing the Austrian transmission network.In this figure the actual value of criticality as well as the prediction error are displayed for the `Candidate` test series. The real values of criticality are drawn as solid lines in the figure. The prediction errors are drawn as dashed lines. The results for the the reference model MMDC_SET are drawn in blue. The results for HEU_LU are drawn in green and the results for HEU_APPROX are drawn in orange.

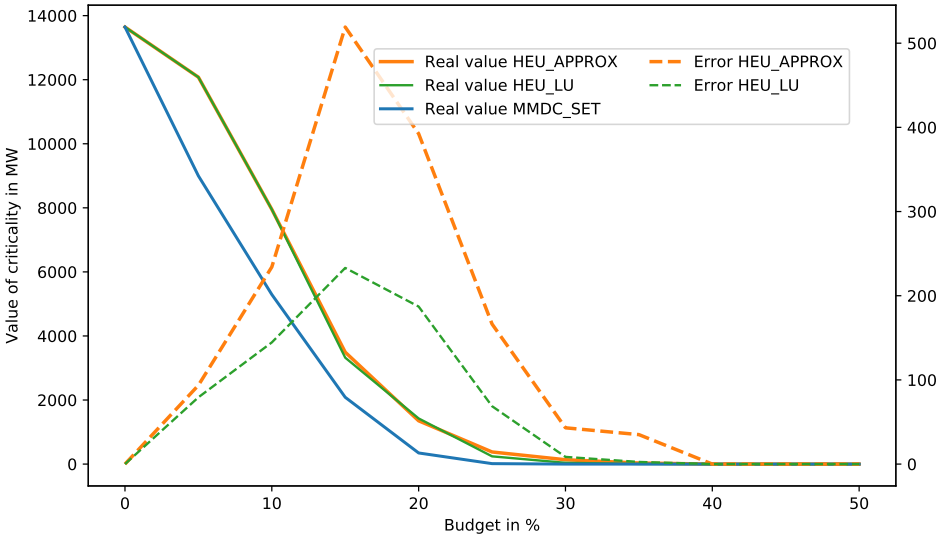


Figure 6.14.: Average value of criticality for the heuristics HEU_APPROX and HEU_LU in comparison with reference model MMDC_SET for optimizing the Austrian transmission network.In this figure the actual value of criticality as well as the prediction error are displayed for the `Budget` test series.The real values of criticality are drawn as solid lines in the figure. The prediction errors are drawn as dashed lines. The results for the the reference model MMDC_SET are drawn in blue. The results for HEU_LU are drawn in green and the results for HEU_APPROX are drawn in orange.

of candidate edges the prediction error is relatively high. This is because a Budget of 30% is not enough to cure the critical edges of the Austrian transmission network. If the budget would be high enough the error would decrease as can be seen when looking at the `Budget` test series. For the `Budget` test series the prediction error is zero for both heuristics if the budget is to small to add edges for all networks. As can be seen in Figure 6.14 the prediction error increases with increasing budget until most criticalities can be cured. If most criticalities are cured the prediction error decreases. For some networks the prediction error remains quite high even for larger budgets. One example for this is the Slovak transmission network. Figure 6.12 shows the prediction error for both heuristics for optimizing the Slovak transmission network as dashed lines. The lines for the Heuristic HEU_APPROX are orange and for HEU_LU the lines are green. As discussed in the previous Section, even our reference model MMDC_SET has some prediction error for the Slovak transmission network as previously not critical edges become critical during optimization. For both heuristics HEU_LU and HEU_APPROX we can observe the same effect.

We summarize in table 6.3 which Hypothesis where correct on different networks for the different test series. We summarize in the table all hypothesis concerning the heuristics HEU, HEU_LU and HEU_APPROX from Section 6.3. If a hypothesis is supported by the test data we use the symbol ✓. If the test data opposes a hypothesis we use the symbol **x**. If we cannot draw any conclusion concerning the hypothesis from the test data we use the symbol ∼. We do not include the Irish and Slovenian transmission network as both have no critical edges for all tested snapshots. For general hypothesis not specifying the test series we write the results as (`Budget`,`Candidate`) to differ between both test series. In the table we can see that most of our hypothesis are supported in the evaluation. Only for Hypothesis 18 the results are mixed. The hypothesis is that HEU_APPROX cures less criticality than HEU and HEU_LU. The heuristics performed equivalently on the networks for which the hypothesis is not supported.

6.4. Mathematical Model with Graph Flow

In this section we evaluate the reference model MMDC_SET and both heuristics HEU_LU and HEU_APPROX in comparison to the mathematical model using graph flow introduced in Section 5.3.2. For short we refer to the mathematical model using graph flow as MM_GRAPH.

The model MM_GRAPH models the optimal solution for the problem, if the power flow is not recalculated after the insertion of candidate edges. The model MM_GRAPH has the same input and uses the same information as the heuristic HEU_LU. Therefore MM_GRAPH computes the optimal solution the heuristics could achieve given the simplification of not recalculating the power flow when adding edges. First we discuss the model generation and optimization times of the model MM_GRAPH in comparison to the heuristics HEU_LU and HEU_APPROX and the reference model MMDC_SET. Then we discuss the value of criticality of the optimized network for all for model as well as the occurring prediction error.

6.4.1. Computation Times

We have the following hypothesis for this evaluation we expect the hypothesis for both test series `Budget` and `Candidate`. Model MM_GRAPH is a simplified model in comparison to the reference model MMDC_SET. Because of this we have less constraints and less variables, which leads to Hypothesis 21.

Table 6.3.: This table summarizes in which networks which Hypothesis from Section 6.3 are correct for the different test series. If a hypothesis is supported by the test data we use the symbol ✓. If the test data opposes a hypothesis we use the symbol **x**. If we cannot draw any conclusion concerning the hypothesis from the test data we use the symbol ∼. We do not include the Irish and Slovenian transmission network as both have no critical edges for all tested snapshots. For general hypothesis not specifying the test series we write the results as (`Budget`,`Candidate`) to differ between both test series.

country	H. 15	H. 16	H.17	H. 18	H. 19	H. 20
Austria	(✓|✓)	✓	✓	(✓|✓)	(✓|✓)	(✓|✓)
Belgium	(✓|✓)	✓	✓	(x|x)	∼	(✓|✓)
Bulgaria	(✓|✓)	✓	✓	(x|x)	∼	(✓|✓)
Croatia	(✓|✓)	✓	✓	(x|x)	∼	(✓|✓)
Czech Republic	(✓|✓)	✓	✓	(x|x)	∼	(✓|✓)
Denmark	(✓|✓)	✓	✓	(✓|✓)	(✓|✓)	(✓|✓)
Hungary	(✓|✓)	✓	✓	(✓|x)	∼	(✓|✓)
Netherlands	(✓|✓)	✓	✓	(✓|✓)	(✓|✓)	(✓|✓)
Norway	(✓|✓)	✓	✓	(✓|✓)	(✓|✓)	(✓|✓)
Poland	(✓|✓)	✓	✓	(✓|✓)	(✓|✓)	(✓|✓)
Portugal	(✓|✓)	✓	✓	(✓|✓)	(✓|✓)	(✓|✓)
Romania	(✓|✓)	✓	✓	(x|✓)	(∼|✓)	(✓|✓)
Slovakia	(✓|✓)	✓	✓	(✓|✓)	(x|✓)	(✓|✓)
Sweden	(✓|✓)	✓	✓	(✓|✓)	(✓|✓)	(✓|✓)
Switzerland	(✓|✓)	✓	✓	(x|x)	∼	(✓|✓)

Hypothesis 21. *Concerning the computation times we expect the model MM_GRAPH to have a shorter model generation and optimization time than the reference model MMDC_SET.*

We observe for the heuristic HEU_LU long optimization times. We therefore expect Hypothesis 22 to be true.

Hypothesis 22. *We expect the model MM_GRAPH to have shorter optimization times than HEU_LU.*

The heuristic HEU_APPROX has because of the value approximation less computations than HEU_LU. We know that HEU_APPROX is fast for optimizing the TNEP-CCE problem because of the approximations. We expect Hypothesis 23 to be true for this reason.

Hypothesis 23. *We expect the heuristic HEU_APPROX to have shorter optimization times than the model MM_GRAPH, especially on larger transmission networks.*

We know that the model generation times for the heuristic HEU_APPROX and the heuristic HEU_LU are really small and barely increase for larger networks. Therefore we add Hypothesis 24 to this list.

Hypothesis 24. *We expect the model MM_GRAPH to have larger model generation times. We expect this especially for larger networks and increasing number of candidate edges.*

We observe for all networks, that the model MM_GRAPH has a shorter model generation time than the reference model MMDC_SET as we expected in Hypothesis 21. In comparison

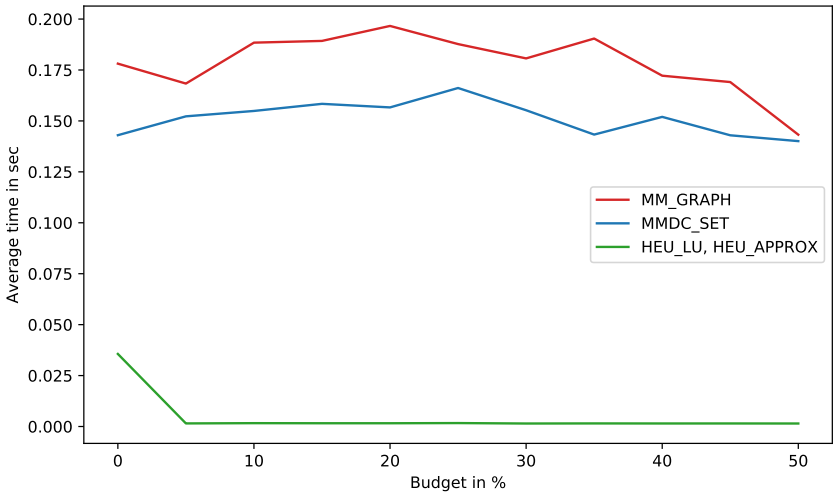


Figure 6.15.: Model generation times of all optimization methods for optimizing the Polish transmission network. The model generation time for MMDC_SET is drawn in blue. The model generation time for HEU_LU and HEU_APPROX is drawn in green and the time for MM_GRAPH is drawn in red.

to the model generation time of the heuristics the model MM_GRAPH has a longer model generation time on larger networks while being faster on small networks. This is some what contrary to what we expect in Hypothesis 24. In the hypothesis we expect the model generation time of the heuristics to be shorter on all networks. On small and medium sized networks the differences between the model generation times are about 0.01 sec or smaller and are compared to the differences in optimization times not that relevant. The model generation times the largest network can be seen in Figure 6.15. This figure shows the model generation times for the Polish transmission network. The model generation time for the model MMDC_SET is drawn as blue line, the generation time for the heuristic HEU_LU an HEU_APPROX are drawn in green and the model generation time for MM_GRAPH are drawn as red lines. We can see in Figure 6.15 clearly, that the model generation time for HEU_LU and HEU_APPROX is the fastest. We can also see that the model generation time for MMDC_SET is longer compared to the model generation time for MM_GRAPH.

For the optimization times we observe, that the heuristic HEU_LU and the reference model MMDC_SET perform generally worse than the heuristic HEU_APPROX and the model MM_GRAPH. These observation support the Hypotheses 21 and 22. Interestingly, the heuristic HEU_LU performs better than the reference model MMDC_SET for smaller budgets in the `Budget` test series. The observe the same behavior for the heuristic HEU_APPROX and the model MM_GRAPH. For smaller budgets the heuristic HEU_APPROX has a shorter optimization time on most networks than the model MM_GRAPH. This means Hypothesis 23 is supported for smaller budgets. For larger budgets we generally observe the model MM_GRAPH to have shorter optimization times. This observation contradicts Hypothesis 23 for larger budgets. We observe that the budget for which HEU_APPROX and MM_GRAPH have the same optimization time varies depending on the network size. For larger networks we observe higher budgets and for smaller networks lower budgets for which both models have the same optimization time. A good example for the optimization times can be seen in Figure 6.16. In this figure the result for optimizing the Polish transmission network can be seen. In the figure we see the optimization time for MM_GRAPH drawn in red, for MMDC_SET the optimization

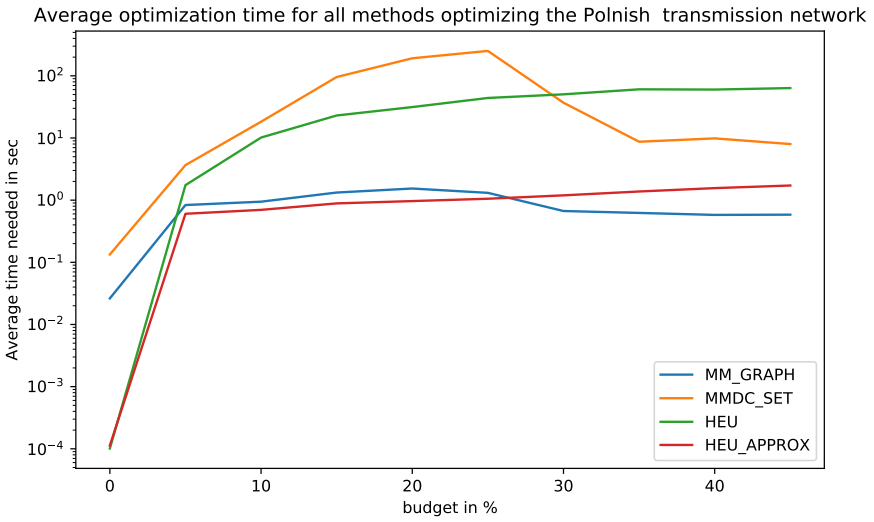


Figure 6.16.: Optimization times for the Polish transmission network for the `Budget` test series. The optimization time for MMDC_SET is drawn in blue, for MM_GRAPH the optimization time is drawn in red, for HEU_LU the optimization time is drawn in green and for HEU_APPROX the optimization time is drawn in orange.

time is drawn in blue, for HEU_LU the optimization time is drawn in green and for HEU_APPROX the optimization time is drawn in orange. We can see in the figure that the optimization time for HEU_APPROX is shorter than the one of MM_GRAPH up to a budget of 25%. We can see a similar behavior for HEU_LU and MMDC_SET in the figure. The optimization time of HEU_LU is shorter than the one of MMDC_SET up to a budget of 30%.

For the test series `Candidate` we see generally the same behavior for all optimization models, with the small difference that the computation times for all models steadily increase for an increasing number of candidate edges. This behavior can clearly be seen in Figure 6.17, which shows the optimization times for the `Candidate` test series on the Portuguese transmission network. In the figure the optimization time for MMDC_SET is drawn in blue, for MM_GRAPH the optimization time is drawn in red, for HEU_LU the optimization time is drawn in green and for HEU_APPROX the optimization time is drawn in orange. In this figure we observe a similar behavior, with the difference that the optimization time for MMDC_SET is longer for all numbers of candidate edges compared to the optimization times of the other models.

6.4.2. Result Quality

Considering the result quality we have the following hypothesis for both test series. Because MM_GRAPH as the same input data as both heuristics but solves the problem optimally for the given data. We expect the results for MM_GRAPH to be better or equally good. This is captured in Hypotheses 25, 26 and 27.

Hypothesis 25. *We expect the model MM_GRAPH to cure more criticality than both heuristics but less than the reference model MMDC_SET.*

Hypothesis 26. *For the prediction error we expect the prediction error of the heuristic HEU_LU and the model MM_GRAPH to be nearly the same.*

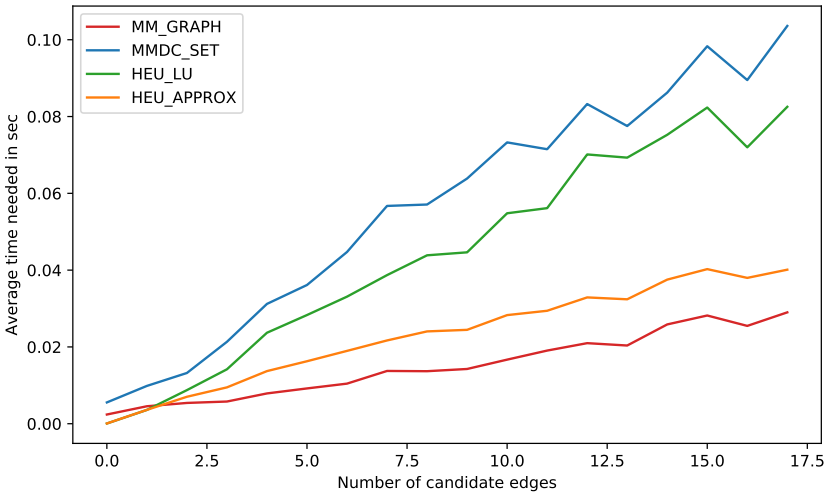


Figure 6.17.: Optimization times for the `Candidate` test series on the Portuguese transmission network. The optimization time for MMDC_SET is drawn in blue, for MM_GRAPH the optimization time is drawn in red, for HEU_LU the optimization time is drawn in green and for HEU_APPROX the optimization time is drawn in orange.

Hypothesis 27. *We expect heuristic HEU_APPROX to have over all the highest prediction error.*

Because the reference model models the DC power flow approximation, we expect Hypothesis 28 to be true.

Hypothesis 28. *We expect heuristic MMDC_SET to have over all the lowest prediction error.*

For both test series and all networks we can see the model MM_GRAPH curing more criticalities than both heuristics. This supports Hypothesis 25. Interestingly we do not see a lot of difference between the real value of criticality for the model MM_GRAPH and the reference model MMDC_SET. On average the difference between the real values of criticality for both models is less than one percent of the initial criticality on each network. This observation is not contrary to our Hypothesis 25 but we expected to see larger differences between the real value of criticality of the models MMDC_SET and MM_GRAPH. A good example for this behavior are the results for the `Budget` test series on the Swiss transmission network, shown in Figure 6.18. In the figure the real value for MMDC_SET is drawn as dashed blue line, the value for MM_GRAPH is drawn as solid red line, the value for HEU_LU is drawn as solid green line and the value for HEU_APPROX is drawn as dashed orange line. In the figure it can be seen that both models MM_GRAPH and the reference model MMDC_SET have relatively similar real values of criticality. Just around 10 and around 20% of total cost of the candidate edges we see small differences between the two models. For the Swiss transmission network we observe the largest difference of both models with about 43 MW for a budget of 20% with an average initial criticality of 4578 MW.

Interesting is, that the predicted value of criticality for the model MM_GRAPH is significantly higher than the real value of criticality for all networks. Which is captured in the prediction error of the model. We observe the higher the predicted value of criticality for

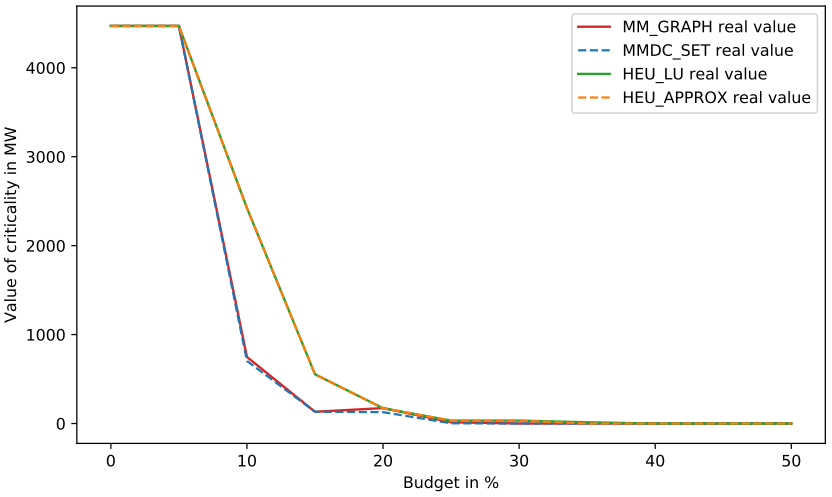


Figure 6.18.: Real value of criticalities for all optimization methods on the Swiss transmission network for the `Budget` test series. In this figure the real value for MMDC_SET is drawn as dashed blue line, the value for MM_GRAPH is drawn as solid red line, the value for HEU_LU is drawn as solid green line and the value for HEU_APPROX is drawn as dashed orange line.

the model MM_GRAPH for both test series. For the prediction errors both test series are quite interesting. For comparison of both test series Figure 6.19 shows the errors for the `Candidate` test series while Figure 6.20 shows the errors for the `Budget` test series. In both figures the error for MMDC_SET is drawn in blue, the error for MM_GRAPH is drawn in red, the error for HEU_LU is drawn in green and the error for HEU_APPROX is drawn in orange. For the `Candidate` test series we chose the budget to be fixed to 30% of the total cost of candidate edges. In this case is is not possible to cure all critical edges given this budget and the original set of candidate edges. Therefore it is expected that the error in Figure 6.19 does not reduce with increasing number of candidate edges.

For most of the networks from the test data set we see comparable results to the ones displayed in Figure 6.19 and Figure 6.20. Meaning that both series show errors of zero if no candidate edges can be added. For the `Budget` test series we have most of the time no error for budgets over 35% of total costs of all candidate edges. Also the reference model MMDC_SET performs best for all tested networks with having the lowest average prediction error. This behavior supports Hypothesis 28. On average over all networks the heuristic HEU_APPROX has the highest prediction error, which support Hypothesis 27. For the heuristic HEU_LU and the model MM_GRAPH we can not conclude that one outperforms the other considering the prediction error. While in Figure 6.20 the error of the model MM_GRAPH is on average larger than for the heuristic HEU_APPROX, we can for example observe the opposite for the Swedish transmission network in Figure 6.21. In Hypothesis 26 we expected the models MM_GRAPH and HEU_LU to have nearly the same prediction error. While this hypothesis is not supported when looking at some transmission networks, we think that on average over all transmission networks the prediction errors of both models are quite similar.

We summarize in the table 6.4 to conclusion for each hypothesis on each network for the different test series. If a hypothesis is supported by the test data we use the symbol ✓. If the test data opposes a hypothesis we use the symbol **x**. If we cannot draw any conclusion concerning the hypothesis from the test data we use the symbol ∼. We do not include

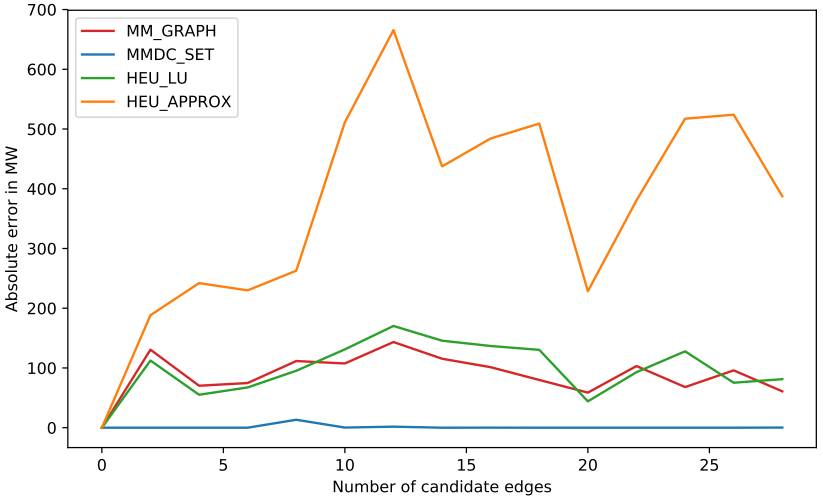


Figure 6.19.: Average absolute error for all optimization methods on the Austrian transmission network for the `Candidate` test series. For each algorithm one curve is displayed showing the average absolute error, meaning the difference between real value of criticality and the predicted one. In this figure the error for MMDC_SET is drawn in blue, the error for MM_GRAPH is drawn in red, the error for HEU_LU is drawn in green and the error for HEU_APPROX is drawn in orange.

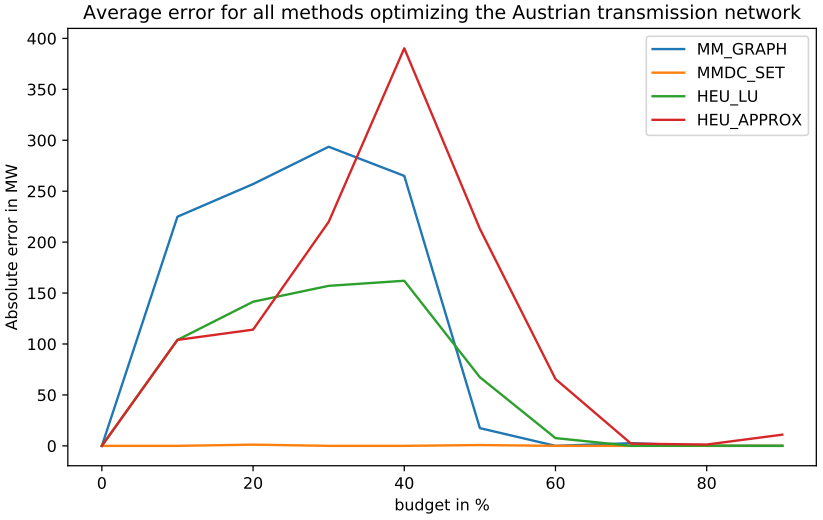


Figure 6.20.: Average absolute error for all optimization methods on the Austrian transmission network for the `Budget` test series. For each algorithm one curve is displayed showing the average absolute error, meaning the difference between real value of criticality and the predicted one. In this figure the error for MMDC_SET is drawn in blue, the error for MM_GRAPH is drawn in red, the error for HEU_LU is drawn in green and the error for HEU_APPROX is drawn in orange.

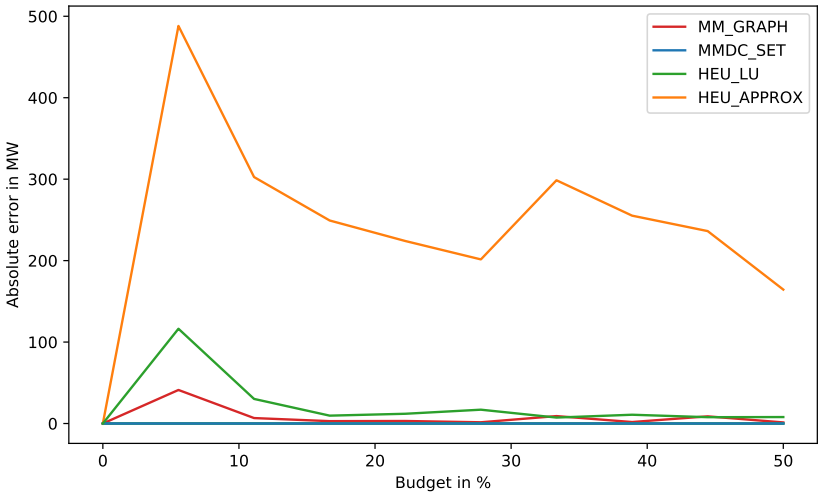


Figure 6.21.: Average absolute error for all optimization methods on the Swedish transmission network for the `Budget` test series. For each algorithm one curve is displayed showing the average absolute error, meaning the difference between real value of criticality and the predicted one. In this figure the error for MMDC_SET is drawn in blue, the error for MM_GRAPH is drawn in red, the error for HEU_LU is drawn in green and the error for HEU_APPROX is drawn in orange.

the Irish and Slovenian transmission network as both have no critical edges for all tested snapshots. For general hypothesis not specifying the test series we write the results as (`Budget`,`Candidate`) to differ between both test series. In the table we can see that our hypothesis are correct in most cases. Hypothesis 23 is for most networks in the test series not supported. The hypothesis is that, we expect the heuristic HEU_APPROX to have shorter optimization times than the model MM_GRAPH especially on larger transmission networks. The reason for the hypothesis being wrong for most networks is most likely that we have rather small networks in our test data set. It would be interesting to observe the results for this hypothesis on large transmission networks. We observe on the larger networks in out test data set for the `Budget` test series, that HEU_APPROX has faster optimization times for smaller budgets, meaning budgets up to 30%. We observe that the budgets until which HEU_APPROX has faster optimization times than MM_GRAPH increases with increasing network size. Therefore we suspect, that the HEU_APPROX has generally faster optimization times on large transmission networks.

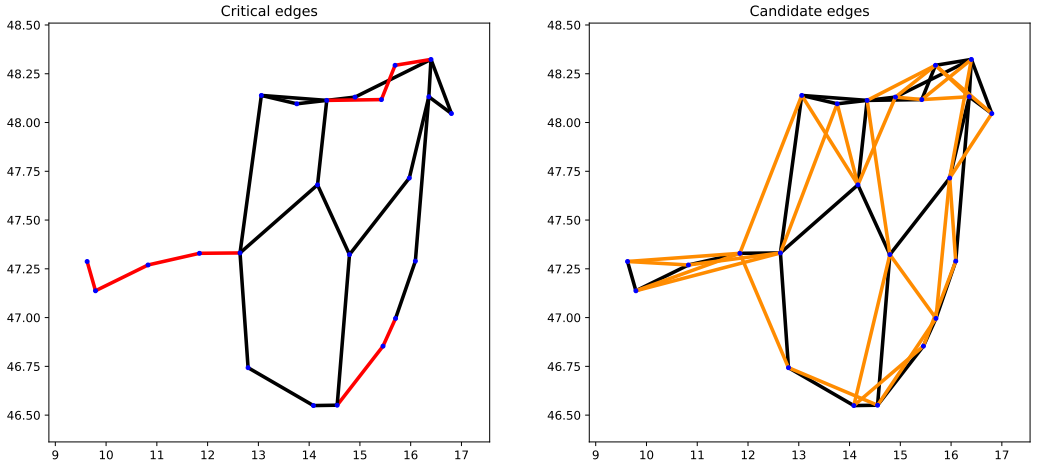
Table 6.4.: This table summarizes in which networks which Hypothesis from Section 6.4 are correct for the different test series. If a hypothesis is supported by the test data we use the symbol ✓. If the test data opposes a hypothesis we use the symbol **x**. If we cannot draw any conclusion concerning the hypothesis from the test data we use the symbol ∼. We do not include the Irish and Slovenian transmission network as both have no critical edges for all tested snapshots. For general hypothesis not specifying the test series we write the results as (`Budget`,`Candidate`) to differ between both test series.

country	H. 21	H. 22	H.23	H. 24	H. 25	H. 26	H. 27	H. 28
Austria	(✓|✓)	(✓|✓)	(x|x)	(*sim*|✓)	(✓|✓)	(x|✓)	(✓|✓)	(✓|✓)
Belgium	(✓|✓)	(✓|✓)	(x|x)	(✓|✓)	(∼|✓)	(x|✓)	(x|x)	(✓|✓)
Bulgaria	(✓|✓)	(✓|✓)	(x|x)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)
Croatia	(✓|✓)	(✓|✓)	(x|x)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)
Czech Republic	(✓|✓)	(∼|∼)	(∼|∼)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)
Denmark	(✓|✓)	(✓|✓)	(x|x)	(∼|x)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)
Hungary	(✓|✓)	(∼|∼)	(∼|∼)	(x|x)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)
Netherlands	(✓|✓)	(✓|✓)	(✓|✓)	(x|x)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)
Norway	(✓|✓)	(✓|✓)	(x|x)	(x|x)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)
Poland	(✓|✓)	(✓|✓)	(∼|✓)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)	(✓|✓)
Portugal	(✓|✓)	(✓|✓)	(x|x)	(✓|✓)	(✓|✓)	(x|✓)	(✓|✓)	(✓|✓)
Romania	(✓|✓)	(✓|✓)	(∼|∼)	(x|x)	(✓|✓)	(x|x)	(✓|✓)	(✓|✓)
Slovakia	(✓|✓)	(✓|✓)	(x|x)	(✓|✓)	(✓|✓)	(x|x)	(✓|✓)	(✓|✓)
Sweden	(✓|✓)	(✓|✓)	(x|x)	(x|x)	(✓|✓)	(x|x)	(✓|✓)	(✓|✓)
Switzerland	(✓|✓)	(✓|✓)	(x|x)	(x|x)	(✓|✓)	(x|x)	(✓|✓)	(✓|✓)

6.5. Properties of the Optimized Networks

At last we discuss the network properties of the optimized networks for all introduced optimization methods. An optimized network is hereby the result of optimizing a network using any optimization method and building the proposed edges. We hope to obtain some information on the performance of our methods on real-world transmission networks by simulating the optimized networks for all optimization methods using the DC power flow approximation. By simulating the optimized networks we get information about the feasibility of our proposed extension for the given load distribution.

As discussed in the beginning of this chapter we calculate a DC power flow approximation on all optimized networks. For all evaluated networks we had no case, in which the optimized network had no feasible power flow for the given load distribution. However there are some differences between the optimized networks, resulting from using different optimization methods on the same transmission network, which we want to discuss in this section.

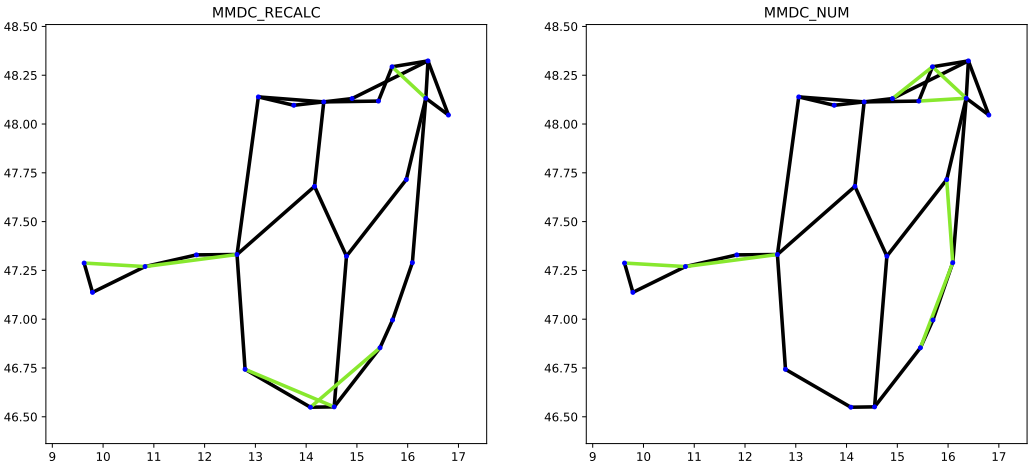


(a) Critical edges of the Austrian transmission network. Critical edges are marked red and not critical edges are marked black in the figure. The unit of the x-axis is given in longitude and the y-axis in latitude in geocoordinates.

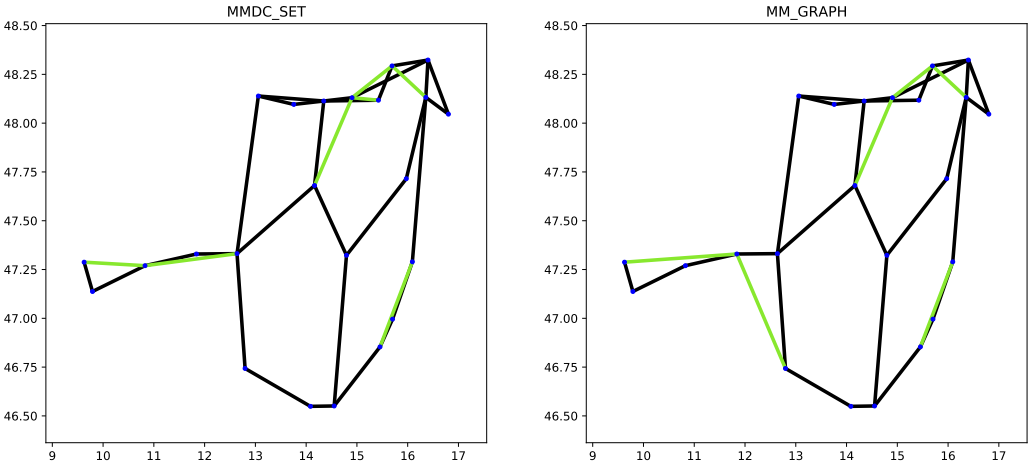
(b) Candidate edges of the Austrian transmission network. Candidate edges are marked orange and original edges are marked black in the figure. The unit of the x-axis is given in longitude and the y-axis in latitude in geocoordinates.

Figure 6.22.: Figures showing the critical edges and the candidate edges of the Austrian transmission network for the 1st January 2013 at 13:00.

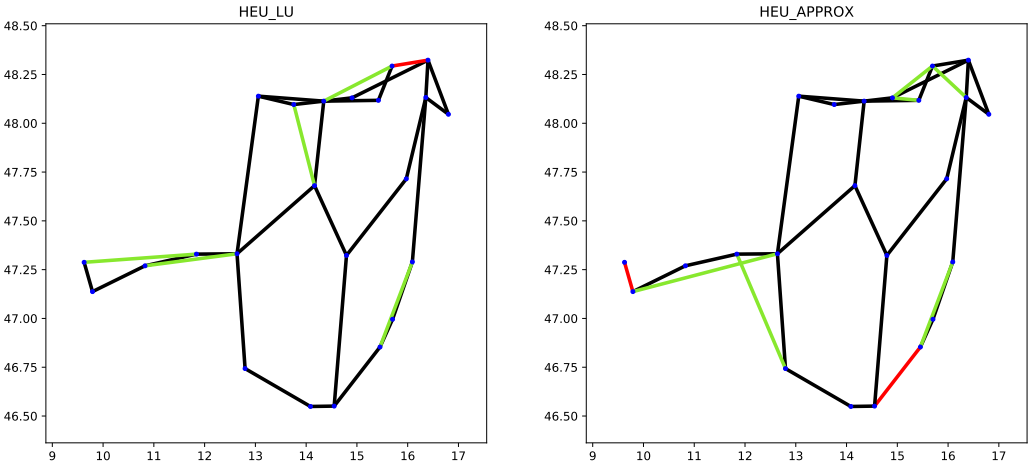
The most significant difference between the optimized networks of different optimization methods is, that they have different topologies. We introduced three methods using the DC power flow approximation MMDC_SET, MMDC_NUM and MMDC_RECALC, one mathematical model solving the TNEP-CCE problem MM_GRAPH and the versions of a heuristic for solving the TNEP-CCE problem HEU, HEU_LU and HEU_APPROX. For HEU and HEU_LU we observe the same networks as the only difference between both optimization methods is the use of a lookup table in HEU_LU to avoid calculating the same value multiple times. Except for HEU and HEU_LU we observe for all optimization methods differences in choice of candidate edges depending on the network. As an example to show these different choices in candidate edges we show the results of the optimization of the Austrian transmission network on the 1st January 2013 at 13:00 with a budget of 25%. In Figure 6.22a the critical edges and in Figure 6.22b the candidate edges for this snapshot of the Austrian transmission network are depicted. In Figures 6.23a and 6.23b we can see the results for optimizing the network using the methods MMDC_RECALC and MMDC_NUM. In Figure 6.23c the results for the optimization with the model



(a) Results for MMDC_RECALC. (b) Results for MMDC_NUM.



(c) Results for MMDC_SET. (d) Results for MM_GRAPH.



(e) Results for HEU_LU. (f) Results for HEU_APPROX.

Figure 6.23.: Optimized networks resulting from optimizing the Austrian transmission network for the 1st January 2013 at 13:00 with all introduced optimization methods. Critical edges are marked red, not critical edges are marked black and added edges are marked green in each figure. The unit of the x-axis is given in longitude and the y-axis in latitude in geocoordinates.

MMDC_SET can be seen. The results for the model MM_GRAPH are depicted in Figure 6.23d. The results for the heuristic HEU_LU and HEU_APPROX can be seen in Figures 6.23e and 6.23f. All normal network edges in the figure are displayed in black. Critical edges in the networks are displayed in red. The candidate edges are displayed in orange. All candidate edges added to the network by any optimization method are displayed in green.

We can clearly see a difference between all optimization results. All methods add different candidate edges to the network and while all critical edges are cured for MMDC_RECALC, MMDC_NUM, MMDC_SET and MM_GRAPH we have still critical edges for HEU_LU and HEU_APPROX. The choice of different edges has also an effect on the maximum power flow the network can support. We do not consider or optimize the maximum power flow for any of the optimization models. However, the maximum power flow is an interesting measure for comparing different expansions of the same transmission network. For this example we had an initial maximum power flow of ca. 18,679 MW. The maximum power flow increases for all optimization methods in this example. The highest maximum power flow we observe in this example for the optimized networks using the MMDC_NUM and HEU_LU methods both are ca. 25,598 MW. The difference between both power flows is less than 0.1 MW which is quite interesting. The lowest maximum power flow we observe for the MMDC_RECALC model with ca. 23,544 MW. Of course this is just one example and therefore we cannot draw any conclusion from these results for the maximum power flows.

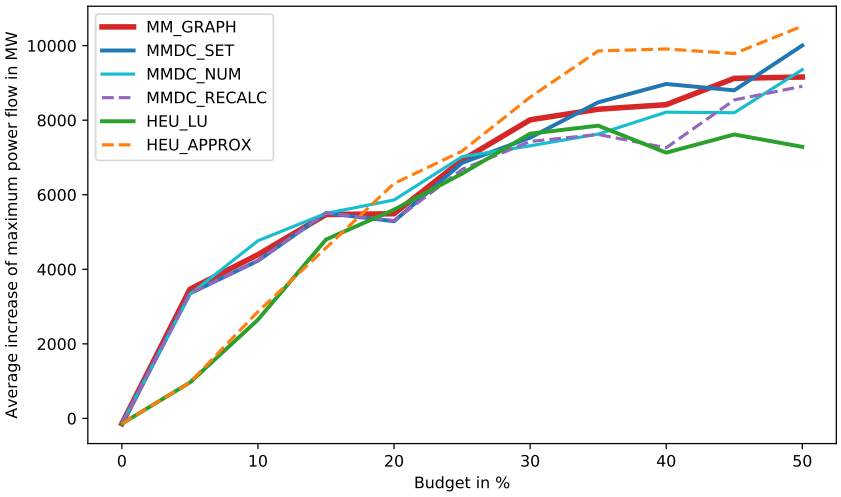


Figure 6.24.: Results for the maximum power flow calculations on the optimized Austrian networks. These are the results of the `Budget` test series. In the figure we see the increase of maximum power flow for each model. We display the increase in maximum power flow for MM_GRAPH(red), MMDC_SET (dark blue), for MMDC_NUM (light blue) and for MMDC_RECALC (purple). For the heuristics we display the increase in maximum power flow for HEU_LU (green) and for HEU_APPROX (orange).

To obtain better data we evaluated the maximum power flows for all methods using the budget test series. In Figure 6.24 we see the results of the maximum power flow computations on the Austrian transmission network. In the figure we display the changes of maximum power flow relative to the original maximum power flow of the network for different budgets. As mentioned above the Austrian transmission network has an initial maximum power flow of ca. 18,679 MW. In the figure we display the changes in

maximum power flow as solid dark blue line for MMDC_SET, as solid light blue line for MMDC_NUM and as dashed purple line for MMDC_RECALC. For the heuristics we used a solid green line to display the results for HEU_LU and a dashed orange line for HEU_APPROX. The results for MM_GRAPH are displayed as solid red line. In the figure we see, that the maximum power flow for each method increases with increasing budget. This can be expected as each method adds more candidate edges with increasing budget. It is interesting to observe that for budgets of less than 15% we can observe similar behavior for similar optimization methods. This means that HEU_LU and HEU_APPROX have similar maximum power flows for budgets lower than 15%, as well as all mathematical models have similar maximum power flows for small budgets. For larger budgets than 15% we observe the tendency of HEU_APPROX providing the largest increase in power flow to the Austrian transmission network. For HEU_LU we observe the tendency to provide the lowest increase in maximum power flow. The maximal difference between the additional maximum power flow provided using HEU_LU and using HEU_APPROX is ca. $2,700$ MW, this is about 14% of the initial power flow on the Austrian transmission network. This difference is quite large especially because HEU_LU and HEU_APPROX use the same routine to add edges to the network but based on different values. We have no explanation for this difference between the maximum power flow of the networks optimized using HEU_LU and HEU_APPROX. As mentioned before we do not optimize or model the maximum power flow in any of our models.

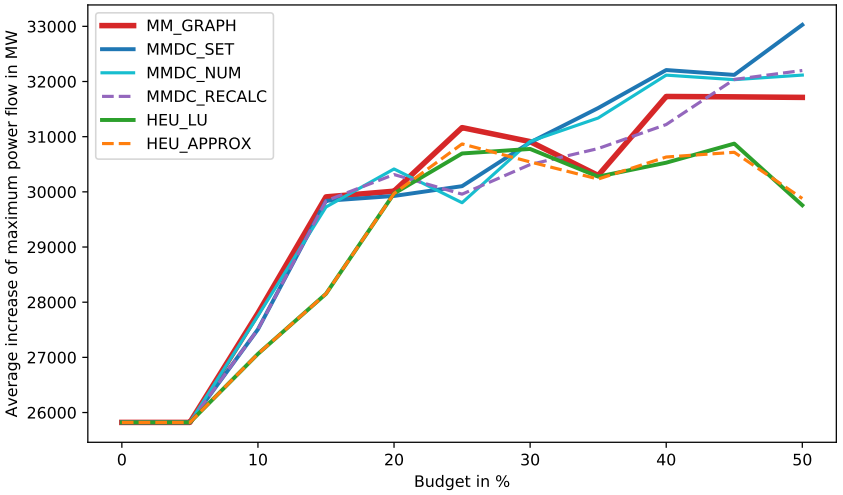


Figure 6.25.: Results for the maximum power flow calculations on the optimized Swiss networks. These are the results of the `Budget` test series. In the figure we see the increase of maximum power flow for each model. We display the increase in maximum power flow for MM_GRAPH(red), MMDC_SET (dark blue), for MMDC_NUM (light blue) and for MMDC_RECALC (purple). For the heuristics we display the increase in maximum power flow for HEU_LU (green) and for HEU_APPROX (orange).

On all networks we observe an increasing maximum power flow for all methods with increasing budget. But the maximum power flow of the methods in comparison with each other changes from network to network. For example we can see in Figure 6.25 the results for the Swiss transmission network. We use the same colors and line styles in this figure as before. We can observe that for the Swiss transmission network both heuristics HEU_LU and HEU_APPROX have a very similar and the lowest maximum power flow for most budgets. The other methods perform quite similar for budgets below 20%. For budgets

above 20% the maximum power flows of these other methods diverge. It is not possible for those other methods to conclude which one performs best.

For most tested networks we observe that HEU_LU and HEU_APPROX have the lowest maximum power flow for budgets smaller than 15%. For budgets larger than 15% we cannot observe any reoccurring pattern besides the increase in maximum power flow for increasing budgets for all optimization methods.

In conclusion we have no method which performs remarkably better than the other methods. It seems to depend on the network topology which method provides the largest increase in maximum power flow. This is not surprising as we do not consider the maximum power flows during the creation of our optimization methods.

7. Conclusion

In Section 5.2 of this thesis we define and introduce the problem TNEP-CCE. This problem formalizes the problem of expanding a transmission network with the objective to cure critical edges. We show that the problem is NP-complete, even for instances with planar graphs and single critical edges.

We introduce different solving methods for this problem.

The first solving method, introduced in Section 5.3.1, is to formulate the problem as Mixed-Integer Linear Program while considering changes in power flow caused by adding candidate edges to the transmission network. We discuss two different versions of the MILP. The first version MMDC_RECALC recalculates the critical edges during the optimization and the second simplified version MMDC_SET uses a set of critical edges as input. To observe the effect of the exact objective formulation, we reformulate the objective of the version MMDC_SET. While the original objective minimizes the sum of all criticalities, the reformulated objective minimizes the number of critical edges in the network. We refer to the reformulated version as MMDC_NUM.

To further research the problem and introduce faster heuristic solution methods we introduce the simplification of ignoring power flow changes. Based on this simplification we reformulate the MILP MMDC_SET to use the initial power flow without updating it. This simplified version is called MM_GRAPH and is introduced in Section 5.3.2. Additional to MM_GRAPH we introduce in Section 5.5.3.1 a heuristic HEU to solve TNEP-CCE which is based on the idea of dynamic programming. For this heuristic we introduce modifications to speed up the optimization process. The first modification HEU_LU uses a lookup table and the second modification HEU_APPROX further simplifies the problem. The version HEU_APPROX approximates the value of criticality which is cured by a set of candidate edges by using precaluclated values for each candidate edge.

Based on the result of the evaluation, presented in Chapter 6, we come to following conclusions:

For an application, requiring a solution of the problem TNEP-CCE without time limitations, we recommend using the MILP MMDC_RECALC. Even though the model MMDC_RECALC has long optimization times it produces the best results in comparison to the other optimization methods.

The model MMDC_SET provides for most of the tested networks similar results considering the quality of the solution compared to MMDC_RECALC. The usage of a given set of

candidate edges leads to significant speed ups in comparison to the optimization time of MMDC_RECALC. We recommend using this model to achieve high accuracy while reducing the optimization times compared to MMDC_RECALC. Thus it can be useful for testing purposes, for example to adjust parameters of candidate networks before optimizing the network with the MMDC_RECALC model.

For applications with time limitations we recommend using models with faster optimization times and less accuracy. Generally we recommend using the model MM_GRAPH. The model MM_GRAPH has great result quality relative to its optimization times. Its optimization time does not exceed five seconds on any of the tested networks. Thus we consider MM_GRAPH to be suitable even for usage in applications with direct user interaction.

We do not recommend using the model HEU in any case. This model has a high optimization time and produces low quality results compared to the models MMDC_SET or MM_GRAPH.

Generally, we do not recommend using the models HEU_LU and HEU_APPROX as well. However, the results of the evaluation especially for the model HEU_APPROX were quite interesting. Because of the properties of the problem TNEP-CCE introduced in Section 5.5, we expected that the value approximation used in HEU_APPROX would lead to a significant decrease in result quality. But the cases, in which the quality of the result would suffer due to the value approximation, do not occur in the tested transmission networks as frequently as expected.

The largest network we tested our models on has 48 nodes and 84 edges. We can only speculate how they would behave on larger networks. We suspect that the optimization and model building times of the MILPs will increase further with increasing network size. We observe that the model HEU_APPROX has the shortest optimization time on the largest networks in our test data set for budgets between 0% and 25%. The fraction of budgets for which HEU_APPROX was the fastest model increases with increasing network size for our test data. Because of that, we think that HEU_APPROX will probably have the shortest optimization time of all our models on networks considerably larger than the ones in our test data set. We suspect that the model HEU_APPROX can therefore be quite useful for optimizing large networks, such as the European transmission network, for interactive applications.

7.1. Future Work

The focus of this thesis is the introduction and examination of the problem TNEP-CCE. Because of the time limitations for master theses, it was not possible to implement and test all ideas we have to solve the problem TNEP-CCE.

Further speed ups for the heuristic HEU and its modifications might be reached by using a maximum flow algorithm optimized to handle dynamic graphs. As we constantly add edges to the original network this could lead to speed ups, as previous computations could be reused. Maximum flow algorithms for dynamic graphs are for example researched in the context of computer vision. One example for this is a dynamic flow algorithm proposed by Kohli et al. [KT07].

Another possibility to speed up the heuristic HEU is to reduce the number of candidate edges considered for curing each critical edge. We already discussed in Section 5.5 that we can tell whether a candidate edge will cure some criticality of a critical edge (a, b). We can do this by looking at its position in the residual graph resulting from the maximum a,b-flow computation for checking the criticality of (a, b). If a candidate edge connects

the subgraph containing a and the subgraph containing b of the residual graph, we know that it will cure some criticality. For all other edges we know that they will not cure any criticality for this critical edge. These properties can be used in the heuristic HEU to speed up value computations.

Another direction of research can be to apply meta heuristics and other approaches commonly used in the field of Transmission Network Expansion Planning and apply them to TNEP-CCE. Some examples of meta heuristics to try are tabu search, neural networks or the frog-leap algorithm as briefly discussed in Section 3.

We focus on the steady-state version of the TNEP-CCE problem. This means we handle one load distribution at a time and propose network expansions based only on this distribution. But it is well known that network load distributions change over time. We also observe different critical edges for different days and different hours of the same day for the same network. Thus it might be interesting to formulate a dynamic version of TNEP-CCE and optimize a network by considering load distributions over a time span. This could for example include the weighing of critical edges depending on the percentage of time when they are critical.

Bibliography

[AAC10] N. Alguacil, J. M. Arroyo, and M. Carrión. Transmission network expansion planning under deliberate outages. In *Handbook of Power Systems I*, pages 365–389. Springer, 2010.

[ASEA02] T. Al-Saba and I. El-Amin. The application of artificial intelligent tools to the transmission expansion problem. *Electric Power Systems Research*, 62(2):117–126, 2002.

[BCGVH14] R. Bent, C. Coffrin, R. Gumucio, and P. Van Hentenryck. Transmission network expansion planning: Bridging the gap between ac heuristics and dc approximations. In *2014 Power Systems Computation Conference*, pages 1–8. IEEE, 2014.

[Bel57] R. Bellman. *Dynamic programming*. Courier Corporation, 1957.

[BHS18] T. Brown, J. Hörsch, and D. Schlachtberger. PyPSA: Python for Power System Analysis. *Journal of Open Research Software*, 6(4), 2018.

[BHS19] T. Brown, J. Hörsch, and D. Schlachtberger. Pypsa documentation, 2019.

[BPG01] S. Binato, M. Pereira, and S. Granville. A new benders decomposition approach to solve power transmission network design problems. *IEEE Transactions on Power Systems*, 16(2):235–240, 2001.

[BV15] D. Bienstock and A. Verma. Strong np-hardness of ac power flows feasibility. *arXiv preprint arXiv:1512.07315*, 2015.

[Chr00] R. Christie. Power systems test case archive, university of washington. *Electrical Engineering. https://www2. ee. washington. edu/research/pstca*, 2000.

[CZP10] F. Cadini, E. Zio, and C.-A. Petrescu. Optimal expansion of an existing electrical power transmission network by multi-objective genetic algorithms. *Reliability Engineering & System Safety*, 95(3):173–181, 2010.

[DEA73] Y.P. Dusonchet and A. El-Abiad. Transmission planning using discrete dynamic optimizing. *IEEE Transactions on Power Apparatus and Systems*, (4):1358–1371, 1973.

[dPP27] T. de Paula Peixoto. Graph tool - efficient network analysis, 2.27.

[DSODOB01] E. L. Da Silva, J.M. A. Ortiz, G. C. De Oliveira, and S. Binato. Transmission network expansion planning under a tabu search approach. *IEEE Transactions on Power Systems*, 16(1):62–68, 2001.

[EH72] O. I. Elgerd and H.H. Happ. Electric energy systems theory: an introduction. *IEEE Transactions on Systems, Man, and Cybernetics*, (2):296–297, 1972.

[EK72] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[ESH11] M. Eghbal, T. K. Saha, and K. N. Hasan. Transmission expansion planning by meta-heuristic techniques: a comparison of shuffled frog leaping algorithm, pso and ga. In *Power and energy society general meeting*, pages 1–8. IEEE, 2011.

[FF56] L. R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[Fou19a] Python Software Foundation. The python standard library, 2019.

[Fou19b] Python Software Foundation. The python standard library: Numeric and mathematical modules: random, 2019.

[GJ78] M. R. Garey and D.S. Johnson. Strong np-completeness results: Motivation, examples, and implications. *Journal of the ACM (JACM)*, 25(3):499–508, 1978.

[GJ79] M. R. Garey and D. S. Johnson. A guide to the theory of np-completeness. *Computers and Intractability*, 1979.

[GO18] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018.

[HHK13] R. Hemmati, R.-A. Hooshmand, and A. Khodabakhshian. State-of-the-art of transmission expansion planning: Comprehensive review. *Renewable and Sustainable Energy Reviews*, 23:312–319, 2013.

[HVBG+10] D. Heide, L. Von Bremen, M. Greiner, C. Hoffmann, M. Speckmann, and S. Bofinger. Seasonal optimal mix of wind and solar power in a future, highly renewable europe. *Renewable Energy*, 35(11):2483–2489, 2010.

[KT07] P. Kohli and P.H.S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2079–2088, 2007.

[McC12] Dr. J. McCalley. Lecture notes in steady state-analysis, lecture the dc power flow equations, 2012.

[MHKS14] P. J. Menck, J. Heitzig, J. Kurths, and H. J. Schellnhuber. How dead ends undermine power grid stability. *Nature communications*, 5, 2014.

[MPS10] L. Moulin, M. Poss, and C. Sagastizábal. Transmission expansion planning with re-design. *Energy systems*, 1(2):113–139, 2010.

[OO11] D. Oeding and B. R. Oswald. *Elektrische kraftwerke und netze*, volume 6. Springer, 2011.

[RSTW12] M. Rohden, A. Sorge, M. Timme, and D. Witthaut. *Phys. Rev. Lett.*, 109, 2012.

[RWTMO17] M. Rohden, D. Witthaut, M. Timme, and H. Meyer-Ortmanns. Curing critical links in oscillator networks as power flow models. *New Journal of Physics*, 19(1):013002, 2017.

[SBP+08] I. Simonsen, L. Buzna, K. Peters, S. Bornholdt, and D. Helbing. Transient dynamics increasing network vulnerability to cascading failures. *Physical review letters*, 100(21), 2008.

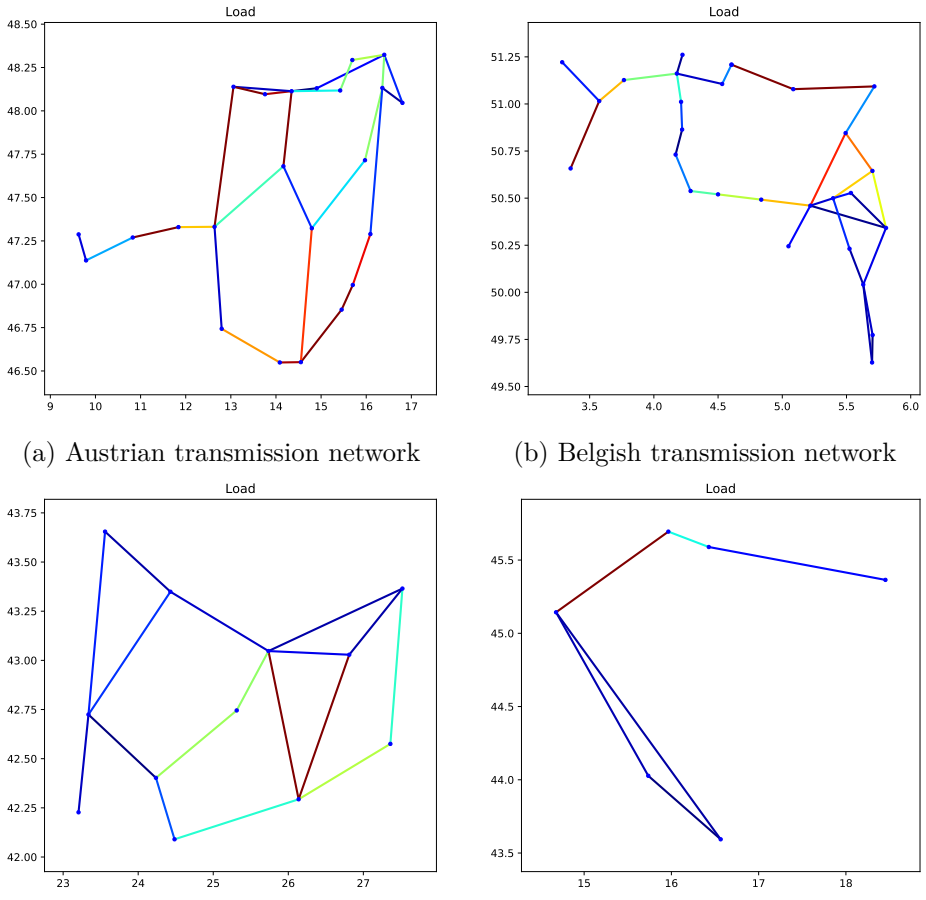
[WRZ+16a] D. Witthaut, M. Rohden, X. Zhang, S. Hallerberg, and M. Timme. Critical links and nonlocal rerouting in complex supply networks. *Phys. Rev. Lett.*, 116:138701, Mar 2016.

[WRZ+16b] D. Witthaut, M. Rohden, X. Zhang, S. Hallerberg, and M. Timme. Supporting information accompanying the manuscript critical links and nonlocal rerouting in complex supply networks. *Phys. Rev. Lett.*, 116:138701, Feb 2016.

Appendix

A. Test Data: Topologies of Transmission Networks

The following figures show each of the tested transmission networks with a powerflow analysis on the 1st Feburary 2013 at 16:00.



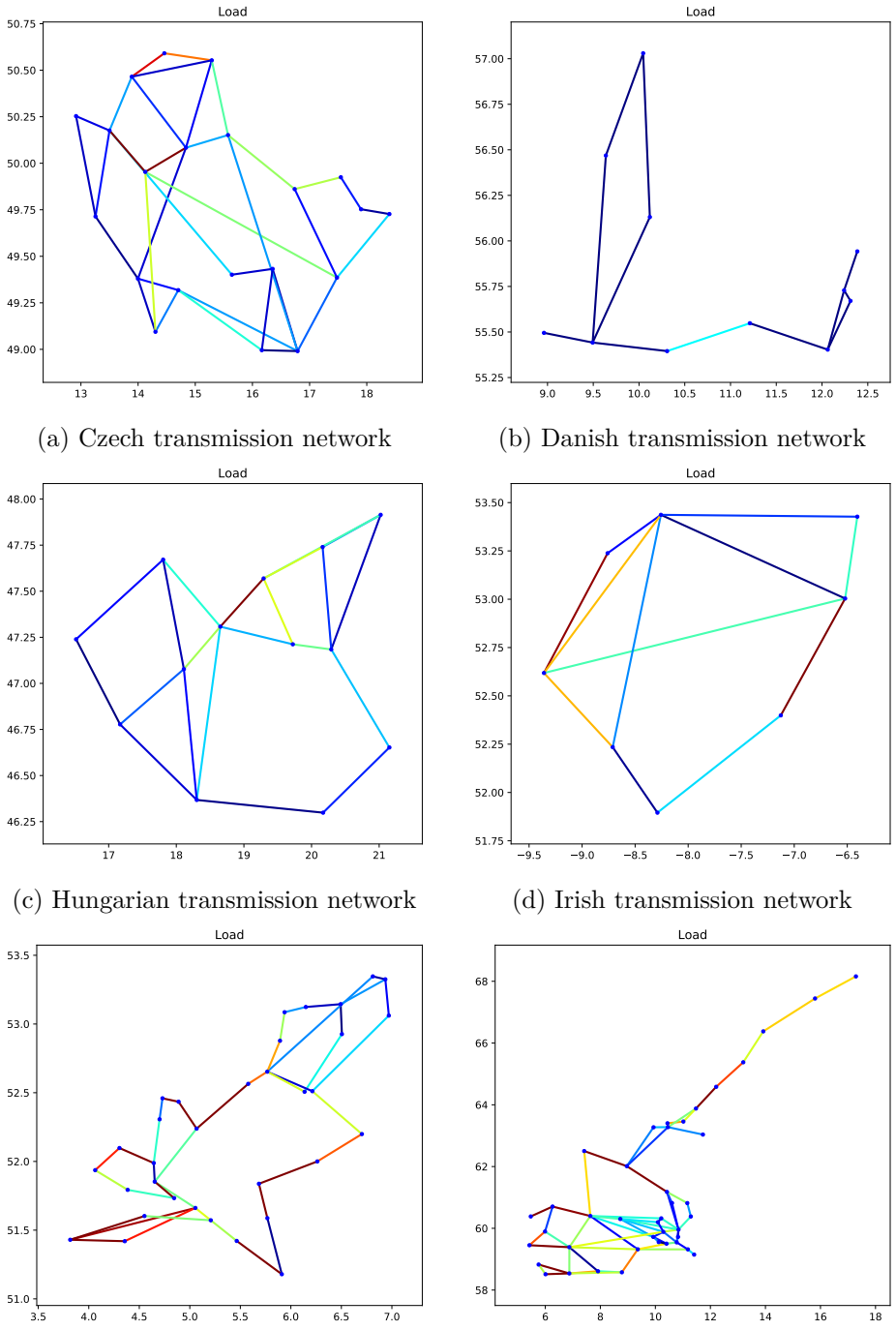
(a) Austrian transmission network

(b) Belgish transmission network

(c) Bulgarian transmission network

(d) Croatian transmission network

Figure A.1.: Load on different transmission networks on the 1st Feburary 2013 at 16:00. The x-axis is given in degrees longitude and the y-axis in degrees latitude.



(a) Czech transmission network

(b) Danish transmission network

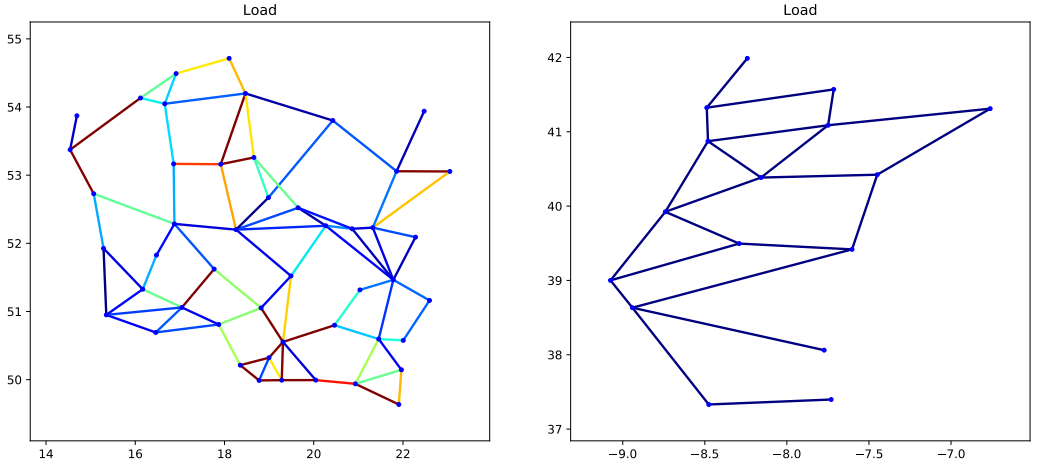
(c) Hungarian transmission network

(d) Irish transmission network

(e) Optimal powerflow for the Dutch transmission network on 1st Februrary 2013 at 16:00

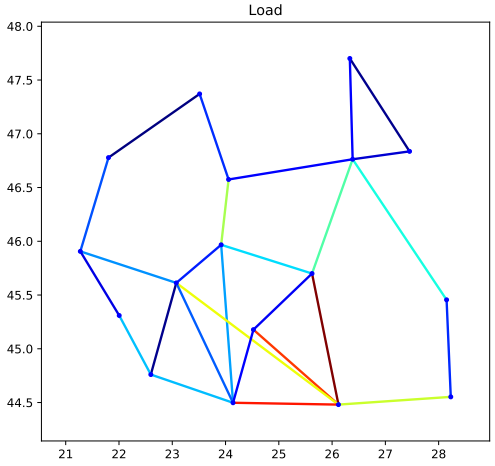
(f) Optimal powerflow for the Norwegian transmission network on 1st Februrary 2013 at 16:00

Figure A.2.: Load on different transmission networks on the 1st Feburary 2013 at 16:00. The x-axis is given in degrees longitude and the y-axis in degrees latitude.

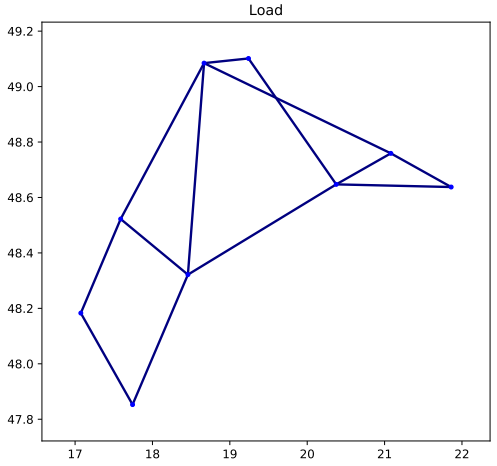


(a) Polnish transmission network

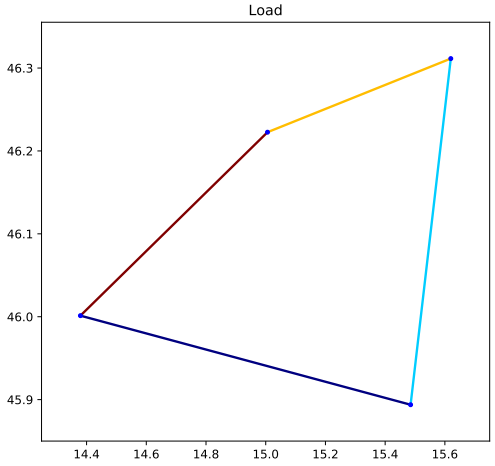
(b) Portuguese transmission network



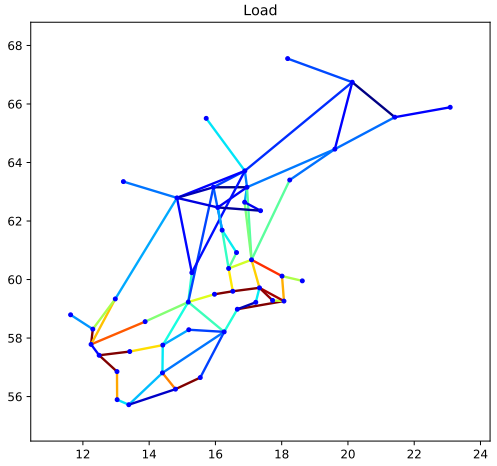
(c) Romanian transmission network



(d) Slovak transmission network



(e) Slovenian transmission network



(f) Swedish transmission network

Figure A.3.: Load on different transmission networks on the 1st Feburary 2013 at 16:00. The x-axis is given in degrees longitude and the y-axis in degrees latitude.