

ETH Zurich - Department of Mathematics - Institute for Operations
Research

Master Thesis - Network reliability problems with
applications in electricity networks

March 5, 2007

Christian Balderer

Professor: Prof. Dr. H.-J. Lüthi
Supervisors: M. Guarisco, R. Zenklusen

Abstract

Reliable electricity supply is a fundamental need in industrialized countries. Therefore, it is crucial to be able to keep electric power systems in a secure state, even in case of disturbances. For this reason, electric power systems have been designed with redundancies and are able to guarantee a reliable electricity supply most of the time. However, large blackouts still occur all over the world and cause great economic damage to the affected regions.

The goal of our work is to develop repair strategies that try to minimize the risk of blackouts due to cascading failures. Given a power transmission system which suffers from several line failures, we try to determine the lines that should be repaired first. We develop a realistic blackout model in order to understand the dynamics of blackouts. Based on the insights from the blackout model, we define several abstract repair problems. These problems not only lead to a concrete repair strategy for electric power systems, but are also of theoretical interest on their own. We apply the derived strategy to our blackout model and conclude that it provides a good starting point for the development of further repair strategies.

Contents

1	Introduction	4
1.1	Motivation	4
1.2	Outline	4
2	Flow models	6
2.1	Network flow model	6
2.2	DC power flow model	6
3	Blackouts	9
3.1	Cascading failures	9
3.2	Load shedding	10
3.3	Blackout model	11
3.3.1	Flow model	11
3.3.2	Initial failure	11
3.3.3	Load shedding	12
3.3.4	Cascading failures	12
3.3.5	Simulation procedure	13
3.4	Computational results	13
3.4.1	The reliability test system	14
3.4.2	Simulation results	15
4	Repair problems	19
4.1	Static repair problem	20
4.1.1	Hardness	20
4.1.2	Special cases	22
4.1.3	Approximation algorithm	24
4.2	Dynamic repair problem	24
4.2.1	Hardness	26
4.2.2	Approximation algorithm	27
4.3	DC repair problem	31
4.3.1	Hardness	32
4.3.2	Maximum flow strategy	34
5	Blackout prevention	37
5.1	Blackout prevention model	37
5.2	Computational results	37
5.2.1	Without repair	38
5.2.2	Random repair	39
5.2.3	Maximum flow strategy	39
6	Implementation	42

7 Conclusion	43
7.1 Summary	43
7.2 Outlook	43
A Classic reliability models	45
A.1 Hardness results	45
A.2 Existing algorithms	46
A.2.1 Inclusion-exclusion	46
A.2.2 Disjoint products	47
A.2.3 Restricted classes	47
A.3 Algorithms for acyclic and nearly acyclic graphs	48
A.3.1 Acyclic directed graphs	48
A.3.2 Nearly acyclic directed graphs	50

1 Introduction

1.1 Motivation

In industrialized countries, a reliable supply of electrical energy is taken for granted. In order to guarantee a stable energy supply even in case of non-serious failures of electrical equipment, the underlying electric power network has been designed with redundancies. Nevertheless, major blackouts of the transmission grid occur all over the world. They are typically caused by a sequence of cascading failures and may be evidence of a critically loaded transmission system [5]. Furthermore, the increasing trade in electricity - a consequence of the liberalization of the energy markets - has led to an additional load of the existing infrastructure. As a consequence, the reliable operation and maintenance of the electric power grid at minimum cost is an increasingly demanding task.

The goal of our work is to develop repair strategies that minimize the risk of cascading failures. A power operator has generally not enough time to repair failed lines once a cascade has started, because cascading failures typically evolve in time scales of seconds and minutes. Consequently, we focus on repair strategies during normal operation. Even during normal operation, there are typically some lines that are not operating due to random failures or maintenance work. While the system may still have enough capacity to transmit the power demand, the average loading and therefore the blackout risk increases with every failed line [5].

Unfortunately, it is very difficult to quantify the overall risk of blackouts due to cascading failures [5]. Hence, we first develop a simple blackout model that allows us to improve our understanding of cascading failures and will later serve as a test environment for our repair strategies. Second, we consider some closely related problems on the classical network flow model in order to get some initial ideas for reasonable repair strategies

1.2 Outline

We start by introducing the two network models used throughout this thesis in section 2. In section 3, we define a power system blackout model, inspired by [11]. We present and analyze simulation results obtained from our model. In section 4, we define three basic repair problems. Whereas the first two are based on the classical network flow model, the third one is defined in terms of a more realistic power flow model. In section 5, we apply the ideas from section 4 to our power system blackout model defined in section 3 and present simulation results. In section 6, we briefly discuss the implementation of the blackout model and the repair strategies. In section 7, we summarize our

results and present some open problems. We give a little survey of the classical network reliability models in appendix A.

2 Flow models

An exact model of a power transmission grid would be given by the set of physical equations that describe an electric power network in all its details. However, solving these equations requires a lot of computational effort and the solution contains much more information than we will actually need. Our goal is to find a model that is as simple as possible, as we are only interested in a crude estimation of the power flow on the lines of a specific network in the following sections.

We propose two different models for this purpose. Both of them model the power network as a graph, whereas they impose different constraints on the set of feasible flows. The first model is the well known *network flow model*. The second model is based on a physical description of the power system and is known as *DC power flow model*.

2.1 Network flow model

In this model, a *network* is a directed graph $G = (V, E)$ with nonnegative capacities $u(e)$ on each edge $e \in E$ and two distinguished vertices, namely a source vertex $s \in V$ and a sink vertex $t \in V$. A *flow* f assigns a value $f(e)$ to each edge $e \in E$. For each $v \in V$, we write $f^+(v)$ for the total flow on edges leaving v and $f^-(v)$ for the total flow on edges entering v . A flow is feasible if it satisfies the *capacity constraints* $0 \leq f(e) \leq u(e)$ for each edge and the conservation constraints $f^+(v) = f^-(v)$ for each node $v \notin \{s, t\}$. The *value* $val(f)$ of a flow f is the net flow $f^-(t) - f^+(t)$ into the sink. A *maximum flow* is a feasible flow with maximum value.

2.2 DC power flow model

The *DC power flow model* is derived in detail in [16]. We give a short summary of the derivation and present the basic ideas. In this model, a *transmission grid* is an undirected graph $G = (V, E)$ with nonnegative capacities $u(e)$ and reactances $x(e)$ for each edge $e = \{i, k\} \in E$ and limits p_i^{min} and p_i^{max} on the net real power p_i for each node $i \in V = \{1, \dots, n\}$. In the following, the nodes are referred to as *buses* and the edges are referred to as *lines*. The set of buses V can be partitioned into a set of *generators* V_G and a set of *loads* V_L . The buses with positive net real power belong to V_G , whereas the ones with negative net real power belong to V_L .

The derivation of the model is based on the alternate current (AC) power flow problem, where we are given the real and reactive power at the buses and we have to solve for the voltage magnitudes and angles. Given the voltage magnitudes and angles, we can calculate the actual power flow on

the lines. The power flow problem can be stated as a set of nonlinear equations, representing network operations limits and physical laws. Due to the nonlinearity of these equations, they are typically solved using an iterative method like the *Newton-Raphson* method. As we are only interested in an estimate of the power flow on a line, we can alternatively solve an approximate version of the power flow equations.

With the assumptions and simplifications described in [16], the power flow equations can be linearized and written as a set of linear equations of the form

$$\mathbf{p} = \mathbf{B}\theta \quad (1)$$

where $\mathbf{p} = (p_1, \dots, p_n)$ is the vector of net real power, \mathbf{B} the admittance matrix representing the properties of the power lines and $\theta = (\theta_1, \dots, \theta_n)$ the vector of voltage angles. Given the admittance matrix and the power injections at the nodes we can solve the linear system for the voltage angles.

For a line $e = \{i, k\} \in E$, the flow f_{ik} from i to k can be expressed in terms of the voltage angles θ_i and θ_k as

$$f_{ik} = \frac{\theta_i - \theta_k}{x(e)} = -f_{ki}$$

This means that the flow vector \mathbf{f} with entries f_{ik} can be written as

$$\mathbf{f} = \mathbf{A}\theta$$

for an appropriate matrix A .

Together with equation (1), we get a linear relationship between the flow vector \mathbf{f} and the vector of net real power \mathbf{p} .

$$\mathbf{f} = \mathbf{A}\mathbf{B}^{-1}\mathbf{p}$$

These linear equations are commonly referred to as *DC power flow equations*.

A flow \mathbf{f} is *feasible* if it satisfies the capacity constraints $-u(\{i, k\}) \leq f_{ik} \leq u(\{i, k\})$ as well as the DC power flow equations $\mathbf{f} = \mathbf{A}\mathbf{B}^{-1}\mathbf{p}$ for some vector of power injections \mathbf{p} . Additionally, we can impose constraints on the power injections of the form $p_i^{min} \leq p_i \leq p_i^{max}$. The *value* $val(\mathbf{f})$ of a flow is the sum of the net flows out of all generators, i.e. $\sum_{i \in V_G} p_i$. A *maximum flow* is a feasible flow with maximum value.

Unfortunately, not all feasible flows in the sense defined above are feasible in reality. For instance, the linearization is only valid for small differences of voltage angles between connected buses.

This model is known as the DC power flow model, because it can be interpreted as the model for a network of resistors. In this dual view, θ is the vector of voltages, \mathbf{B} the conductance matrix and p the vector of net current injections [15].

3 Blackouts

The recent research in the area of blackout risk and cascading failures has started with the analysis of 15 years of blackout data from the North American Electric Reliability Corporation (NERC) [8]. The analysis has shown that large blackouts are much more likely than expected from Gaussian statistics. The data reveal that the distribution of the blackout size approximately follows a power law dependence. This means that the probability density function (pdf) for some measure of blackout size x is of the form

$$f(x) = x^{-k}$$

The observed exponent is approximately -1, which means that the risk of large blackouts is not negligible.

A *blackout* is defined as a cascading event in which the load shed is larger than a small value, typically 10^{-5} times the total power demand [10]. As cascading failures are the usual mechanism for large blackouts [11], we will use the terms interchangeably, depending on the context. In order to improve the understanding of the dynamics of cascading failures, Dobson, Carreras et al. developed a power system blackout model [11]. Their simulations show a strong dependence between the system loading and the average blackout size. At a critical loading, the resulting blackout distribution confirms the power law dependence observed from the NERC blackout data.

We give a more detailed introduction into cascading failures below and develop a model inspired by [11] that allows to simulate such cascading events.

3.1 Cascading failures

The content of this subsection is mainly based on the dissertation of Marek Zima [6] and an article of Ian Dobson [5]. The former deals with the security of electric power systems in general, whereas the latter gives a good overview of the recent research on cascading failures and blackout risk.

A *cascading failure* is a process, in which an initiating event increases the stress of other system components, resulting in a possible overloading above the limits for which they were designed. The outage of overloaded components happens on two different time scales, denoted by steady-state (slow) and transient (quick) progression. The transient progression includes phenomena like voltage instability, frequency and power oscillations and its time scale is in the order of seconds. The steady-state progression is usually guided by line failures due to overload and its time scale is in the order of

minutes [6].

The initial event is assumed to be a random component failure, like a lightning strike or a short circuit caused by a tree falling on a transmission line. Failures due to overload mainly happen for the following two reasons. If the power flow exceeds the so called short term emergency limit, the line protection system will automatically disconnect the line from the system. If the line is overloaded but still below the short term emergency limit, the temperature of the line starts to increase which may finally cause the line to sag into a tree [6].

The risk of cascading failures highly depends on the loading of the system. Consider for instance a system at very low loading. In case of a line failure, the operating margins of the remaining lines are large enough to compensate the failure, preventing a blackout due to cascading failures. For low loading, the line failures can therefore be assumed to be independent and the total number of failed lines should follow a binomial distribution. The binomial distribution has exponential tails, and large blackouts are therefore unlikely in this situation. If the system is at very high load, a single line failure will cause other lines to overload, resulting in a sequence of cascading failures with high probability. The observed power law dependence corresponds to a loading that lies somewhere in between these two extremes. This loading level is denoted as *critical loading*, as it represents a critical transition of the blackout distribution function [5].

Today's power systems are in general operated according to the $N - 1$ criterion. The $N - 1$ criterion says that no other component should be overloaded after the outage of a single component. This is not enough to totally prevent blackouts, as it is possible that multiple random outages happen at the same time. Real world blackouts typically occur as a consequence of a complicated sequence of very unlikely events, as it was the case for the 2003 blackout in the United States and Canada [7].

3.2 Load shedding

If the power demand in an area becomes greater than the supply, it is necessary to cut off certain loads from the system. The load shedding is mainly triggered by automatic protection systems like *underfrequency* and *undervoltage* relays. Underfrequency devices will for instance automatically shed load if the frequency drops to a predefined level or with a predefined rate of change. Additionally the power system operator is able to manually shed load in order to bring the transmission system back into a stable state [6].

3.3 Blackout model

Our blackout model is based on the DC power flow model and only accounts for the steady-state evolution of the system. The only failures we model are the random initial failures and the line failures due to overload. In the following subsections, we describe the different parts of our blackout model in detail.

3.3.1 Flow model

We model the power transmission grid by an undirected graph $G = (V, E)$, where $V = \{1, \dots, n\}$ is the set of buses and $E \subseteq \binom{V}{2}$ the set of transmission lines. We denote the cardinality of the set of buses by n and the cardinality of the set of lines by m . For each node $i \in V$ we are given the real power injected p_i . For each line $e = \{i, j\} \in E$ we are given the reactance $x(e)$ as well as short and long term power limits $u^s(e)$ and $u^l(e)$ respectively.

We use the DC power flow model from section 2.2 in order to compute the power flow on the transmission lines. First, we have to set up the admittance matrix \mathbf{B} , where the entries of the matrix are given by:

$$B_{ij} = \begin{cases} \sum_{k:\{i,k\} \in E} \frac{1}{x(\{i,k\})} & \text{if } i = j \\ -\frac{1}{x(\{i,j\})} & \text{if } i \neq j \text{ and } \{i,j\} \in E \\ 0 & \text{if } i \neq j \text{ and } \{i,j\} \notin E \end{cases}$$

Together with the vector $\mathbf{p} = (p_1, \dots, p_n)$ of net power injections, we can write the DC power flow equations:

$$\mathbf{p} = \mathbf{B}\theta$$

where $\theta = (\theta_1, \dots, \theta_n)$ is the vector of voltage angles. The matrix \mathbf{B} is singular by construction. To make the system solvable, we remove one row of the system of equations, choose the bus associated with that row as our reference bus r , and set $\theta_r = 0$. From the vector θ of voltage angles we directly get the power flow on each line as indicated in section 2.2. We denote the set of loads and the set of generators by V_L and V_G respectively.

3.3.2 Initial failure

The initial failure is modeled by an independent probability of failure ρ for each line. The number of initially failed lines therefore follows a binomial distribution with expectation ρn and variance $n\rho(1 - \rho)$.

3.3.3 Load shedding

In response to an outage, it may be necessary to shed load in order to bring the system back to a stable state. This process involves operator decisions as well as system protection devices, as described in section 3.2. We model this process by a linear program that tries to minimize the load shedding given the line limits and the DC power flow equations.

$$\min \sum_{i \in V_L} (p'_i - p_i)$$

subject to the DC flow equations for some vector θ of voltage angles

$$(p'_1, \dots, p'_n)^\top = \mathbf{B}(\theta_1, \dots, \theta_n)^\top$$

and to the line flow limits

$$-u^s(\{i, j\}) \leq \frac{\theta_i - \theta_j}{x(\{i, j\})} \leq u^s(\{i, j\}) \quad \forall i, j \in V$$

and the requirement that the reduced load must lie between the original load and zero

$$p_i \leq p'_i \leq 0 \quad \forall i \in V_L$$

and the generator limits

$$0 \leq p'_i \leq p_i \quad \forall i \in V_G$$

The solution vector (p'_1, \dots, p'_n) of this linear program serves as the new vector of net power.

3.3.4 Cascading failures

We model the line failures due to overload by an independent probability of failure $\sigma(e)$ for each line. The probability is zero for lines which carry a flow below the long term limit and increases linearly from 0 to 1 between the long and the short term limit:

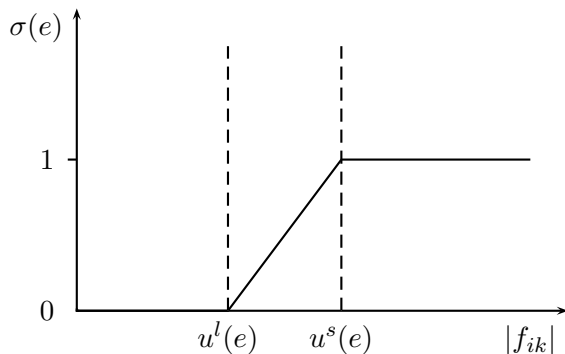
$$\sigma(\{i, j\}) = \begin{cases} 0 & \text{if } |f_{ij}| \leq u^l(\{i, j\}) \\ \frac{|f_{ij}| - u^l(\{i, j\})}{u^s(\{i, j\}) - u^l(\{i, j\})} & \text{if } |f_{ij}| > u^l(\{i, j\}) \end{cases}$$

where $f_{ij} = \frac{\theta_i - \theta_j}{x(\{i, j\})}$ is the power flow from bus i to bus j . See figure 1 for an illustration.

This function models the different reasons for line failures due to overload. Lines that would be loaded above their short term limit in reality, will be forced to be right at the short term limit by the linear program.

According to our function, the failure probability will be equal to 1 for such a line. If a line is loaded above its long term limit but below its short term limit, it may start to overheat, increasing the risk of a flash over towards the ground. This risk is modeled by the linear increase of the failure probability between the long and the short term emergency limit.

Figure 1: The loading dependence of the failure probability $\sigma(e)$ of line $e = \{i, k\}$.



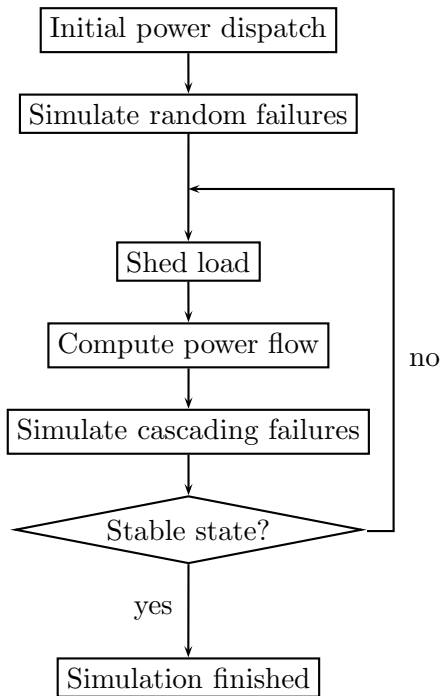
3.3.5 Simulation procedure

We are now able to describe the complete simulation procedure. We start with a feasible power dispatch. This dispatch is expected to be $N - 1$ secure, but may already form a critical loading. First, we simulate the initial failure by flipping a biased coin for each line. Each line fails independently with probability ρ . After this initial event, the power demand and supply may not be in balance anymore and we shed some load using the linear program from section 3.3.3. We then compute the power flow for the new power dispatch and calculate the resulting probability of failure $\sigma(e)$ for each line $e \in E$. For each line with nonzero probability of failure, we flip a biased coin in order to decide whether the line fails or not. If no line was tripped in three consecutive iterations we decide that the system has reached a stable state and we stop the simulation run. See figure 2 for a flow chart of the simulation procedure.

3.4 Computational results

We have performed numerous simulations with our blackout model. We try to interpret the results and compare them with the results found in the literature.

Figure 2: Blackout simulation model.



3.4.1 The reliability test system

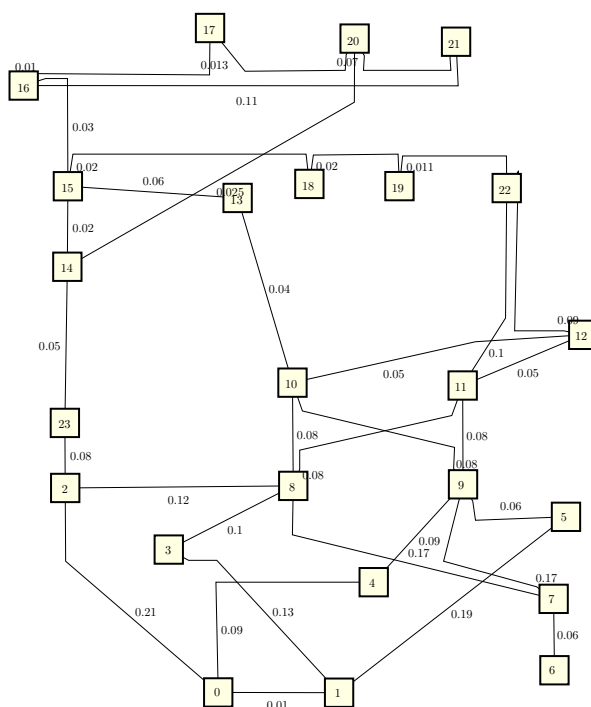
We have performed our simulations on the three area version of the IEEE Reliability Test System 1996 [4]. This artificial test system has 72 buses and 108 lines and was designed for power system reliability evaluation studies. It is composed of three identical copies of the grid shown in figure 3. An advantage of the Reliability Test System is that we have complete bus and line information. Especially we are given short and long term emergency limits for all lines, which is important for our simulation. Furthermore, we have a realistic initial power dispatch. As the DC power flow model does not incorporate any power losses, we have to slightly adjust the generation values of the given power dispatch such that the total power supply matches the demand. We have done this by scaling down the output of all generators by an appropriate factor.

We have simulated different levels of system loading by increasing the power demand and supply at the same rate. We denote this rate by *load factor* in the following. Equivalently we could have reduced the line limits by the inverse factor. The effect would be the same, namely that the operat-

ing margin, i.e. the difference between the capacity and the flow, decreases for every line.

Another parameter that we can vary is the initial failure probability ρ . However, the value of this parameter is closely related to the loading factor mentioned above. If the initial loading is very low, the number of initial failures has to be very high in order to start a cascade. Conversely, for a heavily loaded system, a very low number of initial failures will suffice. We have used $\rho = 0.01$ for all simulations.

Figure 3: One area of the Reliability Test System 1996.

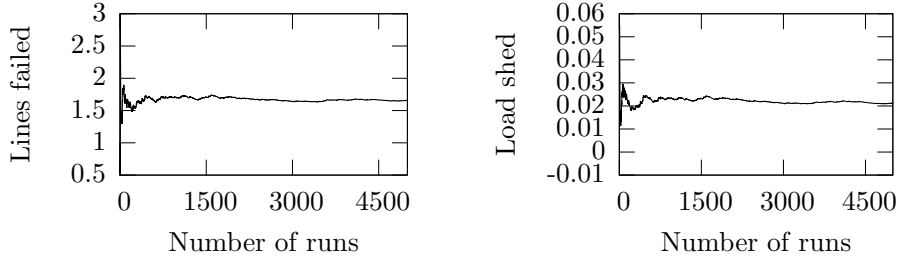


3.4.2 Simulation results

We have mainly looked at two different measures for blackout size. The first one is the number of failed lines at the end of a simulation run. The second one is the fraction of load shed at the end of a run. First, we have checked how fast these two measures converge. We have performed 5000 simulation runs and updated the average of each of the two measures after each run. Figure 4 shows that the average converges quickly for both measures.

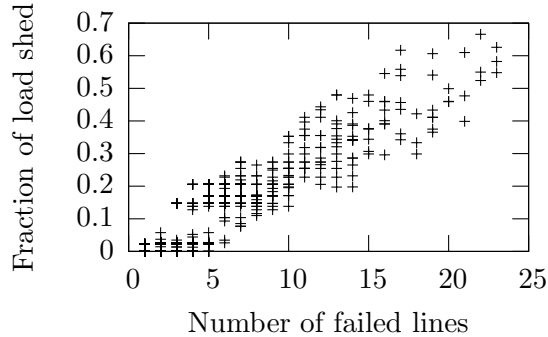
We expect that the two measures are closely correlated. In order to es-

Figure 4: Convergence of average values.



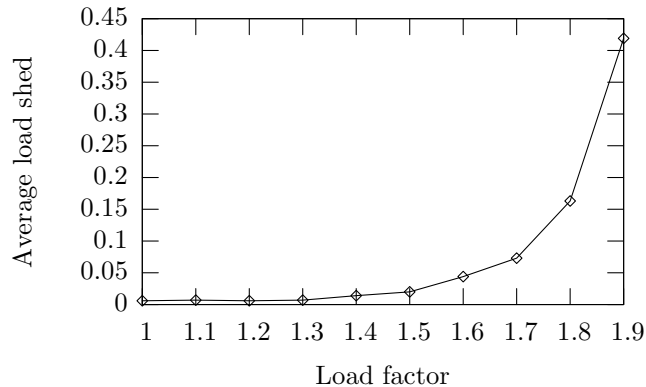
to estimate the actual correlation, we have run the simulation 5000 times and stored the number of failed lines and the fraction of load shed for every run. We have visualized this data in the scatterplot shown in figure 5, with the number of failed lines on the x-axis and the fraction of load shed on the y-axis. The two measures are closely correlated as expected. Nevertheless, there is quite a big range of load shed for a specific number of failed lines. This indicates that there are lines of different importance for the overall reliability of the power transmission grid.

Figure 5: Scatterplot of the number of failed lines and the fraction of load shed.



Furthermore, we have looked at the average blackout size for different loading factors. We have argued in section 3.1 that the average blackout size strongly increases at a critical loading. For this purpose we have calculated the average fraction of load shed over 5000 runs for the loading factors $1, 1.1, \dots, 1.9$. From figure 6, we see that the average fraction of load shed indeed strongly increases around a loading factor of 1.5. We conclude that the loading corresponding to this loading factor can be seen as critical. For simulations at low and high loading, we will use the loading factors 1 and 1.9 respectively.

Figure 6: Loading dependence of blackout size, measured as average load shed.



We have as well created histograms for both the number of failed lines and the fraction of load shed. The histograms are each based on 10000 simulation runs and serve as an estimation of the probability density function for the two measures. Figures 7 through 9 show log-log plots of the estimated pdf of the number of failed lines and log-linear plots of the estimated pdf of the fraction of load shed at low, critical and high loading. We compare the estimated pdf of the number of failed lines at the different loading levels. The distribution at low loading has an exponential tail, as expected. The estimated pdf at critical loading is linear over several orders of magnitude in the log-log plot. This confirms the assumption of a power law dependence, as $f(x) = x^{-k}$ corresponds to a straight line with slope $-k$ in the log-log space. The estimated pdf at high loading shows that large blackouts occur with high probability.

Figure 7: Blackout size at low loading.

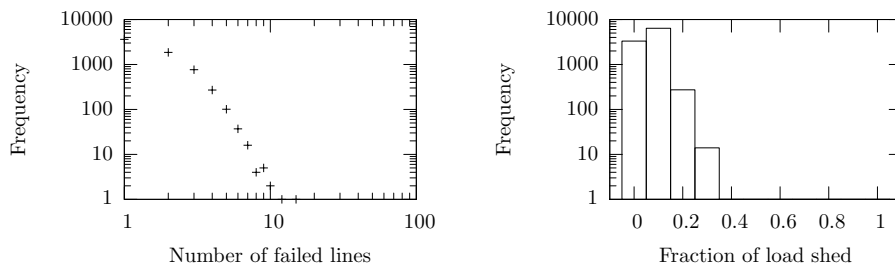


Figure 10 shows a histogram of the line loadings, i.e. the fraction between the flow and the capacity, at critical loading before the initial failures. We can see that there are several lines which are loaded up to a fraction of 0.8 of their long term emergency limit. The operating margin for these lines may

Figure 8: Blackout size at critical loading.

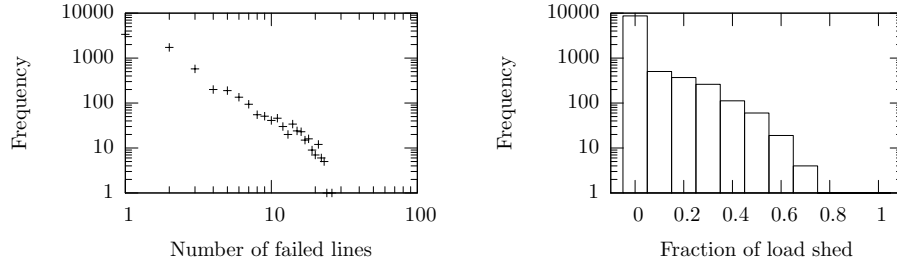
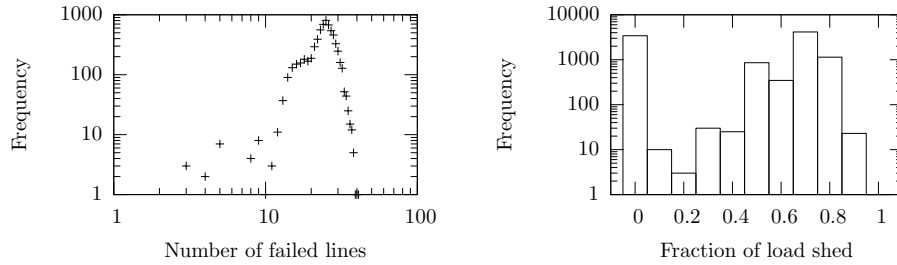
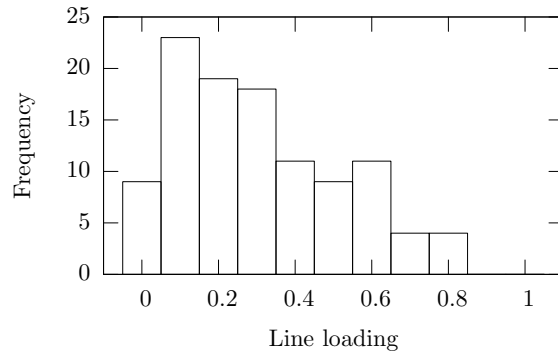


Figure 9: Blackout size at high loading.



not be large enough anymore to compensate the random initial failures.

Figure 10: Line loading, i.e. the fraction between flow and capacity, at critical loading.



4 Repair problems

Our final goal is to find strategies to react to emerging failures that minimize the blackout risk. As mentioned in the introduction, it is not clear how to analytically quantify the blackout risk for a specific state of a power transmission system. Besides, there is basically no theoretical framework for DC power flows.

Hence, we first consider some closely related problems based on the network flow model described in section 2.1. Network flows provide an abstraction for any kind of flow and have been studied extensively in theory [12]. Even though it is not clear if results for network flows are directly applicable to power transmission grids, we will at least get a better understanding of our original problem.

We will start by considering a network where some of the edges are not operating, which is modeled by setting their capacity temporarily to zero. For some value of flow, the minimum cut represents a critically loaded region of the network. The minimum cut is critical in the sense that if we continuously increase the value of the flow, it will be the edges of the minimum cut which reach their capacity limit first, and can then be considered as overloaded. As our final goal is to minimize the risk of failures due to overload, a natural objective is to repair the set of lines that maximizes the minimum cut, or equivalently that maximizes the value of a maximum flow, constraint to some budget. The budget models the fact that we cannot repair all the edges for the short time, due to limited repair capacities. We will call this problem the *static repair problem* and discuss it in section 4.1.

In section 4.2, we extend the problem and ask for an ordered set of edges to repair. A repair team may not be able to repair the whole set of edges at the same time, but in a sequential order. The order on the set of edges will allow to schedule the repair activities in an optimal way. The objective in this extended problem is to maximize the sum of the maximum flows over time. We will refer to this problem as the *dynamic repair problem*.

In section 4.3 we consider again the static repair problem, but this time based on the DC power flow model. That is, given a transmission grid we try to find the set of edges that maximizes the value of the maximum DC power flow.

All problems will show to be \mathcal{NP} -hard. We will prove their hardness and define and analyze simple approximation algorithms for all three problems.

4.1 Static repair problem

First, we have to extend the model from section 2.1 in order to incorporate the cost for repairing an edge. For this purpose we assign a fixed cost $c(e)$ to every edge and express the cost of a flow f by $\sum_{e \in E: f(e) > 0} c(e)$. The static repair problem can now be stated as follows. Given a network $G = (V, E)$ and a partition of E into two subsets E_f and E_o of failed and operating edges respectively, a static repair strategy has to decide on a set of edges $E_r \subseteq E_f$ to repair constraint to the budget $\sum_{e \in E_r} c(e) \leq C$. The goal is to repair the set of edges that maximizes the flow in the resulting graph $G = (V, E_o \cup E_r)$. Setting the cost of a all operating edges to zero, this is equivalent to the following network design problem.

Definition 1. (MAXFLOWFIXEDCOST) *Given a directed graph $G = (V, E)$ with capacities $u(e) \in \mathbb{Z}^+$ and cost $c(e) \in \mathbb{Z}_0^+$ associated with every edge $e \in E$, distinguished vertices s and t and a budget $C \in \mathbb{Z}^+$, find a subset $A \subseteq E$ of cost $\sum_{e \in A} c(e) \leq C$ such that the value of a maximum flow from s to t is maximal in $G_A \equiv (V, A)$.*

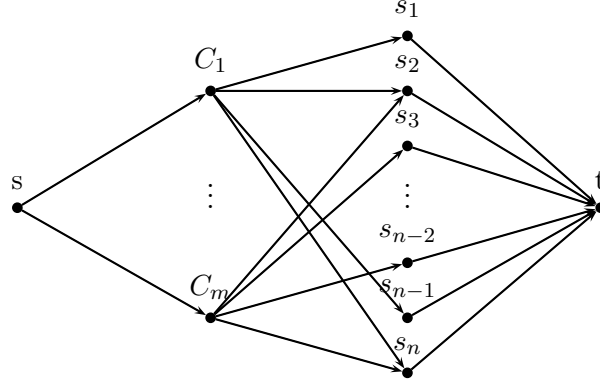
The decision version of this problem is well known (ND32 in [13]) and has already been proven to be strongly \mathcal{NP} -complete. We give a proof based on a reduction from the MINSETCOVER problem below.

4.1.1 Hardness

Theorem 1. *The decision problem corresponding to MAXFLOWFIXEDCOST is strongly \mathcal{NP} -complete, even for $c(e) \in \{0, 1\}$.*

Proof. We proof the theorem by a reduction from MINSETCOVER. Given an instance of MINSETCOVER, that is a collection \mathcal{C} of subsets of a finite set S , we construct the following auxiliary network. We introduce a vertex for every subset $C_i \in \mathcal{C}$ and for every element $s_i \in S$ and additional source and sink vertices s and t . We first construct a bipartite graph with the subset vertices on one side, and the element vertices on the other side as shown in figure 11. We add an edge (C_i, s_j) of capacity 1 and zero cost if and only if $s_j \in C_i$. Finally, we connect the source with the vertices C_i by edges of capacity $|S|$ and unit cost and the the vertices s_i with the sink by edges of unit capacity and zero cost. It is now easy to see, that a set cover of size k can be transformed into a flow f with $val(f) = |S|$ and cost k and vice versa. \square

Figure 11: The reduction from MINSETCOVER to MAXFLOWFIXEDCOST



We can state the MAXFLOWFIXEDCOST problem as the following mixed integer programming problem.

$$\max F$$

such that

$$\sum_{u:(v,u) \in E} f(v,u) - \sum_{u:(u,v) \in E} f(u,v) = \begin{cases} F & \text{if } v = s \\ -F & \text{if } v = t \\ 0 & \text{if } v \in V \setminus \{s, t\} \end{cases}$$

$$0 \leq f(e) \leq x(e)c(e) \quad \forall e \in E$$

$$\sum_{e \in E} x(e)c(e) \leq C$$

$$x(e) \in \{0, 1\} \quad \forall e \in E$$

Even though mixed integer programming problems cannot be solved efficiently in general, this formulation may help to evaluate the quality of the approximation algorithms developed below.

Prior to stating a result about the hardness of approximating MAXFLOWFIXEDCOST, we have to recall a result about the MAXCOVER problem. The MAXCOVER problem is closely related to MINSETCOVER. But instead of covering all the elements with the minimum number of subsets, we are given an integer k and try to maximize the number of covered elements using at most k subsets.

Theorem 2. *Unless $NP \subseteq DTIME(n^{O(\log \log n)})$, there is no approximation algorithm with ratio better than $(1 - \frac{1}{e})$ for the MAXCOVER problem of size n .*

A proof of this theorem is given in [17]. We can directly establish a hardness result for the approximation of the MAXFLOWFIXEDCOST problem by using the same reduction as in the proof of theorem 1.

Corollary 1. *Unless $NP \subseteq DTIME(n^{O(\log \log n)})$, there is no approximation algorithm with ratio better than $(1 - \frac{1}{e})$ for the MAXFLOWFIXEDCOST problem.*

Proof. Assume that we have an approximation algorithm for MAXFLOWFIXEDCOST with ratio $\alpha > (1 - \frac{1}{e})$. Given an instance of the MAXCOVER-AGE problem, we construct the auxiliary network G as in the proof of theorem 1. We then apply our α -approximation algorithm with budget constraint C equal to k to the network G . By assumption, we get a solution $E' \subseteq E$ that allows a maximum flow f with $val(f) \geq \alpha F_{opt}$, where F_{opt} is the value of an optimal solution. This directly implies, that the set $\{C_i | (s, C_i) \in E'\}$ covers at least $\alpha |S_{opt}| > (1 - \frac{1}{e}) |S_{opt}|$ elements, where S_{opt} is the optimal solution for MAXCOVER-AGE. Thus we have constructed an α -approximation algorithm for the MAXCOVER-AGE problem, which is impossible unless $NP \subseteq DTIME(n^{O(\log \log n)})$ by theorem 2. \square

This implies that there is no polynomial time approximation scheme (PTAS) for the MAXFLOWFIXEDCOST problem. We will therefore concentrate on constant-factor approximation algorithms and restricted versions of the problem in the following.

4.1.2 Special cases

In case of a unit capacity network, i.e. $u(e) = 1$ for all edges $e \in E$, the decision version of the MAXFLOWFIXEDCOST is in \mathcal{P} . In order to proof that claim we state an algorithm that runs in polynomial time. First, we recall a property of the MINCOSTFLOW problem. In the MINCOSTFLOW problem, we are given a network $G = (V, E)$ with a cost per unit of flow $c(e)$ associated with every edge $e \in E$ and a demand F and we are looking for a flow f with value F and minimum cost $C = \sum_{e \in E} f(e)c(e)$. For a more rigorous definition of the problem and a proof of the following lemma 1, see for instance [12].

Lemma 1. *If all edge capacities and the demand are integer, the MINCOSTFLOW problem always has an integer minimum cost flow.*

For the special case of a unit capacity network, this implies that we have $f(e) \in \{0, 1\}$ for every edge $e \in E$. In this case, a solution to the minimum cost flow problem directly corresponds to the set of edges $E_{mc} \equiv \{e \in E | f(e) = 1\}$ and therefore to a solution for the MAXFLOWFIXEDCOST problem.

Lemma 2. *Given a unit capacity network $G = (V, E)$ with costs $c(e)$ associated with every edge $e \in E$ and a solution f_{mc} for MINCOSTFLOW with demand F and cost C , the set $A = \{e \in E | f(e) = 1\}$ is a solution for MAXFLOWFIXEDCOST with value F and cost C . Conversely, given a solution A for MAXFLOWFIXEDCOST with value F and cost C , the flow f with*

$$f(e) = \begin{cases} 1 & e \in A \\ 0 & e \notin A \end{cases}$$

is a solution for MINCOSTFLOW with demand F and cost C .

Proof. (\Rightarrow) Let f be a solution for the MINCOSTFLOW problem with demand F and cost C . The set $A = \{e \in E | f(e) = 1\}$ is a solution for MAXFLOWFIXEDCOST with value F and cost C , as $G_A \equiv (V, A)$ allows the flow f_{mc} with cost $\sum_{e \in A} c(e) = \sum_{e \in E} f(e)c(e) = C$ by assumption.

(\Leftarrow) Let A be a solution for the MAXFLOWFIXEDCOST problem with cost C and value F . This implies that there is a flow f with cost C and value F on $G_A \equiv (V, A)$ and therefore also on $G = (V, E)$ as $A \subseteq E$. \square

We are now able to state the claimed polynomial time algorithm for MAXFLOWFIXEDCOST. The idea is to find the maximal integer demand F such that the corresponding solution for MINCOSTFLOW has costs less than or equal to C . As shown in lemma 2, we can directly transform this solution for the MINCOSTFLOW problem into an optimal solution for the MAXFLOWFIXEDCOST problem with cost constraint C .

The maximal integer demand F for MINCOSTFLOW that allows a solution with costs less than or equal to C can be determined in the following way. First, compute a maximum flow f' with cost $\sum_{e \in E} f'(e)c(e) \leq C$, which can be done by solving a linear program. Second, we use the integer part of the value of flow f' , i.e. $F_{max} = \lfloor val(f') \rfloor$, as demand value for the MINCOSTFLOW problem. This value is maximal in the sense described above, as:

- i. the costs of the MINCOSTFLOW with target flow F_{max} are less than or equal to C .
- ii. the costs of the MINCOSTFLOW with target flow $F_{max} + 1 > val(f')$ are greater than C .

Hence, we can state the following theorem.

Theorem 3. *The constructive problem corresponding to MAXFLOWFIXEDCOST restricted to unit capacity networks is in \mathcal{P} .*

4.1.3 Approximation algorithm

We consider again the general case with arbitrary integer capacities. We present an algorithm that is based on the optimal solution for the unit capacity case. We denote the minimum and maximum capacities in the network by U_{min} and U_{max} respectively, i.e. $U_{min} = \min_{e \in E} u(e)$ and $U_{max} = \max_{e \in E} u(e)$. Given a network $G = (V, E)$ with arbitrary integer capacities, we set all capacities to U_{min} , but leave the cost function unchanged. We then compute the optimal solution for the MAXFLOWFIXEDCOST problem on this unit capacity network. This solution is a feasible solution for the original network G as well, and we analyze its quality below.

Algorithm 1 MaxFlowFixedCostApprox(G,C)

Require: Graph $G = (V, E)$ with cost function $c : E \rightarrow \mathbb{Z}_0^+$, budget C
Ensure: An U_{min}/U_{max} -approximation for MAXFLOWFIXEDCOST
 $f' \leftarrow$ maximum flow with costs less than or equal to C ;
 $F_{max} \leftarrow \lfloor val(f') \rfloor$;
 $f \leftarrow$ minimum cost flow with demand F_{max} ;
return $E = \{e \in E \mid f(e) = 1\}$

Theorem 4. *Algorithm 1 is an $\frac{U_{min}}{U_{max}}$ -approximation algorithm for MAXFLOWFIXEDCOST, where $U_{min} \equiv \min_{e \in E} u(e)$ and $U_{max} \equiv \max_{e \in E} u(e)$.*

Proof. Let A_{opt} be an optimal solution for MAXFLOWFIXEDCOST, with cost C and value F_{opt} . By setting all capacities to U_{min} , the capacity of any cut is reduced at most by a factor of $\frac{U_{min}}{U_{max}}$. On this unit capacity network, the set A_{opt} is thus a solution for MAXFLOWFIXEDCOST with cost C and a value of at least $F_{opt} \frac{U_{min}}{U_{max}}$. Therefore, the value of an optimal solution cannot be worse than that. \square

4.2 Dynamic repair problem

Given a graph $G = (V, E)$ and a partition of E into two subsets E_f and E_o of failed and operating edges and a budget C , we are not only interested in a set E_r of edges to repair, but also in an order on that set. The goal is to achieve a maximum flow over time. That means we sum up the values of the maximum achievable flows after each repair step. Setting $E_f = E$, this corresponds to the following new optimization problem.

Definition 2. (MAXSUMFLOW) *Given a directed graph $G = (V, E)$ with capacities $u(e) \in \mathbb{Z}^+$ and costs $c(e) \in \mathbb{Z}_0^+$ associated with every edge $e \in E$, distinguished vertices $s \in V$ and $t \in V$, a budget $C \in \mathbb{Z}^+$ and a time span $T \in \mathbb{Z}^+$, find a subset $A \subseteq E$ of cost $\sum_{e \in A} c(e) \leq C$ and an order $\pi : A \rightarrow \{1, \dots, |A|\}$ such that $\sum_{i=1}^T MaxFlow(G_i, s, t)$ is maximized, where*

$G_i \equiv (V, A_i)$ for $A_i \equiv \{e \in A \mid \pi(e) \leq i\}$ and $MaxFlow(G, s, t)$ is the value of a maximum flow between s and t in G .

In case that the set of operating edges E_o is not empty at the beginning, we simply replace A_i by $A'_i \equiv A_i \cup E_o$, G_i by $G'_i \equiv (V, A'_i)$ and define the rest as above. Furthermore, we could model different repair times $t(e) \in \mathbb{Z}^+$ by splitting up each edge into $t(e)$ subedges with cost equal to $\frac{c(e)}{t(e)}$. In the following, we will restrict ourselves to unit cost networks with budget $C = T$.

First, we want to emphasize that the optimal solution to MAXFLOW-FIXEDCOST is not necessarily identical to the set of edges repaired in an optimal solution for MAXSUMFLOW. Figure 12 provides a very simple example where the sets actually differ. Given the budget $T=4$, the optimal solution for MAXFLOWFIXEDCOST are the upper four edges, leading to a solution of value equal to 2 for MAXSUMFLOW for any order on that set, as indicated in figure 13. The optimal solution for the MAXSUMFLOW problem, however, is the set of the lower two edges and has a value equal to 3, for any order. Nevertheless, in general the optimal solution for MAXFLOW-FIXEDCOST combined with a clever order provides a good approximation for MAXSUMFLOW as we will proof in theorem 6.

Figure 12: Dynamic versus static repair problem.

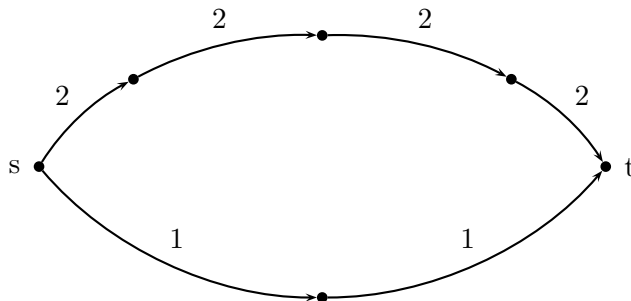
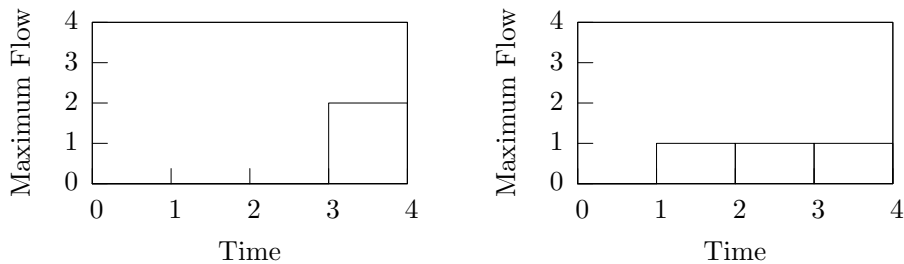


Figure 13: The values of the two different solutions for MAXSUMFLOW.



We can find an optimal solution for MAXSUMFLOW by solving the following mixed integer programming problem.

$$\max \sum_{i=1}^T F_i$$

such that

$$\sum_{u:(v,u) \in E} f_i(v,u) - \sum_{u:(u,v) \in E} f_i(u,v) = \begin{cases} F_i & \text{if } v = s \\ -F_i & \text{if } v = t \\ 0 & \text{else} \end{cases} \quad i \in \{1, \dots, T\}$$

$$0 \leq f_i(e) \leq x_i(e)c(e) \quad \forall e \in E, i \in \{1, \dots, T\}$$

$$x_i(e) \leq x_{i+1}(e) \quad \forall e \in E, i \in \{1, \dots, T-1\}$$

$$\sum_{e \in E} x_i(e) \leq i \quad i \in \{1, \dots, T\}$$

$$x_i(e) \in \{0, 1\} \quad \forall e \in E, i \in \{1, \dots, T\}$$

4.2.1 Hardness

We show that the decision problem corresponding to MAXSUMFLOW is strongly \mathcal{NP} -complete, using a reduction from the Hamiltonian path problem.

Theorem 5. *The decision problem corresponding to MAXSUMFLOW is strongly \mathcal{NP} -complete.*

Proof. We show a reduction from the directed Hamiltonian path problem. We construct an auxiliary graph, that allows a particular solution for MAXSUMFLOW if and only if the original graph contains a Hamiltonian path, as shown in figure 14. Let $G = (V, E)$ be the original graph and let $n = |V|$ be the number of vertices. The vertex set of the auxiliary graph is composed of n copies of the vertex set V , n auxiliary vertices t_1, \dots, t_n and source and sink vertices s and t . Let $V^i = \{v_1^i, \dots, v_n^i\}$ be the i th copy of the vertex set. For every edge $e = (v_k, v_l)$ in the original graph we introduce an edge $e = (v_k^i, v_l^{i+1})$ with capacity $u(e) = n + 1$ for $i = 1, \dots, n - 1$ in the auxiliary graph. Furthermore, we add edges $e = (v_k^i, t_k)$ of unit capacity for all copies of every vertex v_k . Finally, we add an edge of capacity $u(e) = n + 1$ between the source s and the first copy of each node and between the last copy of each node and the sink t and edges of unit capacity between the auxiliary vertices t_i and the sink t . More formally, the network $G_{aux} = (V_{aux}, E_{aux})$

is defined as follows.

$$\begin{aligned}
V_{aux} &= \{v_k^i | v_k \in V, i = 1, \dots, n\} \cup \{t_k | k = 1, \dots, n\} \cup \{s, t\} \\
E_{aux} &= E_{int} \cup E_{ext} \\
E_{int} &= \{(v_k^i, v_l^{i+1}) | (v_k, v_l) \in E, i = 1, \dots, n-1\} \\
&\quad \cup \{(s, v_i^1) | i = 1, \dots, n\} \cup \{(v_i^n, t) | i = 1, \dots, n\} \\
E_{ext} &= \{(v_k^i, t_k) | v_k \in V, i = 1, \dots, n\} \cup \{(t_k, t) | k = 1, \dots, n\}
\end{aligned}$$

The capacity function $u_{aux} : E_{aux} \rightarrow \mathbb{Z}$ can be written as follows:

$$u(e) = \begin{cases} 1 & \text{if } e \in E_{ext} \\ n+1 & \text{if } e \in E_{int} \end{cases}$$

We will show that there is a solution for MAXSUMFLOW in the network G_{aux} with budget $T = n-1$ and $E_o = E_{ext} \cup \{(s, v_i^1) | i = 1, \dots, n\} \cup \{(v_i^n, t) | i = 1, \dots, n\}$ with a value of $\sum_{i=1}^{n+1} i = \frac{(n+1)(n+2)}{2}$ if and only if the original graph contains a Hamiltonian cycle.

(\Rightarrow) Let A_{opt} with ordering π_{opt} be a solution to the MAXSUMFLOW problem on the network defined above. The only way to achieve a solution with a value of $\sum_{i=1}^{n+1} i$ is to increase the flow by one with every repair operation. This implies that in every step, a copy of a new vertex gets connected to the source, as the capacity of the edges (t_i, t) is equal to one. This is only possible if the solution corresponds to a Hamiltonian path in the original graph.

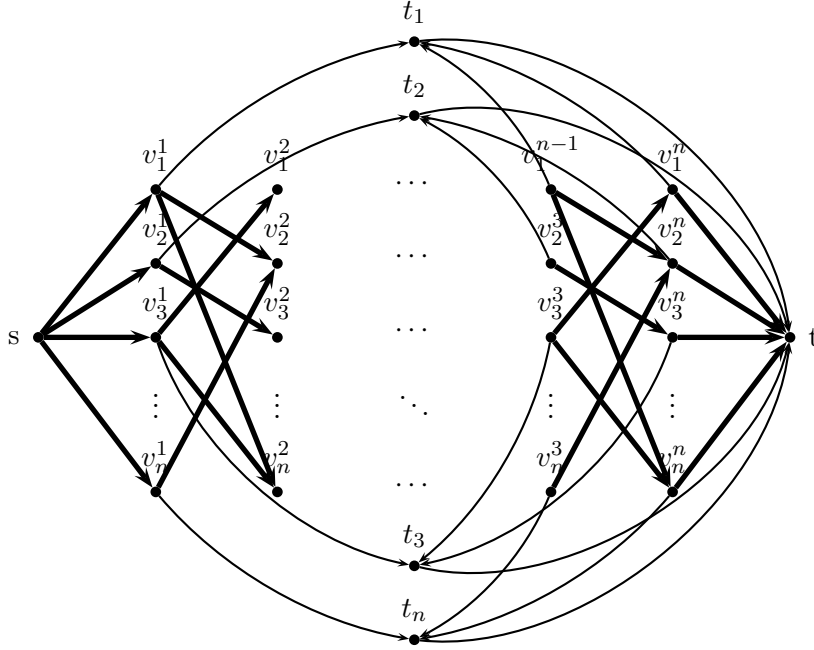
(\Leftarrow) Let $h : V \rightarrow \{1, \dots, n\}$ be a Hamiltonian path. We construct the a solution to the MAXSUMFLOW problem with $A = \{(v_k^i, v_l^{i+1}) \in E | h(v_k) = i \wedge h(v_l) = i+1\}$ and the order $\pi((v_k^i, v_l^{i+1})) = h(v_k)$. This solution forms an st -path in the auxiliary graph and has the claimed value. \square

4.2.2 Approximation algorithm

For the MAXSUMFLOW problem, the unit capacity case appears to be difficult as well, even though we do not provide a hardness proof. Instead, we state an approximation algorithm and give some evidence for the difficulty of the problem.

The basic idea for our first approximation algorithm is to start with an optimal solution for the MAXFLOWFIXEDCOST problem with the same budget constraint and to define a reasonable order on that set.

Figure 14: The reduction from TSP to MAXSUMFLOWFIXEDCOST



Theorem 6. *Algorithm 2 is a $\frac{1}{2} \left(1 - \frac{1}{F}\right)^2$ -approximation for the MAXSUMFLOW problem restricted to unit capacity networks, where F is the value of an optimal solution for MAXFLOWFIXEDCOST.*

Proof. In unit capacity networks, the value of a maximum flow is equal to the number of edge disjoint paths. As a consequence, we can represent each maximum flow by a set \mathcal{P} of edge disjoint paths. We denote by F the value of the solution to the MAXFLOWFIXEDCOST with constraint T . Let f_i be the value of the flow after inserting the i th path P_i and t_i the number of edges inserted so far. We can express the slope of the line connecting the points (t_i, f_i) and (T, F) as $a_i = \frac{F - f_i}{T - t_i}$. We claim that the piece-wise linear curve connecting the points (t_i, f_i) is a concave function, which allows us to give a lower bound for the corresponding solution of MAXSUMFLOW.

Algorithm 2 SumFlow(G, T)

Require: Graph $G = (V, E)$, time budget T

Ensure: A solution for MAXSUMFLOW

$A_{OPT} \leftarrow \text{MaxFlowFixedCostApprox}(G, T)$;

$A \leftarrow A_{OPT}$;

$i \leftarrow 0$;

while A not empty **do**

$P \leftarrow$ shortest path in $G = (V, A)$;

$A \leftarrow A \setminus P$;

for all edge $e \in P$ **do**

$\pi(e) = i$;

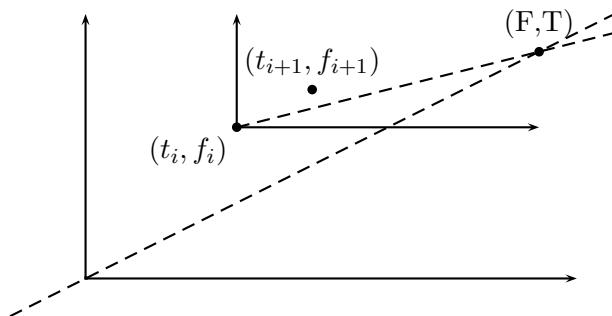
$i \leftarrow i + 1$;

end for

end while

return (A_{OPT}, π) ;

Figure 15: Concavity of the pice-wise linear function.



In order to proof that claim we show that given a point $p_i = (t_i, f_i)$ with corresponding slope a_i , the next point $p_{i+1} = (t_{i+1}, f_{i+1})$ lies above the line with slope a_i and origin p_i , i.e.

$$\begin{aligned} a_i(t_{i+1} - t_i) + f_i &= a_i \cdot \text{length}(P_{i+1}) + f_i \\ &\leq a_i \frac{T - t_i}{F - f_i} + f_i \\ &= a_i \frac{1}{a_i} + f_i \\ &= f_i + 1 = f_{i+1} \end{aligned}$$

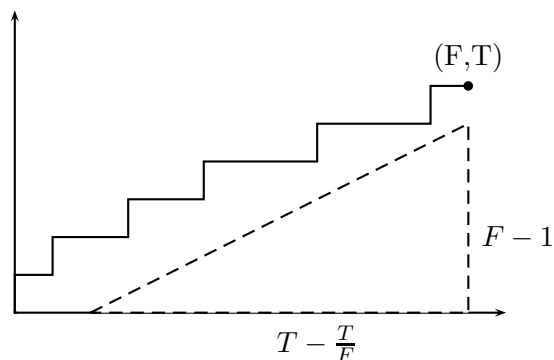
For the second step we used the fact that the shortest path cannot be longer than the length of the average path, which can be expressed by $\frac{T-t_i}{F-f_i}$ in unit

capacity network. See figure 15 for an illustration of this proof. We can now use the fact that the value of the corresponding solution to the MAX-SUMFLOW problem is lower bounded by the area of the triangle indicated in figure 16. We can express this area as:

$$\frac{1}{2} \left(T - \frac{T}{F} \right) (F - 1) = \frac{FT}{2} \left(1 - \frac{1}{F} \right)^2$$

Together with the upper bound of FT for the optimal solution we get the claimed approximation ratio of $\frac{1}{2} \left(1 - \frac{1}{F} \right)^2$ which converges to $\frac{1}{2}$ for $F \gg 0$. \square

Figure 16: A lower bound for the approximation algorithm 2.

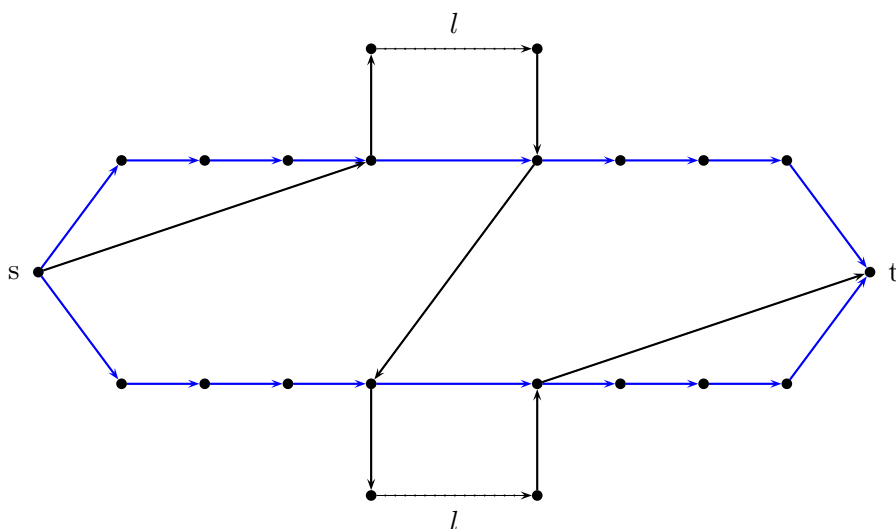


We have to make two remarks concerning the algorithm 2 and the proof of theorem 6.

First, we did not specify how to extract the path set \mathcal{P} defined in the proof. For the correctness of the proof an arbitrary decomposition into disjoint paths is fine, but it is unclear what an optimal decomposition would be. For instance, it is not optimal to iteratively extract the currently shortest path from the graph as we do in algorithm 2. The network shown in figure 17 is an instance where this strategy is not optimal. All edges are assumed to have unit capacity, and the dotted edges with label l represent a sequence of l edges, where l is some large number. If we first extract the shortest path, the two remaining paths contain each one of the sequences with l edges. If we insert these paths ordered by increasing length we get quickly one unit of flow, but have to wait very long for the second unit. However, if we first insert the two blue paths and then the very long remaining path, we get two units of flow within a reasonable time, leading to a bigger sum of flow over time. We claim that already this *path extraction* problem is hard. That is,

given a graph, it is hard to determine the insertion order that leads to the maximal sum of flows over time. The hardness of this problem would imply the hardness of the MAXSUMFLOW problem as well.

Figure 17: The path extraction problem.



Second, we could slightly improve the ratio to $\frac{1}{2} \left(1 - \frac{1}{F} + \frac{1}{T^2}\right)^2$ by selecting the area for the lower bound more carefully.

Furthermore, we can extend these results to networks with arbitrary integer capacities.

Theorem 7. *Algorithm 2 is a $\frac{U_{min}}{2U_{max}} \left(1 - \frac{U_{max}}{U_{min}F}\right)^2$ -approximation for the MAXSUMFLOW problem, where F is the value of an optimal solution for MAXFLOWFIXEDCOST given the same budget constraint.*

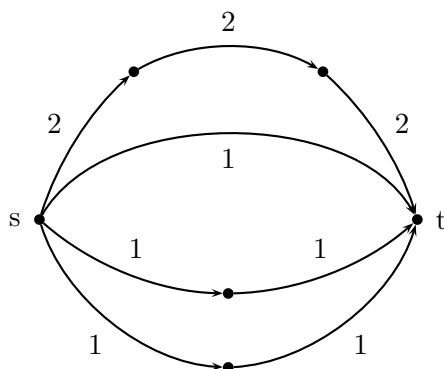
Proof. The value of the optimal solution is still bounded by FT . The difference to the unit capacity case is that we are not anymore able to compute an optimal solution for MAXFLOWFIXEDCOST, but only an $\frac{U_{min}}{U_{max}}$ -approximation. Using the same arguments as above, we can lower bound the value of the approximation by $\frac{FT}{2} \frac{U_{min}}{U_{max}} \left(1 - \frac{U_{max}}{U_{min}F}\right)^2$, proving the claim. \square

4.3 DC repair problem

An optimal solution for the MAXFLOWFIXEDCOST problem maximizes the classical network flow but not necessarily the DC power flow. Let us consider

for instance the artificial network in figure 18, where the labels represent the capacities, and the cost and reactances are assumed to be equal to one for all lines. For the budget $C = 4$, the optimal solution for the MAXFLOW-FIXEDCOST are the upper two paths leading to a maximum flow of value 3. The value of a maximum DC flow using these two paths is only $\frac{4}{3}$ due to the DC flow constraints. The capacity constraint for the path of length 1, i.e. $\theta_s - \theta_t \leq 1$, limits the voltage angle difference to 1 and therefore the flow on the path of length 3 to a value equal to $\frac{\theta_s - \theta_t}{3} \leq \frac{1}{3}$. The optimal solution for the DC flow would be the lower two paths, which allow a flow of value equal to 2.

Figure 18: An artificial network, illustrating the difference between network flow and DC power flow.



We will now define an equivalent version of the MAXFLOWFIXEDCOST problem for DC flows.

Definition 3. (MAXDCFLOWFIXEDCOST) *Given the undirected graph $G = (V, E)$ of a transmission grid with capacities $u(e) \in \mathbb{Z}^+$, costs $c(e) \in \mathbb{Z}_0^+$ and reactances $x(e) \in \mathbb{R}^+$ associated with every line $e \in E$, the set of loads $V_L \subseteq V$ and generators $V_G \subseteq V$ and a budget $C \in \mathbb{Z}^+$, find a subset $A \subseteq E$ of cost $\sum_{e \in A} c(e) \leq C$ such that the maximum DC flow is maximal in $G_A = (V, A)$.*

4.3.1 Hardness

It is not surprising that this problem is \mathcal{NP} -hard as well. We give a reduction from the PARTITION problem that was inspired by [3]. The PARTITION problem is defined as follows. Given a set of integers $C = \{c_1, c_2, \dots, c_n\}$,

with $\sum_{i=1}^n c_i = D$, is there a subset $I \subseteq N = \{1, 2, \dots, n\}$ such that

$$\sum_{i \in I} c_i = \sum_{i \in N-I} c_i = \frac{D}{2}$$

The PARTITION problem is well known to be \mathcal{NP} -complete [13].

Theorem 8. *The decision problem corresponding to MAXDCFLOWFIXEDCOST is \mathcal{NP} -hard.*

Proof. We proof the theorem by a reduction from PARTITION. Given an instance of MINSETCOVER, that is a set of integers $C = \{c_1, c_2, \dots, c_n\}$, we construct an auxiliary graph $G(C)$ as in figure 19, where the labels $l(e)$ correspond to the length of the edges. We call a section of the auxiliary graph corresponding to an element c_i a *block* and denote it by B_i , as indicated in figure 20. We will use the fact, that any two edge-disjoint st-paths in $G(C)$ directly correspond to a partition of the set C . This is true, as any two edge-disjoint paths going through a block B_i must be either use the two parallel or the the two diagonal edges. Therefore, one and only one edge with label c_i is used in the two paths and the two paths naturally correspond to a partition of C into two sets. In particular, two edge-disjoint paths of equal length correspond to a solution for the PARTITION problem.

We can view the auxiliary graph $G(C)$ as a DC network with line reactances $x(e)$ equal to $l(e)$. We set the cost $c(e)$ and capacity $u(e)$ equal to one for all lines. The claim is now that there is a solution to the PARTITION problem if and only if the value of an optimal solution for MAXDCFLOWFIXEDCOST with budget $4n + 2$ is equal to 2.

First, we observe that the value of an optimal solution cannot be greater than 2. Moreover, such a solution must consist of two edge-disjoint paths, as the maximum flow is bounded to 1 else. For two edge-disjoint paths with equal resistance of $x_{tot} = \frac{D}{2}$ we get indeed a flow of value 2. Lets consider a solution with two edge-disjoint paths with equivalent reactance x_1 and x_2 , where $x_1 + x_2 = D$. The capacity constraints limits the maximum voltage angle difference to $x_{min} = \min(x_1, x_2)$. The value of the resulting flow is thus equal to $1 + \frac{\min(x_1, x_2)}{\max(x_1, x_2)}$ which takes its maximum for $\min(x_1, x_2) = \max(x_1, x_2) = \frac{D}{2}$ and is strictly smaller else. This completes the proof as we have shown that given an algorithm for MAXDCFLOWFIXEDCOST, we can efficiently decide the PARTITION problem. □

Figure 19: The reduction from PARTITION to MAXDCFLOWFIXEDCOST.

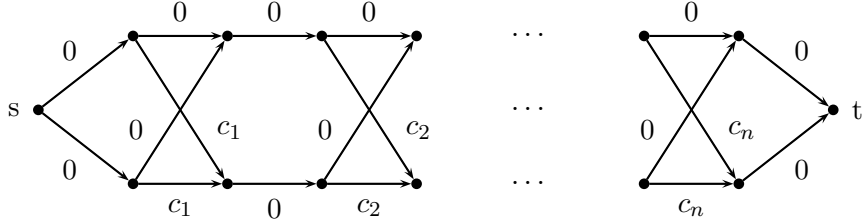
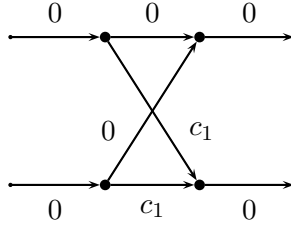


Figure 20: The block B_i .



4.3.2 Maximum flow strategy

We will try to apply algorithm 1, which approximately solves the MAXFLOW-FIXEDCOST problem, to the MAXDCFLOWFIXEDCOST problem. The hope is that an optimal solution for DC power flow is not too different from the one for network flow, even though we have showed that they are not the same in general. However, the value of the minimum cut provides an upper bound on the value of a DC flow solution. This means that a minimum cut of value v is necessary for a DC flow of value v , but as the example based on figure 18 shows, is not sufficient.

Before we can apply algorithm 1, we have to transform the undirected transmission grid with multiple loads and generators into a directed graph with a single source and sink. First, we make the graph directed by replacing each undirected edge $e = \{i, j\}$ by two directed edges $e_1 = (i, j)$ and $e_2 = (j, i)$ with the same cost and capacity as the original edge e . Second, we introduce a super source s and edges from s to all generators and a super sink t with edges from all loads. We set the capacity of an edge (s, v) to the capacity of the generator v and the capacity of an edge (v, t) to the amount of power consumed at load v . If no such bounds are given, we set the capacity to infinity. We set the costs for these auxiliary edges to zero as we do not have to pay for the use of a generator or load. Figures 21 and

22 illustrate the described transformation, where the loads are labeled by l_i and the generators are labeled by g_i .

Figure 21: A simple artificial power transmission grid.

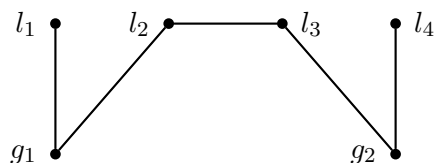
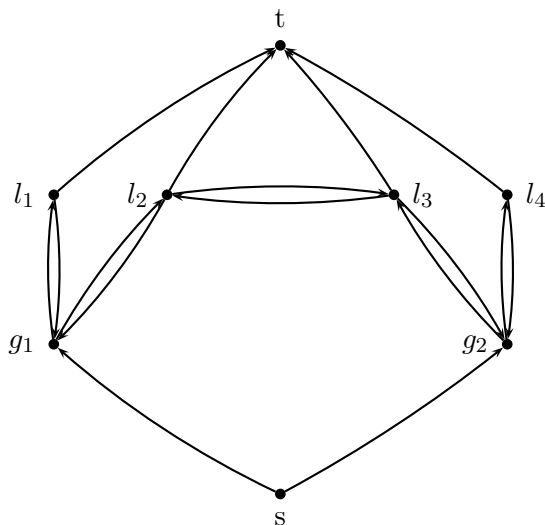


Figure 22: The auxiliary network corresponding to the grid from figure 21.



In other words, the above transformation together with algorithm 1 for MAXFLOWFIXEDCOST defines an algorithm for MAXDCFLOWFIXEDCOST. We will refer to this algorithm as the *maximum flow strategy* in the following. In section 5 we will use this strategy and comment on the quality of the obtained solutions.

The setting for the capacities of the auxiliary edges connecting the super source and sink described above only makes sense for a heavily damaged transmission grid. As long as the transmission grid is able to deliver all the generated power, the minimum cut will lie between the super source and

the generators and the algorithm would not be able to denote a set of edges to repair. There are several possibilities to deal with this problem. A first attempt is to simply multiply the generator and load capacities with a constant factor. For a large enough factor this will cause the minimum cut to be shifted away from the super source, as the transmission grid will reach its maximum capacity sooner or later. For a very high factor this is equivalent to setting the generator capacity and the load demand to infinity. However, this has the negative effect that all generators and loads are treated equally, independent of their actual capacity. If the capacities vary strongly, setting the capacities to infinity might therefore not be reasonable. For real networks, it requires some testing in order to find the appropriate capacities.

5 Blackout prevention

In this section, we combine the results of the previous sections. We define a new model, where a repair process gets interleaved with random failures. This is intended to model the work of a repair team that continuously fixes lines that have previously failed due to random events. The goal of the model is to test the ability of the maximum flow strategy defined in section 4.3.2 to prevent cascading failures.

5.1 Blackout prevention model

We use a discrete time horizon and let at each point in time fail one line in expectation. We flip a coin for each line and trip it with a failure probability of $\rho = \frac{1}{m}$, where m is the number of lines. At the same time, the repair team can deterministically repair exactly one line at each point in time. The objective of the repair team is to determine the line that minimizes the blackout risk, when being repaired. This corresponds exactly to the static repair problem with a budget equal to one, as defined in section 4. We can therefore use the maximum flow strategy in order to select a line to repair.

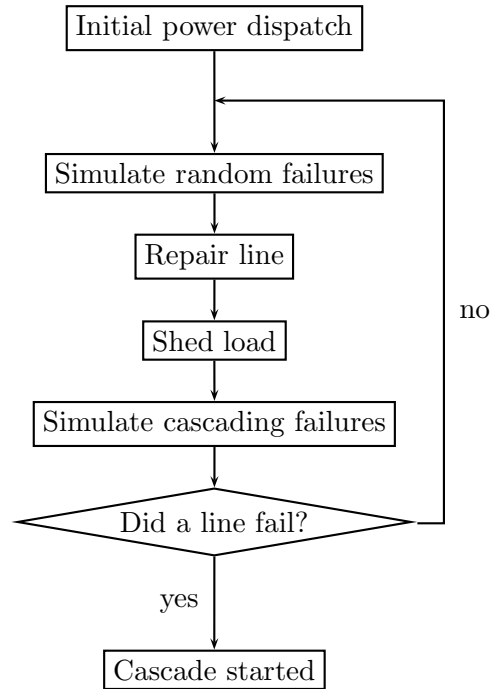
At each point in time we first simulate the random outage as described above. We then apply the maximum flow strategy and repair the selected line. After each such simulation step we apply the linear program from section 3.3.3, determine the load shedding, and calculate the new line failure probabilities $\sigma(e)$ as described in section 3.3.4. We flip a coin for each line with non zero failure probability $\sigma(e)$ in order to decide whether the line fails due to overloading or not. If a line fails, we assume that a cascade has started and stop the simulation run, else we continue with the next iteration. Figure 23 show a flow chart of this simulation procedure.

At first sight, one might think that this process will run forever for any repair strategy. However, from random walk theory we know that the variance of the number of failed lines will increase over time. This means that the simulation will always end within finite time, simply due to the fluctuations of the random process.

5.2 Computational results

We have performed a number of simulations of the blackout prevention model on the Reliability Test System introduced in section 3.4.1. The goal of this section is to evaluate the quality of the maximum flow strategy proposed in section 4.3.2. The principal measure for the quality of a repair

Figure 23: Blackout prevention model



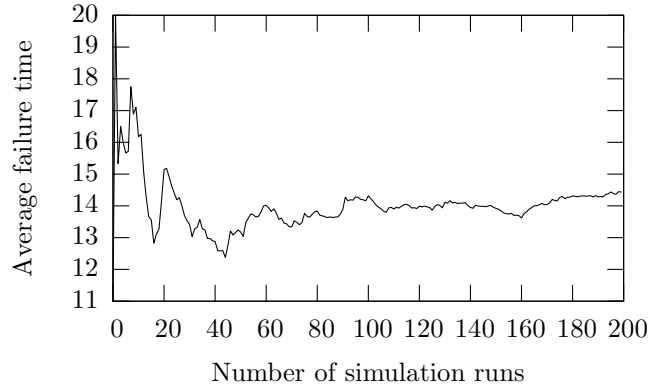
strategy is the number of time steps elapsed until the first line gets tripped due to overloading. We will refer to this number as the *failure time* in the following. Clearly, this number highly depends on the loading level of the system. In a highly loaded system, the first failure of a line will cause other lines to be overloaded, leaving no time to the repair team to react. In case of a low loading, the repair team is expected to keep the system in a stable state for quite some time. But as mentioned above, the expected value of failed lines increases over time and eventually a cascading failure will start.

5.2.1 Without repair

First, we have run the blackout prevention simulation without repairing any lines and measured the failure time for each run. Figure 24 shows that the average of the failure times converges after approximately 200 simulation runs.

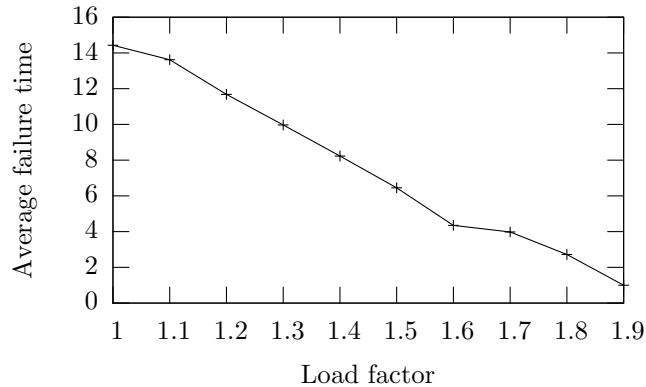
In figure 25, we have plotted the average failure time over 200 simulation runs for different loading levels. We observe that the average failure time decreases linearly with increasing loading level. This is not surprising as the

Figure 24: Convergence of the average failure time.



average line loading is proportional to the load factor.

Figure 25: Loading dependence of the average failure time.



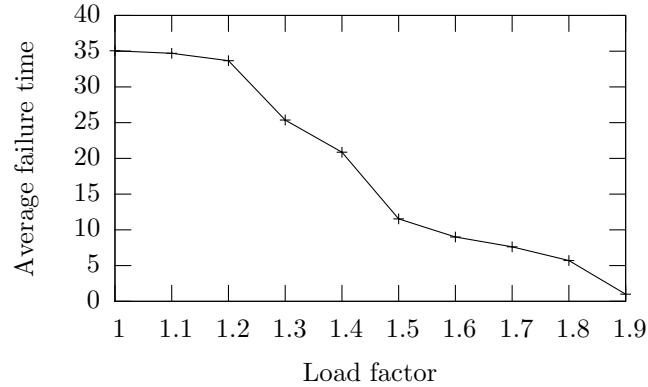
5.2.2 Random repair

In order to get a crude benchmark for our repair strategy, we want to compare it with a random repair strategy. In this strategy, we simply choose one of the failed lines at random and repair it. The average failure time for different loading levels, again calculated over 200 runs, is shown in figure 26. As expected, the average failure times are significantly higher than without repair.

5.2.3 Maximum flow strategy

Finally, we have applied the maximum flow strategy described in section 4.3.2. As mentioned in the description of the algorithm we have to decide

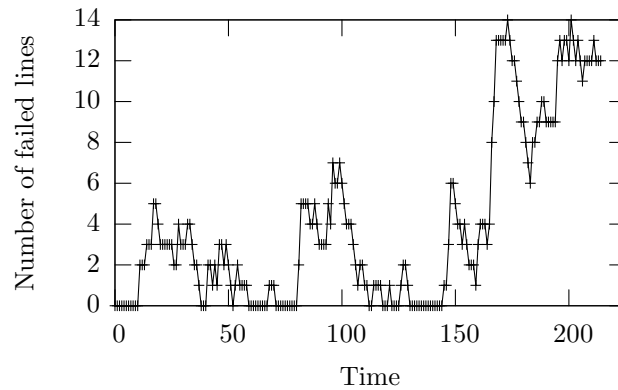
Figure 26: Average failure times for the random repair strategy for different loading factors.



on the capacities of the auxiliary lines. We have done the simulations below with the capacities of the auxiliary lines between the super source and the generators set to the generator capacities and the capacities of the auxiliary lines between the loads and the super sink set to the load demands multiplied by two.

Figure 27 shows a typical evolution of the number of failed lines over time for low load, i.e. a load factor equal to 1. The repair process can avoid a cascade for quite a long time, but eventually an accumulation of line failures causes a line to fail due to overload.

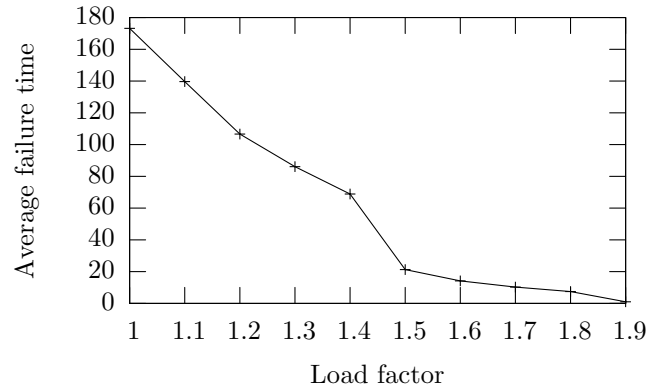
Figure 27: Evolution of the number of failed lines during a simulation run.



As for the random repair strategy, we have run the maximum flow strategy for different loading levels. Clearly, we are interested, whether the maximum flow strategy performs better than the simple random strategy. In

figure 28 we see that at low loading, the average failure time is nearly 180 for the maximum flow strategy, whereas it is 35 for the random strategy. First of all, this shows that the average failure time is strongly influenced by the repair strategy. Second, it shows that our strategy does perform quite well. It seems to be a reasonable objective to maximize the minimum cut of the underlying network. We can conclude that the maximum flow strategy provides indeed a good starting point for the further development of repair strategies.

Figure 28: Average failure times for the maximum flow strategy for different loading factors.



6 Implementation

For the computational results, we have implemented the various models and algorithms in C++. For the graph algorithms we have used the LEDA library [2]. LEDA is a C++ class library for efficient data types and algorithms and provides a wide range of commonly used algorithms in the field of graph- and network problems. For the linear programming problems we have used the CPLEX solver [1]. CPLEX provides an object oriented C++ library that offers classes to model and solve different classes of optimization problems. Both libraries require a valid license and are not available as open source.

We give an overview of the content of the source files that can be found on the CD-ROM enclosed with this thesis. For more detailed information, we refer to the documentation on the CD-ROM.

<i>PowerGrid.cpp</i>	Defines the class <code>PowerGrid</code> that models the power transmission grid. Among others, the class offers methods to calculate the DC power flow for a given vector of power injections, to perform a load shedding and to trip and repair transmission lines.
<i>RepairProcess.cpp</i>	Defines the class <code>RepairProcess</code> that models the action of the repair team and offers a method that applies the maximum flow strategy, given a budget and an instance of <code>PowerGrid</code> .
<i>BlackoutModel.cpp</i>	Provides a function that implements the flow chart shown in figure 2 and makes use of the classes <code>PowerGrid</code> and <code>RepairProcess</code> .
<i>PreventionModel.cpp</i>	Provides a function that implement the flow chart shown in figure 23 and makes use of the classes <code>PowerGrid</code> and <code>RepairProcess</code> .
<i>Utilities.cpp</i>	Provides various utility functions, for instance for the graphical output in LEDA.
<i>main.cpp</i>	Provides the main function that reads command line inputs and calls the simulation functions provided by <i>BlackoutModel.cpp</i> and <i>PreventionModel.cpp</i> .

Furthermore, we have implemented the method proposed in [18] for the exact computation of the st-reliability in an acyclic directed graph. The corresponding function can be found in the file *st_reliability.cpp*.

7 Conclusion

In this section, we provide a summary of our results and point out some open problems.

7.1 Summary

In order to improve our understanding of the dynamics of cascading failures, we have first defined a realistic blackout model based on the DC power flow equations. We have investigated the model on the three area IEEE Reliability Test System 1996. The simulation results show that the average blackout size strongly increases at a critical loading, which is in agreement with the observations from real blackout data and confirms the results from [11].

As a possible source of ideas for repair strategies, we have defined two abstract repair problems in terms of discrete optimization problems. We have provided hardness proofs and an approximation algorithms for both problems. We have also defined a very similar repair problem based on the DC power flow model and shown how to apply the algorithms developed for network flows in this more realistic setting. This has led to a repair strategy that we denote by maximum flow strategy, as it tries to maximize the flow in an auxiliary network.

Finally, we have evaluated our repair strategy by testing its ability to prevent blackouts due to cascading failures. For this purpose, we have developed a new model, where a repair team can fix a line after each occurrence of random failures. We have evaluated our strategy again on the three area IEEE Reliability Test System 1996. The results show that with the maximum flow strategy, we can delay the start of a cascading failure up to five times longer than by repairing a line at random.

From the simulation results, we conclude that the maximum flow strategy provides a good starting point for the development of repair strategies in electricity networks.

7.2 Outlook

There are many open problems concerning both, the theory and the application. The following list is not exhaustive, but provides some examples of interesting problems.

- We do not provide a hardness proof for both, the MAXSUMFLOW problem restricted to unit capacity networks and the briefly mentioned

path extraction problem. We believe that both problems can be shown to be \mathcal{NP} -hard by reductions similar to the ones given in this thesis.

- There might be better approximation algorithms for the MAXFLOW-FIXEDCOST problem, taking the different capacities into account. An approach based on capacity scaling might be a good starting point.
- The proposed maximum flow strategy can be used as a starting point to develop further repair strategies based on network flow.

A Classic reliability models

The classical model used to describe reliability problems in networks is called the *probabilistic graph*. A probabilistic graph $G = (V, E)$ is a set V of n vertices together with a collection E of m directed or undirected edges. There is a probability of operation $p(e)$ associated with each edge and we make the assumption that these probabilities are statistically independent. We simulate the failure process by flipping a biased coin for each edge in order to determine whether the edge has failed or is operating. The goal of the most common reliability problems is to calculate the probability that some connectedness property does hold in the resulting graph.

The most commonly studied problems in undirected graphs are the *two-terminal* and the *all-terminal reliability* problems. In the latter, the goal is to determine the probability that the graph is still connected, whereas in the former we determine the probability that two particular vertices are still connected. Both problems are special cases of the more general k -terminal reliability problem, where we seek the probability that for a set T of k specified target nodes, the graph contains a path between each pair of nodes. In directed graphs, the mentioned problems are stated in a similar way. In the directed analog of the k -terminal reliability problem, we are given a source node s , and a set T of $k - 1$ target nodes and need to determine the probability that there is a path from s to each node in T . If there is only one target node t , we denote the problem as the *st-reliability* problem. The special case $T = V \setminus \{s\}$ is referred to as *reachability* problem.

A.1 Hardness results

All problems mentioned in the previous section have been shown to be $\#P$ -complete. The basic proof idea is the same for all the reliability problems. We will present this idea briefly and refer to [20] for the actual proofs.

In the following, we assume that all operation probabilities are equal to p . Furthermore we need the following definitions. A *state* of the graph is a set $S \subseteq E$ of operating edges. Every reliability problem is characterized by a structure function $\phi : 2^E \leftarrow \{0, 1\}$ which determines for every state whether it is a fail or an operating state. The probability of being in a state S of cardinality i is therefore $P[S] = p^i(1 - p)^{m-i}$ and we can write any reliability in the following form:

$$Rel_\phi(G; p) = \sum_{i=0}^m Pr[S] \phi(S) \quad (2)$$

If we denote by N_i the number of operating states of cardinality i under the

structure function ϕ we can rewrite the previous expression as

$$Rel_\phi(G; p) = \sum_{i=0}^m N_i p^i (1-p)^{m-i} \quad (3)$$

The problem of determining the sequence (N_0, N_1, \dots, N_m) , referred to as *path-set numbers* for a given graph G and a structure function $\phi()$ is known as the *functional reliability* problem.

The idea of the proofs is now as follows:

1. Show that some difficult problem (i.e an \mathcal{NP} -hard or $\#P$ -complete) is polynomial time reducible to the the functional reliability problem.
2. Show that the functional reliability problem is polynomial time reducible to the standard formulation of the corresponding reliability problem.

Whereas the first step depends on the actual reliability problem, the second step is always the same. Given an algorithm for a reliability problem, we solve the problem for m different values of operational probabilities p_1, \dots, p_m on all the edges. This allows to set up a system of m equations for the m unknowns N_i which determines the values of the N_i 's exactly.

A.2 Existing algorithms

The most general strategy to solve reliability problems is complete state enumeration. We simply consider all possible states of a given graph and sum up the probabilities of all operational ones. The running time is clearly poor as this algorithm requires the enumeration of all possible 2^m states.

If we have no information at all about the structure of the operational states that is the best we can do. Nevertheless, the operational states in reliability problems satisfy a property that we will denote by *coherence*: Every superset of an operational state is an operational state and every subset of a failed state is a failed state as well. This means that it satisfies to consider only the sets of minimal operational or failed states, denoted as *minpaths* and *mincuts* in the following.

A.2.1 Inclusion-exclusion

This observation immediately leads to the family of inclusion-exclusion algorithms. Given the set of minpaths P_1, \dots, P_h , we define E_i as the event that all edges in minpath P_i are operational. Let $R(G)$ be some reliability problem on a given graph G . It now holds that:

$$R(G) = Pr[E_1 \vee E_2 \cdots \vee E_h] \quad (4)$$

These events are not disjoint and we can apply the principle of inclusion-exclusion to get the following expression for the reliability:

$$R(G) = \sum_{j=1}^h (-1)^{j+1} \sum_{I \subseteq \{1, \dots, h\}, |I|=j} Pr[E_I] \quad (5)$$

where E_I is the event that all paths P_i with $i \in I$ are operational. The running time of a naive implementation is indeed superexponential in the number n of nodes of the graph as already the number h of minpaths may be exponential. The key idea to avoid the superexponential behavior is to consider only a relevant set of states. Introducing the concept of domination, Satyanarayana and Hagstrom developed one of the most efficient algorithms within this family [22].

A.2.2 Disjoint products

Another way of dealing with the fact that the events E_i are not disjoint is the following. Instead of using the events E_i directly we define a new set of disjoint events $D_i = E_i \cap \overline{E_1} \cap \dots \cap \overline{E_{i-1}}$. We now get a much simpler expression for the reliability:

$$Rel(G) = \sum_{i=1}^h Pr[D_i] \quad (6)$$

Following this strategy, Ball and Nemhauser [23] give an algorithm for the all-terminal reliability which is polynomial in the number of minpaths.

In a similar way, but using disjoint sets of mincuts, Provan and Ball [24] give an algorithm which is polynomial in the number of mincuts. As a function of n , the running time is in $O(3^n)$, which is an improvement over complete state enumeration which has complexity $O(2^m)$.

A.2.3 Restricted classes

The hardness results suggest that efficient algorithms for the reliability problems on general graphs do not exist. However, there are efficient algorithms for several special classes of graphs. We will give some examples in this subsection. An important element of these algorithms is the reduction of the size of the graph through transformations. A transformation that preserves reliability and the independence between the edges is called a *reduction*. Among the most common reductions there are the *series* and *parallel* reductions. A series reduction at node v is possible if v is of degree two. Let $e = \{u, v\}$ and $f = \{v, w\}$ be the two edges incident to v with operation probabilities p_e and p_f . Then we can replace e and f by the edge $g = \{u, w\}$ with operation probability $p_g = p_e p_f$ and remove the node v from the graph. In a parallel

reduction we replace two parallel edges $e = \{u, v\}$ and $f = \{u, v\}$ by a single edge $g = \{u, v\}$ with operation probability $p_g = 1 - (1 - p_e)(1 - p_f)$.

We first consider the class of *series-parallel* graphs, a subclass of the class of planar graphs. An undirected graph is *series-parallel* if and only if it has no subgraph homeomorphic to the four-node complete graph K_4 . Intuitively a graph is series-parallel if it is reducible to a single edge by series and parallel reductions. However, the problem is that the series reduction is only applicable if the node to be eliminated is not a target node. Without this special case, the solution for all reliability problems would be trivial. Satyanarayana and Wood [25] introduced a set of additional reduction rules which resolve this problem and lead to linear time algorithms for all reliability problems on series-parallel graphs.

There are efficient algorithms for several other subclasses of planar graphs, such as *cube-free* or *normal* graphs [19]. But for general planar graphs, the reliability problems remain difficult.

Another interesting subclass of graphs are acyclic directed graphs (digraphs). While the st-reliability remains $\#\mathcal{P}$ -complete, we can solve reachability efficiently. The key observation is that in order to construct a pathset for reachability, it is necessary that each target node has at least one operational incoming edge. For acyclic digraphs this condition is sufficient as well. This leads to the following expression for reachability, where $Pred(v)$ is the set of arcs entering v .

$$\prod_{v \neq s} \left[1 - \prod_{a \in Pred(v)} (1 - p_a) \right] \quad (7)$$

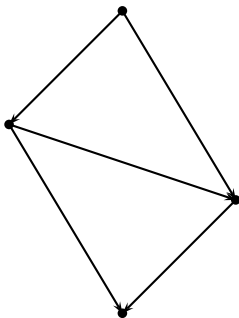
A.3 Algorithms for acyclic and nearly acyclic graphs

We next focus on the st-reliability problem in acyclic and nearly acyclic graphs. Despite the fact that this problem was shown to be theoretically difficult, it may be possible to exploit the special structure in order to get more efficient algorithms than for the general case.

A.3.1 Acyclic directed graphs

Bein et al. have developed an optimal reduction of two-terminal directed acyclic graphs (st-dags) to series parallel graphs [18]. An *st-dag* is defined as an directed acyclic graph with exactly one source and one sink. A directed acyclic graph is series parallel if and only if it does not contain a subgraph homeomorphic from the *interdictive graph* shown in figure 29. The authors define a notion of *reduction complexity* $\mu(G)$, a measure for

Figure 29: The interdictive graph (IG).



how nearly series-parallel an st-dag is. They then show how to calculate the st-reliability in time only exponential in the *reduction complexity* of a graph.

In order to define the reduction complexity we need a new type of reduction called *node reduction* which is a slight generalization of the series reduction defined before. A node reduction at v can occur if v has in-degree or out-degree 1. Assume v has in-degree 1 and let $e = \{u, v\}$ be the only incoming edge and let $f_1 = \{v, w_1\}, \dots, f_k = \{v, w_k\}$ be the outgoing edges. We replace the edges $\{e, f_1, \dots, f_k\}$ by $\{g_1, \dots, g_k\}$, where $g_i = \{u, w_i\}$ and remove v from the graph. The case where v has out-degree 1 is symmetric. Note that this reduction does introduces dependencies between the operational probabilities of the nodes and is therefore only applicable in case we know that e is operational.

Definition 4. For G an st-dag, the reduction complexity $\mu(G)$ is defined as the minimum number of node reductions sufficient (along with series and parallel reductions) to reduce G to a single edge.

Using the fact that the st-reliability in series-parallel graphs is computable in linear time $O(m)$, the authors give an $O(m2^{\mu(G)})$ algorithm for a general st-dag G . For this purpose let G' be the graph after the node reduction replacing $\{e, f_1, \dots, f_k\}$ by $\{g_1, \dots, g_k\}$ and setting $p_{g_i} = p_{f_i}$ and $G'' = G - \{e, f_1, \dots, f_k\}$. So G'' corresponds to the case that e fails and G' is derived from G under the condition that e is operating. We can then apply the following recurrence:

$$R(G) = (1 - p_e)R(G') + p_e R(G'') \quad (8)$$

It remains to prove that in every case a node reduction is possible and easy to find and that the resulting graphs after $\mu(G)$ steps are indeed series parallel. We refer to the original paper for the details.

A.3.2 Nearly acyclic directed graphs

The algorithm presented in the previous section is efficient for graphs with a constant reduction complexity c . Along these lines, we try to find a similar reduction from a nearly acyclic graph to a *reasonable* number of acyclic graphs. We expect this number of acyclic graphs to be exponential in some measure of distance between acyclic and cyclic graphs. This is suggested by the fact that the reachability is solvable in linear time for acyclic digraphs but is $\#\mathcal{P}$ -complete for general digraphs.

A first immediate observation is that the graph of the strongly connected components forms an acyclic digraph. We could therefore use the cardinality of the set of edges contained in some connected component $E_{cc} \subseteq E$ (that is, both endpoints of the edge are in the same component) as a measure of *cyclic complexity*. This quantity is 0 for acyclic graphs and m for strongly connected graphs.

We propose a very simple algorithm which leads to $2^{|E_{cc}|}$ calls to a subroutine $R_{acyclic}$ for acyclic graphs. We first define the notion of an *event*. An *event* is an assignment of a subset of the edges to either 0 or 1. For an event α , $G[\alpha]$ is the graph obtained by replacing the edges e with $\alpha(e) = 1$ by arcs with operational probability 1, removing the edges with $\alpha(e) = 0$ and by contracting the cycles with all edges having an operational probability of 1 to a simple node. The algorithm is written down in pseudo-code below.

Algorithm 3 R_{simple}

Require: Graph $G = (V, E)$, Algorithm $R_{acyclic}$

Ensure: Some reliability measure $R(G)$

$R(G) \leftarrow 0.$

for all $\alpha \in \{0, 1\}^{E_{cc}}$ **do**

$p(\alpha) \leftarrow \prod_{\alpha(e)=1} p_e \prod_{\alpha(e)=0} (1 - p_e)$

$R(G) \leftarrow R(G) + p(\alpha)R_{acyclic}(G[\alpha])$

end for

return $R(G)$

It is obvious that this algorithm is not optimal in general. Assume for instance that our graph G is a simple cycle C_n . In this case we have $E_{cc} = E$ and we get 2^m (trivial) instances of acyclic graphs, whereas we can find a linear number of instances as we will see later. What we are looking for is a minimal set \mathcal{A} of exhaustive and mutually disjoint events, such that $G[\alpha]$ is acyclic for all $\alpha \in \mathcal{A}$. By exhaustive we mean that for every possible state $S \subseteq E$ there is an event $\alpha \in \mathcal{A}$ such that $\alpha(e) = 1$ implies that $e \in S$ and $\alpha(e) = 0$ implies $e \notin S$. By disjoint we mean that for all $\alpha_1, \alpha_2 \in \mathcal{A}$, there is an edge e with $\alpha_1(e) \neq \alpha_2(e)$. Given such a set we can express the reliability

of G as:

$$R(G) = \sum_{\alpha \in \mathcal{A}} Pr[\alpha] R(G[\alpha]) \quad (9)$$

Consider again the cycle C_n with edges $E = \{e_1, \dots, e_n\}$. The set of events $\mathcal{A} = \{\{e_1 \rightarrow 0\}, \{e_1 \rightarrow 1, e_2 \rightarrow 0\}, \dots, \{e_1 \rightarrow 0, e_2 \rightarrow 0, \dots, e_n \rightarrow 1\}, \{e_1 \rightarrow 0, e_2 \rightarrow 0, \dots, e_n \rightarrow 0\}\}$ meets the requirements described above and has only $m + 1$ elements by construction.

This reasoning leads to the following recursive algorithm. Given an arbitrary order on the edges, compute the strong components of the graph, from which we can easily compute the set of all edges contained in a cycle. As long as this set is not empty, select the smallest edge e (with respect to our order) and proceed recursively with the two graphs $G' = G[e \rightarrow 1]$ and $G'' = G[e \rightarrow 0]$ and return the reliability $R(G) = p_e R(G') + (1 - p_e) R(G'')$. If the set is empty, the graph is acyclic and we can compute the reliability given an algorithm for acyclic graphs.

Algorithm 4 R_{cyclic}

Require: Graph $G = (V, E)$, Algorithm $R_{acyclic}$

Ensure: Some reliability measure $R(G)$

if There is a cycle $C \subseteq G$ **then**

$e \leftarrow$ smallest edge contained in C

return $p_e R_{cyclic}(G[e \rightarrow 1]) + (1 - p_e) R_{cyclic}(G[e \rightarrow 0])$

else

return $R_{acyclic}(G)$

end if

In case of C_n , this algorithm ends indeed up with the $m + 1$ events described above. The number of possible calls to $R_{acyclic}$ is still bounded by $2^{|E_{cc}|}$, but we can give a bound with respect to the number and size of cycles contained in the graph as well. For this purpose we introduce another measure of cycle complexity. Assume that the graph contains k cycles C_1, \dots, C_k and let l_1, \dots, l_k denote their lengths. We can then use the product of the lengths $\prod_{i=1}^k l_i$ as our measure. For acyclic graphs this measure is again 0, while for the complete graph we get number that is double exponential in n .

Theorem 9. *The number of acyclic instances generated algorithm 4 is $O(\prod_{i=1}^k l_i)$.*

Proof. In order to end up with an acyclic graph, each event has to assign in every cycle at least one edge to 0 (failed) or all edges to 1 (operational). While in the first case the cycle disappears, we can contract the cycle to a single node in the second case. We claim that during the execution of our algorithm, in every cycle either

1. exactly one edge e_i gets assigned to 0 and the edges e_j with $j < i$ get assigned to 1, or
2. all edges get assigned to 1.

If we can establish this property we have proved the claimed complexity as each cycle C_i can cause at most $l_i + 1$ different assignments on its edges. The claim can be proved easily: Assume that no edge gets assigned to 0 and not all edges get assigned to 1 for some cycle, then the resulting graph does still contain that cycle. Furthermore, a cycle can not cause two or more edges to fail, as it does not exist anymore after the failure of one edge. Therefore the only possible cases are the ones claimed above. \square

References

- [1] <http://www.ilog.com/products/cplex/>
- [2] <http://www.algorithmic-solutions.com/>
- [3] B. Yang, S.Q.Zheng, E. Lu, "Finding two Disjoint Paths in a Network with Normalized α -Min-Sum Objective Function.", Proceedings of the 17th IASTED International Conference on Parallel and Distributed Computing and System, 2005.
- [4] "The IEEE Reliability Test System - 1996", IEEE Transactions on Power Systems, Volume 14, No. 3, 1999.
- [5] I. Dobson, "Where is the edge for cascading failure?: challenges and opportunities for quantifying blackout risk.", IEEE Power Engineering Society General Meeting, Tampa USA, 2007.
- [6] M. Zima, "Contributions to Security of Electric Power Systems", Dissertation ETH No. 16492, 2006.
- [7] "Final Report on the August 14, 2004 Blackout in the United States and Canada: Causes and Recommendations", U.S.-Canda Power System Outage Task Force, 2004.
- [8] B. A. Carreras, D. E. Newman, I. Dobson, A. B. Poole, "Initial Evidence for Self-Organized Criticality in Electric Power System Blackouts", Proceedings of the 33th Hawaii International Conference on System Science, 2000.
- [9] J. Chen, J.S. Thorp, M. Parashar, "Analysis of Electric Power System Disturbance Data", Proceedings of the 34th Hawaii International Conference on System Sciences, 2001.
- [10] B. A. Carreras, V. E. Lynch, M. L. Sachtjen, "Modeling Blackout Dynamics in Power Transmission Networks with Simple Structure", Hawaii International Conference on System Sciences, 2001.
- [11] B. A. Carreras, D. E. Newman, I. Dobson, "Critical points and transitions in an electric power transmission model for cascading failure blackouts", Chaos, Volume 12, Number 4, 2002.
- [12] R. K. Ahuja, T. L. Magnanti and James B. Orlin, "Network Flows, Theory, Algorithms, and Applications.", Prentice-Hall, Inc, 1993.
- [13] Michael R. Garey and David S. Johnson, "Computers and Intractability: A guide to the theory of NP-Completeness", 1979.

- [14] S. Krumke, H. Noltemeier and R. Ravi, "Flow Improvement and Network Flows with Fixed Costs", OR'98 Zurich.
- [15] G. Andersson, "Modelling and Analysis of Electric Power Systems", Lecture Notes, 2006.
- [16] A. J. Wood and B. F. Woolenber, "Power Generation, Operation and Control", John Wiley and Sons Inc., 1996.
- [17] U. Feige, "A threshold of $\ln n$ for Approximating Set Cover", Journal of the ACM 45, 634-652, 1998.
- [18] W. W. Bein, J. Kamburovski and M. F. M Stallmann, "Optimal Reduction of Two-Terminal Directed Acyclic Graphs", SIAM Journal of Computing Vol. 21 No. 6, 1992.
- [19] T. Politof and A. Satyanarayana, "A Linear-Time Algorithm to Compute the Reliability of Planar Cube-Free Networks.", IEEE Transactions on Reliability, Vol. 39, No. 5, 1990.
- [20] C. J. Colbourn, "The Combinatorics of Network Reliability", Oxford University Press, 1987.
- [21] N. G. Hingorani and L. Gyugyi. "Understanding FACTS concepts and technology of flexible AC transmission systems", IEEE Press, New York, 2000.
- [22] A. Satyanarayana and J. N. Hagstrom, "Combinatorial properties of directed graphs useful in computing network reliability", Networks 11, 357-366, 1981.
- [23] M. O. Ball and G. L. Nemhauser, "Matroids and a Reliability Analysis Problem", Mathematics of Operations Research 4, 132-143, 1979.
- [24] J. S. Provan and M. O. Ball, "Computing Network Reliability in Time Polynomial in the Number of Cuts", 1983.
- [25] A. Satyanarayana and R. K Wood, "A Linear-Time Algorithm for Computing K-Terminal Reliability in Series-Parallel Networks", SIAM Journal of Computing Vol. 14, 1985.