THE UNIVERSITY *of* EDINBURGH

Design of Reliable Aerospace System Architecture

Lukas Matthias Schäfer

Doctor of Philosophy
University of Edinburgh
June 27, 2018

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text. Chapters 2 and 3 are based on publications [48] and [49], respectively, of which I am the main author. This work has not been submitted for any other degree or professional qualification.

Place, Date	Lukas Matthias Schäfer

Acknowledgements

For the last few years, I give great thanks to my supervisor Sergio. His support, encouragement, and mathematical guidance have led me to this point. Furthermore, I want to thank my friends and office mates Karen, Jenny and Simon for their help. I want to also thank my mother for her support, my brothers for their help and I am thankful for the support of my girlfriend Angela, especially for stopping me from throwing in the towel at the end.

I also want to thank Vassili Srithammvanh from the Airbus Group for his support over the years. My PhD was funded by an EPSRC Industrial CASE studentship (EP/L509420/1) in partnership with Airbus Group.

Lay Summary

The design of systems is an important field in aircraft architecture engineering. Especially if it has to do with the safety of an aircraft. The research for this dissertation was done in cooperation with AIRBUS Group France and they are very interested in the optimal design of aircraft architecture systems. Therefore, this dissertation addresses how to optimally design the door management system (DMS) which is an aircraft architecture system. The function of the DMS is to check if all doors are properly closed and the cabin has the correct pressure. As the DMS is part of the pressurization system of the aircraft, it is obvious why it is considered to be safety-critical. For safety-critical systems, reliability and redundancy are important issues that have to be considered while designing them.

Reliability is the probability of a system not failing. Imagine a gas network system which has to transport gas from one point to another by using pipeline systems. The reliability of such a gas network is the probability of it being able to transport the gas. Redundancy has also something to do with the functionality of a system. Redundancy means that every function of a system is implemented twice with both implementations not sharing any hardware. Consider again the gas network example. It is redundant if there exists at least two pipelines systems that connect the same two points and these two pipeline systems share no pipeline or crossing point. Therefore, if one pipeline system has an error, the other is not affected and gas can still be transported between the end points of the pipeline systems.

Hence, the objective of the dissertation is to examine how to optimally design the DMS under consideration of reliability and redundancy. Optimality can mean different things. It can mean that the system is supposed to weigh as little as possible, to be as cheap as possible or the reliability to be as high as possible.

To address this objective, the thesis is split into three smaller problems. The first problem is to find the most efficient calculation method for the reliability of the DMS that also can be used in designing it in the best possible way. The second problem is to consider only redundancy and to develop methods for designing an optimal DMS so that it is redundant. The last problem is to consider both redundancy and reliability and to develop methods for designing an optimal DMS so that it is redundant and has a certain reliability.

Abstract

Reliability and redundancy of safety-critical network systems is a paramount issue in system engineering. Be it in evaluating existing network systems or solving optimization problems for designing network systems, it is important to consider reliability and redundancy. This dissertation is in collaboration with AIRBUS Group, France, and they are very interest in the optimal design of safety-critical aircraft architecture systems which have to consider reliability and redundancy. To address the problem of optimally designing such systems, we chose to focus on one specific aircraft architecture system the door management system. It checks if all doors are properly closed and the cabin has the correct pressure. It is a safety-critical system since it is part of the pressurization system of an aircraft.

To optimally design the DMS while considering reliability, a suitable reliability evaluation algorithm is necessary. In this dissertation, we begin by proposing a suitable reliability evaluation algorithm for a type of non series-parallel network system which includes the DMS and which can be used in an optimization model. The reliability evaluation algorithm is based on a simplification of the probability principle of inclusion-exclusion formula for intersections of unions. The simplification exploits the presence of many repeated events and has many fewer terms, which significantly reduces the number of operations needed. We compare its computational efficiency against the sum of disjoint products method KDH88 for a simple artificial example and for the DMS.

Afterwards, we introduce the first MILP model for the DMS with k-redundancy. As the model is too difficult to be solved efficiently by standard MILP solvers, we discuss the issues of solving the model with general solving methods such as branch-and-bound and branch-and-price. We introduce specialized branching rules and new heuristics to solve the DMS problem with k-redundancy more efficiently and show results of computational tests which compare the specialized solving algorithms with general solving algorithms for example instances of the DMS problem.

Lastly, we discuss the problems of considering reliability in MI(N)LP models for the DMS and how the new reliability evaluation algorithm can be used. In this discussion, we give different MI(N)LP models for the DMS problem with redundancy and reliability. Moreover, we propose a new heuristic for the DMS problem with redundancy and reliability. It is based on branch-and-bound, the Dantzig-Wolfe decomposition and on the new reliability evaluation algorithm. We show results of computational tests of the new heuristic for example instances of the DMS problem and discuss its validity.

Contents

Chapter 1

Introduction

1.1 Motivation

Reliability and redundancy of safety-critical network systems is a paramount issue in system engineering. Be it in evaluating existing network systems or solving optimization problems for designing network systems, it is important to consider reliability and redundancy. First we would like to define what reliability and redundancy of a network system are.

The reliability of a network system is its probability of not failing. Failure of a network system happens if it cannot perform all of its functions. For example, a data information network with multiple points fails if it cannot transfer data between any two points of the system. In safety-critical network systems such as an aircraft safety system, the reliability should be at least $(1 - 10^{-9})$ based on data from AIRBUS Group France.

Redundancy of a network system is given when every function of the network system is implemented at least twice. This means that at least two implementations of a function do not share any hardware components in the network system. Redundancy is mainly used to increase the reliability of the system and because evaluation and calculation of reliability is very hard, sometimes k-redundancy is considered for network systems. A network system is k-redundant if each function of the network system is implemented k times and all implementations are disjoint.

In this dissertation, we are addressing the reliability calculation and the design of a door management system (DMS), which is a safety-critical network system in aircraft architecture engineering, through optimization while considering redundancy and/or reliability. We concentrate on this specific safety-critical network system as the work for this PhD thesis is in cooperation with AIRBUS Group France for whom this is a very important problem.

The functions of a DMS are to check the status of doors and locks in an aircraft, and send their status information to on-board computers and from these on-board computers to pressurization regulators in the aircraft. Because the pressurization of an aircraft is very important, the DMS is a safety-critical network system. Figure 1.1 shows a simple DMS with one door and Figure 1.2 shows an example for a DMS with multiple doors.

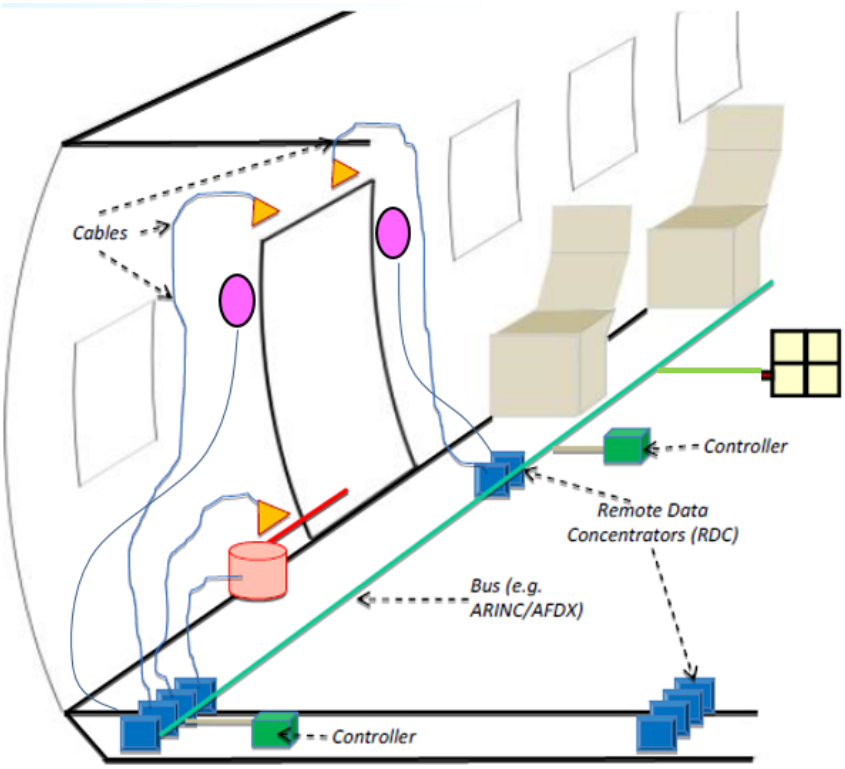

Figure 1.1: DMS for one door.

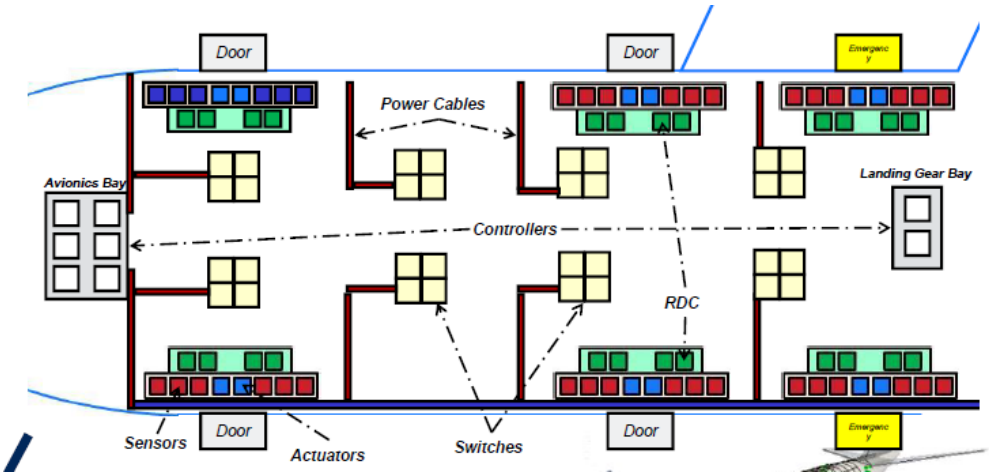

Figure 1.2: Example space for the DMS.

To consider reliability and k-redundancy in optimization models of network systems is a hard problem and there has been considerable research in the literature for many different network systems but not yet for the DMS. Not only are network system optimization models most often mixed integer problems (MIP), but both redundancy and reliability create even greater challenges. For example, k-redundancy brings symmetry to the MIP formulation, which creates problems for most solvers. But k-redundancy is still the smaller problem and can be reasonably dealt with. On the other hand, to consider reliability is much harder and to this day in most of the cases it is not possible to solve a reliability optimization model exactly without certain restrictive assumptions on the system. This is the case because reliability calculations are non-linear and also non-convex and this makes the optimization problem a non-convex mixed integer non-linear problem (MINLP), which can be considered as one of the hardest kind of optimization problem today.

Furthermore, most reliability calculation or evaluation methods are not suitable to be used in MINLP formulations and when they can be included in the formulation as an MINLP constraint, the number of variables and constraints needed is too large for the model to be solved.

Therefore, one of the main parts of this PhD thesis is to find and develop a reliability calculation algorithm that can be used in an MIP formulation for the DMS problem. The other main parts are how to consider redundancy and reliability in an MIP formulation for the DMS problem.

1.2 Objective, Contribution and Structure

As mentioned before, we address in this thesis three issues that arise when considering reliability and k-redundancy in MI(NL)Ps of network systems, specifically the DMS. They are:

- Finding the most suitable reliability calculation/evaluation method for the DMS, which can also be used in MIPs.
- How to consider k-redundancy in an MILP formulation for the DMS and how to solve it most efficiently.
- How to consider reliability in an MILP or MINLP formulation for the DMS and how to solve it most efficiently.

The thesis is also structured into three parts, with each one addressing one of these issues.

We start in Chapter 2 by giving an overview of reliability calculation and evaluation methods and discuss their suitability for MIP formulation. Afterwards we introduce our own reliability calculation method, which uses a simplification of inclusion-exclusion for intersections of unions. The simplification of inclusion-exclusion for intersections of unions is proven in Proposition 1 and is the main result of this chapter. Furthermore, it is shown how to implement our new method and we compare it with another similar existing method KDH88 [19]. The results of this chapter are published in [48] of which I am the main author.

In Chapter 3, we first introduce for the first time an MILP formulation for the k-redundancy DMS problem. For this problem, we explore two approaches to solve the MILP problem and discuss why general solving methods are not suitable and efficient. The first approach is an branch-and-bound algorithm with a new branching rule and heuristic which are tailored to the DMS problem with k-redundancy. It is discussed why the general branching rules and heuristics are not suitable and we show good performance of our algorithm with a computational study. This research is part of the paper [49], of which I am the main author. The second approach is a branch-and-price algorithm. We show that this approach is not efficient, but it helped to inspire the main idea for a heuristic of the DMS problem with redundancy and reliability which is the main result of the following chapter.

After having considered only k-redundancy, we start to also consider reliability. In Chapter 4 we address the DMS problem with redundancy and reliability. We first give MINLP and MILP formulations for the DMS problem with redundancy and reliability and discuss why these cannot be solved at the moment. Both of these formulations use our new reliability calculation method. Since we cannot solve these formulations to optimality at the moment, we developed a heuristic that gives a good reliability feasible solution by considering certain characteristics of reliability calculation of network systems. It is based on the Dantzig-Wolfe decomposition, branch-and-bound and our new reliability calculation method. This heuristic is introduced in Section 4.2.

Finally, in Chapter 5 we summarize the objective, methodology and results of our work and draw our conclusions.

Chapter 2

Reliability Calculation Method

In this chapter, we discuss the best way to calculate the reliability of the DMS which can also be used in MIP formulations. This chapter is structured as follows.

In Section 2.1 we first give an overview of existing reliability calculation and evaluation methods and discuss their lack of suitability to be used in MIP formulations. For this reason, we developed our own reliability calculation method which is based on a new simplification of the probability principle of inclusion-exclusion for intersections of unions. This new method is the main result of this chapter and is introduced in Section 2.2 as Proposition 1. In Section 2.2 we also compare the time complexity of the new method with the time complexity of the classical probability principle of inclusion-exclusion for intersections of unions. The two following subsections, 2.2.1 and 2.2.2, contain the proofs for Proposition 1 and the time complexity of the method.

To implement the reliability calculation method, we need a generating algorithm for a certain type of set which we later introduce as C_k sets. Such an algorithm is introduced in Section 2.3 with a proof of correctness.

With this, we were able to run computational tests for the new reliability calculation method. We implemented the method in Python and compared it to the KDH88 method which we also implemented in Python. The results of these tests are shown in Section 2.4.

The last section of this chapter is Section 2.5 in which we discuss the applicability of our new method and its limitations.

2.1 Reliability Evaluation/Calculations Methods

In this section we will give an introduction to reliability evaluation and calculation methods. As mentioned before, reliability of a network system is the probability of the system not failing. It is a critical issue in different fields such as computer networks, information networks or gas networks. In particular, reliability of safety-critical network systems ([44],[42]) is an important topic in system engineering. In most practical situations, the reliability of a complex network system (e.g. a system that is not series-parallel) has to be calculated exactly [22]. There are several methods to calculate or evaluate the reliability of a complex system which have been developed in recent decades. For static systems, these include reliability block diagram models [29], fault tree models, and binary decision diagram models. A general introduction to these methods can be found in [44]. For time-dependent systems, modeling techniques such as Markov models [31], dynamic fault tree models [6] and Petri net models [61] have been used.

Reliability calculations of complex systems can also be divided into systems with multi-state components, where the components of a system operate in any of several intermediate states with various effects on the entire system performance ([39], [38],[36],[43], [56] and [34]), or with binary-state components, where either a component works perfectly or not at all. Furthermore, reliability of complex systems with specific graph structures, like systems with a hypercube structure ([30],[33]), have received attention over the last few years. However most of these methods

consider systems with two-terminal nodes or k-terminal nodes where all k-nodes have to be connected. The DMS is a different type of complex system with a specific structure that has multiple functions with multiple start and end nodes. This will be explained in more detail later on.

For considering reliability in an optimization model for the DMS, reliability calculation methods are methods that consider binary state component and the specific graph structure of the network. Methods that are based on the classical probability principle of inclusion-exclusion, whose formula is

$$P\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \left((-1)^{i+1} \sum_{\substack{J \subseteq \{1,\ldots,n\}, \\ |J|=i}} P\left(\bigcap_{j \in J} A_j\right) \right), \tag{2.1}$$

with probability events $A_1, \ldots, A_n$, belong to this category.

Practical reliability calculations often involve very long expressions when the probability principle of inclusion-exclusion formula (2.1) is used. Therefore, there are many approaches in the literature on general network reliability calculations to simplify it such as, for example, partitioning techniques [12] and the sum of disjoint products (SDP) methods ([1],[19],[4], [41], [45], [60]). SDP methods are the most often used approaches, with recent results in [59], [58], [57],[9] and [51].

There is a problem with all of these methods. To simplify the classical probability principle of inclusion-exclusion formula with these methods, the exact system structure is needed and not only a general graph structure. However when we build the optimization model/MIP formulation, we do not have the exact structure of the network system because the exact system structure is the result of the optimization, and, therefore, these methods are not suitable.

We also found approximations, lower or upper bounds, of the probability principal of inclusion-exclusion ([7, 10]) as a reliability evaluation method unsuitable. This is the case because the approximations are not generally monotone increasing or decreasing with the exact reliability value. We provide an example of this behavior for a lower bound approximation in Appendix A.1.

This shows us that we need a new reliability calculation method that does not use the exact system structure. The method should only simplify the probability principle of inclusion-exclusion based on a general graph structure of the system. We propose such a method in the following section.

2.2 Simplification of Inclusion-Exclusion for Intersections of Unions

We propose our new approach to simplify the probability principle of inclusion-exclusion without needing the exact system structure, and to apply it to the calculation of the reliability of complex network systems in system engineering. To begin, we introduce the complex network systems under consideration which are generalized from the network structure of the DMS.

In system engineering, most network systems have multiple functions that have to be performed and these are not always independent (e.g., they share components). Reliability can be increased if different sets of components in the network can perform the same function. Therefore, these functions are implemented multiple times in the network system through different sets of components, and calculation of the reliability of the network system becomes a very complex task. Figure 2.1 shows an example complex network. It is a redundant DMS network with three doors.

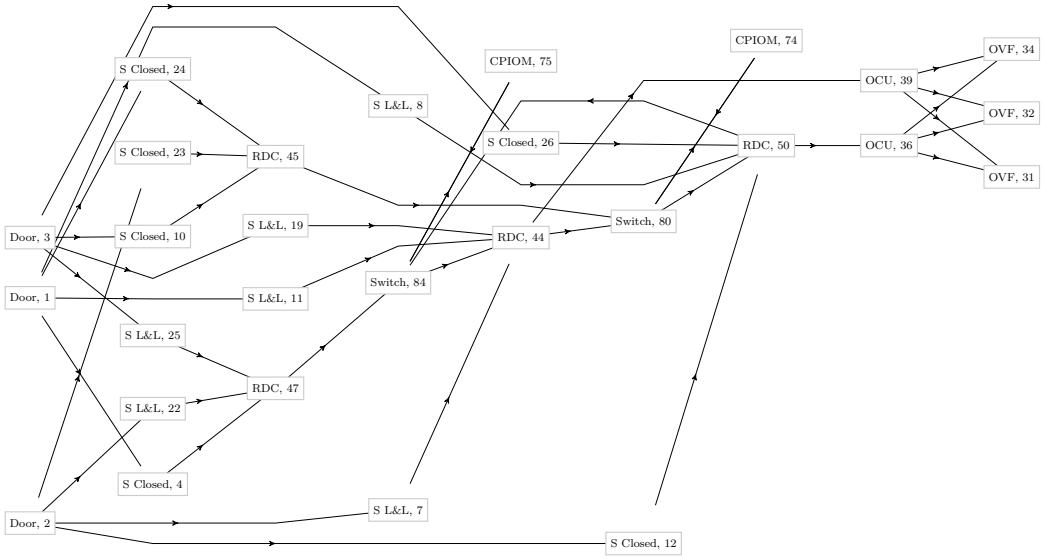

Figure 2.1: Redundant DMS network with three doors.

We assume that all failure probabilities of the individual components are known exactly. We do not consider the case when these probabilities are known only approximately (e.g., either by estimation or a confidence interval). If the different components of the network are independent of each other, then we can easily calculate the reliability of a set of components. Through this we can calculate the reliability of one implementation of a function, which is defined as the probability of the event that one implementation of the function does not fail. Finally, the probability of an intersection of such events can be calculated easily. However, if full independence cannot be assumed, then the calculation becomes very expensive, usually prohibitively so, as we demonstrate now.

Let n be the number of functions in the system and t_i be the number of implementations of function i in the system. Let $F_i,\ i \in \{1,\dots,n\}$, be the event that function i of a system does not fail in a specific period of time and $F_{ij},\ j \in \{1,\dots,t_i\}$, be the event that implementation j of function i does not fail in a specific period of time. Let $\mathcal{F} = \{F_1,\dots,F_n\}$ be the set of all functions and $\mathcal{F}_i = \{F_{i1},\dots,F_{it_i}\}$ be the set of all implementations of function i. Furthermore, let R be the event that the system does not fail. The reliability of the system, $P(R)$, is the probability that no function in $\mathcal{F}$ fails. A function $F \in \mathcal{F}$ does not fail if at least one of its implementations does not fail. Therefore,

$$P(R) = P\left(\bigcap_{i=1}^{n} F_i\right) = P\left(\bigcap_{i=1}^{n}\left(\bigcup_{j=1}^{t_i} F_{ij}\right)\right).$$

Because the different functions and implementations may not be independent, $P(R)$ is not easily calculable. In order to work on this expression, first we need to establish some notations. Let

$$W = \{1,\dots,t_1\} \times \ldots \times \{1,\dots,t_n\}, \text{ and}$$
$$B_w = \bigcap_{i=1}^{n} F_{iw_i} \text{ for } w = (w_1,\dots,w_n) \in W,$$

where $w_i \in \{1,\dots,t_i\}$ represents the implementation index of function i. We then have that

$$P(R) = P\left(\bigcap_{i=1}^{n}\left(\bigcup_{j=1}^{t_i} F_{ij}\right)\right) = P\left(\bigcup_{w\in W}\left(\bigcap_{i=1}^{n} F_{iw_i}\right)\right) = P\left(\bigcup_{w\in W} B_w\right). \tag{2.2}$$

Now the probability principle of inclusion-exclusion can be used and it follows that

$$P(R) = \sum_{t=1}^{|W|} \left((-1)^{t+1} \sum_{\substack{I \subseteq W, \\ |I|=t}} P\left(\bigcap_{j \in I} B_j \right) \right). \tag{2.3}$$

The number of summands in (2.3), which is equal to the number of possible intersections of B_w's, is $\sum_{t=1}^{|W|} \binom{|W|}{t} = 2^{|W|} - 1$ with $|W| = \prod_{i=1}^{n} |\mathcal{F}_i|$. Therefore, we have a doubly exponential computational complexity. Table 2.1 shows the number of summands for different values of the number of functions and implementations, with the assumption that every function has the same number of implementations.

$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{F}_i\rvert$	Summands
2	2	15
2	3	5.11×10^{2}
2	4	6.55×10^{4}
3	2	2.55×10^{2}
3	3	1.34×10^{8}
3	4	1.84×10^{17}
4	2	6.55×10^{4}
4	3	2.41×10^{24}
5	2	4.29×10^{9}
5	3	1.41×10^{73}

Table 2.1: Number of summands in the probability principle of inclusion-exclusion formula.

As can be seen, even for a small number of functions and implementations, the calculation of $P(R)$ becomes very expensive. However, note that there are many (a priori different) terms that, when the intersection of the sets is calculated, lead to the same intersection set, that is,

$$\exists I, J \subseteq W : \; I \neq J \; \wedge \bigcap_{i \in I} B_i = \bigcap_{j \in J} B_j. \tag{2.4}$$

For example, let $\mathcal{F} = \{F_1, F_2\}$, $\mathcal{F}_1 = \{F_{11}, F_{12}\}$ and $\mathcal{F}_2 = \{F_{21}, F_{22}\}$. It follows that

$$P(R) = P\left((F_{11} \cap F_{21}) \cup (F_{11} \cap F_{22}) \cup (F_{12} \cap F_{21}) \cup (F_{12} \cap F_{22}) \right).$$

It can be seen, for example, that

$$\begin{aligned} B_{(1,1)} \cap B_{(2,2)} &= (F_{11} \cap F_{21}) \cap (F_{12} \cap F_{22}) \\ &= F_{11} \cap F_{12} \cap F_{21} \cap F_{22} \\ &= (F_{11} \cap F_{21}) \cap (F_{11} \cap F_{22}) \cap (F_{12} \cap F_{21}) \\ &= B_{(1,1)} \cap B_{(1,2)} \cap B_{(2,1)}. \end{aligned}$$

Therefore, it seems natural to determine which combinations lead to the same intersection set and then simplify the formula. Utilizing this idea, we obtain the following result.

Proposition 1. *Let $\mathcal{F} = \{F_1, \ldots, F_n\}$ and let $\mathcal{F}_i = \{F_{i1}, \ldots, F_{it_i}\}$, $i \in \{1, \ldots, n\}$, be sets of events such that $F_i = \bigcup_{j=1}^{t_i} F_{ij}$. Let $R = \bigcap_{i=1}^{n} F_i$, $m = \sum_{i=1}^{n} t_i$ and, given $k \in \{n, n+1, \ldots, m\}$, let*

$$\begin{aligned} C_k = \{ \mathbf{E} = \{E_1, \ldots, E_k\} : \; & E_u = F_{i(u)j(u)} \text{ for some } i(u) \in \{1, \ldots, n\}, \; j(u) \in \{1, \ldots, t_{i(u)}\}, \\ & u \in \{1, \ldots, k\}, \; \{i(l) \mid l \in \{1, \ldots, k\}\} = \{1, \ldots, n\} \text{ and } E_p \neq E_q \text{ for } p, q \in \{1, \ldots, k\} \end{aligned}$$

and $p \neq q\}$.

That is, C_k *is the family of sets* $\mathbf{E} = \{E_1, \ldots, E_k\}$ *of implementations not failing where every function* i, $i \in \{1, \ldots, n\}$, *is implemented at least once. Then*

$$P(R) = \sum_{k=n}^{m} \left((-1)^{k-n} \sum_{\mathbf{E} \in C_k} P\left(\bigcap_{j=1}^{k} E_j \right) \right). \tag{2.5}$$

The proof for Proposition 1 can be found in Section 2.2.1. A generating algorithm for sets C_k, $k \in \{n, n+1, \ldots, m\}$, is given in Section 2.3.

Table 2.2 shows the total number of summands that we obtain when we use the result stated in Proposition 1 for different numbers of functions and implementations where we again assume that every function has the same number of implementations. These calculations require summing over $k \in \{n, \ldots, m\}$ the cardinalities $|C_k|$, but Lemma 3, stated and proved in Section 2.2.2, shows

$$\sum_{k=n}^{m} |C_k| = \prod_{i=1}^{n} \left(2^{|\mathcal{F}_i|} - 1 \right), \tag{2.6}$$

making the calculations possible. Therefore, the expression given by (2.5) has an exponential computational complexity with a linear exponent.

$\lvert\mathcal{F}\rvert$	2	2	2	3	3	3	4	4	5	5
$\lvert\mathcal{F}_i\rvert$	2	3	4	2	3	4	2	3	2	3
Summands	9	49	225	27	343	3375	81	2401	243	16807

Table 2.2: Number of summands in formula (2.5).

When comparing Tables 2.1 and 2.2, we can see an enormous reduction in the number of terms involved to calculate the same value of $P(R)$. In the following, we prove Proposition 1 and its time complexity.

2.2.1 Proof of Proposition 1

In this section, we prove Proposition 1 after stating and proving an auxiliary result (Lemma 2). As we mentioned earlier (see (2.4)), there are different subsets of W in (2.3) that lead to the same intersection set and, therefore, the same probability. The first result of the following lemma enables us to count how many different subsets of W with the same cardinality $t \in \{1, \ldots, |W|\}$ lead to the same intersection set. The second result of Lemma 2 gives us the coefficient of an intersection set $\mathbf{E}$ in formula (2.5).

Lemma 2. *Let* $t, n \in \mathbb{N}^+$ *and let* $A_1, \ldots, A_n$ *be non-empty sets with* $A_i \cap A_j = \emptyset \ \forall i \neq j$, $A = \bigcup_{i=1}^{n} A_i$, $k = |A|$ *and* $D = A_1 \times \ldots \times A_n$. *Furthermore, let us define* $s(e) = \{e_1, \ldots, e_n\}$ *for* $e = (e_1, \ldots, e_n) \in D$ *and, given* $I \subseteq A$ *and* $\mathcal{A} = (A_1, \ldots, A_n)$, *let*

$$p(I, \mathcal{A}) = \prod_{i=1}^{n} |A_i \cap I| = |(A_1 \cap I) \times \ldots \times (A_n \cap I)|.$$

Let $c(\mathcal{A}, t)$ *be the total number of non-empty subsets* $\{e^1, \ldots, e^t\} \subseteq D$ *of cardinality* t *such that* $\bigcup_{\ell=1}^{t} s(e^\ell) = A$. *Then*

1. $c(\mathcal{A}, t) = \sum_{i=0}^{k-n} (-1)^i \sum_{\substack{I \subseteq A, \\ |I| = k-i}} \binom{p(I, \mathcal{A})}{t}$, (2.7)

2. $\sum_{t=1}^{|D|} (-1)^{t-1} c(\mathcal{A}, t) = (-1)^{k-n}$. (2.8)

Proof.
1. First we prove (2.7) by induction over $k \geq n$. We use $\binom{i}{j} = 0$ if $j > i$ several times throughout the proof. The first time it is needed is to see that for all t with $t > |D|$, we have that $c(\mathcal{A}, t) = 0$ because there exist no subsets of D with cardinality strictly greater than $|D|$. Furthermore, if $t > |D|$ and because $p(I, \mathcal{A}) \leq |D|$, the right-hand side of (2.7) is 0. Therefore, (2.7) holds for $t > |D|$. In the following, we assume that $t \leq |D|$.

If $k = n$, then $|A_1| = \ldots = |A_n| = 1$ and $|D| = 1$ and we can assume $t = 1$. Thus

$$\sum_{i=0}^{k-n} (-1)^i \sum_{\substack{I \subseteq A, \\ |I| = k-i}} \binom{p(I, \mathcal{A})}{t} = (-1)^0 \binom{1}{1} = 1.$$

Moreover, $|D| = 1$ means that there exists exactly one vector $e \in D$ and $s(e) = A$. Therefore, $c(\mathcal{A}, 1) = 1$ and the formula is correct for $k = n$.

Let us assume that (2.7) holds for $n, k, t \in \mathbb{N}^+$ with $k \geq n$. We will show that it also holds for $k + 1$. For this, let $A_1, \ldots, A_n$ be non-empty sets with $A_i \cap A_j = \emptyset$ for all $i \neq j$, $A = \bigcup_{i=1}^n A_i$ with $|A| = k + 1$ and $\mathcal{A} = (A_1, \ldots, A_n)$. We can write the number of non-empty subsets $\{e^1, \ldots, e^t\} \subseteq D$ of cardinality t from D with $\bigcup_{l=1}^t s(e^l) = A$ as the number of subsets $\{e^1, \ldots, e^t\} \subseteq D$ of cardinality t minus the number of subsets $\{e^1, \ldots, e^t\} \subseteq D$ of cardinality t with $\bigcup_{i=1}^t s(e^i) = A \backslash J$ for all non-empty sets $J \subseteq A$, $|J| \leq k + 1 - n$ (because $|A \backslash J| \geq n$) and $p(A \backslash J, \mathcal{A}) = \prod_{i=1}^n |(A_i \cap (A \backslash J))| = \prod_{i=1}^n |(A_i \backslash J)| \neq 0$. Since for these sets $A \backslash J$ we have that $|A \backslash J| \leq k$, we can use the induction hypothesis to obtain that the number of such sets, for each J, is $c(\mathcal{A} \backslash J, t)$, where we define $\mathcal{A} \backslash J = (A_1 \backslash J, \ldots, A_n \backslash J)$. Therefore, using that $|D| = p(A, \mathcal{A})$ we have that

$$\begin{aligned} c(\mathcal{A}, t) &= \binom{|D|}{t} - \sum_{j=1}^{k+1-n} \left(\sum_{\substack{J \subseteq A, \ |J| = j, \\ p(A \backslash J, \mathcal{A}) \neq 0}} c(\mathcal{A} \backslash J, t) \right) \\ &= \binom{p(A, \mathcal{A})}{t} - \sum_{j=1}^{k+1-n} \left(\sum_{\substack{J \subseteq A, \ |J| = j, \\ p(A \backslash J, \mathcal{A}) \neq 0}} \left(\sum_{i=0}^{k+1-j-n} (-1)^i \sum_{\substack{I \subseteq A \backslash J, \\ |I| = k+1-j-i}} \binom{p(I, \mathcal{A} \backslash J)}{t} \right) \right). \end{aligned}$$

We can drop the condition $p(A \backslash J, \mathcal{A}) \neq 0$, because if $p(A \backslash J, \mathcal{A}) = 0$ then there exists $i \in \{1, \ldots, n\}$ such that $A_i \backslash J = \emptyset$ and therefore we have for all $I \subseteq A \backslash J$ that $p(I, \mathcal{A} \backslash J) = 0$. Thus, by dropping the condition only zeros are added to the sum. Furthermore based on the definition of function p, we know that for all $I, J \subseteq A$:

$$p(I, \mathcal{A} \backslash J) = \prod_{i=1}^n |(A_i \backslash J) \cap I| = \prod_{i=1}^n |A_i \cap (I \backslash J)| = p(I \backslash J, \mathcal{A}).$$

Therefore

$$\begin{aligned} c(\mathcal{A}, t) =& \binom{p(A, \mathcal{A})}{t} - \sum_{j=1}^{k+1-n} \left(\sum_{J \subseteq A, \ |J| = j} \left(\sum_{i=0}^{k+1-j-n} (-1)^i \sum_{\substack{I \subseteq A \backslash J, \\ |I| = k+1-j-i}} \binom{p(I \backslash J, \mathcal{A})}{t} \right) \right) \\ =& \binom{p(A, \mathcal{A})}{t} - \sum_{j=1}^{k+1-n} \left(\sum_{J \subseteq A, \ |J| = j} \left(\sum_{i=0}^{k+1-j-n} (-1)^i \sum_{\substack{I \subseteq A, \ J \subseteq I, \\ |I| = k+1-i}} \binom{p(I \backslash J, \mathcal{A})}{t} \right) \right) \end{aligned}$$

$$= \binom{p(A,\mathcal{A})}{t} - \sum_{j=1}^{k+1-n} \left(\sum_{i=0}^{k+1-j-n} (-1)^i \left(\sum_{J\subseteq A,\ |J|=j} \sum_{\substack{I\subseteq A,\ J\subseteq I,\\ |I|=k+1-i}} \binom{p(I\backslash J,\mathcal{A})}{t} \right) \right). \quad (2.9)$$

Next, we define $A[\ell] = \{I \subseteq A \ : \ |I| = \ell\},\ \ell \in \mathbb{N}^+$, and we have that $\forall \ell \in \{1,\dots,|A|-1\}\ \forall L \in A[\ell]\ \forall j \in \{1,\dots,|A|-\ell\}$:

$$\begin{aligned} &|\{(I,J) \in A[\ell+j] \times A[j] \ : \ J \subseteq I,\ I\backslash J = L\}| \\ &= |\{(I,J) \in A[\ell+j] \times A[j] \ : \ L \cap J = \emptyset,\ I = L \cup J\}| \\ &= |\{(L \cup J, J) \in A[\ell+j] \times A[j] \ : \ L \cap J = \emptyset\}| \\ &= |\{J \in A[j] \ : \ J \subseteq A\backslash L\}| \\ &= \binom{|A|-\ell}{j}. \end{aligned} \quad (2.10)$$

The last step is known from the general result for unordered sampling without replacement in combinatorics. Equation (2.10) can now be used to rewrite (2.9) by summing over sets $L = I\backslash J$ with coefficients $\binom{|A|-\ell}{j}$ instead of summing over J and I separately. We can now write that

$$\begin{aligned} (2.9) =& \binom{p(A,\mathcal{A})}{t} \\ &- \sum_{j=1}^{k+1-n} \left(\sum_{\ell=0}^{k+1-j-n} (-1)^\ell \sum_{\substack{L\subseteq A,\\ |L|=k+1-j-\ell}} \left(\binom{|A|-(k+1-j-\ell)}{j} \binom{p(L,\mathcal{A})}{t} \right) \right) \\ =& \binom{p(A,\mathcal{A})}{t} - \sum_{j=1}^{k+1-n} \left(\sum_{\ell=0}^{k+1-j-n} (-1)^\ell \sum_{\substack{L\subseteq A,\\ |L|=k+1-j-\ell}} \left(\binom{j+\ell}{j} \binom{p(L,\mathcal{A})}{t} \right) \right). \end{aligned} \quad (2.11)$$

Now let $\hat{\ell} \in \{1,\dots,k+1-n\}$ and $L \in A[k+1-\hat{\ell}]$. It holds that

$$\forall j \in \{1,\dots,\hat{\ell}\}\ \exists \ell \in \{0,\dots,k+1-n-j\} : \ j+\ell = \hat{\ell}$$

and that L appears exactly once for all possible combinations $j + \ell = \hat{\ell}$ with the coefficients $(-1)^{\hat{\ell}-j}\binom{\hat{\ell}}{j} = (-1)^{\hat{\ell}+j}\binom{\hat{\ell}}{j}$ in (2.11). Therefore, we can take the sum over the sets $L \in A[k+1-\hat{\ell}]$ for $\hat{\ell} \in \{1,\dots,k+1-n\}$ and it holds that

$$\begin{aligned} (2.11) =& \binom{p(A,\mathcal{A})}{t} - \sum_{\hat{\ell}=1}^{k+1-n} \left(\sum_{\substack{L\subseteq A,\ p(L,\mathcal{A})\neq 0,\\ |L|=k+1-\hat{\ell}}} \left(\sum_{j=1}^{\hat{\ell}} (-1)^{\hat{\ell}+j} \binom{\hat{\ell}}{j} \right) \binom{p(L,\mathcal{A})}{t} \right) \\ =& \binom{p(A,\mathcal{A})}{t} - \sum_{\hat{\ell}=1}^{k+1-n} \left(\sum_{\substack{L\subseteq A,\ p(L,\mathcal{A})\neq 0,\\ |L|=k+1-\hat{\ell}}} (-1)^{\hat{\ell}} \left(\sum_{j=1}^{\hat{\ell}} (-1)^{j} \binom{\hat{\ell}}{j} \right) \binom{p(L,\mathcal{A})}{t} \right). \end{aligned} \quad (2.12)$$

Finally, by using the following known result from combinatorics

$$\sum_{i=0}^{n} (-1)^i \binom{n}{i} = \sum_{i=1}^{n} (-1)^i \binom{n}{i} + 1 = 0, \quad (2.13)$$

we have that

$$
\begin{aligned}
(2.12) =& \binom{p(A,\mathcal{A})}{t} - \sum_{\hat{\ell}=1}^{k+1-n} \left(\sum_{\substack{L\subseteq A,\\ |L|=k+1-\hat{\ell}}} (-1)^{\hat{\ell}} (-1) \binom{p(L,\mathcal{A})}{t} \right) \\
=& \binom{p(A,\mathcal{A})}{t} + \sum_{\hat{\ell}=1}^{k+1-n} \left(\sum_{\substack{L\subseteq A,\\ |L|=k+1-\hat{\ell}}} (-1)^{\hat{\ell}} \binom{p(L,\mathcal{A})}{t} \right) \\
=& \sum_{\hat{\ell}=0}^{k+1-n} \left(\sum_{\substack{L\subseteq A,\\ |L|=k+1-\hat{\ell}}} (-1)^{\hat{\ell}} \binom{p(L,\mathcal{A})}{t} \right).
\end{aligned}
$$

This completes the proof for (2.7).

2.) Now we will prove the second result, (2.8), for $n \geq 1$ and $k \geq 1$ with $n \leq k$ by induction over $m = k - n$. Let $n \geq 1$, $k \geq 1$ and let $A_1, \ldots, A_n$ be non-empty sets with $A_i \cap A_j = \emptyset$ for all $i \neq j$, $A = \bigcup_{i=1}^{n} A_i$ with $|A| = k$, $\mathcal{A} = (A_1, \ldots, A_n)$ and $D = A_1 \times \ldots \times A_n$. We can rewrite (2.8) by using (2.7) as follows:

$$
\begin{aligned}
\sum_{t=1}^{|D|} (-1)^{t-1} c(\mathcal{A}, t) &= \sum_{t=1}^{p(A,\mathcal{A})} \left((-1)^{t-1} \sum_{i=0}^{k-n} \left(\sum_{\substack{I\subseteq A,\\ |I|=k-i}} (-1)^{i} \binom{p(I,\mathcal{A})}{t} \right) \right) \\
&= \sum_{t=1}^{p(A,\mathcal{A})} \left((-1)^{t-1} \sum_{i=0}^{k-n} \left(\sum_{\substack{I\subseteq A,\ p(I,\mathcal{A})\neq 0,\\ |I|=k-i}} (-1)^{i} \binom{p(I,\mathcal{A})}{t} \right) \right) \\
&= \sum_{i=0}^{k-n} (-1)^{i} \left(\sum_{\substack{I\subseteq A,\ p(I,\mathcal{A})\neq 0,\\ |I|=k-i}} \sum_{t=1}^{p(I,\mathcal{A})} (-1)^{t-1} \binom{p(I,\mathcal{A})}{t} \right). \qquad (2.14)
\end{aligned}
$$

Next we use (2.13) again and thus,

$$
\begin{aligned}
(2.14) &= \sum_{i=0}^{k-n} (-1)^{i} \left(\sum_{\substack{I\subseteq A,\ p(I,\mathcal{A})\neq 0,\\ |I|=k-i}} (-1) \left(\sum_{t=0}^{p(I,\mathcal{A})} (-1)^{t} \binom{p(I,\mathcal{A})}{t} - 1 \right) \right) \\
&= \sum_{i=0}^{k-n} (-1)^{i} \left(\sum_{\substack{I\subseteq A,\ p(I,\mathcal{A})\neq 0,\\ |I|=k-i}} (-1)^{2} \right) \\
&= \sum_{i=0}^{k-n} (-1)^{i} \left(\sum_{\substack{I\subseteq A,\ p(I,\mathcal{A})\neq 0,\\ |I|=k-i}} 1 \right).
\end{aligned}
$$

If we now modify the external sum on the previous expression to start with $i = n$, it follows that

$$
\sum_{t=1}^{|D|} (-1)^{t-1} c(\mathcal{A}, t) = (-1)^{k} \sum_{i=n}^{k} (-1)^{i} \left(\sum_{\substack{I\subseteq A,\ p(I,\mathcal{A})\neq 0,\\ |I|=i}} 1 \right). \qquad (2.15)
$$

If $m = 0$ and therefore $n = k$, then

$$\sum_{t=1}^{|D|}(-1)^{t-1}c(\mathcal{A},t) = (-1)^k \sum_{i=n}^{k}(-1)^i \left(\sum_{\substack{I\subseteq A,\ p(I,\mathcal{A})\neq 0,\\ |I|=i}} 1 \right)$$
$$= (-1)^n \cdot (-1)^n \cdot 1 = 1 = (-1)^{k-n}.$$

As a consequence, equation (2.8) holds for $m = 0$.

We assume now that it holds for $n \geq 1$ and $k \geq 1$ with $m = k - n$ and $m \geq 0$. Without loss of generality, we show that it also holds for $m + 1$ with $m + 1 = (k + 1) - n$ by fixing n. For this, let $A_1, \ldots, A_n$ be non-empty sets with $A_i \cap A_j = \emptyset$ for all $i \neq j$, and $A = \bigcup_{i=1}^{n} A_i$ with $|A| = \sum_{i=1}^{n} |A_i| = k$. Furthermore, let $A_1^*, A_2^*, \ldots, A_n^*$ be non-empty sets with $A^* = \bigcup_{i=1}^{n} A_i^*$ and $|A^*| = k + 1$. Without loss of generality, let $A_2 = A_2^*, \ldots, A_n = A_n^*$ and $A_1 = A_1^* \backslash \{\delta\}$ with $\delta \notin \bigcup_{i=2}^{n} A_i$. Also, let $\hat{A}_1, \ldots, \hat{A}_{n-1}$ be non-empty sets with $\hat{A}_1 = A_2, \ldots, \hat{A}_{n-1} = A_n$ and $\hat{A} = \bigcup_{i=1}^{n-1} \hat{A}_i$. By using (2.15), it holds that

$$\sum_{t=1}^{p(A^*,\mathcal{A}^*)} (-1)^{t-1}c(\mathcal{A}^*,t) = (-1)^{k+1} \sum_{i=n}^{k+1}(-1)^i \left(\sum_{\substack{I\subseteq A^*,\ p(I,\mathcal{A}^*)\neq 0,\\ |I|=i}} 1 \right)$$
$$= (-1) \left((-1)^k \sum_{i=n}^{k+1}(-1)^i \left(\sum_{\substack{I\subseteq A^*,\ p(I,\mathcal{A}^*)\neq 0,\\ |I|=i}} 1 \right) \right).$$

The sum over the subsets I can be split by considering whether the subsets contain δ or not:

$$= (-1) \left(\left((-1)^k \sum_{i=n}^{k+1}(-1)^i \left(\sum_{\substack{I\subseteq A^*,\ p(I,\mathcal{A}^*)\neq 0,\\ |I|=i,\ \delta\notin I}} 1 \right) \right) + \left((-1)^k \sum_{i=n}^{k+1}(-1)^i \left(\sum_{\substack{I\subseteq A^*,\ p(I,A^*)\neq 0,\\ |I|=i,\ \delta\in I}} 1 \right) \right) \right)$$
$$= (-1) \left((-1)^k \sum_{i=n}^{k+1}(-1)^i \left(\sum_{\substack{I\subseteq A^*\backslash\{\delta\},\ p(I,\mathcal{A}^*\backslash\{\delta\})\neq 0,\\ |I|=i}} 1 \right) \right)$$
$$+ (-1) \left((-1)^k \sum_{i=n}^{k+1}(-1)^i \left(\sum_{\substack{I\subseteq A^*,\ p(I,A^*)\neq 0,\\ |I|=i,\ \delta\in I}} 1 \right) \right).$$

Using that $A^*\backslash\{\delta\} = A$ and that no subset of A can be of cardinality $k + 1$, we can rewrite the first sum:

$$= (-1) \left(\left((-1)^k \sum_{i=n}^{k}(-1)^i \left(\sum_{\substack{I\subseteq A,\ p(I,\mathcal{A})\neq 0,\\ |I|=i}} 1 \right) \right) + \left((-1)^k \sum_{i=n}^{k+1}(-1)^i \left(\sum_{\substack{I\subseteq A^*,\ p(I,\mathcal{A}^*)\neq 0,\\ |I|=i,\ \delta\in I}} 1 \right) \right) \right)$$
$$= (-1) \left(\sum_{t=1}^{p(A,\mathcal{A})} (-1)^{t-1}c(\mathcal{A},t) + \left((-1)^k \sum_{i=n}^{k+1}(-1)^i \left(\sum_{\substack{I\subseteq A^*,\ p(I,\mathcal{A}^*)\neq 0,\\ |I|=i,\ \delta\in I}} 1 \right) \right) \right),$$

where we use (2.15) to obtain this last equality. Finally, by using the induction hypothesis for $m = k - n$:

$$= (-1) \left((-1)^{k-n} + \left((-1)^k \sum_{i=n}^{k+1}(-1)^i \left(\sum_{\substack{I\subseteq A^*,\ p(I,\mathcal{A}^*)\neq 0,\\ |I|=i,\ \delta\in I}} 1 \right) \right) \right)$$

$$= (-1)^{k+1-n} + (-1)\left((-1)^k \sum_{i=n}^{k+1} (-1)^i \left(\sum_{\substack{I \subseteq A^*,\ p(I,\mathcal{A}^*)\neq 0,\\ |I|=i,\ \delta \in I}} 1 \right)\right).$$

Thus to prove that (2.8) holds for $m+1$, we only have to show that

$$(-1)^k \sum_{i=n}^{k+1} (-1)^i \left(\sum_{\substack{I \subseteq A^*,\ p(I,\mathcal{A}^*)\neq 0,\\ |I|=i,\ \delta \in I}} 1 \right) = 0. \tag{2.16}$$

First, we rewrite the left-hand side of (2.16) as follows:

$$\begin{aligned}
&(-1)^k \sum_{i=n}^{k+1} (-1)^i \left(\sum_{\substack{I \subseteq A^*,\ p(I,\mathcal{A}^*)\neq 0,\\ |I|=i,\ \delta \in I}} 1 \right) \\
=&(-1)^k \sum_{i=n}^{k+1} (-1)^i \left(\sum_{\substack{I \subseteq A^*\setminus\{\delta\},\ p(I\cup\{\delta\},\mathcal{A}^*)\neq 0,\\ |I|=i-1}} 1 \right) \\
=&(-1)^k \sum_{i=n-1}^{k} (-1)^{i+1} \left(\sum_{\substack{I \subseteq A^*\setminus\{\delta\},\ p(I\cup\{\delta\},\mathcal{A}^*)\neq 0,\\ |I|=i}} 1 \right).
\end{aligned} \tag{2.17}$$

The following observations are needed to rewrite (2.17) further.

1. First note that $A^*\setminus A_1^* = \hat{A}$, and that for all $I \subseteq \hat{A}$:

$$\begin{aligned}
p(I \cup \{\delta\}, \mathcal{A}^*) &= \prod_{i=1}^{n} |A_i^* \cap (I \cup \{\delta\})| \\
=&|\{\delta\}| \cdot \prod_{i=2}^{n} |A_i^* \cap I| = \prod_{i=1}^{n-1} |\hat{A}_i \cap I| = p(I, \hat{\mathcal{A}}).
\end{aligned} \tag{2.18}$$

2. In addition, $A^*\setminus\{\delta\} = A$ and for all $I \subseteq A$ it holds that

$$p(I \cup \{\delta\}, \mathcal{A}^*) = \prod_{i=1}^{n} |A_i^* \cap (I \cup \{\delta\})\,| = (|A_1 \cap I| + 1) \prod_{i=2}^{n} |A_i^* \cap I|.$$

   Therefore,

$$p(I \cup \{\delta\}, \mathcal{A}^*) \neq 0 \Leftrightarrow \prod_{i=2}^{n} |A_i^* \cap I| = \prod_{i=1}^{n-1} |\hat{A}_i \cap I| = p(I, \hat{\mathcal{A}}) \neq 0.$$

3. Moreover, it holds that

$$\begin{aligned}
&\forall I \subseteq A:\ p(I, \hat{\mathcal{A}}) \neq 0\ \wedge\ |I \cap A_1| \neq 0 \\
\Leftrightarrow\quad &p(I, \hat{\mathcal{A}}) \cdot |I \cap A_1| = p(I, \mathcal{A}) \neq 0.
\end{aligned} \tag{2.19}$$

Now we can rewrite (2.17) by splitting the sum over subsets I by considering whether or not the

subset is disjoint with A_1.

$$
\begin{aligned}
&(-1)^k \sum_{i=n-1}^{k} (-1)^i \left(\sum_{\substack{I\subseteq A^*\setminus\{\delta\},\ p(I\cup\{\delta\},\mathcal{A}^*)\neq 0,\\ |I|=i}} 1 \right) \\
&= (-1)^k \sum_{i=n-1}^{k} (-1)^i \left(\sum_{\substack{I\subseteq A^*\setminus A_1^*,\ p(I\cup\{\delta\},\mathcal{A}^*)\neq 0,\\ |I|=i}} 1 \right) \\
&\quad + (-1)^k \sum_{i=n}^{k} (-1)^i \left(\sum_{\substack{I\subseteq A^*\setminus\{\delta\},\ p(I\cup\delta,\mathcal{A}^*)\neq 0,\\ |I|=i,\ I\cap A_1\neq\emptyset}} 1 \right).
\end{aligned}
\tag{2.20}
$$

By using (2.18), we can rewrite the sum over $I \subseteq A^*\setminus A_1^* = \hat{A}$ and, by using (2.19), the sum over $I \subseteq A^*\setminus\{\delta\} = A$. Furthermore, we can change the upper limit of the first sum in (2.20) to $|\hat{A}| = k - |A_1|$. Therefore, we can write that (2.20) is

$$
\begin{aligned}
&= (-1)^k \sum_{i=n-1}^{k-|A_1|} (-1)^i \left(\sum_{\substack{I\subseteq \hat{A},\ p(I,\hat{A})\neq 0,\\ |I|=i}} 1 \right) + (-1)^k \sum_{i=n}^{k} (-1)^i \left(\sum_{\substack{I\subseteq A,\ p(I,A)\neq 0,\\ |I|=i}} 1 \right) \\
&= (-1)^{|A_1|} (-1)^{k-|A_1|} \sum_{i=n-1}^{k-|A_1|} (-1)^i \left(\sum_{\substack{I\subseteq \hat{A},\ p(I,\hat{A})\neq 0,\\ |I|=i}} 1 \right) \\
&\quad + (-1)^k \sum_{i=n}^{k} (-1)^i \left(\sum_{\substack{I\subseteq A,\ p(I,A)\neq 0,\\ |I|=i}} 1 \right).
\end{aligned}
\tag{2.21}
$$

Further, as $(k-|A_1|)-(n-1) < m+1$ and $k-n < m+1$, we can use the induction hypothesis on both sums and we use (2.15) to obtain that

$$
(2.21) = \left((-1)^{|A_1|} (-1)^{k-|A_1|-(n-1)} + (-1)^{k-n} \right) = \left((-1)^{k-(n-1)} + (-1)^{k-n} \right) = 0.
$$

Hence, we have proved that

$$
\sum_{t=1}^{|D|} (-1)^{t-1} c(\mathcal{A}, t) = (-1)^{k-n} \tag{2.22}
$$

holds for any $n \geq 1$ and $k \geq 1$ with $k - n \geq 0$ which completes the proof of Lemma 2. □

We can now prove our main result, Proposition 1.

Proof of Proposition 1.

Since $R = \bigcap_{F\in\mathcal{F}} F$, it follows that

$$
P(R) = P\left(\bigcap_{i=1}^{n} F_i \right) = P\left(\bigcap_{i=1}^{n} \left(\bigcup_{j=1}^{t_i} F_{ij} \right) \right). \tag{2.23}
$$

Let

$$W = \{1, \ldots, t_1\} \times \ldots \times \{1, \ldots, t_n\},$$
$$B_w = \bigcap_{i=1}^{n} F_{iw_i} \quad \text{and}$$
$$\mathfrak{B}_w = \{F_{1w_1}, \ldots, F_{nw_n}\} \text{ for } w = (w_1, \ldots, w_n) \in W.$$

We can rewrite (2.23) as

$$P(R) = P\left(\bigcup_{w \in W}\left(\bigcap_{i=1}^{n} F_{iw_i}\right)\right) = P\left(\bigcup_{w \in W} B_w\right)$$

as in (2.3). Using the probability principle of inclusion-exclusion (2.1), it holds that

$$P(R) = \sum_{t=1}^{|W|}\left((-1)^{t+1} \sum_{\substack{I \subseteq W, \\ |I|=t}} P\left(\bigcap_{j \in I} B_j\right)\right). \tag{2.24}$$

Based on the definition of $B_w,\ w \in W$, we know that

$$\forall I \subseteq W\ \exists k \in \{n, n+1, \ldots, m\} \text{ and } \exists\ \mathbf{E} = \{E_1, \ldots, E_k\} \in C_k :$$
$$\bigcap_{j \in I} B_j = \bigcap_{i=1}^{k} E_i. \tag{2.25}$$

We define for all $k \in \{n, n+1, \ldots, m\}$ and $\mathbf{E} \in C_k$:

$$D_{\mathbf{E}} = \{B_w \ : \ \mathfrak{B}_w \cap \mathbf{E} = \mathfrak{B}_w \text{ and } w \in W\}.$$

Furthermore, for all $k \in \{n, \ldots, m\},\ \mathbf{E} \in C_k$ and $\ell \in \{1, \ldots, |D_{\mathbf{E}}|\}$, let us define

$$T(\mathbf{E}, \ell) = \left\{I \subseteq W \ : \ |I| = \ell \text{ and } \bigcup_{i \in I} \mathfrak{B}_i = \mathbf{E}\right\} \text{ and}$$
$$t(\mathbf{E}, \ell) = |T(\mathbf{E}, \ell)|.$$

If we use (2.25), we can rewrite (2.24) to a sum over $\mathbf{E} \in C_k,\ k = n, \ldots, m$, where the coefficients are $\sum_{i=1}^{|W|}(-1)^{i+1} t(\mathbf{E}, i)$. Furthermore, we can rewrite it as $\sum_{i=1}^{|D_{\mathbf{E}}|}(-1)^{i+1} t(\mathbf{E}, i)$, because $t(\mathbf{E}, \ell)$ is zero for $\mathbf{E} \in C_k, k = n, \ldots, m$ and $\ell \geq 1$ if $|D_{\mathbf{E}}| < \ell$. Hence, we have that (2.24) is

$$= \sum_{k=n}^{m}\left(\sum_{E \in C_k}\left(\sum_{i=1}^{|D_E|}(-1)^{i+1} t(E, i)\right) P\left(\bigcap_{j=1}^{k} E_j\right)\right) \tag{2.26}$$

In addition, let $\mathcal{E} = (\mathcal{F}_1 \cap \mathbf{E}, \ldots, \mathcal{F}_n \cap \mathbf{E})$. Based on the first result (2.7) from Lemma 2, we know that $t(\mathbf{E}, \ell) = c(\mathcal{E}, \ell)$ for all $\mathbf{E} \in C_k,\ k \in \{n, n+1, \ldots, m\}$ and $\ell \in \{1, \ldots, |D_{\mathbf{E}}|\}$. This gives us that (2.25) is

$$= \sum_{k=n}^{m}\left(\sum_{E \in C_k}\left(\sum_{i=1}^{|D_E|}(-1)^{i+1} c(\mathcal{E}, i)\right) P\left(\bigcap_{j=1}^{k} E_j\right)\right)$$

and using (2.8) from Lemma 2, it holds that

$$= \sum_{k=n}^{m} \left(\sum_{E \in C_k} (-1)^{k-n} P\left(\bigcap_{j=1}^{k} E_j \right) \right)$$
$$= \sum_{k=n}^{m} \left((-1)^{k-n} \sum_{E \in C_k} P\left(\bigcap_{j=1}^{k} E_j \right) \right).$$

This completes the proof and we have shown that (2.5) holds. □

Now we going to prove the time complexity.

2.2.2 Proof of Time Complexity

Lemma 3 states the time complexity of the new formula in Proposition 1 and the proof for this Lemma is given next.

Lemma 3. *Let $n \geq 1$, $\mathcal{F}_i = \{F_{i1}, \ldots, F_{it_i}\}$, $i \in \{1, \ldots, n\}$, be sets of cardinality t_i, $t_i \geq 1$ and $\mathbf{F} = (\mathcal{F}_i, \ldots, \mathcal{F}_n)$. Let $m \coloneqq \sum_{i=1}^{n} t_i$ and, given $k \in \{n, n+1, \ldots, m\}$, let*

$$C_k^{\mathbf{F}} \coloneqq \{\mathbf{E} = \{E_1, \ldots, E_k\} : \ E_u = F_{i(u)j(u)} \text{ for some } i(u) \in \{1, \ldots, n\}, \ j(u) \in \{1, \ldots, t_{i(u)}\},$$
$$u \in \{1, \ldots, k\}, \ \{i(l) \mid l \in \{1 \ldots, k\}\} = \{1, \ldots, n\} \text{ and } E_u \neq E_v \text{ for } u \neq v \in \{1, \ldots, k\}\}.$$

Lastly, let $\mathfrak{c}(\mathbf{F}) \coloneqq \sum_{k=n}^{m} |C_k^{\mathbf{F}}|$. It holds that

$$\mathfrak{c}(\mathbf{F}) = \sum_{k=n}^{m} |C_k^{\mathbf{F}}| = \prod_{i=1}^{n} \left(2^{|\mathcal{F}_i|} - 1 \right). \tag{2.27}$$

Proof. We prove (2.27) by induction. Let $n \geq 1$ and $t_i \geq 1$, $i \in \{1, \ldots, n\}$. We use induction on $\ell = m - n$. Because all t_i, $i \in \{1, \ldots, n\}$, are greater than 1, $m \geq n$ and we can first assume $\ell = 0$ with $m = n$. With $m = n$, we have that $t_1 = \ldots = t_n = 1$. Therefore $C_n = \{\{F_{11}, \ldots, F_{n1}\}\}$ and $\mathfrak{c}(\mathbf{F}) = \sum_{k=n}^{m} |C_k| = |C_n| = 1$. Also

$$\prod_{i=1}^{n} \left(2^{|\mathcal{F}_i|} - 1 \right) = \prod_{i=1}^{n} 1 = 1$$

and (2.27) holds for $\ell = 0$.

We assume now that (2.27) holds for $n \geq 1$ and $t_i \geq 1$, $i \in \{1, \ldots, n\}$, with $\ell = m - n$ and $\ell \geq 0$. Without loss of generality, we show it also holds for $\ell + 1$ with $\ell + 1 = (m + 1) - n$ by fixing n. Let $n \geq 1$, $\mathcal{F}_i = \{F_{i1}, \ldots, F_{it_i}\}$, for $i \in \{1, \ldots, n\}$ be sets of cardinality t_i, $t_i \geq 1$ and $m = \sum_{i=1}^{n} t_i$. We then know that $\mathfrak{c}(\mathbf{F}) = \prod_{i=1}^{n} \left(2^{|\mathcal{F}_i|} - 1\right)$. Without loss of generality, let $\mathcal{F}_i^* = \mathcal{F}_i$, $i \in \{1, \ldots, n-1\}$, $\mathcal{F}_n^* = \mathcal{F}_n \cup \{\delta\}$, $\delta \notin \mathcal{F}_n$, and $\mathbf{F}^* = (\mathcal{F}_1^*, \ldots, \mathcal{F}_n^*)$. Furthermore, let $\hat{\mathcal{F}}_i = \mathcal{F}_i$, $i \in \{1, \ldots, n-1\}$ and $\hat{\mathbf{F}} = (\hat{\mathcal{F}}_1, \ldots, \hat{\mathcal{F}_{n-1}})$. For $k \in \{n, \ldots, m+1\}$, we can split the set $C_k^{\mathbf{F}^*}$ by considering whether or not δ is contained in $\mathbf{E}$. We have that

$$C_k^{\mathbf{F}^*} = \{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E}\} \cup \{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \notin \mathbf{E}\} \text{ and}$$
$$|C_k^{\mathbf{F}^*}| = |\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E}\}| + |\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \notin \mathbf{E}\}|.$$

For $k \in \{n, \ldots, m\}$, $\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \notin \mathbf{E}\} = C_k^{\mathbf{F}}$ and $\{\mathbf{E} \in C_{m+1}^{\mathbf{F}^*} : \delta \notin \mathbf{E}\} = \emptyset$. By the induction hypothesis,

$$\sum_{k=n}^{m+1} |\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \notin \mathbf{E}\}| = \sum_{k=n}^{m} |C_k^{\mathbf{F}}| = \prod_{i=1}^{n} \left(2^{|\mathcal{F}_i|} - 1 \right) = \mathfrak{c}(\mathbf{F}). \tag{2.28}$$

The set $\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E}\}$, $k \in \{n, \dots, m+1\}$, can be further split into two sets by considering if, for $E \in C_k^{\mathbf{F}^*}$, $\mathcal{F}_n \cap \mathbf{E} = \emptyset$ or $\mathcal{F}_n \cap \mathbf{E} \neq \emptyset$. This gives

$$\begin{aligned}
&\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E}\} \\
&= \{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} = \emptyset\} \cup \{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} \neq \emptyset\} \text{ and hence} \\
&|\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E}\}| \\
&= |\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} = \emptyset\}| + |\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} \neq \emptyset\}|.
\end{aligned}$$

Let $\hat{m} = \sum_{i=1}^{n-1} t_i$. The set $\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} = \emptyset\}$, $k \in \{n, \dots, \hat{m}+1\}$, can also be rewritten as

$$\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} = \emptyset\} = \{\mathbf{E} \cup \{\delta\} : \mathbf{E} \in C_{k-1}^{\hat{\mathbf{F}}}\}.$$

Therefore it holds that

$$|\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} = \emptyset\}| = |\{\mathbf{E} \cup \{\delta\} : \mathbf{E} \in C_{k-1}^{\hat{\mathbf{F}}}\}| = |C_{k-1}^{\hat{\mathbf{F}}}|.$$

Next, since $|\left(\bigcup_{i=1}^{n} \mathcal{F}_i^*\right) \cap \mathcal{F}_n| = \hat{m}+1$, we have that $\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} = \emptyset\} = \emptyset$ for $k \in \{\hat{m}+2, \dots, m+1\}$. Because $\hat{m}+1-(n-1) < (m+1)-n = \ell+1$, we can use the induction hypothesis to see that

$$\begin{aligned}
\sum_{k=n}^{m+1} |\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} = \emptyset\}| &= \sum_{k=n}^{\hat{m}+1} |C_{k-1}^{\hat{\mathbf{F}}}| \\
&= \sum_{k=n-1}^{\hat{m}} |C_k^{\hat{\mathbf{F}}}| = \prod_{i=1}^{n-1} \left(2^{|\hat{\mathcal{F}}_i|} - 1\right) = \mathfrak{c}(\hat{\mathbf{F}}).
\end{aligned} \tag{2.29}$$

Now we consider the sets $\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} \neq \emptyset\}$, $k \in \{n, \dots, m\}$. For $k = n$, let $\mathbf{E} \in C_k^{\mathbf{F}^*}$ and we have that $|\mathbf{E} \cap \mathcal{F}_i^*| = 1$ for $i \in \{1, \dots, n\}$. Therefore,

$$\{\mathbf{E} \in C_n^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} \neq \emptyset\} = \emptyset.$$

For $k \in \{n+1, \dots, m+1\}$ it holds that

$$\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} \neq \emptyset\} = \{\mathbf{E} \cup \{\delta\} : \mathbf{E} \in C_{k-1}^{\mathbf{F}}\}$$

and thus it follows that

$$|\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} \neq \emptyset\}| = |\{\mathbf{E} \cup \{\delta\} : \mathbf{E} \in C_{k-1}^{\mathbf{F}}\}| = |C_{k-1}^{\mathbf{F}}|. \tag{2.30}$$

By using the induction hypothesis again,

$$\begin{aligned}
\sum_{k=n}^{m+1} |\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} \neq \emptyset\}| &= \sum_{k=n+1}^{m+1} |C_{k-1}^{\mathbf{F}}| = \sum_{k=n}^{m} |C_k^{\mathbf{F}}| \\
&= \prod_{i=1}^{n} \left(2^{|\mathcal{F}_i|} - 1\right) = \mathfrak{c}(\mathbf{F}).
\end{aligned}$$

We can now write

$$\begin{aligned}
\mathfrak{c}(\mathbf{F}^*) &= \sum_{k=n}^{m} |C_k^{\mathbf{F}^*}| \\
&= \sum_{k=n}^{m} |\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \notin \mathbf{E}\}| + \sum_{k=n}^{m} |\{\mathbf{E} \in C_k^{\mathbf{F}^*} : \delta \in \mathbf{E} \text{ and } \mathcal{F}_n \cap \mathbf{E} = \emptyset\}|
\end{aligned}$$

$$+\sum_{k=n}^{m}\left|\{\mathbf{E}\in C_k^{\mathbf{F}^*}:\delta\in\mathbf{E}\text{ and }\mathcal{F}_n\cap\mathbf{E}\neq\emptyset\}\right|$$

and using (2.28), (2.29) and (2.30), we obtain that

$$\begin{aligned}
&= 2\mathfrak{c}(\mathbf{F})+\mathfrak{c}(\hat{\mathbf{F}}) = 2\prod_{i=1}^{n}\left(2^{|\mathcal{F}_i|}-1\right)+\prod_{i=1}^{n-1}\left(2^{|\mathcal{F}_i|}-1\right)\\
&= \left(2\left(2^{|\mathcal{F}_n|}-1\right)+1\right)\prod_{i=1}^{n-1}\left(2^{|\mathcal{F}_i|}-1\right) = \prod_{i=1}^{n-1}\left(2^{|\mathcal{F}_i|}-1\right)*\left(2^{|\mathcal{F}_n|+1}-1\right)\\
&= \prod_{i=1}^{n}\left(2^{|\mathcal{F}_i^*|}-1\right).
\end{aligned}$$

This shows that (2.27) holds for $\ell+1$ and thus, by induction, completes the proof. □

2.3 Generating Algorithm for Sets C_k

In this section, we provide a generating algorithm for the elements of sets C_k in Proposition 1. Furthermore, we include an example of how the algorithm works for $n=2$ and $t_1=t_2=3$.

The algorithm only needs the inputs n and $t_i,\ i\in\{1,\dots,n\}$ and is described in Algorithm 1. The correctness of Algorithm 1 is proven in the Subsection 2.3.1.

Algorithm 1 Generating Algorithm

Require: n and $t=\{t_1,\dots,t_n\}$.
1: Create sets $H_i=\{F_{i,p}\mid p\in\{1,\dots,t_i\}\},\ i\in\{1,\dots,n\}$ and let $m=\sum_{i=1}^{n}t_n$.
2: **for** $k\in\{n,\dots,m\}$
3: Set $E=\{\}$.
4: RECURSION(k,n,E)
5: **end for**
6: **procedure** RECURSION(l,s,E)
7: Set $ub_s=\min\{l-(s-1),t_s\}$ and $lb_s=\max\{1,l-\sum_{b=1}^{s-1}t_b\}$.
8: **for** $i\in\{lb_s,\dots,ub_s\}$
9: Create all combinations $Comb_{si}$ of H_s of length i.
10: **for** $J\in Comb_{si}$
11: Set $\hat{E}=E\cup J$.
12: **if** $s>1$
13: RECURSION$(l-i,s-1,\hat{E})$
14: **end if**
15: **else**
16: Output $\hat{E}$ as element of C_k
17: **end else**
18: **end for**
19: **end for**

We now present an example of how to generate elements of C_k sets and the formula from Proposition 1 for $n=2$ and $t_1=t_2=3$. Furthermore, we show how the algorithm works by generating elements of C_3.

1. First, we define the two sets $H_1=\{F_{(1,1)},F_{(1,2)},F_{(1,3)}\}$ and $H_2=\{F_{(2,1)},F_{(2,2)},F_{(2,3)}\}$.
2. Next, we set $E=\emptyset$ and start the Recursion procedure with RECURSION$(3,2,E)$.
3. The next step is to calculate ub_2 and lb_2. In this case we have $ub_2=\min\{(3-(2-1),3\}=2$ and $lb_2=\max\{1,3-(2-1)\times 3\}=1$.

4. For $i = 1$, we create all possible combinations $Comb_{21}$ of H_2 of length 1 and obtain $Comb_{21} = H_2$.

5. We choose $J = \{F_{(2,1)}\}$ and we have that $\hat{E} = E \cup J = \{F_{(2,1)}\}$.

6. As $s = 2 > 1$, we start the recursion procedure with $\textsc{Recursion}(2, 1, \hat{E})$.

7. We have to calculate lb_1 and ub_2. We have that $lb_1 = ub_1 = 2$.

8. Therefore, $i = 2$ and we create $Comb_{12} = \{\{F_{(1,1)}, F_{(1,2)}\}, \{F_{(1,1)}, F_{(1,3)}\}, \{F_{(1,2)}, F_{(1,3)}\}\}$.

9. First, we choose $J = \{F_{(1,1)}, F_{(1,2)}\}$ and we have that $\hat{E} = \{F_{(2,1)}, F_{(1,1)}, F_{(1,2)}\}$.

10. As $s = 1$, $\hat{E}$ is an element of C_3 and we iterate over the elements of $Comb_{12}$. We also obtain $\{F_{(2,1)}, F_{(1,1)}, F_{(1,3)}\}$ and $\{F_{(2,1)}, F_{(1,2)}, F_{(1,3)}\}$ which are elements of C_3.

After running the algorithm for all $k \in \{2, \ldots, 6\}$, we obtain the following sets $C_k, k \in \{2, \ldots, 6\}$.

$$C_2 = \{\{F_{(1,1)}, F_{(2,1)}\}, \{F_{(1,1)}, F_{(2,2)}\}, \{F_{(1,1)}, F_{(2,3)}\}, \{F_{(1,2)}, F_{(2,1)}\}, \{F_{(1,2)}, F_{(2,2)}\},$$
$$\{F_{(1,2)}, F_{(2,3)}\}, \{F_{(1,2)}, F_{(2,1)}\}, \{F_{(1,2)}, F_{(2,2)}\}, \{F_{(1,2)}, F_{(2,3)}\}\},$$

$$C_3 = \{\{F_{(2,1)}, F_{(1,1)}, F_{(1,2)}\}, \{F_{(2,1)}, F_{(1,1)}, F_{(1,3)}\}, \{F_{(2,1)}, F_{(1,2)}, F_{(1,3)}\}, \{F_{(2,2)}, F_{(1,1)}, F_{(1,2)}\},$$
$$\{F_{(2,2)}, F_{(1,1)}, F_{(1,3)}\}, \{F_{(2,2)}, F_{(1,2)}, F_{(1,3)}\}, \{F_{(2,3)}, F_{(1,1)}, F_{(1,2)}\}, \{F_{(2,3)}, F_{(1,1)}, F_{(1,3)}\},$$
$$\{F_{(2,3)}, F_{(1,2)}, F_{(1,3)}\}, \{F_{(2,1)}, F_{(2,2)}, F_{(1,1)}\}, \{F_{(2,1)}, F_{(2,2)}, F_{(1,2)}\}, \{F_{(2,1)}, F_{(2,2)}, F_{(1,3)}\},$$
$$\{F_{(2,1)}, F_{(2,3)}, F_{(1,1)}\}, \{F_{(2,1)}, F_{(2,3)}, F_{(1,2)}\}, \{F_{(2,1)}, F_{(2,3)}, F_{(1,3)}\}, \{F_{(2,2)}, F_{(2,3)}, F_{(1,1)}\},$$
$$\{F_{(2,2)}, F_{(2,3)}, F_{(1,2)}\}, \{F_{(2,2)}, F_{(2,3)}, F_{(1,3)}\}\},$$

$$C_4 = \{\{F_{(2,1)}, F_{(1,1)}, F_{(1,2)}, F_{(1,3)}\}, \{F_{(2,2)}, F_{(1,1)}, F_{(1,2)}, F_{(1,3)}\}, \{F_{(2,3)}, F_{(1,1)}, F_{(1,2)}, F_{(1,3)}\},$$
$$\{F_{(2,1)}, F_{(2,2)}, F_{(1,1)}, F_{(1,2)}\}, \{F_{(2,1)}, F_{(2,2)}, F_{(1,1)}, F_{(1,3)}\}, \{F_{(2,1)}, F_{(2,2)}, F_{(1,2)}, F_{(1,3)}\},$$
$$\{F_{(2,1)}, F_{(2,3)}, F_{(1,1)}, F_{(1,2)}\}, \{F_{(2,1)}, F_{(2,3)}, F_{(1,1)}, F_{(1,3)}\}, \{F_{(2,1)}, F_{(2,3)}, F_{(1,2)}, F_{(1,3)}\},$$
$$\{F_{(2,2)}, F_{(2,3)}, F_{(1,1)}, F_{(1,2)}\}, \{F_{(2,2)}, F_{(2,3)}, F_{(1,1)}, F_{(1,3)}\}, \{F_{(2,2)}, F_{(2,3)}, F_{(1,2)}, F_{(1,3)}\},$$
$$\{F_{(2,1)}, F_{(2,2)}, F_{(2,3)}, F_{(1,1)}\}, \{F_{(2,1)}, F_{(2,2)}, F_{(2,3)}, F_{(1,2)}\}, \{F_{(2,1)}, F_{(2,2)}, F_{(2,3)}, F_{(1,3)}\}\},$$

$$C_5 = \{\{F_{(2,1)}, F_{(2,2)}, F_{(1,1)}, F_{(1,2)}, F_{(1,3)}\}, \{F_{(2,1)}, F_{(2,3)}, F_{(1,1)}, F_{(1,2)}, F_{(1,3)}\},$$
$$\{F_{(2,2)}, F_{(2,3)}, F_{(1,1)}, F_{(1,2)}, F_{(1,3)}\}, \{F_{(2,1)}, F_{(2,2)}, F_{(2,3)}, F_{(1,1)}, F_{(1,2)}\},$$
$$\{F_{(2,1)}, F_{(2,2)}, F_{(2,3)}, F_{(1,1)}, F_{(1,3)}\}, \{F_{(2,1)}, F_{(2,2)}, F_{(2,3)}, F_{(1,2)}, F_{(1,3)}\}\}, \text{ and}$$

$$C_6 = \{\{F_{(2,1)}, F_{(2,2)}, F_{(2,3)}, F_{(1,1)}, F_{(1,2)}, F_{(1,3)}\}\}.$$

To show the sum of (2.5), which is the new formula of the simplification of the inclusion-exclusion principle, for $n = 2$ and $t_1 = t_2 = 3$, we use a simple artificial example that is also used later in the computational tests. We assume that each implementation j of i, $i \in \{1, 2\}$ and $j \in \{1, \ldots, 3\}$, corresponds to an elementary component a_{ij} which is independent of all other components and has reliability r_{ij}. Therefore, for $\mathbf{E} \in C_k$, $k \in \{2, \ldots, 6\}$, we find that

$P(\mathbf{E}) = \prod_{F_{(i,j)} \in E} r_{ij}$ and for R we have that

$$
\begin{aligned}
P(R) &= r_{21}r_{11} + r_{21}r_{12} + r_{21}r_{13} + r_{22}r_{11} + r_{22}r_{12} + r_{22}r_{13} + r_{23}r_{11} + r_{23}r_{12} + r_{23}r_{13} \\
&+ r_{21}r_{11}r_{12} + r_{21}r_{11}r_{13} + r_{21}r_{12}r_{13} + r_{22}r_{11}r_{12} + r_{22}r_{11}r_{13} + r_{22}r_{12}r_{13} + r_{23}r_{11}r_{12} + r_{23}r_{11}r_{13} \\
&+ r_{23}r_{12}r_{13} + r_{21}r_{22}r_{11} + r_{21}r_{22}r_{12} + r_{21}r_{22}r_{13} + r_{21}r_{23}r_{11} + r_{21}r_{23}r_{12} + r_{21}r_{23}r_{13} + r_{22}r_{23}r_{11} \\
&+ r_{22}r_{23}r_{12} + r_{22}r_{23}r_{13} + r_{21}r_{11}r_{12}r_{13} + r_{22}r_{11}r_{12}r_{13} + r_{23}r_{11}r_{12}r_{13} + r_{21}r_{22}r_{11}r_{12} + r_{21}r_{22}r_{11}r_{13} \\
&+ r_{21}r_{22}r_{12}r_{13} + r_{21}r_{23}r_{11}r_{12} + r_{21}r_{23}r_{11}r_{13} + r_{21}r_{23}r_{12}r_{13} + r_{22}r_{23}r_{11}r_{12} + r_{22}r_{23}r_{11}r_{13} \\
&+ r_{22}r_{23}r_{12}r_{13} + r_{21}r_{22}r_{23}r_{11} + r_{21}r_{22}r_{23}r_{12} + r_{21}r_{22}r_{23}r_{13} + r_{21}r_{22}r_{11}r_{12}r_{13} + r_{21}r_{23}r_{11}r_{12}r_{13} \\
&+ r_{22}r_{23}r_{11}r_{12}r_{13} + r_{21}r_{22}r_{23}r_{11}r_{12} + r_{21}r_{22}r_{23}r_{11}r_{13} + r_{21}r_{22}r_{23}r_{12}r_{13} + r_{21}r_{22}r_{23}r_{11}r_{12}r_{13}.
\end{aligned}
$$

2.3.1 Proof of Correctness of Algorithm 1

In this section, we prove in Lemma 4 that the sets C_k created by Algorithm 1 are equal to the sets C_k from Proposition 1.

Lemma 4. *Let $n \in \mathbb{N}^+$, $t = \{t_1, \ldots, t_n\} \in \mathbb{N}^{+n}$ and $m = \sum_{i=1}^{n} t_i$. Furthermore, let $\widehat{C}_k,\ k \in \{n, \ldots, m\}$, be the sets created with Algorithm 1 and for all $k \in \{n, \ldots, m\}$ let*

$$
\begin{aligned}
C_k = \{&\mathbf{E} = \{E_1, \ldots, E_k\} :\ E_u = F_{i(u)j(u)} \text{ for some } i(u) \in \{1, \ldots, n\},\ j(u) \in \{1, \ldots, t_{i(u)}\}, \\
&u \in \{1, \ldots, k\},\ \{i(l) \mid l \in \{1 \ldots, k\}\} = \{1, \ldots, n\} \text{ and} \\
&E_p \neq E_q \text{ for } p, q \in \{1, \ldots, k\} \text{ and } p \neq q\}.
\end{aligned}
$$

We then have that $\widehat{C}_k = C_k$ for all $k \in \{n, \ldots, m\}$.

Proof. Without loss of generality, let $k \in \{n, \ldots, m\}$ be fixed. First, we prove that $\widehat{C}_k \subseteq C_k$. We begin by showing that $lb_s \leq ub_s$ for all $s \in \{1, \ldots, n\}$ in the algorithm. We first choose $s = n$ and, hence, $l = k$. As seen before, $ub_n = \min\{k - n + 1, t_n\}$ and $lb_n = \max\{1, k - \sum_{b=1}^{n-1} t_b\}$. We know that $1 \leq t_n,\ 1 \leq k - n + 1$, and $k - \sum_{i=1}^{n-1} t_i \leq k - (n-1)$. Also,

$$
k \leq m = \sum_{i=1}^{n} t_i \Leftrightarrow k - \sum_{i=1}^{n-1} t_i \leq t_n.
$$

Hence, $lb_n \leq ub_n$. Without loss of generality, let $s \in \{1, \ldots, n-1\}$ and let $\hat{t}_n, \ldots, \hat{t}_{s+1}$ be the value chosen with $lb_i \leq \hat{t}_i \leq ub_i,\ i \in \{s+1, \ldots, n\}$. Therefore $l = k - \sum_{b=s+1}^{n} \hat{t}_b$. We know that $1 \leq t_s$ and $k - \sum_{b=s+1} \hat{t}_b - \sum_{b=1}^{s-1} t_b \leq k - \sum_{b=s+1}^{n} \hat{t}_b - (s-1)$. Furthermore, we know that

$$
\begin{aligned}
& k - \sum_{b=s+2}^{n} \hat{t}_b - \sum_{b=1}^{s} t_b \leq lb_{s+1} \leq \hat{t}_{s+1} \\
\Rightarrow & k - \sum_{b=s+2}^{n} \hat{t}_b - \hat{t}_{s+1} - \sum_{b=1}^{s-1} t_b - t_s \leq 0 \\
\Rightarrow & k - \sum_{b=s+1}^{n} \hat{t}_b - \sum_{b=1}^{s-1} t_b \leq t_s \text{ and} \\
& \hat{t}_{s+1} \leq ub_{s+1} \leq k - \sum_{b=s+2}^{n} \hat{t}_b - ((s+1) - 1) \\
\Rightarrow & 0 \leq k - \sum_{b=s+1}^{n} \hat{t}_b - (s-1) - 1 \\
\Rightarrow & 1 \leq k - \sum_{b=s+1}^{n} \hat{t}_b - (s-1).
\end{aligned}
$$

Hence, $lb_s \leq ub_s$. This gives us that $lb_i \leq ub_i$ for all $i \in \{1, \ldots, n\}$.
Let $\hat{E} \in \widehat{C}_k$, $\hat{H}_i = \hat{E} \cap H_i$ and $\hat{t}_i = |\hat{H}_i|$, $i \in \{1, \ldots, n\}$. We know that $1 \leq \hat{t}_i \leq t_i$, $i \in \{1, \ldots, n\}$. To prove that $\hat{E} \in C_k$, we just have to show $|\hat{E}| = k$. We have proven that $lb_s \leq ub_s$, $s \in \{1, \ldots, n\}$, and we know that $lb_s \leq \hat{t}_i \leq ub_s$. Hence,

$$lb_1 = \max\{1, k - \sum_{b=2}^{n} \hat{t}_b\} \leq \hat{t}_1 \leq ub_1 = \min\{k - \sum_{b=2}^{n} \hat{t}_b, t_1\}$$
$$\Rightarrow \hat{t}_1 = k - \sum_{b=2}^{n} \hat{t}_b$$
$$\Rightarrow k = \sum_{b=1}^{n} \hat{t}_b = |\hat{E}|.$$

This gives us that $\widehat{C}_k \subseteq C_k$. Now we prove that $C_k \subseteq \widehat{C}_k$.
Let $\mathbf{E} = \{E_1, \ldots, E_k\} \in C_k$. We know that

$$\forall l \in \{1, \ldots, k\}\ \exists \hat{i} \in \{1, \ldots, n\}\ \exists \hat{j} \in \{1, \ldots, t_{\hat{i}}\} : E_l = F_{\hat{i}\hat{j}}.$$

Furthermore, let $\hat{H}_i = \{F_{ij} \mid \exists l \in \{1, \ldots, k\}\ \exists i \in \{1, \ldots, n\}\ \exists j \in \{1, \ldots, t_i\} : E_l = F_{ij}\}$ and $\hat{t}_i = \left|\hat{H}_i\right|$ for all $i \in \{1, \ldots, n\}$. We know that $\sum_{i=1}^{n} \hat{t}_i = k$ and $\hat{t}_i \in \{1, \ldots, t_i\}$. In the recursion procedure of the algorithm with $l = k$, $s = n$, and $E = \{\}$, we have that
$ub_n = \min\{k - n + 1, t_n\}$ and $lb_n = \max\{1, k - \sum_{b=1}^{n-1} t_b\}$.
As $1 \leq \hat{t}_n$, $k - \sum_{i=1}^{n-1} t_i \leq k - \sum_{i=1}^{n-1} \hat{t}_i = \hat{t}_n$, $\hat{t}_n \leq t_n$ and $\hat{t}_n = k - \sum_{b=1}^{n-1} \hat{t}_i \leq k - (n-1)$, we have that $lb_n \leq \hat{t}_n \leq ub_n$ and, therefore, $\hat{t}_n$ can be chosen in the algorithm and $\hat{H}_n$ is a combination of elements in H_n of length $\hat{t}_n$.

Without loss of generality, let $s \in \{1, \ldots, n-1\}$. We then have that
$ub_s = \min\{k - \sum_{b=s+1}^{n} \hat{t}_n - (s-1), t_s\}$ and $lb_s = \max\{1, k - \sum_{b=s+1}^{n} \hat{t}_n - \sum_{b=1}^{s-1} t_n\}$.
We know that $1 \leq \hat{t}_s$, $k - \sum_{b=s+1}^{n} \hat{t}_n - \sum_{b=1}^{s-1} t_n \leq k - \sum_{b=s+1}^{n} \hat{t}_n - \sum_{b=1}^{s-1} \hat{t}_n = \hat{t}_s$, $\hat{t}_s \leq t_s$ and $\hat{t}_s = k - \sum_{b=s+1}^{n} \hat{t}_n - \sum_{b=1}^{s-1} \hat{t}_n \leq k - \sum_{b=s+1}^{n} \hat{t}_n - \sum_{b=1}^{s-1} 1 \leq k - \sum_{b=s+1}^{n} \hat{t}_b - (s-1)$ and, therefore, $lb_s \leq \hat{t}_s \leq ub_s$. Hence $\hat{t}_1, \ldots, \hat{t}_n$ can be chosen in the recursion of the algorithm and as $\hat{H}_i \subseteq H_i$ with cardinality $\hat{t}_i$ for all $i \in \{1, \ldots, n\}$, we know that $\mathbf{E} \in \widehat{C}_k$ and $C_k \subseteq \widehat{C}_k$.
This proves that $C_k = \widehat{C}_k$ for all $k \in \{n, \ldots, m\}$.

□

2.4 Computational Tests

To test the efficiency of our proposed reliability calculation method, we run computational tests on simple artificial system examples and a DMS application from aircraft architecture engineering. For our computational tests, we compare the number of summands and the computational time of the SDP method KDH88 from [19], with the classic probability principle of inclusion-exclusion (2.3) and our proposed method. We are using KDH88 by applying it on (2.2).KDH88 algorithm is described in Appendix A.2. All methods are implemented in Python 2.7 and run on an Intel Core i7-6600U with CPU 2.60GHz and 32GB memory.

2.4.1 Simple Artificial System

First we compare the methods for some examples of a simple artificial system. We set all $t_i = t$, and assume that each implementation j of i, $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, t\}$, corresponds to an elementary component a_{ij} which has a behavior independent from all other components and has the common elementary reliability r. We then have a system of $t \times n$ components. We run computational tests for $n \in \{2, \ldots, 6\}$, $t \in \{2, \ldots, 4\}$ and different values of r. Because all components are independent, we know that the reliability R can be calculated with $R(t, n, r) = ((1-(1-r)^t)^n)$.

		Proposed		KDH88		Classic I-E	
n	t	Summands	Time (s)	Summands	Time (s)	Summands	Time (s)
2	2	9	0.00018	4	0.00023	15	0.00034
2	3	49	0.00071	9	0.00121	511	0.00440
2	4	225	0.00266	16	0.00269	65535	0.54116
3	2	27	0.00052	8	0.00075	255	0.00242
3	3	343	0.00672	27	0.00933	1.34e+8	> 300
3	4	3375	0.03884	64	0.10125	1.84e+19	> 300
4	2	81	0.00120	16	0.00199	65535	0.55845
4	3	2401	0.02959	81	0.17708	2.41e+24	> 300
4	4	50625	0.64799	256	7.66231	1.15e+77	> 300
5	2	243	0.00358	32	0.01169	4.29e+9	> 300
5	3	16807	0.23287	243	5.48717	1.41e+73	> 300
5	4	759375	11.28866	1024	> 300	1.79e+308	> 300
6	2	729	0.01087	64	0.07497	1.84e+19	> 300
6	3	117649	1.65340	729	174.17662	2.82e+219	> 300

Table 2.3: Computational results for examples of simple artificial systems.

To check the validity of the methods and their implementation, we compared the calculated reliability against $R(t, n, r)$. Table 2.3 shows the number of summands and computational time for all three methods.

Table 2.3 shows that the number of summands for our method is far greater compared to KDH88. However our method is computationally more efficient. This is because it is much more computationally expensive to calculate the few disjoint products/summands with KDH88 than all summands that are needed in our method. Furthermore the summands that are created by KDH88 are not simple products like the summands in our method. However, both methods are far more efficient and have fewer summands than the classic probability principle of inclusion-exclusion.

2.4.2 Door Management System

Our next computational test is based on the DMS. As mentioned earlier, the DMS is a safety-critical system which checks the status of doors, regulates the locks and relays information to on-board computers and pressurization regulators.

The functionality of the DMS for each door can be seen as a function of the systems. Hence, suppose the aircraft has n doors and consider the event set $\mathcal{F} = \{F_1, \ldots, F_n\}$ where F_i is the event "The functionality of the DMS for door i in the system does not fail". Let an implementation of the functionality of a door be a set of components and the corresponding connections which can check the status of a door, regulate the locks and relay information to on-board computers and pressurization regulators. Moreover, suppose that no subset of these components and connections can be removed without losing functionality. Because the DMS is a safety-critical system that has to be redundant, there are at least two implementations for every door. Let $\mathcal{F}_i = \{F_{i1}, \ldots, F_{it_i}\}$ and $t_i \geq 2$, $i \in \{1, \ldots, n\}$, be event sets where F_{ij} is the event "The functionality of implementation j of door i does not fail". Lastly, let R be the event "The DMS system fails for no door". With these event sets, we can calculate the reliability $P(R)$ of the DMS with the formula from

Proposition 1:

$$P(R) = \sum_{k=n}^{m} \left((-1)^{k-n} \sum_{\mathbf{E} \in C_k} P\left(\bigcap_{j=1}^{k} E_j \right) \right). \tag{2.31}$$

The calculation of $P(\bigcap_{j=1}^{k} E_j)$ for $\mathbf{E} = \{E_1, \ldots, E_k\} \in C_k,\ k \in \{n, n+1, \ldots, m\}$ is simple. Let T_c be the event that component c of the DMS system does not fail. Since we are considering a static system, we know the probability $P(T_c) = a_c$ with $a_c \in (0,1)$. Let $\mathcal{T}_{F_{ij}}$ be the set of components of implementation j for door i. Since we assume that all components have independent failures for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, t_i\}$, it follows that

$$P\left(F_{ij}\right) = P\left(\bigcap_{c \in \mathcal{T}_{F_{ij}}} T_c \right) = \prod_{c \in \mathcal{T}_{F_{ij}}} P\left(T_c\right).$$

Therefore, for all $k \in \{n, n+1, \ldots, m\}$ and for all $\mathbf{E} \in C_k$, we have that

$$P\left(\bigcap_{j=1}^{k} E_j \right) = P\left(\bigcap_{j=1}^{k} \left(\bigcap_{c \in \mathcal{T}_{E_j}} T_c \right) \right) = P\left(\bigcap_{c \in \bigcup_{j=1}^{k} \mathcal{T}_{E_j}} T_c \right) = \prod_{c \in \bigcup_{j=1}^{k} \mathcal{T}_{E_j}} P\left(T_c\right).$$

If z is the number of components of the system, we know that each summand of (2.31) is a product of at most z factors.

We ran computational tests for different numbers of doors n, implementations t and number of components/connections. We tested them for our proposed method and the KDH88 method, but we did not run them for the classic probability principle of inclusion-exclusion (2.3) because of its much worse performance already discussed. For every combination of n and t, we ran 100 different systems. All systems have a different numbers of components and different reliability values. An example system can be found in Appendix A.3 as well as a link to the data of all other DMS systems. Table 2.4 shows the minimum, maximum and median number of components/connections, the minimum, maximum and median number of summands and minimum, maximum and median execution times for both methods and every combination of n and t.

				Proposed				KDH88					
(n,t)	Components			Summands	Time (s)			Summands			Time (s)		
	Min	Max	Med		Min	Max	Med	Min	Max	Med	Min	Max	Med
$(2,2)$	42	65	56.0	9	0.00017	0.00037	0.00020	4	5	5.0	0.00028	0.00073	0.00036
$(2,3)$	57	81	72.0	49	0.00087	0.00128	0.00103	13	20	17.0	0.00252	0.00421	0.00332
$(3,2)$	57	76	69.0	27	0.00056	0.00070	0.00061	7	13	11.0	0.00102	0.00215	0.00174
$(3,3)$	68	96	84.0	343	0.00662	0.00857	0.00760	34	87	59.0	0.02927	0.09249	0.06568
$(4,2)$	76	101	90.0	81	0.00187	0.00231	0.00208	10	28	21.0	0.00744	0.01381	0.01074
$(4,3)$	83	119	102.5	2401	0.05573	0.07409	0.06377	81	339	187.0	0.42229	3.23434	1.88462
$(5,2)$	88	128	111.0	243	0.00599	0.00807	0.00704	19	67	45.0	0.02448	0.11609	0.08271
$(5,3)$	96	146	125.0	16807	0.45604	0.60943	0.54729	110	1067	619.0	12.21276	95.74859	59.34907

Table 2.4: Computational results for the DMS.

We obtained similar results as in the other computational tests. With our proposed method there is a constant number of summands for the 100 different systems, but this value is much larger than the number of summands when using KDH88. Also, the minimum and maximum number of summands for the different systems when using KDH88 is very different. This shows that the efficiency of KDH88 is highly dependent on the exact structure of the system. Furthermore, we can see that our method is computationally much more efficient, especially for larger systems.

2.5 Discussion

The new expression (Proposition 1) reduces considerably the computational effort needed to calculate the reliability. It has been shown that the formula obtained can be applied to the reliability calculation of a certain kind of complex network systems and it decreases the computational time significantly. It has also been shown that the time complexity is reduced from doubly exponential to exponential with linear exponent. Moreover, we have provided a comparison with the general SDP method KDH88 from [19] and showed that our method is computationally more efficient than KDH88 for our examples.

The complex network systems we consider are systems with multiple functions, which means they have multiple start and end nodes in the system. If a system with only one function is considered, the simplification does not take effect and the number of terms of the probability principle of inclusion-exclusion does not decrease. We also considered that we have multiple implementations for each function. For a system with only one implementation per function, our proposed calculation method does not have any advantages. In all the other cases, that is, multiple functions and multiple implementations per function, the expression proposed in this paper can be applied very effectively.

Another limitation of our method is also one that KDH88 and other SDP methods have. The different paths in the system that represent the implementations for each function and the failure probability of all components in the system have to be known.

And as it was our main goal, it must be noted that the result introduced does not only give the option to calculate reliability more efficiently. It also allows to formulate optimization problems of complex network systems to include the exact reliability of the system without depending solely on heuristics to solve it.

Chapter 3

Redundancy Model and Methods

In this chapter we address the issue of how to consider k-redundancy in an MILP formulation for the DMS and how to solve it most efficiently. This chapter is structured as follows.

We start by showing in what kind of MILP problem category the DMS problem with redundancy falls and present a short overview of previous research in that field. We then introduce a new MILP formulation of the DMS problem with redundancy in Section 3.2. In Section 3.3 we propose a solving method which is based on branch-and-bound. It includes a new branching rule and heuristic that are tailored to the new formulation of the problem. We explore another approach in Section 3.4 which is based on branch-and-price. The branch-and-price approach is not efficient, but inspired the main idea for a reliability heuristic which is the main result of Chapter 4. A short introduction to the general branch-and-bound and branch-and-price algorithm can be found in Appendix B.1.

In the last section of this chapter, Section 3.5, we discuss authenticity of our problem instances and the practicability of the new method for the industry.

3.1 Motivation and Previous Research

In this section, we explain the DMS in detail and how it can be formulated as an MILP. Furthermore, we present an overview of previous research and we show why our results are a new contribution to the literature.

The functions of a DMS are to check the status of doors and locks and send their status information to on-board computers and to pressurization regulators in the aircraft. For our MILP formulation, we consider the gathering of the doors status and sending the status information to on-board computers as one function and sending the information to the pressurization regulators as a second function. In a DMS, the two functions are implemented in the system through sets of units and cable connections between them. Like every safety-critical system, a DMS has to adhere to safety and architectural constraints and regulations. One of the most important regulations is that the system has to be k-redundant. A system is k-redundant if for every function of the system it has k disjoint subsystems that can fulfil this function.

Therefore, a DMS is a system with multiple doors, two functions for each door and both functions have to be k-redundant for each door. Our goal is to build an MILP formulation for such a system. To achieve this goal, we first have to consider one main architectural restriction. This restriction is the space restriction of an aircraft. Only specific locations in an aircraft can be used to install units for the DMS and also only certain cable connections are available. For our formulation, we represent all possible locations and all possible cable connections as a directed graph $G = (N, A)$. Figure 3.1 shows a set of possible locations for units and its corresponding graph which includes possible connections. Not all possible connections are shown in the figure.

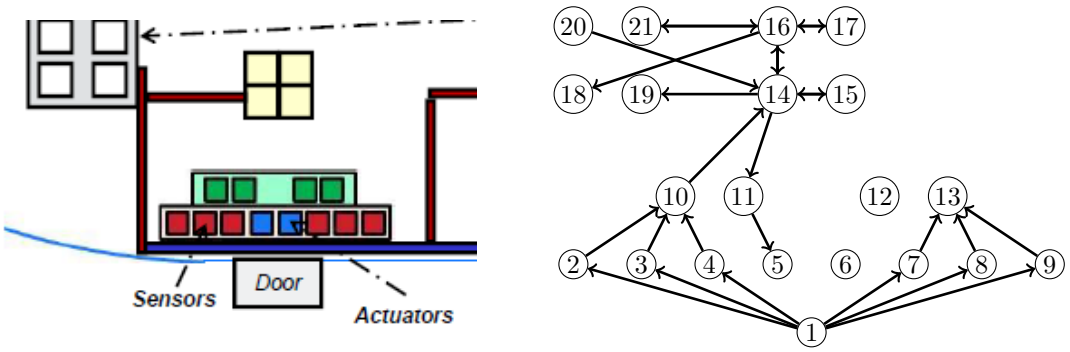

Figure 3.1: View of a set of locations and connections and their corresponding graph.

Therefore, any DMS design would be a subgraph G' of G. We will refer to the complete DMS as the *overall system* in the rest of this chapter. To formulate our problem we cannot only consider the overall system. We must look at every function which must be k-redundant in the overall system. As mentioned before, functions are sets of units and cable connections. Hence, we can also see every function as a subgraph of G'. In the following, we refer to these subgraphs as subsystems.

Hence, we have k-redundancy for the functions when we have k node/arc-disjoint paths for each function. In our formulation, we consider node- and arc-disjoint paths. k-node/arc-disjoint path problems have been widely researched ([25],[23],[52],[13],[16],[18]) and are applied in many different fields, as for example routing ([32],[47]) and social networks ([17],[46]).

The problem studied in this chapter is not a mere case of the previous researched problems. It is a problem with multiple subsystems that have to have k node/arc-disjoint paths and all subsystems are part of the overall system and do not have to be pairwise disjoint. Also, only the start and end unit type of a function is fixed and not the location of the start and end unit.

In the following, we will introduce an MILP formulation of the DMS problem with redundancy.

3.2 MILP Formulation of the DMS Problem with Redundancy

We propose an MILP formulation for the DMS problem with redundancy which is the first in literature. For the formulation, we first consider the limited space in an aircraft. As mentioned before, only certain locations are available for installing units and there is also a limited number of possible cable connections in an aircraft. These locations and possible connections can be seen as a directed graph $G = (N, A)$. To write the formulation concisely, we need the following notation. We first define the sets.

3.2.1 Sets

We need to know for the formulation which unit types are available and at which locations they can be placed. Also, we have to consider that a type of unit may come in different models. For example, we have two different models for a switch, where one model has 24 ports and the other one only 16 ports. This information is contained in the following sets:

- U is the set of different unit types.
- $U^i,\ i \in N$, is the set of different unit types that can be put at locations i.
- $N^u,\ u \in U$, is the set of different locations where a unit of type u can be set.
- $M^u,\ u \in U$, is the set of different models available for unit type u.

In Figure 3.2 we can see an example of these sets.

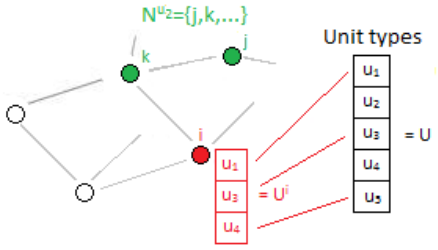

Figure 3.2: Example for sets U, U^i, and N^u.

We define subsets of U, N, and A for each function, since each function has their own architectural and safety restrictions. For the DMS problem, let D be the set of doors, let F be the set of functions, and let $P = \{1, \ldots, k\}$ be the set of disjoint paths needed. We then define the following sets:

- $U_f \subset U,\ f \in F$, is the set of unit types that can be used for function f.
- $N_f = \{i \in N \mid U_f \cap U^i \neq \emptyset\},\ f \in F$, is the set of locations that can be used for function f.
- $N_f^u = \{i \in N_f \mid u \in U^i\},\ f \in F, u \in U_f$, is the set of locations that can be used for function f and unit u.
- $U_f^i = \{u \in U_f \mid u \in U^i\},\ f \in F,\ i \in N_f$, is the set of unit types that can be used at location i for function f.
- $A_f = \{(i,j) \in A \mid i,j \in N_f\},\ f \in F$, is the set of connections (arcs) that can be used for function f.
- $F^u = \{f \in F \mid u \in U_f\},\ u \in U$, is the set of functions that can use a unit of type u.

In the following, if the subscript $f \in F$ or superscript $u \in U, i \in N$ is missing, it means it is the union over the missing index. For example $U_f = \bigcup_{i \in N} U_f^i$.

There are architectural constraints that restrict which units can be connected or even which units must have a connection from another unit for a function $f \in F$. The following sets contain this information:

- $C_f(u),\ f \in F,\ u \in U_f$, is the set of unit types to which a unit of type u can connect to for function f.
- $C_f^+(u),\ f \in F,\ u \in U_f$, is the set of unit types to which a unit of type u must connect to for function f.
- $C_f^-(u),\ f \in F,\ u \in U_f$, is the set of unit types from which a unit of type u must have a connection from for function f.
- $W_f^i(u) = U_f^i \cap C_f(u)$, is the set of unit types that can be set at location i and can have a connection from a unit of type u for function f.

Additional sets that contain information of the architectural restrictions for the functions are:

- $U_f^b,\ f \in F$, is the set of unit types that are not a start or end unit type for function f.
- $U^{s,e} = \{\ (s_f, e_f) \mid s_f,\ e_f \in U_f$ are a start unit and an end unit for some $f \in F\}$, is the set of pairs of start and end units for functions.
- $V_f^+(i) := \{k \in N_f : (i,k) \in A_f\},\ i \in N_f$, is the set of locations that location i can connect to for function $f \in F$.

- $V_f^-(i) := \{k \in N_f : (k,i) \in A_f\},\ i \in N_f$, is the set of locations that location i can connect from for function $f \in F$.

- $V_f^+(i,u) = V_f^+(i) \cap \{k \in N \mid W_f^k(u) \neq \emptyset\}$ is the set of locations that location i can connect to and where units can be placed to which unit u can be connected to.

- $V_f^-(j,\hat{u}) = V_f^-(j) \cap \{k \in N \mid \{u \in U_f^k \mid \hat{u} \in C_f(u)\} \neq \emptyset\}$ is the set of locations that location j can have a connection from and where units can be placed from which unit $\hat{u}$ can be connected from.

3.2.2 Decision variables

We will now define the decision variables of the MILP. First, we need variables that represent the overall system. These are assignment variables that represent which unit type and model is installed at which location and variables that represent which cable connections are used.

- $t_{ium} \in \{0,1\},\ i \in N,\ u \in U^i,\ m \in M^u$, is a binary variable that takes value 1 if unit type u and model m is set at location i and 0 otherwise.

- $x_{iju\hat{u}} \in \{0,1\},\ (i,j) \in A, u \in U^i,\ \hat{u} \in W^j(u)$, is a binary variable that takes value 1 if locations i and j are connected and units u and $\hat{u}$ are installed at i and j, respectively. It is 0 otherwise.

We need the same kind of variables for every path in a subsystem. For each path p of the subsystem for function f and door d, let $(d,f,p) \in D \times F \times P$ refer to it. The required decision variables for every $(d,f,p) \in D \times F \times P$ are:

- $t_{iu}^{dfp} \in \{0,1\},\ i \in N_f,\ u \in U_f^i$, a binary variable that takes value 1 if a unit of type u is used at location i, and

- $x_{iju\hat{u}}^{dfp} \in \{0,1\},\ (i,j) \in A_f,\ u \in U_f^i,\ \hat{u} \in W_f^j(u)$, a binary variable that takes value 1 if locations i and j are connected and unit u and $\hat{u}$ are used at i and j, respectively. It is 0 otherwise.

3.2.3 Constraints

In this subsection we describe the constraints of the MILP formulation. With the overall system and the subsystems, we have two levels of assignment variables which have to be synchronized. For example, if a unit or cable connection is installed in the overall system, there has to be a subsystem which uses it. And also if a unit or cable connection is used for a subsystem, it has to be installed in the overall system. Constraints (3.1) to (3.4) force this synchronization. Furthermore, constraints (3.3) and (3.4) impose that the subsystems have k node/arc-disjoint paths.

In detail, constraint (3.1) forces at least one subsystem to use a unit u at location i if it is located there in the overall system.

$\forall i \in N,\ \forall u \in U^i$:

$$\sum_{m \in M^u} t_{ium} \leq \sum_{\substack{d \in D\ p \in P, \\ f \in F(u)}} t_{iu}^{dfp}, \tag{3.1}$$

Constraint (3.2) forces at least one subsystem to use a connection (i, j) if it is used in the overall system.

$\forall (i,j) \in A,\ \forall u \in U^i,\ \forall \hat{u} \in W^j(u)$:

$$x_{iju\hat{u}} \leq \sum_{\substack{d\in D,\ p\in P,\\ f\in F\ \mid\ (i,j)\in A_f,\\ u\in U_f^i,\ \hat{u}\in W_f^j(u)}} x_{iju\hat{u}}^{dfp}, \tag{3.2}$$

Constraint (3.3) forces a unit u be installed at location i in the overall system if it was used in at least one subsystem. Moreover, it imposes that the subsystems have k node-disjoint paths since t_{ium} are binary variables.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall i \in N_f,\ \forall u \in U_f^i\ ,\ \forall \hat{f} \in F^u$:

$$\sum_{m\in M^u} t_{ium} \geq t_{iu}^{dfp} + \sum_{\hat{p}\in P\setminus\{p\}} t_{iu}^{d\hat{f}\hat{p}}, \tag{3.3}$$

Constraint (3.4) forces connection (i, j) be installed in the overall system if it was used in at least one subsystem. It also imposes that the subsystems have k arc-disjoint paths, because $x_{iju\hat{u}}$ are binary variables.

$\forall d \in D,\ \forall f \in F,\ \forall (i,j) \in A_f,\ \forall u \in U_f^i,\ \forall \hat{u} \in W_f^j(u)$:

$$x_{iju\hat{u}} \geq \sum_{p\in P} x_{iju\hat{u}}^{dfp}. \tag{3.4}$$

Constraints (3.1) to (3.4) provide also the option to lift the integer requirement on t_{ium} and $x_{iju\hat{u}}$ variables. Computational tests showed that lifting the integer requirement did not improve solution time.

In the following, we present flow constraints which create the needed paths of the subsystems. First, constraint (3.5) imposes that no more than the maximum number of connections E_{fu}^{out} can go out of unit u at location i for a function $f \in F$. It also synchronizes the connection and location variables of the subsystems.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ i \in N_f,\ \forall u \in U_f^i$:

$$\sum_{\substack{j\in V_f^+(i,u),\\ \hat{u}\in W_f^j(u)}} x_{iju\hat{u}}^{dfp} \leq E_{fu}^{out} t_{iu}^{dfp}, \tag{3.5}$$

For connections arriving at a location, we use constraint (3.6).

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ j \in N_f,\ \forall \hat{u} \in U_f^j$:

$$\sum_{\substack{i\in V_f^-(j,\hat{u}),\\ u\in U_f\ \mid\ \hat{u}\in C_f(u)}} x_{iju\hat{u}}^{dfp} \leq E_{fu}^{in} t_{j\hat{u}}^{dfp}. \tag{3.6}$$

The following constraints impose that a flow arrives and leaves if a unit u at a location i is used for a function $f \in F$. Constraint (3.7) imposes that a flow arrives at a unit. If a unit u at location i is used for a function f, at least $T_u^{f,in}$ connections have to arrive at location i for f. Constraint (3.8) imposes that the flow continues as long as it does not arrive at the end unit. This

means that if a unit u at location i is used for a subsystem, at least $T_u^{f,out}$ connections have to leave location i for a function $f \in F$. We do not need subtour and graph elimination constraints, because of the architecural restriction of the network system.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F, \forall i \in N_f\ :$

$$\sum_{u \in (U_f^{\mathrm{b}} \cup \{e_f\}) \cap U_f^i} T_u^{f,\mathrm{in}} t_{iu}^{dfp} \leq \sum_{\substack{\ell \in V_f^-(i),\ u \in (U_f^{\mathrm{b}} \cup \{e_f\}) \cap U_f^i, \\ \hat{u} \in U_f\ \mid\ u \in C_f(\hat{u})}} x_{\ell i \hat{u} u}^{dfp}, \tag{3.7}$$

$$\sum_{u \in (U_f^{\mathrm{b}} \cup \{s_f\}) \cap U_f^i} T_u^{f,\mathrm{out}} t_{iu}^{dfp} \leq \sum_{\substack{\ell \in V_f^+(i),\ u \in (U_f^{\mathrm{b}} \cup \{s_f\}) \cap U_f^i, \\ \hat{u} \in W_f^{\ell}(u)}} x_{i \ell u \hat{u}}^{dfp}. \tag{3.8}$$

The last flow constraint is (3.9). It imposes that a start unit and an end unit are installed for all k paths of each function $f \in F$.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall u_f \in \{s_f, e_f\}\ :$

$$\sum_{i \in N_f^{u_f}} t_{iu_f}^{dfp} = 1. \tag{3.9}$$

We have now presented all general flow and synchronization constraints that are needed. In the following, we present problem specific architectural constraints for the different functions.

The first function specific architectural constraints are constraints (3.10) and (3.11). Constraint (3.10) imposes that if a unit u is installed at location i for function $f \in F$, then it connects to all unit types which it has to have a connection to for function $f \in F$. Set $C_f^+(u)$ contains the unit types to which unit u has to have a connection to.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall i \in N_f,\ \forall u \in U_f^i,\ \forall \hat{u} \in C_f^+(u)\ :$

$$t_{iu}^{dfp} \leq \sum_{j \in V_f^+(i) \cap N_f^{\hat{u}}} x_{iju\hat{u}}^{dfp}. \tag{3.10}$$

Constraint (3.11) imposes that if a unit u is installed at location i for a function $f \in F$, then it has a connection from all unit types it has to have a connection from. Set $C_f^-(u)$ contain the unit types which unit u has to have a connection from.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall j \in N_f,\ \forall u \in U_f^j,\ \forall \hat{u} \in C_f^-(u)\ :$

$$t_{ju}^{dfp} \leq \sum_{i \in V_f^-(j) \cap N_f^{\hat{u}}} x_{ij\hat{u}u}^{dfp}. \tag{3.11}$$

The last architectural constraint for subsystems is (3.12). In some cases, specific functions have to be connected. For example, the end unit of a function f_1 has to be also the start unit of another function f_2. Therefore, we need for some $f_1, f_2 \in F$ that

$\forall i \in N^{e_{f_1}}, \forall d \in D,\ \forall p \in P$ with $e_{f_1} = s_{f_2}$:

$$t_{ie_{f_1}}^{df_1p} = t_{is_{f_2}}^{df_2p}. \tag{3.12}$$

For the DMS problem, we need that the two functions are connected. As the information

of the sensors are saved in a specific CPIOM/computer, the data flow of that information to a pressurization valve can also only leave this specific CPIOM/computer. Therefore, constraint (3.12) is needed for the formulation.

Other than for the subsystem, we also need architectural constraints for the overall system. The first architectural constraint is constraint (3.13), which imposes that at most one unit is installed at any location of the system.

$\forall i \in N$:

$$\sum_{\substack{u \in U^i, \\ m \in M^u}} t_{ium} \leq 1. \tag{3.13}$$

Constraint (3.14) imposes that only as many connections arrive and leave as ports are available on a unit in the overall system.

$\forall i \in N,\ \forall u \in U^i$:

$$\sum_{\substack{\ell \in V^-(i,u), \\ \hat{u} \in U^\ell \ \mid \ u \in C(\hat{u})}} x_{\ell i \hat{u} u} + \sum_{\substack{\ell \in V^+(i,u), \\ \hat{u} \in W^k(u)}} x_{i \ell u \hat{u}} \leq \sum_{m \in M^u} t_{ium} E_{um}. \tag{3.14}$$

Until now, we provided an MILP formulation for a general system with multiple sets of functions and k-redundancy for these functions. The following constraint is specific for the DMS problem. It is a safety regulation constraint. In the DMS we always have two unit types named outflow valve (OVF) and outflow valve control unit (OCU). The requirement is that every OVF must have at least a connection from two different OCUs in the overall system. This is imposed through constraint (3.15):

$\forall i \in N^{OVF}$:

$$2 \sum_{m \in M^{OVF}} x_{i(OVF)m} \leq \sum_{j \in N^{OCU} \cap V^-(i)} x_{ji(OCU)(OVF)}. \tag{3.15}$$

3.2.4 Tightening Constraints

Constraints (3.1) to (3.15) represent a valid formulation for the DMS problem with k-redundancy. Furthermore, we could reduce the number of decision variables and synchronization constraints by aggregating the decision variables $x_{iju\hat{u}}$ to x_{ij} and the corresponding constraints and still have a valid formulation. However we encountered large gaps between the LP relaxation and the MILP solution during computational tests with the current constraints and the aggregation of decision variables. This lead to long computational times and poor convergence in the branch-and-bound algorithm. Therefore, we dis-aggregate the connection variable of the overall system and also propose the following constraints (3.16)-(3.21) to decrease the gap between the LP relaxation and the MILP solution.

Constraints (3.16) and (3.17) strengthen the flow of the different paths in the subsystems through sets $C_f^+(u)$ and $C_f^-(u)$ and the parameters E_{fu}^{in} and E_{fu}^{out}. $C_f^+(u)$ and $C_f^-(u)$ are the sets that contain information on which unit must connect to which unit and which unit must connect from which other unit,respectively. Furthermore, E_{fu}^{in} and E_{fu}^{out} are the parameters that contain the information what the maximum number of connections arriving or leaving a unit u is for function $f \in F$.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall u \in U_f$:

$$\sum_{\substack{\hat{u} \in \{\overline{u} \ \mid \ u \in C_f^+(\overline{u})\}, \\ i \in N_f^{\hat{u}}}} t_{i\hat{u}}^{dfp} \leq E_{fu}^{in} \sum_{i \in N_f^u} t_{iu}^{dfp} \tag{3.16}$$

$$\sum_{\substack{\hat{u} \in \{\overline{u} \ \mid \ u \in C_f^-(\overline{u})\}, \\ i \in N_f^{\hat{u}}}} t_{i\hat{u}}^{dfp} \leq E_{fu}^{out} \sum_{i \in N_f^u} t_{iu}^{dfp} \tag{3.17}$$

Constraints (3.18) to (3.21) synchronize decision variable sets t_{ium} and $x_{iju\hat{u}}$.

$\forall (i,j) \in A,\ \forall u \in U^i$:

$$\sum_{\hat{u} \in W^j(u)} x_{iju\hat{u}} \leq \sum_{m \in M^u} t_{ium}, \tag{3.18}$$

$\forall (i,j) \in A,\ \forall \hat{u} \in U^j$:

$$\sum_{u \in U^i \ \mid \ \hat{u} \in C(u)} x_{iju\hat{u}} \leq \sum_{m \in M^u} t_{j\hat{u}m}. \tag{3.19}$$

$\forall i \in N\ \ \forall u \in (U^{\mathrm{b}} \cup \bigcup_{f \in F} \{e_f\}) \cap U^i$:

$$\sum_{m \in M^u} t_{ium} \leq \sum_{\substack{\ell \in V^-(i,u), \\ \hat{u} \in U^\ell \ \mid \ u \in C(\hat{u})}} x_{\ell i \hat{u} u} \tag{3.20}$$

$\forall i \in N\ \ \forall u \in (U^{\mathrm{b}} \cup \bigcup_{f \in F} \{s_f\}) \cap U^i$:

$$\sum_{m \in M^u} t_{ium} \leq \sum_{\substack{\ell \in V^+(i,u), \\ \hat{u} \in W^j(u)}} x_{i\ell u\hat{u}}. \tag{3.21}$$

The utility of constraints (3.16)-(3.21) was tested and the results can be seen in Table 3.2 in Section 3.3.3. It can be clearly seen that the additional constraints and decision variables reduce the gap significantly and help to solve the problem faster.

3.2.5 Objective Functions

We introduce here several objective functions which are used for computational tests. The following parameters are used:

- c_{um}: cost of a unit of type $u \in U$ and model $m \in M^U$,
- c_0: cost of cable for a meter,
- w_{um}: weight of a unit of type $u \in U$ and model $m \in M^U$,
- w_0: weight of cable for a meter,
- pow_{um}: power usage of a unit of type $u \in U$ and model $m \in M^U$,
- ℓ_{ij}: distance between locations i and j in meters,
- $\gamma_c,\ \gamma_w,\ \gamma_{pow}$: weight for the respective parameter in a combination of different objectives.

The different objective functions that we consider for minimization are the following:

1. Total of weight and cost (obj1):

$$\min \left[\sum_{i \in N} \sum_{\substack{u \in U^i, \\ m \in M^u}} t_{ium} \left(c_{um} + w_{um} \right) \right] + \sum_{(i,j) \in A} x_{ij} \ell_{ij} \left(c_0 + w_0 \right), \tag{3.22}$$

2. Total number of units placed (obj2):

$$\min \left[\sum_{i \in N} \sum_{\substack{u \in U^i, \\ m \in M^u}} t_{ium} \right], \tag{3.23}$$

3. Total length of cable used (obj3):

$$\min \sum_{(i,j) \in A} x_{ij} l_{ij}, \tag{3.24}$$

4. Weighted total of weight, cost, and power (obj4):

$$\begin{aligned} \min & \left[\sum_{i \in N} \sum_{\substack{u \in U^i, \\ m \in M^u}} t_{ium} \left(\gamma_c c_{um} + \gamma_w w_{um} + \gamma_{pow} pow_{um} \right) \right] \\ & + \sum_{(i,j) \in A} x_{ij} \ell_{ij} \left(\gamma_c c_0 + \gamma_w w_0 \right). \end{aligned} \tag{3.25}$$

3.3 Branch-and-bound Approach

In this section, we use the MILP formulation of the previous Section 3.2 and build a solving method using branch-and-bound algorithm (see Subsection B.1.1). During the first computational tests, we noticed that the standard branching rules implemented in CPLEX (e.g. pseudo cost or strong branching) do not work well and, hence, we developed a branching rule that is tailored to our formulation. Also, we found that the CPLEX default heuristics do not work efficiently for the problem. This is a problem that is often seen in location problems as for example in [24]. To solve our problem efficiently, we developed a simple heuristic which is also tailored to our formulation.

In the following subsections, we present our new branching rule and heuristic and, furthermore, show computational results that compare our branch-and-bound method that is tailored to the problem and the general branch-and-bound method that CPLEX uses.

3.3.1 Branching Rule

The following new branching rule allows us to solve the DMS problem with redundancy and reliability much more efficiently. In the literature, see [3], branching is usually done on integer decision variables which have a fractional value for the current linear relaxation. Computational tests have shown that the general branching strategies are not efficient. These strategies would branch on decision variables that represent specific locations or connections for the overall system and subsystems. However these would not affect the gap between linear relaxation and MILP solution immediately. This is the case since we have symmetry for different paths of a function in a subsystem or a unit can be set at other location in the overall system without affecting the objective value.

The two main points of our branching rules are that:

- Whenever possible, we branch on the decision variables sets $t_u,\ t_{um}$ and t_u^f based on the following conditions, and
- If no suitable decision variable is found among $t_u,\ t_{um}$ and t_u^f, general branching rules of CPLEX are used.

3.3.1.1 Sets, decision variables and constraints

These are the sets, decision variables and constraints needed for the new branching rule. First, the sets are

- $\mathcal{U}_f^+ = \left\{u \in U_f \ :\ C_f(u) \cap \left(\bigcup_{f' \in F\backslash\{f\}} C_{f'}(u)\right) = \emptyset\right\}$, $f \in F$, is the set of units which cannot connect to the same units for other functions $\hat{f} \in F\backslash\{f\}$ as for $f \in F$. It is a subset of U_f.
- $\mathcal{U}_f^- = \left\{u \in U_f \ :\ \left\{\overline{u} \ :\ u \in C_f(\overline{u}) \text{ and } u \in \bigcup_{f' \in F\backslash\{f\}} C_{f'}(\overline{u})\right\} = \emptyset\right\},\ f \in F$, is the set of units which cannot connect from the same units for other functions $\hat{f} \in F\backslash\{f\}$ as for $f \in F$. It is a subset of U_f.
- $E_f^+(u) = \left\{\hat{u} \in U_f \mid u \in C_f^+(\hat{u})\right\},\ f \in F,\ u \in \mathcal{U}_f^+$, is the set of units that must connect to unit u for function f.
- $E_f^-(u) = \left\{\hat{u} \in U_f \mid u \in C_f^-(\hat{u})\right\},\ f \in F,\ u \in \mathcal{U}_f^-$, is the set of units that must connect from unit u for function f.

Note that $E_f^+(u) = \emptyset$ for $f \in F,\ u \in U_f\backslash\mathcal{U}_f^+$ and $E_f^-(u) = \emptyset$ for $f \in F,\ u \in U_f\backslash\mathcal{U}_f^-$. The decision variables for the new branching and pruning rule are:

- $t_u \in \{0, 1, \ldots, |N^u|\},\ u \in U^i$, which is the total number of units of type u used,
- $t_{um} \in \{0, 1, \ldots, |N^u|\},\ u \in U^i,\ m \in M^u$, which is the total number of units of type u and model m used,
- $t_u^f \in \{0, 1, \ldots, |N^u|\},\ f \in F,\ u \in U_f^i$, which is the total number of units of type u used for function f,
- $t_{iu}^f \in \{0, 1\},\ f \in F,\ u \in U_f^i$, which takes value 1 if a unit of type u is used for function f at location i, and 0 otherwise.

Lastly, the needed constraints are synchronization constraints between the new branching variables and the previous variables that represent the network system. The following constraints (3.26)-(3.29) impose the synchronization between decision variables t_{ium} and t_{iu}^{dfp} and the branching variables $t_u,\ t_{um},\ t_u^f$, and t_{iu}^f.

$\forall u \in U,\ \forall m \in M^u$:

$$\sum_{i \in N^u} t_{ium} = t_{um}, \tag{3.26}$$

$\forall u \in U$:

$$\sum_{m \in M^u} t_{um} = t_u, \tag{3.27}$$

$\forall d \in D,\ \forall f \in F,\ \forall i \in N_f,\ \forall u \in U_f^i$:

$$\sum_{p \in P} t_{iu}^{dfp} \leq t_{iu}^f, \tag{3.28}$$

$\forall f \in F,\ \forall u \in U_f$:

$$\sum_{i \in N_f} t_{iu}^f = t_u^f. \tag{3.29}$$

3.3.1.2 Branching rule algorithm

In the following, we present the conditions and rules on which the branching variable is chosen from the decision variable sets $t_u,\ t_{um}$ and t_u^f. Let $t_u^*,\ t_{um}^*$ and $t_u^{f,*}$ be the values of the corresponding variables for the solution of the current linear relaxation.

First, we branch on a variable t_u if it is integer infeasible. If all t_u variables are integer feasible, we check the t_{um} variables and branch on one of them if it is integer infeasible. The gap between lower and upper bound of the branch-and-bound algorithm closes quicker than through the general branching strategies, because the objective functions are based on cost, weight or power usage of the overall system and the number of cables is strongly linked to the number of installed units

The two previously proposed branching rules are for closing the lower and upper bound gap. The following branching rules do not close the lower and upper bound gap of the branch-and-bound algorithm. They are needed for the pruning rule which we propose later. The goal of the next two branching rules is that for all $u \in U$ the lower bound t_u^{lb} is equal to the upper bound t_u^{ub} for the decision variables $t_u,\ u \in U$. If t_u^* is greater than the lower bound t_u^{lb} for a t_u, then we branch on this t_u by creating a new node with $t_u^{lb} \geq t_u^*$ and another with $t_u^{ub} \leq t_u^* - 1$. If $t_u^* = t_u^{lb}$ for all $u \in U$, we check if t_u^* equals the upper bound t_u^{ub}. If this is not the case for a unit type $u \in U$, then we branch on t_u by creating a new node with $t_u^{lb} \geq t_u^* + 1$ and another with $t_u^{ub} \leq t_u^*$.

The goal of the last branching rule is that variables t_u^f are equal to their lower bound $t_u^{f,lb}$. Therefore, if $t_u^{f,*} \neq t_u^{f,lb}$ for a unit type $u \in U$, then we branch on $t_u^{f,*}$ by creating a new node with $t_u^{f,lb} = \lceil t_u^{f,*} \rceil$ and another with $t_u^{f,ub} = \lceil t_u^{f,*} \rceil - 1$.

Through the previous mentioned branching rules, we obtained that $t_u^{lb} = t_u^{ub}$ for all $u \in U$, all t_{um} variables are integer feasible and all t_u^f variables are equal to their lower bound $t_u^{f,lb}$. This enables us to use the following pruning and branching rule. Before introducing it, we first explain it is important for our problem.

A linear relaxation of the MILP does not guarantee that the redundancy requirement is met. For example, the solution of a linear relaxation can have

$$t_{iu}^{dfp,*} = t_{iu}^{df\hat{p},*} = 0.5$$

for $d \in D,\ f \in F,\ i \in N_f, u \in U_f^i,\ p, \hat{p} \in P$ and $p \neq \hat{p}$. Through this, a system based on a linear relaxation solution may have a combination of units given by t_u and t_{um} that does not provide enough ports for all paths if they are integer feasible. A small example is in Appendix B.2. It is very hard to prune nodes with such a linear relaxation solution through normal branching. Unfortunately, we did not find constraints or cuts which were able to prevent such solutions from occurring. This is why we use the following pruning and branching rule.

As part of the pruning and branching rule, we first test for all unit types $u \in U$ whose ports are all used in the linear relaxation solution if they have enough ports. We will refer to this test as *validity test* and we explain it later in detail. If a test for a unit $u \in U$ shows that not enough ports are available, we branch on variable t_{um} by creating a new node with $t_{um}^{ub} \geq t_{um}^* - 1$ and another with $t_{um}^{lb} \leq t_{um}^* + 1$, therefore pruning all possible solutions with $t_{um} = t_{um}^*$. If no such $u \in U$ is found, we do not branch on any t_u, t_{um} or t_u^f variable and instead we use the general branching rules implemented in the solver.

For the validity test, we need the assumption that $t_u^{lb} = t_u^{ub}$ for all $u \in U$, all t_{um} variables are integer feasible and all t_u^f variables are equal to their lower bound $t_u^{f,lb}$. Furthermore, we need the sets $E_f^+(u)$ and $E_f^-(u)$ which were defined in Section 3.2.1. Let $\overline{u}$ be the unit type for which we do the validity test. Since $t_{\overline{u}}^{ub} = t_{\overline{u}}^{lb}$, $t_{\overline{u}}^{f,*} = t_{\overline{u}}^{f,lb}$ for all $f \in F$ and constraints (3.10)

and (3.11) hold, we know that at least $\sum_{f\in F}\sum_{\hat{u}\in E_f^-(\overline{u})} t_{\hat{u}}^{f,lb}$ connections have to leave from and at least $\sum_{f\in F}\sum_{\hat{u}\in E_f^+(\overline{u})} t_{\hat{u}}^{f,lb}$ connections have to arrive at the locations where a unit of type $\overline{u}$ is installed in the overall system.

Through the values $t^*_{\overline{u}m}$, $m \in M^U$, of $t_{\overline{u}m}$, we can build all possible combinations of connections that can arrive at and leave from the units of type $\overline{u}$ for the different functions and check if they are feasible. If it is shown that no combination is feasible, then the unit type failed the validity test.

The pruning and branching rules are summarized in Algorithm 2.

Algorithm 2 Branching rule

Require: LP Relaxation (t^*, x^*)
for $u \in U$
  if t^*_u integer infeasible **then**
    Branch on t_u with $t_u \geq \lceil t^*_u \rceil$ and $t_u \leq \lfloor t^*_u \rfloor$
  else if t^*_{um} integer infeasible
    Branch on t_{um} with $t_{um} \geq \lceil t^*_{um} \rceil$ and $t_{um} \leq \lfloor t^*_{um} \rfloor$
  else if $t^*_u \neq t^{lb}_u$
    Branch on t_u with $t_u \geq t^*_u$ and $t_u \leq t^*_u - 1$
  else if $t^*_u \neq t^{ub}_u$
    Branch on t_u with $t_u \geq t^*_u + 1$ and $t_u \leq t^*_u$
  else if $t^{f,*}_u \neq t^{f,lb}_u$
    Branch on t^f_u with $t^f_u \geq \lceil t^{f,*}_u \rceil$ and $t^f_u \leq \lceil t^{f,*}_u \rceil - 1$
  end if
end for
for $u \in U$
  if u fails validity test
    Choose $m \in M^U$ with $t^*_{um} \geq 1$ and branch with $t_{um} \leq t^*_{um} - 1$ and $t_{um} \geq t^*_{um} + 1$, while remembering that t_{um} is integer feasible.
  end if
end for
if No decision variable was chosen for branching
  Use CPLEX standard branching rules
end if

3.3.2 Heuristic I

We introduce in this section the heuristic specifically designed for the DMS problem with redundancy. Before deciding to use a problem specific heuristic, we first tried general heuristics, e.g. feasibility pump [2], that are implemented into solvers such as CPLEX or Gurobi. Through computational tests, we saw that these general heuristics do not find a solution with a low objective value at the root node of the branch-and-bound tree and in most cases they were only able to find a solution with a low objective value only at much later nodes.

One reason why the general heuristics do not work well may be the following. Our model mostly consists of binary variables. Most of them have values that are below 0.5 for a solution of the linear relaxation, because of disjoint constraints and symmetry of paths. In many general heuristics, one step of the heuristic is to round the values of the binary variable. Most of the times this either leads to an infeasible solution or to a bad starting solution for the DMS problem with redundancy since almost all variables are rounded down to 0.

In the following we show an example of how our model behaves in the general MIP heuristic from [2] called feasibility pump. Let y be all variables of the model, y^* be the solution of an LP relaxation, $\tilde{y} = [y^*]$ and $J = \{1, \ldots, |y|\}$. For our model, $\tilde{y}$ is mostly equal to 0. The feasibility pump then tries to close the distance between y and $\tilde{y}$ by solving the model with the distance $\Delta(y, \tilde{y}) = \sum_{j\in J} |y_j - \tilde{y_j}|$ as the objective. The idea is that we obtain a different integer solution $\tilde{\tilde{y}}$ by rounding the new solution y^{**} and repeat this until the distance is 0. However if $\tilde{\tilde{y}} = \tilde{y}$, this

iteration would run infinitely. As our $\tilde{y}$ is mostly equal to 0 and the values of our LP relaxation solutions are below 0.5, we have in most cases that $\tilde{\tilde{y}} = \tilde{y}$. In order to avoid that the loop iterates indefinitely, the feasibility pump algorithm changes $\tilde{y}$ values randomly with the help of a score. However for our model, this leads to a slow convergence and a bad feasible solution. After a few nodes, we have that some variables are fixed to 1 and more values are greater than 0.5. This means that feasibility pump performs better deeper in the branch-and-bound tree. As we would like to obtain fast a solution with a low objective value at the root node, we developed our own problem specific heuristic.

We propose the following heuristic which will help us to solve the DMS problem in an exact way much more efficiently. The main idea of the algorithm is based on greedy algorithms where nodes or arcs of a graph that have the highest value in the LP relaxation are chosen. For our algorithm, the idea is to start with a path of a subsystem and to find an integer feasible solution for it. Afterwards we move to another path of a subsystem. After all paths are made integer feasible, we have also an integer feasible overall system.

Let $S = \{(d, f, p) \in D \times F \times P\}$ be the set of all paths. To decide with which path to start, we calculate

$$v_{dfp} = \left(\sum_{x^{dfp}_{iju\hat{u}} \neq 0} \left(1 - x^{dfp,*}_{iju\hat{u}}\right) \cdot \sum_{x^{dfp}_{iju\hat{u}} \neq 0} 1 \right) \tag{3.30}$$

for all $(d, f, p) \in S$, where $x^{dfp,*}_{iju\hat{u}}$ are the variable values of the linear relaxation. We call v_{dfp} *violation of the path*. Let $(\hat{d}, \hat{f}, \hat{p})$ be the path with the smallest violation. To obtain an integer feasible solution for decision variables of $(\hat{d}, \hat{f}, \hat{p})$, we look at the connection variables $x^{\hat{d}\hat{f}\hat{p}}_{iju\hat{u}}$. We define $1 - x^{\hat{d}\hat{f}\hat{p}}_{iju\hat{u}}$ as the violation of the variable. Now we choose the least violated variable and set it to 1. Afterwards, we solve the new problem and choose the new least violated variable. If all connection variables of the path are binary feasible, which means that they have value 0 or 1, then all location variables of this path are binary feasible because of the synchronization constraints. Now that this path is integer feasible, we choose the new least violated path and repeat this procedure until all the paths are feasible for the integer problem.

We have feasible subsystems when every path is feasible. This does not assure us a feasible overall system since there might still be infeasible fractional variables of the overall system. This is the case since for the synchronization constraint (3.3) the left hand side is a sum over the models of the units and the decision variables t_{ium} of the overall system can still be fractional in the linear relaxation even though all subsystem variables are integer feasible.

Therefore, if we have several variables $t_{ium} > 0$ for a unit u and location i, we choose the model $m \in M^U$ with the most ports E_{um} for unit u, set t_{ium} to 1 and solve the linear relaxation. This is repeated until all variables t_{ium} are integer feasible. By proceeding in this way we obtain a feasible solution that in most cases is better than the solutions obtained by standard heuristics implemented in CPLEX. Particularly, the greater the cardinality of D, F or P is, the better our heuristic works when compared to CPLEX heuristics. Our heuristic is shown in pseudo-code form in Algorithm 3.

3.3.3 Computational Results for Branch-and-Bound

In this subsection, we show our computational results using Heuristic I (Algorithm 3) and the branching rule (Algorithm 2). The study compares solution times of our model with tightening constraints and without (Table 3.2). Then it compares solution times of our solving method and the standard MILP solver of CPLEX for several instances. The instances differ in the number of doors, number of disjoint paths required, configurations of the locations and objective function. The instances are artificial and were built in cooperation with AIRBUS Group.

As mentioned in the introduction, we have two functions in the DMS and therefore $F = \{F1,\ F2\}$. The first function is the information flow from two sensors at a door to a controller

Algorithm 3 Model Heuristic

Require: LP Relaxation (t^*, x^*), K = number of violated variables,
K_f = number of violated variables on flow level f, $S = \{(d, f, p) \in D \times F \times P\}$
while $K_f \neq 0$
$v_{dfp} = \left(\sum_{x_{iu\hat{u}}^{dfp} \neq 0} \left(1 - x_{iju\hat{u}}^{dfp,*}\right) \cdot \sum_{x_{iu\hat{u}}^{dfp} \neq 0} 1\right)$
$(d, f, p)^* = \arg\min_{(d,f,p) \in S}\{v_{dfp}\}$
$S = S \backslash \{(d, f, p)^*\}$
$K_{(d,f,p)^*}$ = number of violated variables in the $(d, f, p)^*$ subsystem
Let $V_{(d,f,p)^*}$ be the set of all $x_{iju\hat{u}}^{(dfp)^*}$ variables in subsystem $(d, f, p)^*$
while $K_{(d,f,p)^*} \neq 0$
$\hat{x} = \arg\min\{x_{iju\hat{u}}^{(dfp)^*,*} - 1\}$
Set $\hat{x} = 1$ and solve LP
$V_{(d,f,p)^*} = V_{(d,f,p)^*} \backslash \{\hat{x}\}$
end while
end while
while $K \neq 0$
$(\hat{i}, \hat{u}) = \arg\min_{i \in N, u \in U, m \in M^u}\{|t_{ium} - 0.5|\}$
$\hat{m} = \arg\max_{m \in M^{\hat{u}}}\{E_{\hat{u}m}\}$
Set $t_{\hat{i}\hat{u}\hat{m}} = 1$ and solve LP
end while

(CPIOM) and the second function is the information flow from that controller to an outflow valve. Also, the controller used in both functions has to be the same for a door and, therefore, constraint (3.12) is used.

The DMS has 8 unit types: door (DO), latch-and-lock sensor (LLS), closed sensor (CS), outflow valve (OVF), outflow valve control unit (OCU), remote data concentrator (RDC), CPIOM, switch (SWT). We also have the following unit sets:

- $U^{s,e} = \{s_1 = DO, s_2 = CPIOM, e_1 = CPIOM, e_2 = OVF\}$,
- $U^{\mathrm{b}} = \{LLS, CS, OCU, RDC, SWT\}$,
- $U_1 = \{DO, LLS, CS, RDC, CPIOM, SWT\}$,
- $U_2 = \{OVG, OCU, RDC, CPIOM, SWT\}$.

Except for the door, all units are available in different models. For example, there can be various kinds of RDC models with different costs, weights or number of ports.

In Table 3.1, the size of the model for different instances is shown. It includes the number of potential locations, continuous variables, integer variables, binary variables, and constraints. For the two door instances, a more complex configuration of locations was used which resulted in a higher number of continuous variables. The more complex configuration of locations was not used for more doors to avoid having too large problems.

Doors	Locations	Continuous Variables	Binary Variables	Integer Variables	Constraints
2	66	3149	13038	21	14020
3	90	1772	21648	21	23468
4	91	1820	27226	21	29262

Table 3.1: Size of the problems for different number of doors and locations.

In the computational tests, a total of 5 different cost and weight sets were used and a time limit of 4 hours was set. All computations were done with a Four Intel Xeon E5-2680 v3 2.5GHz,

192GB RAM and CPLEX 12.5.1. Implementation of the branching rules and heuristic was done in C++. Also, all available cuts in CPLEX were disabled because computational tests showed that they are not efficient for this problem. The problem instances in the tables have the names $i(d, p, loc, \ell)$ with d the number of doors, p the number of paths, loc the number of locations in the configuration of the space, and ℓ the number of the cost and weight set.

In Table 3.2, we solve the DMS problem with and without constraints (3.16)- (3.21). Column "Objective Function" refers to which objective function was used, z_{MIP} is the MIP optimal value of the instance, z_{LP} is the optimal value for the linear relaxation at the root node, Gap is the gap at the root node and Time is the solution time in seconds.

Instance	Objective Function	z_{MIP}	Without			With		
			z_{LP}	Gap	Time	z_{LP}	Gap	Time
$i(2,2,66,1)$	obj1	2721.92	2608.06	4.18	14400	2721.92	0.00	20
$i(2,2,66,1)$	obj2	22.00	17.00	22.73	14400	22 .00	0.00	19
$i(2,2,66,1)$	obj3	32.00	26.00	18.75	14400	32 .00	0.00	22
$i(2,2,66,1)$	obj4	4127.88	3847.19	6.80	14400	4127.88	0.00	16
$i(2,2,66,2)$	obj1	2539.91	2473.28	2.62	14400	2539.91	0.00	13
$i(2,2,66,3)$	obj1	2257.46	2173.33	3.73	14400	2257.46	0.00	11
$i(2,2,66,4)$	obj1	2871.12	2728.16	4.98	14400	2871.12	0.00	13
$i(2,2,66,5)$	obj1	3721.74	3592.35	3.48	14400	3721.74	0.00	17
$i(2,3,66,1)$	obj1	4082.88	3912.09	4.18	14400	4082.88	0.00	98
$i(2,3,66,1)$	obj2	32.00	24.50	23.44	14400	32.00	0.00	177
$i(2,3,66,1)$	obj3	48.00	39.00	18.75	14400	48.00	0.00	94
$i(2,3,66,1)$	obj4	6191.82	5770.78	6.80	14400	6191.82	0.00	110
$i(2,3,66,2)$	obj1	3809.86	3709.93	2.62	14400	3809.86	0.00	290
$i(2,3,66,3)$	obj1	3386.19	3260.00	3.73	14400	3386.19	0.00	144
$i(2,3,66,4)$	obj1	4306.68	4092.24	4.98	14400	4306.68	0.00	167
$i(2,3,66,5)$	obj1	5582.60	5388.54	3.48	14400	5582.60	0.00	197
$i(3,2,90,1)$	obj1	2731.00	2608.50	4.49	14400	2731.00	0.00	57
$i(3,2,90,1)$	obj2	27.00	18.00	33.33	14400	27.00	0.00	72
$i(3,2,90,1)$	obj3	40.00	30.00	25.00	14400	40.00	0.00	164
$i(3,2,90,1)$	obj4	4182.22	3854.49	7.84	14400	4182.22	0.00	148
$i(3,2,90,2)$	obj1	2545.63	2473.28	2.84	14400	2545.63	0.00	252
$i(3,2,90,3)$	obj1	2262.84	2173.33	3.96	14400	2262.84	0.00	117
$i(3,2,90,4)$	obj1	2876.44	2728.16	5.15	14400	2876.44	0.00	355
$i(3,2,90,5)$	obj1	3731.41	3592.36	3.73	14400	3731.41	0.00	183
$i(3,3,90,1)$	obj1	4096.50	3912.75	4.49	14400	4096.50	0.00	758
$i(3,3,90,1)$	obj2	39.00	25.50	34.62	14400	39.00	0.00	2248
$i(3,3,90,1)$	obj3	60.00	45.00	25.00	14400	60.00	0.00	1952
$i(3,3,90,1)$	obj4	6273.33	5781.73	7.84	14400	6273.33	0.00	3486
$i(3,3,90,2)$	obj1	3818.45	3709.93	2.84	14400	3818.45	0.00	4437
$i(3,3,90,3)$	obj1	3394.27	3260.00	3.96	14400	3394.27	0.00	1792
$i(3,3,90,4)$	obj1	4314.66	4092.24	5.15	14400	4314.66	0.00	1218
$i(3,3,90,5)$	obj1	5597.12	5388.54	3.73	14400	5597.12	0.00	3008
$i^*(3,2,90,1)$	obj1	2829.00	2608.50	7.79	14400	2731 .00	3.46	14400
$i^*(3,2,90,1)$	obj2	27.00	18.00	33.33	14400	27.00	0.00	62
$i^*(3,2,90,1)$	obj3	40.00	30.00	25.00	14400	40.00	0.00	59
$i^*(3,2,90,1)$	obj4	4182.22	3854.49	7.84	14400	4182.22	0.00	144
$i^*(3,2,90,2)$	obj1	2644.33	2473.28	6.47	14400	2545.63	3.73	14400
$i^*(3,2,90,3)$	obj1	2336.34	2173.33	6.98	14400	2262.84	3.15	14400
$i^*(3,2,90,4)$	obj1	2886.24	2728.16	5.48	14400	2876.44	0.34	14400
$i^*(3,2,90,5)$	obj1	3869.73	3592.36	7.17	14400	3731.41	3.57	14400
$i(4,2,91,1)$	obj1	2838.08	2626.22	7.46	14400	2740.08	3.45	14400

$i(4,2,91,1)$	obj2	32.00	27.00	15.63	14400	32.00	0.00	752
$i(4,2,91,1)$	obj3	48.00	42.00	12.50	14400	48.00	0.00	855
$i(4,2,91,1)$	obj4	4236.56	3955.87	6.63	14400	4236.56	0.00	65
$i(4,2,91,2)$	obj1	2650.06	2484.73	6.24	14400	2551.36	3.72	14400
$i(4,2,91,3)$	obj1	2341.73	2184.09	6.73	14400	2268.23	3.14	14400
$i(4,2,91,4)$	obj1	2891.56	2738.80	5.28	14400	2881.76	0.34	14400
$i(4,2,91,5)$	obj1	3879.41	3611.71	6.90	14400	3741.09	3.57	14400
Average Computing times					14400			3492

Table 3.2: Analysis of constraints (3.16)- (3.21)

It is evident that the use of constraints (3.16)- (3.21) helps considerably to solve the instances of the DMS problem. In most of the cases, we were able to find an optimal solution at the root node with the tightening constraints. Without them, we were not able to solve any instance in 4 hours. However, there are some instances which cannot be solved within the time limit even with the use of these constraints. It must be noted that for all instances that could not be solved to optimality, the MILP optimal solution was found as the upper bound and the gap could not be closed to prove optimality and the lower bound did not improve after the root node.

Next, we compare the performance of the standard branch-and-bound algorithm of CPLEX ("Standard") and the modification that uses the branching rules and the heuristic that we propose in this chapter ("Proposed"). In both cases, as mentioned earlier, the solver cuts were disabled. Table 3.3 shows the different instances, objective function used, linear relaxation value at the root node (z_{LP}), gap in % and computational time, respectively. Independently on whether optimality is achieved or not, z_{lb} and z_{ub} show the best lower bound and the best upper bound when the run stops. If z_{ub} is in bold face, it means that the optimal solution was found and proven to be optimal.

Instance	Objective Function	z_{LP}	Gap	Standard			Proposed		
				Time	z_{lb}	z_{ub}	Time	z_{lb}	z_{ub}
$i(2,2,66,1)$	obj1	2721.92	0.00	20	2721.92	**2721.92**	22	2721.92	**2721.92**
$i(2,2,66,1)$	obj2	22.00	0.00	19	22.00	**22.00**	24	22.00	**22.00**
$i(2,2,66,1)$	obj3	32.00	0.00	22	32.00	**32.00**	20	32.00	**32.00**
$i(2,2,66,1)$	obj4	4127.88	0.00	16	4127.88	**4127.88**	22	4127.88	**4127.88**
$i(2,2,66,2)$	obj1	2539.91	0.00	13	2539.91	**2539.91**	29	2539.91	**2539.91**
$i(2,2,66,3)$	obj1	2257.46	0.00	11	2257.46	**2257.46**	29	2257.46	**2257.46**
$i(2,2,66,4)$	obj1	2871.12	0.00	13	2871.12	**2871.12**	27	2871.12	**2871.12**
$i(2,2,66,5)$	obj1	3721.74	0.00	17	3721.74	**3721.74**	27	3721.74	**3721.74**
$i(2,3,66,1)$	obj1	4082.88	0.00	98	4082.88	**4082.88**	69	4082.88	**4082.88**
$i(2,3,66,1)$	obj2	32.00	0.00	177	32.00	**32.00**	99	32.00	**32.00**
$i(2,3,66,1)$	obj3	48.00	0.00	94	48.00	**48.00**	94	48.00	**48.00**
$i(2,3,66,1)$	obj4	6191.82	0.00	110	6191.82	**6191.82**	66	6191.82	**6191.82**
$i(2,3,66,2)$	obj1	3809.86	0.00	290	3809.86	**3809.86**	91	3809.86	**3809.86**
$i(2,3,66,3)$	obj1	3386.19	0.00	144	3386.19	**3386.19**	223	3386.19	**3386.19**
$i(2,3,66,4)$	obj1	4306.68	0.00	167	4306.68	**4306.68**	72	4306.68	**4306.68**
$i(2,3,66,5)$	obj1	5582.60	0.00	197	5582.60	**5582.60**	87	5582.60	**5582.60**
$i(3,2,90,1)$	obj1	2731.00	0.00	57	2731.00	**2731.00**	60	2731.00	**2731.00**
$i(3,2,90,1)$	obj2	27.00	0.00	72	27.00	**27.00**	84	27.00	**27.00**
$i(3,2,90,1)$	obj3	40.00	0.00	164	40.00	**40.00**	68	40.00	**40.00**
$i(3,2,90,1)$	obj4	4182.22	0.00	148	4182.22	**4182.22**	47	4182.22	**4182.22**
$i(3,2,90,2)$	obj1	2545.63	0.00	252	2545.63	**2545.63**	70	2545.63	**2545.63**
$i(3,2,90,3)$	obj1	2262.84	0.00	117	2262.84	**2262.84**	73	2262.84	**2262.84**
$i(3,2,90,4)$	obj1	2876.44	0.00	356	2876.44	**2876.44**	65	2876.44	**2876.44**
$i(3,2,90,5)$	obj1	3731.41	0.00	183	3731.41	**3731.41**	68	3731.41	**3731.41**
$i(3,3,90,1)$	obj1	4096.50	0.00	758	4096.50	**4096.50**	168	4096.50	**4096.50**

$i(3,3,90,1)$	obj2	39.00	0.00	2248	39.00	**39.00**	303	39.00	**39.00**
$i(3,3,90,1)$	obj3	60.00	0.00	1952	60.00	**60.00**	218	60.00	**60.00**
$i(3,3,90,1)$	obj4	6273.33	0.00	3486	6273.33	**6273.33**	171	6273.33	**6273.33**
$i(3,3,90,2)$	obj1	3818.45	0.00	4437	3818.45	**3818.45**	243	3818.45	**3818.45**
$i(3,3,90,3)$	obj1	3394.27	0.00	1792	3394.27	**3394.27**	241	3394.27	**3394.27**
$i(3,3,90,4)$	obj1	4314.66	0.00	1217	4314.66	**4314.66**	233	4314.66	**4314.66**
$i(3,3,90,5)$	obj1	5597.12	0.00	3008	5597.12	**5597.12**	244	5597.12	**5597.12**
$i^*(3,2,90,1)$	obj1	2731.00	3.46	14400	2731.00	2829.00	203	2829.00	**2829.00**
$i^*(3,2,90,1)$	obj2	27.00	0.00	62	27.00	**27.00**	83	27.00	**27.00**
$i^*(3,2,90,1)$	obj3	40.00	0.00	59	40.00	**40.00**	66	40.00	**40.00**
$i^*(3,2,90,1)$	obj4	4182.22	0.00	144	4182.22	**4182.22**	57	4182.22	**4182.22**
$i^*(3,2,90,2)$	obj1	2545.63	3.73	14400	2545.63	2644.33	240	2644.33	**2644.33**
$i^*(3,2,90,3)$	obj1	2262.84	3.15	14400	2262.84	2336.34	228	2336.34	**2336.34**
$i^*(3,2,90,4)$	obj1	2876.44	0.34	14400	2876.44	2886.24	243	2886.24	**2886.24**
$i^*(3,2,90,5)$	obj1	3731.41	3.57	14400	3731.41	3869.73	271	3869.73	**3869.73**
$i(4,2,91,1)$	obj1	2740.08	3.45	14400	2740.08	2838.08	264	2838.08	**2838.08**
$i(4,2,91,1)$	obj2	32.00	0.00	752	32.00	**32.00**	122	32.00	**32.00**
$i(4,2,91,1)$	obj3	48.00	0.00	855	48.00	**48.00**	77	48.00	**48.00**
$i(4,2,91,1)$	obj4	4236.56	0.00	65	4236.56	**4236.56**	68	4236.56	**4236.56**
$i(4,2,91,2)$	obj1	2551.36	3.72	14400	2551.36	2650.06	329	2650.06	**2650.06**
$i(4,2,91,3)$	obj1	2268.23	3.14	14400	2268.23	2341.73	271	2341.73	**2341.73**
$i(4,2,91,4)$	obj1	2881.76	0.34	14400	2881.76	2891.56	279 1	2891.561	**2891.56**
$i(4,2,91,5)$	obj1	3741.09	3.57	14400	3741.09	3879.41	323	3879.41	**3879.41**
Average Computing times					3492			131	

Table 3.3: Analysis of new branching rules and heuristic

For the smaller instances with two doors and two paths, the general solver is slightly better in all instances except one. Albeit the new method obtains better solution times with more doors or paths. Therefore, for larger instances our proposed method performs much better. Particularly, it can solve many instances that the general solver cannot solve within the time limit of 4 hours. These are the instances where a gap between linear relaxation at the root node and optimal solution still exists.

3.4 Branch-and-Price Approach

After using the branch-and-bound approach, we want to explore the option to use branch-and-price to solve the DMS problem with redundancy. We want to price the paths of functions and to do this, we have to reformulate certain parts of the MILP formulation from Section 3.2.

3.4.1 Reformulation

We need the following changes to constraints of the MILP formulation and, furthermore, need a new level of variables for paths of the doors. This level is between the overall system level and the function level.

These additional variables are location variables for components and connections for each path of the door. Let $(d,p) \in D \times P$ refer to path p of the subsystem for door d wich includes all functions $f \in F$. The required decision variables for every $(d,p) \in D \times P$ are:

- $t^{dp}_{ium} \in \{0,1\},\ i \in N,\ u \in U^i,\ m \in M^u$, a binary variable that takes value 1 if a unit of type u is installed at location i, and
- $x^{dp}_{iju\hat{u}} \in \{0,1\},\ (i,j) \in A,\ u \in U^i,\ \hat{u} \in W^j(u)$, a binary variable that takes value 1 if locations i and j are connected and unit u and $\hat{u}$ are installed at i and j, respectively. It is 0 otherwise.

We need to synchronize these new decision variables with the location variables of the overall system level and the function level. For this we drop constraints (3.3) and (3.4) and add the following constraints. Constraint (3.31) and (3.32) synchronize the overall system with the path level of the doors and they also create node-disjoint and arc-disjoint paths which are needed for redundancy.

$\forall d \in D\ \forall i \in N\ \forall u \in U^i$

$$\sum_{p \in P, m \in M^u} t^{dp}_{ium} \leq \sum_{m \in M^u} t_{ium}, \tag{3.31}$$

$\forall d \in D\ \forall (i,j) \in A\ \forall u \in U^i\ \forall \hat{u} \in W^j(u)$:

$$\sum_{p \in P} x^{dp}_{iju\hat{u}} \leq x_{iju\hat{u}}. \tag{3.32}$$

The following two constraints (3.33) and (3.34) synchronize the path level with the function level.
$\forall (d,f,p)\ \forall i \in N_f\ \forall u \in U^i_f$

$$t^{dfp}_{iu} \leq \sum_{m \in M^u} t^{dp}_{ium}, \tag{3.33}$$

$\forall (d,f,p)\ \forall (i,j) \in A\ u \in U^i_f, \hat{u} \in W^j_f(u)$:

$$x^{dfp}_{iju\hat{u}} \leq x^{dp}_{iju\hat{u}}. \tag{3.34}$$

These new constraints and variables are needed to obtain a dual solution that proves the optimality of the reduced master problem in branch-and-price. Furthermore, we need constraints like (3.14) which are only for the overall system level in the previous formulation also for the (d,p) level in the pricing problems. Therefore, we have to add constraint (3.35) to the formulation.

$\forall (d,p)\ \forall i \in N,\ \forall u \in U^i$:

$$\sum_{\substack{\ell \in V^-(i,u),\\ \hat{u} \in U^\ell\ \mid\ u \in C(\hat{u})}} x^{dp}_{\ell i\hat{u}u} + \sum_{\substack{\ell \in V^+(i,u),\\ \hat{u} \in W^k(u)}} x^{dp}_{i\ell u\hat{u}} \leq \sum_{m \in M^u} t^{dp}_{ium} E_{um}. \tag{3.35}$$

In the following, we show the master and pricing problems which are necessary for the branch-and-price algorithm.

3.4.2 Master Problem and Pricing Problems

In this subsection, we introduce the master problem and the pricing problems. In the following, we split the constraint matrix of our formulation into three matrices.

- Constraints (3.1), (3.2), (3.31) and (3.32) are represented through matrix S and RHS vector s. These are the synchronization constraints between (d,f,p) variables and overall variables or between (d,p) variables and overall variables. These are part of the master problem.
- Constraints (3.13)-(3.15) and (3.18)-(3.21),are represented through matrix O and RHS vector o. These are synchronization or architectural constraints that only include overall variables and they are also part of the master problem.
- Constraints (3.5)-(3.8), (3.9)-(3.12), (3.16), (3.17), and (3.33)-(3.35) are represented through

matrices K_{dp} and RHS vector k_{dp}, $d \in D,\ p \in P$. These are only constraints that include (d, f, p) and (d, p) variables.

Let t be the vector containing all t_{ium} variables and x be the vector containing all $x_{iju\hat{u}}$ variables. Furthermore, let t_{dp} and x_{dp} be vectors containing all corresponding variables $t^{dp}_{ium},\ x^{dp}_{iju\hat{u}},\ t^{dfp}_{iu}$ and $x^{dfp}_{iju\hat{u}}$ for $d, p \in D \times P$, respectively. Also, let

$$l = (t, x, (t_{dp})_{d,p \in D \times P}, (x_{dp})_{d,p \in D \times P}). \tag{3.36}$$

Let n, n_0 and $n_{dp},\ d, p \in D \times P$ so that we have $l \in \{0,1\}^n, (t, x) \in \{0,1\}^{n_o}$ and $(t_{dp}, x_{dp}) \in \{0,1\}^{n_{dp}},\ d, p \in D \times P$, respectively. We can now write the MILP formulation in the following form:

$$\begin{aligned} \min\ & f(l) && (3.37)\\ s.t.\ & Sl \leq s \\ & O(t, x) \leq o \\ & K_{dp}(t_{dp}, x_{dp}) \leq k_{dp},\ \forall d \in D,\ p \in P \qquad\qquad l \in \{0,1\}^n. \end{aligned}$$

Figure 3.3 shows an example constraint matrix for two doors and three paths. We can identify the matrices O, $K_{dp},\ (d, p) \in D \times P$, and S and see that we have a block diagonal form with connecting constrains in S.

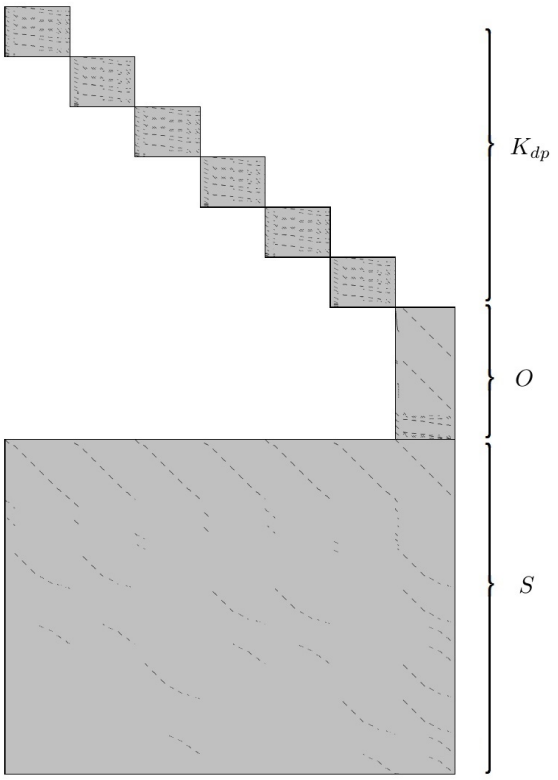

Figure 3.3: Example Constraint matrix for 2 doors and 3 paths.

Let $\mathcal{K}_{dp} = \{(t_{dp}, x_{dp}) \mid K_{dp}(t_{dp}, x_{dp}) \leq k_{dp}\}$, $(d, p) \in D \times P$. Then we can write formulation 3.37 as

$$\begin{aligned} \min\ & f(t, x) && (3.38)\\ s.t.\ & Sl \leq s \\ & O(t, x) \leq o \end{aligned}$$

$$(t_{dp}, x_{dp}) \in \mathcal{K}_{dp},\ \forall d \in D,\ p \in P.$$

To look in more detail at the $\mathcal{K}_{dp}$, we write out the included constraints and see why we can use a result from network flow theory. The first constraints are flow constraints

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ i \in N_f,\ \forall u \in U_f^i$:

$$\sum_{\substack{j \in V_f^+(i,u),\\ \hat{u} \in W_f^j(u)}} x_{iju\hat{u}}^{dfp} \leq E_{fu}^{out} t_{iu}^{dfp}, \tag{3.39}$$

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ j \in N_f,\ \forall \hat{u} \in U_f^j$:

$$\sum_{\substack{i \in V_f^-(j,\hat{u}),\\ u \in U_f \mid \hat{u} \in C_f(u)}} x_{iju\hat{u}}^{dfp} \leq E_{fu}^{in} t_{j\hat{u}}^{dfp}, \tag{3.40}$$

$\forall d \in D,\ \forall p \in P,\ \forall f \in F, \forall i \in N_f$:

$$\sum_{u \in (U_f^b \cup \{e_f\}) \cap U_f^i} T_u^{f,\text{in}} t_{iu}^{dfp} \leq \sum_{\substack{\ell \in V_f^-(i),\ u \in (U_f^b \cup \{e_f\}) \cap U_f^i,\\ \hat{u} \in U_f \mid u \in C_f(\hat{u})}} x_{\ell i \hat{u} u}^{dfp}, \tag{3.41}$$

$$\sum_{u \in (U_f^b \cup \{s_f\}) \cap U_f^i} T_u^{f,\text{out}} t_{iu}^{dfp} \leq \sum_{\substack{\ell \in V_f^+(i),\ u \in (U_f^b \cup \{s_f\}) \cap U_f^i,\\ \hat{u} \in W_f^\ell(u)}} x_{i\ell u\hat{u}}^{dfp}, \tag{3.42}$$

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall u_f \in \{s_f, e_f\}$:

$$\sum_{i \in N_f^{u_f}} t_{iu_f}^{dfp} = 1. \tag{3.43}$$

These constraints create the flow for the path of the functions. And, therefore, every set $\mathcal{K}_{dp}$ contains all possible paths for the two functions of door d. The following constraints are only architectural restrictions, synchronizations and tightening constraints.

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall i \in N_f,\ \forall u \in U_f^i,\ \forall \hat{u} \in C_f^+(u)$:

$$t_{iu}^{dfp} \leq \sum_{j \in V_f^+(i) \cap N_f^{\hat{u}}} x_{iju\hat{u}}^{dfp}. \tag{3.44}$$

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall j \in N_f,\ \forall u \in U_f^j,\ \forall \hat{u} \in C_f^-(u)$:

$$t_{ju}^{dfp} \leq \sum_{i \in V_f^-(j) \cap N_f^{\hat{u}}} x_{ij\hat{u}u}^{dfp}. \tag{3.45}$$

$\forall i \in N^{e_{f_1}}, \forall d \in D,\ \forall p \in P$ with $e_{f_1} = s_{f_2}$:

$$t_{ie_{f_1}}^{df_1p} = t_{is_{f_2}}^{df_2p}. \tag{3.46}$$

$\forall d \in D,\ \forall p \in P,\ \forall f \in F,\ \forall u \in U_f$:

$$\sum_{\substack{\hat{u} \in \{\overline{u} \ \mid \ u \in C_f^+(\overline{u})\}, \\ i \in N_f^{\hat{u}}}} t_{i\hat{u}}^{dfp} \leq E_{fu}^{in} \sum_{i \in N_f^u} t_{iu}^{dfp} \tag{3.47}$$

$$\sum_{\substack{\hat{u} \in \{\overline{u} \ \mid \ u \in C_f^-(\overline{u})\}, \\ i \in N_f^{\hat{u}}}} t_{i\hat{u}}^{dfp} \leq E_{fu}^{out} \sum_{i \in N_f^u} t_{iu}^{dfp} \tag{3.48}$$

$\forall (d, f, p)\ \forall i \in N_f\ \forall u \in U_f^i$

$$t_{iu}^{dfp} \leq \sum_{m \in M^u} t_{ium}^{dp}, \tag{3.49}$$

$\forall (d, f, p)\ \forall (i, j) \in A\ u \in U_f^i, \hat{u} \in W_f^j(u)$:

$$x_{iju\hat{u}}^{dfp} \leq x_{iju\hat{u}}^{dp}. \tag{3.50}$$

$\forall (d, p)\ \forall i \in N,\ \forall u \in U^i$:

$$\sum_{\substack{\ell \in V^-(i,u), \\ \hat{u} \in U^\ell \ \mid \ u \in C(\hat{u})}} x_{\ell i\hat{u}u}^{dp} + \sum_{\substack{\ell \in V^+(i,u), \\ \hat{u} \in W^k(u)}} x_{i\ell u\hat{u}}^{dp} \leq \sum_{m \in M^u} t_{ium}^{dp} E_{um}. \tag{3.51}$$

The paths contained in $\mathcal{K}_{dp}$ are represented by the new decision variables t_{ium}^{dp} and $x_{iju\hat{u}}^{dp}$. Furthermore, we know that every extreme point of these sets are paths for the two functions. Hence, we use $K_{dp} \leq k_{dp}$ as our constraints of the pricing problem. We therefore price the paths of our doors and we can use the following result of network flow theory to formulate the master problem. An extreme point $(t_{\mathfrak{p}dp}, x_{\mathfrak{p}dp})$ of the polytope $conv(\mathcal{K}_{dp})$ is a path $\mathfrak{p}$ which includes both functions in the network. Let $\mathfrak{P}_{dp}$ be the set of all possible paths/extreme points of $\mathcal{K}_{dp}$. We can then rewrite variable vectors (t_{dp}, x_{dp}) as a convex combination of possible paths:

$$(t_{dp}, x_{dp}) = \sum_{\mathfrak{p} \in \mathfrak{P}_{dp}} (t_{\mathfrak{p}dp}, x_{\mathfrak{p}dp}) \lambda_{\mathfrak{p}}$$

$$\sum_{\mathfrak{p} \in \mathfrak{P}_{dp}} \lambda_{\mathfrak{p}} = 1$$

$$\lambda_{\mathfrak{p}} \geq 0,\ \forall \mathfrak{p} \in \mathfrak{P}_{dp}.$$

Furthermore, we split the matrix S to S_o and S_{dp}, $(d, p) \in D \times P$, so that we have

$$Sl \leq s \ \Leftrightarrow \ S_o(t, x) + \sum_{d \in D,\ p \in P} S_{dp}(t_{dp}, x_{dp}) \leq s.$$

We obtain the following master problem that are equivalent to formulation (3.37) and (3.38):

$$\begin{aligned} \min \ & f(t, x) \\ s.t. \ & O(t, x) \leq o \\ & S_o(t, x) + \sum_{d \in D, p \in P} \left(S_{dp} \sum_{\mathfrak{p} \in \mathfrak{P}_{dp}} (t_{\mathfrak{p}dp}, x_{\mathfrak{p}dp}) \lambda_{\mathfrak{p}} \right) \leq s \end{aligned} \tag{3.52}$$

$$\sum_{\mathfrak{p}\in\mathfrak{P}_{dp}} \lambda_{\mathfrak{p}} = 1$$
$$\lambda_{\mathfrak{p}} \in \mathbb{N},\ \forall \mathfrak{p} \in \mathfrak{P}_{dp},\ \forall d \in D,\ p \in P$$
$$(t, x) \in \{0, 1\}^{n_o}$$

To understand the master problem better, we write out the constraints related to constraint matrices O and S and how they were reformulated for (3.52). Constraints $O(t, x) \leq o$ are the overall system flow constraints

$\forall (i, j) \in A,\ \forall u \in U^i$:

$$\sum_{\hat{u}\in W^j(u)} x_{iju\hat{u}} \leq \sum_{m\in M^u} t_{ium}, \tag{3.53}$$

$\forall (i, j) \in A,\ \forall \hat{u} \in U^j$:

$$\sum_{u\in U^i \ \mid\ \hat{u}\in C(u)} x_{iju\hat{u}} \leq \sum_{m\in M^u} t_{j\hat{u}m}, \tag{3.54}$$

$\forall i \in N\ \ \forall u \in (U^{\mathrm{b}} \cup \bigcup_{f\in F}\{e_f\}) \cap U^i$:

$$\sum_{m\in M^u} t_{ium} \leq \sum_{\substack{\ell\in V^-(i,u),\\ \hat{u}\in U^\ell \ \mid\ u\in C(\hat{u})}} x_{\ell i\hat{u}u}, \tag{3.55}$$

$\forall i \in N\ \ \forall u \in (U^{\mathrm{b}} \cup \bigcup_{f\in F}\{s_f\}) \cap U^i$:

$$\sum_{m\in M^u} t_{ium} \leq \sum_{\substack{\ell\in V^+(i,u),\\ \hat{u}\in W^j(u)}} x_{i\ell u\hat{u}}, \tag{3.56}$$

and architecural constraints

$\forall i \in N$:

$$\sum_{\substack{u\in U^i,\\ m\in M^u}} t_{ium} \leq 1, \tag{3.57}$$

$\forall i \in N,\ \forall u \in U^i$:

$$\sum_{\substack{\ell\in V^-(i,u),\\ \hat{u}\in U^\ell \ \mid\ u\in C(\hat{u})}} x_{\ell i\hat{u}u} + \sum_{\substack{\ell\in V^+(i,u),\\ \hat{u}\in W^k(u)}} x_{i\ell u\hat{u}} \leq \sum_{m\in M^u} t_{ium} E_{um}. \tag{3.58}$$

$\forall i \in N^{OVF}$:

$$2 \sum_{m\in M^{OVF}} x_{i(OVF)m} \leq \sum_{j\in N^{OCU}\cap V^-(i)} x_{ji(OCU)(OVF)}. \tag{3.59}$$

No changes were made to these constraints to create the master problem. Now, we consider the constraints related to matrix S. We have the two synchronization constraints (3.60) and (3.61)

between overall system and the subsystems related to the functions of doors.

$\forall i \in N,\ \forall u \in U^i$:

$$\sum_{m \in M^u} t_{ium} \leq \sum_{\substack{d \in D \ \ p \in P, \\ f \in F(u)}} t_{iu}^{dfp}, \tag{3.60}$$

$\forall (i,j) \in A,\ \forall u \in U^i,\ \forall \hat{u} \in W^j(u)$:

$$x_{iju\hat{u}} \leq \sum_{\substack{d \in D,\ p \in P, \\ f \in F \ \mid \ (i,j) \in A_f, \\ u \in U_f^i,\ \hat{u} \in W_f^j(u)}} x_{iju\hat{u}}^{dfp}. \tag{3.61}$$

After reformulation, the constraints are

$\forall i \in N,\ \forall u \in U^i$:

$$\sum_{m \in M^u} t_{ium} \leq \sum_{\substack{d \in D \ \ p \in P, \\ \mathfrak{p} \in \mathfrak{P}_{dp}}} t_{\mathfrak{p}dp} \lambda_{\mathfrak{p}}, \tag{3.62}$$

$\forall (i,j) \in A,\ \forall u \in U^i,\ \forall \hat{u} \in W^j(u)$:

$$x_{iju\hat{u}} \leq \sum_{\substack{d \in D,\ p \in P, \\ \mathfrak{p} \in \mathfrak{P}_{dp}}} x_{\mathfrak{p}dp} \lambda_{\mathfrak{p}}. \tag{3.63}$$

Lastly, we have the two synchronization constraints (3.66) and (3.67) which also enforce that the paths are node- and arc-disjoint.

$\forall d \in D\ \forall i \in N\ \forall u \in U^i$

$$\sum_{p \in P, m \in M^u} t_{ium}^{dp} \leq \sum_{m \in M^u} t_{ium}, \tag{3.64}$$

$\forall d \in D\ \forall (i,j) \in A\ \forall u \in U^i\ \forall \hat{u} \in W^j(u)$:

$$\sum_{p \in P} x_{iju\hat{u}}^{dp} \leq x_{iju\hat{u}}. \tag{3.65}$$

After reformulation, the constraints are

$\forall d \in D\ \forall i \in N\ \forall u \in U^i$

$$\sum_{\substack{p \in P, \\ \mathfrak{p} \in \mathfrak{P}_{dp}}} t_{\mathfrak{p}dp} \lambda_{\mathfrak{p}} \leq \sum_{m \in M^u} t_{ium}, \tag{3.66}$$

$\forall d \in D\ \forall (i,j) \in A\ \forall u \in U^i\ \forall \hat{u} \in W^j(u)$:

$$\sum_{\substack{p \in P, \\ \mathfrak{p} \in \mathfrak{P}_{dp}}} x_{\mathfrak{p}dp} \lambda_{\mathfrak{p}} \leq x_{iju\hat{u}}. \tag{3.67}$$

This gives us the valid master problem (3.52). Let $\boldsymbol{\pi}_{dp}$ be the dual optimal solutions of the the

RMP (reduced master problem) for constraint matrix S_{dp} and π^0_{dp} for convexity constraint $\lambda_{\mathfrak{p}} \geq 0$, for all $\mathfrak{p} \in \mathfrak{P}_{dp}$,$(d,p) \in D \times P$. With the constraints sets $\mathcal{K}_{dp}$ for the pricing problems, we can write the pricing problems for $(d,p) \in D \times P$ as follows:

$$\begin{aligned} \overline{c}^*_{dp} := \min \quad & -\boldsymbol{\pi}^T_{dp} S_{dp}(t_{dp}, x_{dp}) - \pi^0_{dp} \\ s.t. \quad & K_{dp}(t_{dp}, x_{dp}) \leq k_{dp} \\ & (t_{dp}, x_{dp}) \in \{0,1\}^{n_{dp}} \end{aligned}$$

With this, we have enough to implement a general branch-and-price algorithm. Before developing such an algorithm that is tailored to the DMS problem with redundancy, we tried using the generic branch-cut-and-price solver GCG (see [14]). We first tried to solve the linear relaxation with GCG. The solver detected the same structure of the constraint matrix as we did. Unfortunately, even for the smallest problem with 2 doors and 2 paths, the solver did not converge to an optimal solution within one hour. And for larger problems we did not obtain an optimal solution. Even though we knew the outcome before, we tried to solve the MILP with GCG. The outcome was also that we did not obtain a solution in a reasonable time. It seems that the MILP formulation is not suitable for a general column generation/branch-and-price solver. Therefore, we addressed the problem of the slow convergence to optimality as explained in the following subsection.

3.4.3 Solving Dual Problem of Reduced Master Problem

Through computational experiments, we saw that the results from dual solutions obtained through solving the standard primal problem does not prove optimality in the column generation algorithm in a reasonable time as the convergence is very slow and unstable. We observed that this is caused by the symmetry of the location variables in the pricing problem. To reduce the number of iterations needed to prove optimality and converge faster, we tried solving the dual problem directly with a different objective function and replaced variables to remove the symmetry.

Let b be the dual variables of constraints (3.31) and let o be the dual variables of constraints (3.32). For all $d \in D$ and $u \in U$, we reformulated the dual problem and all dual variables $b_{d,i,u}$, $i \in N$, where replaced with $b_{d,u}$ one to one. Also for all $d \in D$, $j \in N$, $u \in U$ and $\hat{u} \in W_j(u)$, we replaced all dual variables $o_{d,i,j,u,\hat{u}}$, $i \in N^u$, with $o_{d,j,u,\hat{u}}$. This cancels the symmetry problem of the pricing problems. Let g be the objective function of the dual problem and z^* the optimal solution of the reduced master problem. We add the constraint $g = z^*$ to the dual problem. With this, we can change the objective function and still have an optimal solution for objective function g. Let y_{dp} be the dual variable associated with the convexity constraint of the (d,p) pricing problem and R be the set of all dual variables associated with the constraints (3.1) and (3.2). We change the objective function of the dual problem to

$$\min \sum_{(d,p) \in D \times P} 100 y_{dp} + \sum_{r \in R} |D| r - \sum_{\substack{d \in D, j \in N, \\ u \in U and u \hat{W}_j(u)}} o_{d,j,u,\hat{u}} - \sum_{d \in D,\ u \in U} b_{d,u},$$

because this gave dual solution with lower dual convexity constraint solution and higher dual solution values for location variables in the pricing problems. This results in in pricing problem solutions wit higher optimal values. Through these changes, we were able to stabilize the convergence. However we were still not able to solve the LP relaxation in a reasonable time compared to the general LP solver of CPLEX. No instance from Section 3.3.3 was solved in less than 10 minutes. We suspect that it still converges to slowly because for most cases the solution of LP relaxations are very fractional.

At this point we stopped to research the branch-and-price approach for the MILP problem. We did not see a possibilty that the approach can be better than our branch-and-bound approach from Section 3.3. Nevertheless, the branch-and-price approach inspired an idea for another heuristic which uses the pricing problems. The heuristic has certain limitations that makes

it also unsuitable to use. However it gave us an idea fo the heuristic of the DMS problem with reliability and redundancy which provides feasible solutions with low objective values. Therefore, we still present the heuristic in the following subsection, even though it is not usable.

3.4.4 Heuristic II

In this subsection a second heuristic is proposed for the DMS problem with redundancy which uses the idea of Dantzig-Wolfe decomposition.

Let the objective function for the DMS problem with redundancy be

$$\min \sum_{i \in N} \sum_{\substack{u \in U^i, \\ m \in M^u}} c_{um} t_{ium} + \sum_{(i,j) \in A} c_{\text{cable}} x_{ij} \ell_{ij}, \tag{3.68}$$

where c_{cable} and c_{um} are objective coefficients.

To obtain a feasible solution for the MILP from solving the pricing problems, we use the following steps:

Let L_{iu} be the number of open ports of unit u at location i. Let $X \in \{0,1\}^{|N| \times |N|}$ and x_{ij} of X is 0 if arc (i,j) is not yet used in any subsystem and 1 otherwise. Also, let $T \in (U \cup \{0\})^{|N|}$ and t_i is $u \in U$ if unit type u is set by a subsystem and 0 otherwise. Lastly, we have to enforce constraint (3.15). For this let

$$\begin{aligned}
H_1 &:= \{i \in N \mid t_i = OVF \text{ and } |x_{*i}|_1 = 1\}, \\
H_> &:= \{i \in N \mid t_i = OVF \text{ and } |x_{*i}|_1 > 1\}, \\
B_j^> &= \{i \in N \mid (i,j) \in A, \sum_{j \in H_>} (X)_{ij}) > 0\} \text{ and} \\
B_j^1 &= \{i \in N \mid (i,j) \in A, \sum_{j \in H_1} (X)_{ij}) > 0\}.
\end{aligned}$$

Moreover choose appropriate $z_1, z_> \in \mathbb{R}_-$ with $z_1 < z_>$ and l. z_1, $z_>$ and l have to be chosen based on the parameters $c_{(OVF)m}$, $c_{(OCU)m}$ and c_{cable}. For example, a better objective value should be achieved by connecting a second OCU to an existing OVF rather than placing a new OVF and connecting a OCU to it or connecting a third OCU to an existing OVF. For all $(d,p) \in D \times P$:

1. Define objective function :

$$\begin{aligned}
v(dp) := &\sum_{i \in N} \sum_{\substack{u \in U^i, \\ m \in M^u}} (l c_{um} + 1) t_{ium}^{dp} + \sum_{\substack{(i,j) \in A \\ \forall u \in U^i \ \forall \hat{u} \in W^j(u)}} (l c_{\text{cable}} + 1) x_{iju\hat{u}}^{dp} \ell_{ij} \\
&+ \sum_{j \in H} \left(\sum_{i \in B_j^1, \hat{u} \in W_i(OVF)} z_1 x_{ij\hat{u}(OVF)} + \sum_{i \in B_j^>, \hat{u} \in W_i(OVF)} z_> x_{ij\hat{u}(OVF)} \right)
\end{aligned}$$

2. Update port constraints (3.35) in K_{dp} with :

$\forall i \in N$ with $t_i \neq 0$:

$$\sum_{\substack{\ell \in \{V^-(i,t_i) \mid (X)_{ij}=0\}, \\ \hat{u} \in U^\ell \mid t_i \in C(\hat{u})}} x_{\ell i \hat{u} t_i}^{dp} + \sum_{\substack{\ell \in \{V^+(i,u) \mid (X)_{ij}=0\}, \\ \hat{u} \in W^k(t_i)}} x_{i \ell t_i \hat{u}}^{dp} \leq \sum_{m \in M^{t_i}} t_{it_i m}^{dp} E_{t_i m} - |x_{*i}| - |x_{i*}| \tag{3.69}$$

3. Solve:

$$\begin{aligned} \min \quad & v(dp) \\ s.t. \quad & K_{dp}(t_{dp}, x_{dp}) \leq k_{dp} \\ & (t_{dp}, x_{dp}) \in \{0,1\}^{n_{dp}} \end{aligned}$$

4. Let (t^*_{dp}, x^*_{dp}) be the optimal solution. For all $\hat{p} \in P \backslash \{p\}$, add constraint

$$\sum_{u \in U^i, m \in M^u} t^{d\hat{p}}_{ium} = 0$$

to $K_{d\hat{p}}$, if $i \in N$ with $\sum_{u \in U^i, m \in M^u} t^{(dp)^*}_{ium} > 0$.

5. Update X, T, H_1, $H_>$, B^1_j and $B^>_j$ and go to next element in $D \times P$.

The heuristic obtains a feasible solution to the DMS with redundancy with certain limitations. One limitation is that lower bounds greater than 0 on the overall location variables cannot be used in the formulations. If we have for example a lower bound on the number of a certain model of a unit, we cannot translate this into a constraint of the pricing problems. Another limitation is that if we have an upper bound on the number of a specific model of a unit, we can add this constraint to the pricing problems. However we cannot ensure that we obtain a feasible solution from the heuristic, as one of the later pricing problems can be infeasible. These limitations make the heuristic unusable. However, it inspired the main idea for the heuristic of the DMS problem with reliability and redundancy in Section 4.2.

3.5 Discussion

In this section, we discuss the authenticity of the used instances and computational results and furthermore the practicability of our results. For the authenticity of the used instances, we have to mention that only artificial data was used due to confidentiality issues. However, it must be noted that the instances solved were built in collaboration with AIRBUS Group so that they were as real as possible. Moreover, the DMS problem with redundancy is just a simplification of the whole real problem because some aspects are not taken into account: for example, the reliability of the system. Nevertheless, having an exact method to solve the DMS problem with redundancy is already a useful tool for engineers in order to analyze the solutions obtained for small-scale configurations because so far only some internal heuristics were available to them.

In this context, we also have to consider the relationship between redundancy and reliability. For example, it is known that a 2-redundant system has a lower reliability than a 3-redundant system. Hence, the k-redundancy formulation can be used to build a heuristic for the problem with reliability by iterating k. By creating a k-redundant system that is reliability feasible (i.e., that meets the reliability threshold) and then a $(k-1)$-redundant system that is not reliability feasible, an aircraft engineer has two systems which can be used to build the needed end system. Therefore, the model can be used in the preliminary design phase of a network in order to enable a quick exploration of the design space for a better initialization of the detailed design phases.

Chapter 4

Reliability Models and Methods

Now that we have presented an MILP formulation and an exact solution method for the DMS problem with redundancy, we discuss how to consider reliability for the DMS problem. As we have already seen in Chapter 2, the calculation of reliability is very hard and computationally expensive even on its own. If reliability has to be considered in an optimization problem it becomes even more complex.

Optimization problems of safety-critical systems that consider redundancy or reliability are called redundancy allocation problems (RAPs) (see [28, 26]). RAPs involve the simultaneous selection of components and a system-level design configuration which can collectively meet all design constraints in order to optimize some objectives such as system cost or reliability. In particular, system reliability has received broad attention in the literature (see [53] for a review). For RAPs, the majority of research has been done in the direction of reliability with respect to subsystems and series-parallel systems (see [15] and [27]). However, some assumptions, such as independency of certain subsystems, were made in most cases to simplify the calculation of reliability. Research has also been conducted into RAPs of aircraft architecture systems. For fly-by-wire airliners, an architecture optimization of a flight-control system which considers the reliability of the system was proposed in [5] and the optimization of an actuation system with regards to redundancy and reliability was presented in [40]. In all of these cases, because of the difficulty of modelling and calculating reliability in complex systems, either independency was assumed for the calculation of the reliability, or a genetic algorithm [8] or colony algorithm [21] was used to design the system. Clearly, the DMS problem with redundancy and reliability is an RAP. Our objective is to make no assumption of independency or to use any approximations. An optimization model for aircraft architecture problems similar to the DMS is proposed in [20], but this uses an approximation for the reliability. As mentioned before, in Appendix A.1 we observe that approximations as used in [20] are not suitable for optimization.

We saw in Chapter 2 that the DMS is also a system where the calculation of reliability is computationally expensive and also non-linear. We would like now to formulate MINLP and MILP problems using the calculation method for the reliability from Chapter 2 and see if they are solvable.

In the following sections, we first propose MINLP and MILP formulations for the DMS problem with redundancy and reliability, and then discuss the current difficulties to solving it exactly in Section 4.1. As mentioned in the previous chapter, the branch-and-price approach helped to inspire the main idea for a heuristic of the DMS problem with redundancy and reliability which is introduced in Section 4.2. It is also based on the reliability calculation method from Section 2.2 and characteristics of the reliability calculation that were observed during the development of the heuristic.

4.1 MILP and MINLP Formulations

In this section we present MILP and MINLP formulations for the DMS problem with reliability and redundancy. Note that we only consider 2-redundancy and not k-redundancy as in Chapter 3.

Both formulations are based on the MILP formulation in Section 3.2 and all sets, parameters, constraints and decision variables can be used with only one small change. As mentioned before, we only require 2-redundancy and not k-redundancy and therefore we have to loosen the redundancy constraints. However more than two paths may be needed to achieve the necessary reliability. Therefore, we define $P = \{1, \ldots, \kappa\},\ \kappa \geq 3$. For both of these reliability formulations, we can use all constraints and variables from the redundancy formulation with two exception. We must relax the redundancy constraints (3.3) and (3.4): we force the first two paths to be disjoint to obtain the 2-redundancy and all other paths are allowed to share components and connections. This results in the following constraints (4.1)-(4.4) instead of (3.3) and (3.4).

$$\forall d \in D,\ \forall f \in F,\ \forall i \in N_f,\ \forall u \in U_f^i\ ,\ \forall \hat{f} \in F^u:$$
$$\sum_{m \in M^u} t_{ium} \geq t_{iu}^{d1f} + t_{iu}^{d2\hat{f}}, \tag{4.1}$$
$$\forall d \in D,\ \forall f \in F,\ \forall (i,j) \in A_f,\ \forall u \in U^i,\ \forall \hat{u} \in W^j(u):$$
$$x_{iju\hat{u}} \geq x_{iju\hat{u}}^{d1f} + x_{iju\hat{u}}^{d2f}, \tag{4.2}$$
$$\forall d \in D,\ \forall p \in P \backslash \{1,2\},\ \forall f \in F,\ \forall i \in N_f,\ \forall u \in U_f^i\ :$$
$$\sum_{m \in M^u} t_{ium} \geq t_{iu}^{dpf}, \tag{4.3}$$
$$\forall d \in D,\ \forall p \in P \backslash \{1,2\},\ \forall f \in F,\ \forall (i,j) \in A_f,\ \forall u \in U^i,\ \forall \hat{u} \in W^j(u):$$
$$x_{iju\hat{u}} \geq x_{iju\hat{u}}^{dpf}. \tag{4.4}$$

The rest of the MILP formulation for the DMS problem with redundancy will be used without changes and this provides us with all necessary architectural restriction and design constraints. We only have to add constraints and variables that are needed to consider the reliability. Before introducing these constraints and variables, we first present the reliability calculation formula from Section 2.2 applied to the DMS problem.

Let R be the event that the system does not fail, and $R_d,\ d \in D$, be the event that the system for door d does not fail. Also let $R_{dp},\ d \in D$ and $p \in P$, be event that the system of path p for door d does not fail. Let $z_d = |D|,\ z_p = |P|,\ \mathcal{R} = \{R_1, \ldots, R_{z_d}\},\ \mathcal{R}_d = \{R_{d1}, \ldots, R_{dz_p}\},\ d \in D$, and $z_{\text{all}} = \sum_{i=1}^{z_d} |\mathcal{R}_i|$. Given $k \in \{z_d, z_d + 1, \ldots, z_{\text{all}}\}$, let

$$C_k = \{\mathbf{E} = \{E_1, \ldots, E_k\} |\ E_u = R_{i(u)j(u)} \text{ for some } i(u) \in \{1, \ldots, n\},\ j(u) \in \{1, \ldots, t_{i(u)}\},$$
$$u = 1, \ldots, k, \{i(1), i(2), \ldots, i(k)\} = \{1, \ldots, n\} \text{ and } E_u \neq E_v,\ u, v \in \{1, \ldots, k\}\}.$$

Using this, we know from Proposition 1 that

$$P(R) = \sum_{k=n}^{m} \left((-1)^{k-n} \sum_{\mathbf{E} \in C_k} P(\mathbf{E}) \right). \tag{4.5}$$

However it is usually the case that we need to consider reliability with a value of around $1 - 10^{-9}$ which creates numerical issues in optimization models. Therefore, we want to look at the probability of the whole system failing, which is $1 - P(R)$. The probability of the whole system failing is around 10^{-9} and it is easier to scale it to a number that does not give numerical issues. The probability of the whole system failing can be written as

$$\begin{aligned} 1 - P(R) &= 1 - \sum_{k=n}^{m} \left((-1)^{k-n} \sum_{\mathbf{E} \in C_k} P(\mathbf{E}) \right) \\ &= \sum_{k=n}^{m} \left((-1)^{k-n} \sum_{\mathbf{E} \in C_k} 1 \right) - \sum_{k=n}^{m} \left((-1)^{k-n} \sum_{\mathbf{E} \in C_k} P(\mathbf{E}) \right) \end{aligned}$$

$$= \sum_{k=n}^{m} \left((-1)^{k-n} \sum_{\mathbf{E} \in C_k} (1 - P(\mathbf{E})) \right). \tag{4.6}$$

In the following, we present decision variables and constraints that are used in both the MILP and the MINLP formulation.

4.1.1 Variables and Constraints for Reliability

From (4.6) we can see that we need to calculate each $P(\mathbf{E})$. To implement this in the formulation, we need variables $t_{\mathbf{E}ium}$ and $x_{\mathbf{E}ij}$ with $i \in N,\ u \in N_i,\ m \in M^u,\ \ j \in V^+(i)$, $\mathbf{E} \in C_k,\ k \in \{z_d, \ldots, z_{\text{all}}\}$, and variables $t_{\mathbf{E}um}$ and $x_{\mathbf{E}}$ with $\mathbf{E} \in C_k,\ k \in \{z_d, \ldots, z_{\text{all}}\}$. For each $\mathbf{E} \in C_k$, these variables represent the subnetwork which is composed of all implementations in $\mathbf{E}$. For these variables to represent the subnetworks, constraints are also needed to synchronize these new variables with the existing location variables of the overall system and the subnetworks of $(d, f, p) \in D \times F \times P$.

Let $k \in \{z_d, \ldots, z_{\text{all}}\}$ and $\mathbf{E} \in C_k$, and $I(\mathbf{E}) = \{(d, p) \mid \exists i \in \{1, \ldots, k\} : E_i = R_{dp}\}$. For z_d, we have the synchronization constraints:

$\forall \mathbf{E} \in C_{z_d},\ \forall (d,p) \in I(\mathbf{E}),\ \forall f \in F,\ \forall i \in N_f,\ \forall u \in U_f^i :$

$$\sum_{m \in M^u} t_{\mathbf{E}ium} \geq t_{iu}^{dfp}, \tag{4.7}$$

$\forall \mathbf{E} \in C_{z_d},\ \forall i \in N,\ \forall u \in U^i :$

$$\sum_{m \in M^u} t_{\mathbf{E}ium} \leq \sum_{\substack{\forall (d,p) \in I(\mathbf{E}), \\ \forall f \in F^u}} t_{iu}^{dfp}, \tag{4.8}$$

$\forall \mathbf{E} \in C_{z_d},\ \forall i \in N,\ \forall u \in U^i,\ \forall m \in M^u :$

$$t_{ium} \geq t_{\mathbf{E}ium}, \tag{4.9}$$

$\forall \mathbf{E} \in C_{z_d},\ \forall (d,p) \in I(\mathbf{E}),\ \forall f \in F,\ \forall (i,j) \in A_f :$

$$x_{\mathbf{E}ij} \geq \sum_{(u,\hat{u}) \in U_f^i \times W_f^j(u)} x_{iju\hat{u}}^{dfp}. \tag{4.10}$$

For $k > z_d$ we know the following:

$$\forall \mathbf{E} \in C_k\ \exists \hat{\mathbf{E}}_{1,\mathbf{E}}, \hat{\mathbf{E}}_{2,\mathbf{E}} \in C_{k-1} :\ I(\mathbf{E}) = I(\hat{\mathbf{E}}_{1,\mathbf{E}}) \cup I(\hat{\mathbf{E}}_{2,\mathbf{E}}).$$

Therefore, for $k > z_d$ we can use the following synchronization constraints.

$\forall k \in \{z_d + 1, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k,\ \forall i \in N,\ \forall u \in U,\ \forall m \in M^u :$

$$t_{\mathbf{E}ium} \geq t_{\hat{\mathbf{E}}_{1,\mathbf{E}}ium}, \tag{4.11}$$

$$t_{\mathbf{E}ium} \geq t_{\hat{\mathbf{E}}_{2,\mathbf{E}}ium}, \tag{4.12}$$

$$t_{\mathbf{E}ium} \leq t_{\hat{\mathbf{E}}_{1,\mathbf{E}}ium} + t_{\hat{\mathbf{E}}_{2,\mathbf{E}}ium}. \tag{4.13}$$

If we were to use the same synchronization constraints for $k > z_d$ as for $k = z_d$, many more constraints would be needed.

The following provides a strengthening of constraints (4.13). Let $\mathbf{E} \in C_k,\ k > z_d$ and for $i \in \{1, \ldots, n\}$ define:

$$\mathbf{E}_i = \{E \in \mathbf{E} \mid \exists j \in \{1, \ldots, t_i\} : E = R_{ij}\}$$

and also define $\alpha_{\mathbf{E}} := \max_{i \in \{1,\ldots,n\}} \{|\mathbf{E}_i|\}$. Furthermore define $\alpha_k = \max_{\mathbf{E} \in C_k} \{a_{\mathbf{E}}\}$. Let $k_1 = \lceil \frac{k}{2} \rceil$

and $k_2 = \lfloor \frac{k}{2} \rfloor$. We then have that if $\alpha_{k_1} + \alpha_{k_2} \geq \alpha_{\mathbf{E}}$ then

$$\exists\ \hat{\mathbf{E}}_{\mathbf{k_1},\mathbf{E}} \in C_{k_1} \text{ and } \exists\ \hat{\mathbf{E}}_{\mathbf{k_2},\mathbf{E}} \in C_{k_2} \text{ with } I(\mathbf{E}) = I(\hat{\mathbf{E}}_{\mathbf{k_1},\mathbf{E}}) \cup I(\hat{\mathbf{E}}_{\mathbf{k_2},\mathbf{E}}).$$

Otherwise

$$\exists\ \hat{\mathbf{E}}_{\mathbf{k_1},\mathbf{E}} \in C_{k_1} \text{ and } \exists\ \hat{\mathbf{E}}_{\mathbf{k_2},\mathbf{E}} \in C_{k_2+1} \text{ with } I(\mathbf{E}) = I(\hat{\mathbf{E}}_{\mathbf{k_1},\mathbf{E}}) \cup I(\hat{\mathbf{E}}_{\mathbf{k_2},\mathbf{E}}).$$

The following constraint is stronger than (4.13):

$$\forall k \in \{z_d + 1, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k,\ \forall i \in N,\ \forall u \in U,\ \forall m \in M^u :$$
$$t_{\mathbf{E}ium} \leq t_{\hat{\mathbf{E}}_{k_1,\mathbf{E}}ium} + t_{\hat{\mathbf{E}}_{k_2,\mathbf{E}}ium}. \tag{4.14}$$

To calculate each $P(\mathbf{E})$ we need the total number of components and connections of the corresponding subnetwork. These are given through the constraints

$$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k,\ \forall u \in U,\ \forall m \in M^u :$$
$$t_{\mathbf{E}um} = \sum_{i \in N^u} t_{\mathbf{E}ium} \tag{4.15}$$

and $\forall k \in \{z_d, \ldots, m\},\ \forall E \in C_k$:

$$x_{\mathbf{E}} = \sum_{(i,j) \in A} x_{\mathbf{E}ij}. \tag{4.16}$$

With these variables and constraints, we could calculate $P(\mathbf{E})$ easily with

$$P(\mathbf{E}) = \prod_{u \in U, m \in M^u} (1 - f_{um})^{t_{\mathbf{E}um}} \times (1 - f_{\text{cable}})^{x_{\mathbf{E}}} \tag{4.17}$$

where f_{cable} is the failure rate of a cable and f_{um} is the failure rate of a model m of a component u. It is easy to see that this is a non-convex (non-linear) function.

To finish our formulation, we only need to formulate (4.17). We will see that the MILP and MINLP formulations require different variables and constraints needed for this. We first propose the last part of the MINLP formulation.

4.1.2 MILNP Formulation

We need variables $r_{\mathbf{E}},\ \mathbf{E} \in C_k,\ k \in \{z_d, \ldots, z_{\text{all}}\}$ to include (4.17) in the MINLP formulation. These will represent the negative logarithm of $P(\mathbf{E})$. We consider the negative of the logarithm, because the logarithm turns a value that is less than, but close to 1 into a negative variable that is close to 0. Furthermore, the failure rates f_{um} and f_{cable} are very small (usually less than $< 10^{-5}$) and hence the values of $r_{\mathbf{E}}$ are also very small, so we scale $r_{\mathbf{E}}$ by 10^{ℓ}, for some $\ell \geq 1$ which depends on the average value of the failure rates. We can then include (4.17) as

$$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k :$$
$$-r_{\mathbf{E}} = \sum_{\substack{u \in U, \\ m \in M^u}} t_{\mathbf{E}um} \left(10^{\ell} \ln(1 - f_{um})\right) + x_{\mathbf{E}} \left(10^{\ell} \ln(1 - f_{\text{cable}})\right). \tag{4.18}$$

To include the probability $1 - P(R)$ of the whole system failing, we need a variable r_{total} which gives the probability of the whole system failing, also scaled by 10^{ℓ}. To include it in the formulation, we use the constraint

$$r_{\text{total}} = \sum_{k=z_d}^{z_{\text{all}}} \left((-1)^{k - z_d} \sum_{\mathbf{E} \in C_k} p_{\mathbf{E}} \right) \tag{4.19}$$

where $p_{\mathbf{E}}$ is $10^l \left(1 - e^{-\frac{r_{\mathbf{E}}}{10^l}}\right)$.

To include $p_{\mathbf{E}}$, we have the constraints $\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k$:

$$(-1)^{k-z_d} = \quad 1 \Rightarrow -p_{\mathbf{E}} + 10^l \left(1 - e^{-\frac{r_{\mathbf{E}}}{10^\ell}}\right) \leq 0, \tag{4.20}$$

$$(-1)^{k-z_d} = -1 \Rightarrow \quad p_{\mathbf{E}} - 10^l \left(1 - e^{-\frac{r_{\mathbf{E}}}{10^\ell}}\right) \leq 0. \tag{4.21}$$

Observe that (4.20) is a concave function, while (4.21) is a convex function. We now explain a linear relaxation that can be used for the constraints (4.20) and (4.21).

To initially construct the best possible linear relaxation, bounds on the variables $p_{\mathbf{E}}$ and $r_{\mathbf{E}}$ are needed. The bounds for both variables are closely related and dependent because $p_{\mathbf{E}}$ is the result of an equation calculation that only involves $r_{\mathbf{E}}$. To obtain the best possible bounds, we solve the problem without (4.20) and (4.21) several times. For every $k \in \{z_d, \ldots, z_{\text{all}}\}$, we choose an element $\mathbf{E} \in C_k$ and minimize and maximize once with the objective function $r_{\mathbf{E}}$. This provides the bounds for $r_{\mathbf{E}}$ and consequently p_E. For the convex constraint (4.21), we use the outer approximation (4.23) and (4.24) as the linear relaxation and use the known convex envelope (4.22) for the concave constraint (4.20). Let us define a function $g : \mathbb{R} \to \mathbb{R}$ by $g(x) = 10^l \left(1 - e^{-\frac{x}{10^\ell}}\right)$. Let $r_{\mathbf{E}}^{lb}$ and $r_{\mathbf{E}}^{ub}$ be the lower and upper bounds of $r_{\mathbf{E}}$, respectively, then:
$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k$:

$$(-1)^{k-z_d} = -1 \Rightarrow f\left(r_{\mathbf{E}}^{lb}\right) + \frac{g\left(r_{\mathbf{E}}^{ub}\right) - g\left(r_{\mathbf{E}}^{lb}\right)}{r_{\mathbf{E}}^{ub} - r_{\mathbf{E}}^{lb}} \left(r_{\mathbf{E}} - r_{\mathbf{E}}^{lb}\right) - p_{\mathbf{E}} \quad \leq 0, \tag{4.22}$$

$$(-1)^{k-z_d} = \quad 1 \Rightarrow p_{\mathbf{E}} - r_{\mathbf{E}} \exp\left(-\frac{r_{\mathbf{E}}^{ub}}{10^l}\right) - p_{\mathbf{E}}^{ub} + r_{\mathbf{E}}^{ub} \exp\left(-\frac{r_{\mathbf{E}}^{ub}}{10^l}\right) \leq 0, \tag{4.23}$$

$$(-1)^{k-z_d} = \quad 1 \Rightarrow p_{\mathbf{E}} - r_{\mathbf{E}} \exp\left(-\frac{r_{\mathbf{E}}^{lb}}{10^l}\right) - p_{\mathbf{E}}^{lb} + r_{\mathbf{E}}^{lb} \exp\left(-\frac{r_{\mathbf{E}}^{lb}}{10^l}\right) \leq 0. \tag{4.24}$$

Another possible way of calculating $p_{\mathbf{E}}$ is by using the power series representation of the exponential function:
$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k$:

$$\begin{aligned} &(-1)^{k-z_d} = 1 \Rightarrow \\ &-p_{\mathbf{E}} \geq r_{\mathbf{E}} - \frac{1}{2 \times 10^\ell} r_{\mathbf{E}}^2 + \frac{1}{6 \times 10^{2\ell}} r_{\mathbf{E}}^3 - \frac{1}{4! \times 10^{3\ell}} r_{\mathbf{E}}^4 + \frac{1}{5! \times 10^{4\ell}} r_{\mathbf{E}}^5, \end{aligned} \tag{4.25}$$

$$\begin{aligned} &(-1)^{k-z_d} = -1 \Rightarrow \\ &p_{\mathbf{E}} \geq -r_{\mathbf{E}} + \frac{1}{2 \times 10^\ell} r_{\mathbf{E}}^2 - \frac{1}{6 \times 10^{2\ell}} r_{\mathbf{E}}^3 + \frac{1}{4! \times 10^{3\ell}} r_{\mathbf{E}}^4. \end{aligned} \tag{4.26}$$

Observe that (4.25) is a concave constraint, while (4.26) is a convex constraint.

We ran tests on these different non-convex mixed integer programming problems for both two and three doors with different MINLP solvers available on the NEOS server. Unfortunately, even when using (4.22) the problem is not tractable and therefore not solvable in a reasonable amount of time. It is not tractable mainly because of the linear part is very large for non-convex MINLPs as seen in Chapter 3 and linear relaxation values of the reliability variables are very poor and fractional. All this slows the convergence considerably.

4.1.3 MILP Formulation

As a alternative to the MINLP formulations for (4.17), we have an MILP formulation. I consists in enumerating all possible combinations of the subsystems, calculating the reliability beforehand and then using them in the formulation. To verify that this approach is computationally possible, we first consider a simpler problem where only units are used for the reliability calculation as this reduces the number of combinations.

For the MILP formulation, we also need the decision variables r_{total} and $r_{\mathbf{E}}$, $\mathbf{E} \in C_k$, $k \in \{z_d, \ldots, z_{\text{all}}\}$, with

$$r_{\text{total}} = \sum_{k=z_d}^{z_{\text{all}}} \left((-1)^{k-n} \sum_{\mathbf{E} \in C_k} r_{\mathbf{E}} \right). \tag{4.27}$$

To create linear constraints to calculate the values of $r_{\mathbf{E}}$, we need all the possible combinations of units. We need the lower and upper bounds t^{lb}_{kmu} and t^{ub}_{kmu}, $u \in U$, $m \in M^u$, of variables $t_{\mathbf{E}um}$, to enumerate all these possible combinations of units for an element $\mathbf{E} \in C_k$, $k \in \{z_d, \ldots, z_{\text{all}}\}$. These can be calculated by minimizing and maximizing the redundancy MILP formulation with all constraints from Subsection 4.1.1 and with $t_{\mathbf{E}Um}$ as the objective, e.g. $\max\{t_{\mathbf{E}um}\}$, $\mathbf{E} \in C_k$, $u \in U$, $m \in M^u$. Since all elements $\mathbf{E} \in C_k$ have the same bounds at first, we only need to run a minimization and maximization for one element $\mathbf{E} \in C_k$ per $k \in \{z_d, \ldots, z_{\text{all}}\}$.

Let $\hat{m} = \max_{u \in U} M^u$. The set of all possible combinations of units for an element $\mathbf{E} \in C_k$, $k \in \{z_d, \ldots, z_{\text{all}}\}$ is

$$K^{\mathbf{E}}_k = \left\{ X \in \mathbb{Z}^{|U| \times \hat{m}} \;\middle|\; t^{lb}_{kij} \leq X_{ij} \leq t^{ub}_{kij}, i \in U,\ j \in M^i \text{ and } t^{lb}_{ku} \leq \sum_{j \in M^i} X_{ij} \leq t^{ub}_{ku}, i \in U, \right.$$
$$\left. \text{and } X_{ij} = -1, i \in U, |M^i| < j \leq \hat{m} \right\}.$$

We define decision variables $d_{\mathbf{E}uma} \in \{0,1\}$, $\mathbf{E} \in C_k$, $k, \in \{z_d, \ldots, z_{\text{all}}\}$, $u \in U$, $m \in M^u$ and $a \in \{t^{lb}_{kum}, \ldots, t^{ub}_{kum}\}$. Variable $d_{\mathbf{E}uma}$ takes value 1 if $t_{\mathbf{E}um} = a$, and 0 otherwise. Furthermore, let us define variables $h_{\mathbf{E}X} \in \{0,1\}$, $k \in \{z_d, \ldots, z_{\text{all}}\}$, $\mathbf{E} \in C_k$, and $X \in K^{\mathbf{E}}_k$. The variable $h_{\mathbf{E}X}$ is 1 if $\sum_{i \in U,\ j \in M^i} d_{Eijx_{ij}} = \sum_{i \in U} |M^i|$, and 0 otherwise. To achieve this, we need the following constraints:

$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k,\ \forall u \in U,\ \forall m \in M^u :$

$$t_{\mathbf{E}um} = \sum_{a \in \{t^{lb}_{kum}, \ldots, t^{ub}_{kum}\}} a d_{\mathbf{E}uma}, \tag{4.28}$$

$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k,\ \forall u \in U,\ \forall m \in M^u :$

$$\sum_{a \in \{t^{lb}_{kum}, \ldots, t^{ub}_{kum}\}} d_{\mathbf{E}uma} = 1, \tag{4.29}$$

$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k,\ \forall u \in U,\ \forall m \in M^u\ \forall a \in \{t^{lb}_{kum}, \ldots, t^{ub}_{kum}\} :$

$$a - t_{\mathbf{E}um} = o^a_+ - o^a_-, \tag{4.30}$$

$$\sum_{a \in \{t^{lb}_{kum}, \ldots, t^{ub}_{kum}\} \setminus \{\hat{a}\}} d_{\mathbf{E}uma} \leq o^a_+ + o^a_-, \tag{4.31}$$

$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k,\ \forall u \in U,\ \forall m \in M^u,\ \forall a \in \{t^{lb}_{kum}, \ldots, t^{ub}_{kum}\} :$

$$d_{\mathbf{E}uma} = \sum_{X \in K^E_k \ |\ x_{ij} = a} h_{\mathbf{E}X}, \tag{4.32}$$

$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k$:

$$\sum_{X \in K_k^{\mathbf{E}}} h_{\mathbf{E}X} = 1. \tag{4.33}$$

Instead of (4.32), we could use the following constraints (4.34), although there are many more of them:

$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k,\ \forall X \in K_k^{\mathbf{E}}$:

$$\sum_{i \in U,\ j \in M^i} d_{\mathbf{E}ijx_{ij}} - \sum_{i \in U} |M^i| + 1 \leq h_{\mathbf{E}X}. \tag{4.34}$$

Let f_{um} be the failure rate of a model m of unit u. We then have that the reliability of an element $\mathbf{E}$ with combination of units X is $r_{\mathbf{E}X} = 1 - \left(\prod_{i \in U, m \in M}(1 - f_{um})^{x_{ij}}\right)$. To have the right value of $r_{\mathbf{E}}$, we need the constraints

$\forall k \in \{z_d, \ldots, z_{\text{all}}\},\ \forall \mathbf{E} \in C_k$:

$$(-1)^{k - z_d} = -1 \Rightarrow r_{\mathbf{E}} \leq \sum_{X \in K_k^{\mathbf{E}}} r_{\mathbf{E}X} h_{\mathbf{E}X}, \tag{4.35}$$

$$(-1)^{k - z_d} = \ 1 \ \Rightarrow r_{\mathbf{E}} \geq \sum_{X \in K_k^{\mathbf{E}}} r_{\mathbf{E}X} h_{\mathbf{E}X}. \tag{4.36}$$

However the number of all possible combinations of units is huge. We show this fact with an example of the DMS with 2 doors and 3 possible paths. The Tables C.1, C.2, C.3, C.4 and C.5 in Appendix C.1 give the lower and upper bounds with the number of models and possible combinations for all $k \in \{2, \ldots, 6\}$. We see that for all k there are $5,346,096$ possible combinations and with $|C_2| = 9,\ |C_3| = 18,\ |C_4| = 15,\ |C_5| = 6$ and $|C_6| = 1$, we have $27,586,608$ $h_{\mathbf{E}X}$ variables in the model.

We ran computational tests on a smaller example with only one model per unit type and this resulted in a problem with approximately 100.000 variables. We saw that the linear relaxation for the reliability value r_{total} is very poor because of fractional h_{EX} values close to 0. Due to this poor linear relaxation value, we were not able to solve the problem, unless the reliability threshold was achieved through simple 2-redundancy. However such solutions are not meaningful. Unfortunately, we were not able to strengthen the linear relaxation and could not solve the problem.

Another approach was to use column generation/branch-and-price to solve the problem and add the h_{EX} variables to the problem slowly and not start with all of them from the beginning. This is not possible, because we would need a non-linear non-convex objective function or a non-linear non-convex constraint in the pricing problems or master problem, respectively. This would result in the same problems as with the previous MINLP formulations.

Having not been able to solve the DMS problem with redundancy and reliability exactly, we now wish to develop a heuristic that offers good feasible solutions inspired by what has been discussed in Section 3.4.

4.2 Heuristic for the DMS Problem with Redundancy and Reliability

In this section, we introduce a new heuristic for the DMS problem with redundancy and reliability that gives good feasible solutions for problems where the reliability threshold is a constraint and not an objective. It is important to note that this is a heuristic that is valid for a DMS with at most three paths. If we wish to consider more paths, changes have to be made to the heuristic.

As mentioned before, the heuristic is based on the branch-and-price approach. It therefore uses an MILP formulation for the master problem and pricing problems. We also use the redundancy

MILP formulation for the DMS problem with redundancy as a basis for the master problem and pricing problems.

As mentioned in Section 3.1, the design for the DMS network can be seen as a sub-graph G' of G where G is a directed graph that represents all possible locations of units and cable connections. We will refer to the DMS network as the *overall system*. However, to formulate our problem we cannot only look at the overall system. We have to look at every function which must be at least 2-redundant in the network system. As mentioned before, functions are sets of units and cable connections that fulfill a certain task. Hence, we can also see every function as a sub-graph of G'. Every functions is implemented multiple times in the network systems and every implementation is represented by a path. If we look at the finished redundancy formulation, we then have flow and architectural constraints for every path of a function of a door. Furthermore, we have constraints that synchronize the subsystems with the overall system. Figure 4.1 shows the constraint matrix of the redundancy formulation for 2 doors and 3 paths.

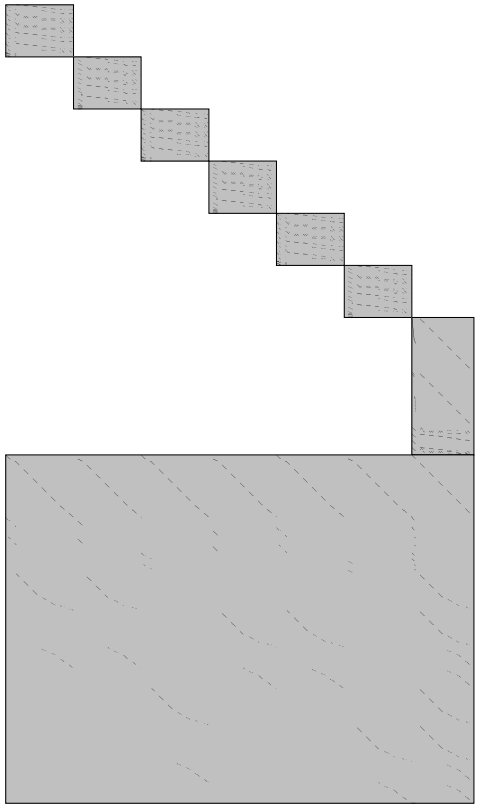

Figure 4.1: Example constraint matrix for 2 doors and 3 paths.

The matrix shows the structure needed for a Dantzig-Wolfe decomposition where we have a block of connecting constraints or, in our case, synchronization constraints and several independent sub-matrices that represent the paths of both functions for a door. Therefore, we can split the problem into a master problem which includes synchronization constraints and the architectural constraints for the overall system, and pricing problems which include flow and architectural constraints of the functions. It has to mentioned here that a pricing problem covers a path for each function of a door. We combine both functions into one pricing problem, because the functions are connected, as mentioned before, and therefore have to be considered together for reliability calculations. We therefore have for example 6 pricing problems for a DMS network with 2 doors and 3 paths.

To consider the reliability in our heuristic and to obtain good solutions which meet the reliability threshold, we have to add additional constraints to the master and sub-problems. Since

we cannot directly implement the reliability of the subsystems and overall system, we use the following five characteristics of the system reliability which help us to obtain solutions with high reliability. These characteristics were observed during computational tests on the reliability calculation of the DMS. They are not proven to be true in general, but helped us to obtain better solutions. In Appendix C.4, we give proofs for a few simplifications of these characteristics.

(a) The first characteristic is that the fewer the units or connections are shared between the different paths of a function, the higher the reliability. Let us assume that we have three paths and two of these are disjoint. This means that the third path is allowed to share units and connections with the other two paths. We observed that the fewer units and connections the third paths shares with the other two, the higher the reliability of the overall system.

(b) Let there be n paths of a function and suppose that the ith path does not share all units or connections with the other paths. Furthermore, let the sum of the failure rates of the units and connections of the different paths be fixed as well as the number of units and connections. The second characteristic is that the higher the sum of the failure rates of the units and connections that the ith path does not share with the other paths, the higher the reliability of the overall system. This is the case since the ith path becomes more disjoint from the other paths and this always increases the reliability of the overall system.

(c) Furthermore, we observed that if the number of units, the number of cables and the sum of failure rates for a path is low, then in most cases the reliability of the overall system is high.

(d) For the fourth characteristic, let us assume that we have three paths. Two of the paths have to be disjoint to obtain a redundant system. The third path can share units and connections with either of the two paths. We observe that if the third path shares approximately the same sum of failure rates with the first as with the second path, the reliability is higher. Possible explanation for this observation is that it the reliability is higher the less reliant the third path is on either of the two other paths.

(e) Finally, let us assume that we have a system where the number of shared units and connections is fixed. We then know that the lower the sum of the failure rates of the overall system, the higher the reliability of the overall system.

Using the idea of the branch-and-price approach, these reliability characteristics and the redundancy MILP formulation, we can build the following heuristic that obtains good feasible solutions for the DMS problem with redundancy and reliability.

The rest of this section is organized as follows: first, the MILP formulation for the master problem and subproblems is given in Subsection 4.2.1. It must be noted that what we previously called pricing problems, we now call subproblems as the idea of pricing is not used in what follows. The heuristic algorithm is presented in Subsection 4.2.2. A computational study is presented in Subsection 4.2.3 and shows the performance of the heuristic. Since there is no exact solution method, we can only give computational times and the range of the objective values, and no comparisons are possible. The instances used in the computational study range from two doors to five doors with different objectives and reliability thresholds.

4.2.1 Master and Subproblem Formulations

For our heuristic, we build a master problem for the overall system and subproblems for the different subsystems in the DMS. This builds on the redundancy MILP formulation and the idea of the Dantzig-Wolfe decomposition. The master problem gives us the architecture of the overall system and synchronization constraints between the overall system and the subsystems. The subproblems are used to obtain the architecture of the subsystems. The master problem is used as a verification tool so that the solutions of the subproblem build a feasible DMS network, and it is also used to optimize the real objective. For the formulations of the master problem

and the subproblems, we define the same sets and parameters as in the redundancy formulation. Descriptions of these sets and parameters can be found in Subsections 3.2.1 and **??**, respectively. The only additional parameters that we need are the failure rates of the different models of units and cable connections:

- f_{um}, the failure rate of the model m of unit type $u \in U$; and
- f_{cable}, the failure rate of a connection.

We first introduce the MILP formulation of the master problem.

4.2.1.1 Formulation of the master problem

The master problem is comprised of synchronization constraints between the overall system and the subsystems, and architectural constraints for the overall system. Before presenting the constraints, we first define the decision variables needed for the formulation.

We need decision variables for the overall system and each of the subsystems which tells us the location at which a unit is installed and which cable connections are used. For the overall system, we have the following decision variables:

- $t_{ium} \in \{0,1\},\ i \in N,\ u \in U^i,\ m \in M^u$, a binary variable which takes value 1 if unit type u and model m is set at location i, and takes value 0 otherwise; and
- $x_{iju\hat{u}} \in \{0,1\},\ (i,j) \in A, u \in U^i,\ \hat{u} \in W^j(u)$, a binary variable which takes value 1 if locations i and j are connected and units u and $\hat{u}$ are installed at i and j, respectively, and takes value 0 otherwise.

The same kind of decision variable sets are defined for each subsystem $(d,p) \in D \times P$:

- $t_{iu}^{dp} \in \{0,1\},\ i \in N,\ u \in U^i$, a binary variable which takes value 1 if a unit of type u is installed at location i , and takes value 0 otherwise; and
- $x_{iju\hat{u}}^{dp} \in \{0,1\},\ (i,j) \in A,\ u \in U^i,\ \hat{u} \in W^j(u)$, a binary variable which takes value 1 if locations i and j are connected and unit u and $\hat{u}$ are installed at i and j, respectively, and takes value 0 otherwise.

As can be seen, we only have a differentiation of the model type included in the overall system decision variables. This is the case because we must include the parameters associated to the models only in the constraints for the overall system.

Lastly, we need the decision variables $r^{dp},\ d \in D$ and $p \in P$, which represent the sum of the failure rates in the different (d,p) subsystems. A path of $d \in D$ covers both functions as they are connected.

With these decision variables, sets and parameters, we can present the constraints of the master problem. We need synchronization constraints which synchronize the decision variables of the overall system and the subsystems. For example, if a unit or cable connection is installed in the overall system, there has to be a subsystem which uses it. Also, the converse is true: if a unit or cable connection is used for a subsystem, it has to be installed in the overall system. Constraints (4.37) to (4.42) force this synchronization. As we only consider a system with at most three paths and 2-redundancy for this heuristic, we have three different sets of synchronization constraints.

First, we have constraints (4.37) and (4.38) that impose that the overall system uses a unit or cable connection, if it is used in a third path of a door.

$\forall d \in D,\ \forall i \in N\ \forall u \in U^i$

$$t_{iu}^{d3} \leq \sum_{m \in M^u} t_{ium}, \tag{4.37}$$

$\forall (i,j) \in A \; u \in U^i, \hat{u} \in W^j(u) \; :$

$$x^{d3}_{iju\hat{u}} \leq x_{iju\hat{u}}. \tag{4.38}$$

The second set of constraints, which include (4.39) and (4.40), impose the same for the first two paths. Additionally, they also impose the redundancy through the binary condition of the decision variables.

$\forall d \in D, \; \forall i \in N \; \forall u \in U^i$

$$t^{d1}_{iu} + t^{d2}_{iu} \leq \sum_{m \in M^u} t_{ium}, \tag{4.39}$$

$\forall (i,j) \in A \; u \in U^i, \hat{u} \in W^j(u) \; :$

$$x^{d1}_{iju\hat{u}} + x^{d2}_{iju\hat{u}} \leq x_{iju\hat{u}}. \tag{4.40}$$

Lastly, constraints (4.41) and (4.42) impose the reverse. That is, if a unit is used in the overall system, then it has to be used in at least one subsystem:

$\forall i \in N \; \forall u \in U^i$

$$\sum_{m \in M^u} t_{ium} \leq \sum_{d \in D, \; p \in P} t^{dp}_{iu}, \tag{4.41}$$

$\forall (i,j) \in A \; \forall u \in U^i \; \forall \hat{u} \in W^j(u) :$

$$x_{iju\hat{u}} \leq \sum_{d \in D, \; p \in P} x^{dp}_{iju\hat{u}}. \tag{4.42}$$

With these constraints, we have synchronized the two levels of decision variables. The following constraints are architectural constraints for the overall system.

Constraint (4.43) imposes that the number of available ports for connections at a location is not exceeded:

$\forall i \in N, \; \forall u \in U^i :$

$$\sum_{\substack{\ell \in V^-(i,u), \\ \hat{u} \in U^\ell \; | \; u \in C(\hat{u})}} x_{\ell i \hat{u} u} + \sum_{\substack{\ell \in V^+(i,u), \\ \hat{u} \in W^k(u)}} x_{i \ell u \hat{u}} \leq \sum_{m \in M^u} t_{ium} E_{um}. \tag{4.43}$$

Additionally, constraint (4.44) is used to enforce that at most one unit is installed at any location.

$\forall i \in N$:

$$\sum_{\substack{u \in U^i, \\ m \in M^u}} t_{ium} \leq 1. \tag{4.44}$$

The formulation described above could be used for several different network systems. The following constraint is specific to the DMS problem and is a safety requirement. The DMS always has two particular unit types: the outflow valve (OVF) and the outflow valve control unit (OCU). We require that every OVF has at least one connection from two different OCUs in the overall system. This is imposed through constraint (4.45):

$\forall i \in N^{OVF}$:

$$2 \sum_{m \in M^{OVF}} x_{i(OVF)m} \leq \sum_{j \in N^{OCU} \cap V^-(i)} x_{ji(OCU)(OVF)}. \tag{4.45}$$

With constraint (4.45), we now have all the synchronization and architectural constraints that are required. The following constraints and decision variables are related to reliability. To consider reliability, we need constraints that give us the value of $r^{dp},\ d \in D$ and $p \in P$, which represent the sum of the failure rates in the different (d,p) subsystems. As mentioned before, we solve the master problem after we have solved the subproblems, and to do this we use the solutions of the subproblems. For the formulation of the following constraints, we use these solutions and define the following sets for each subproblem of the subsystems $(d,p) \in D \times P$:

$$H^l_{(d,p)} = \{(i,u) \in N \times U \mid \text{unit } u \text{ is set at position } i \text{ for the subsystem } (d,p)\},$$
$$H^c_{(d,p)} = \{(i,j) \in A \mid \text{cable connection from } i \text{ to } j \text{ is used for the subsystem } (d,p)\}.$$

We can now define the following failure rate constraints:

$\forall d \in D,\ \forall p \in P$:

$$\sum_{\substack{(i,u) \in H^l_{(d,p)}, \\ m \in M^u}} f_{um} t_{ium} + \sum_{\substack{(i,j) \in H^c_{(d,p)}, \\ u \in U^i \ \hat{u} \in W^j(u)}} f_{cable} x_{iju\hat{u}} = r^{dp}. \tag{4.46}$$

With all constraints and decision variables defined, we now address the objective function of the master problem, for which there are many options. For example, we might wish to minimize the cost or the weight of the overall system. We could also try to include the power consumption into the minimization. For our heuristic, we assume that the objective is based on parameters of the different units and cables and that it is linear.

Let w_{um} and w_c be the objective coefficients of the decision variable types t_{ium} and $x_{iju\hat{u}}$, respectively. The objective function would be:

$$\min \sum_{i \in N,\ u \in U^i,\ m \in M^u} w_{um} t_{ium} + w_c \sum_{\substack{(i,j) \in A, \\ u \in U^i \ \hat{u} \in W^j(u)}} \ell_{ij} x_{iju\hat{u}} \tag{4.47}$$

where ℓ_{ij} is the distance of the connection between locations i and j, where $i,j \in N$. However, when solving the heuristic, it is not efficient to use this as the objective function and has to be included in the constraints. It is not efficient as it does not improve the reliability of the DMS. Therefore, we need a decision variable o^{total} which represents the objective function value of (4.47). This can be done by using the following constraint (4.48).

$$\sum_{i \in N,\ u \in U^i,\ m \in M^u} w_{um} t_{ium} + w_c \sum_{\substack{(i,j) \in A, \\ u \in U^i \ \hat{u} \in W^j(u)}} \ell_{ij} x_{iju\hat{u}} = o^{total}. \tag{4.48}$$

In our heuristic, we minimize the sum of failure rates of all subsystems. This goes back to reliability characteristic (c) mentioned in the introduction. Therefore, we have the following objective function for the master problem:

$$\min \sum_{d \in D,\ p \in P} r^{dp}. \tag{4.49}$$

We now have a complete MILP formulation of the master problem and present the formulation of the subproblems in the following subsection.

4.2.1.2 Formulation of the subproblems

We now introduce the formulation of the subproblems. The sets and parameters needed are the same as for the master problem. For the rest of this subsection, we look at the formulation of the subsystem related to $(d, p) \in D \times P$. We first define the decision variables.

The decision variables we need for the subproblem are similar to the master problem variables. We also need decision variables that represent the units used in the overall system. We do not need a decision variable for the number of connections in the overall system, since this can be obtained through different means. We define:

- $t_{ium} \in \{0, 1\},\ i \in N,\ u \in U^i,\ m \in M^u$, a binary variable that takes value 1 if unit type u and model m is set at location i and 0 otherwise.

We again need decision variables that represent the subsystem (d, p). The only difference to the master problem is that we also need to include the model type in the index of the variable.

- $t^{dp}_{ium} \in \{0, 1\},\ i \in N,\ u \in U^i,\ m \in M^u$, a binary variable that takes value 1 if a unit of type u is installed at location i, and takes value 0 otherwise, and
- $x^{dp}_{iju\hat{u}} \in \{0, 1\},\ (i, j) \in A,\ u \in U^i,\ \hat{u} \in W^j(u)$, a binary variable that takes value 1 if locations i and j are connected and unit u and $\hat{u}$ are installed at i and j, respectively, and takes value 0 otherwise.

Lastly, we need a similar set of decision variables for the different functions $f \in F$.

- $t^f_{iu} \in \{0, 1\},\ i \in N_f,\ u \in U^i_f$, a binary variable that takes value 1 if a unit of type u is installed at location i, and takes value 0 otherwise, and
- $x^f_{iju\hat{u}} \in \{0, 1\},\ (i, j) \in A_f,\ u \in U^i_f,\ \hat{u} \in W^j_f(u)$, a binary variable that takes value 1 if locations i and j are connected and unit u and $\hat{u}$ are installed at i and j, respectively, and takes value 0 otherwise.

Furthermore, we need decision variables that are connected to the different characteristics (a)-(e) of reliability networks. However we will not present these variables here, because it is easier to understand their meaning together with the corresponding constraints.

First, since we have three levels (overall system, subsystems and flows) and their corresponding decision variable sets, we need to synchronize the decision variable sets through constraints. The constraints are built in a similar way to the master problem synchronization constraints.

The following constraint synchronizes the subsystem level to the overall system level:

$\forall d \in D,\ \forall i \in N\ \forall u \in U^i,\ m \in M^u$

$$t^{dp}_{ium} \leq t_{ium}. \tag{4.50}$$

Furthermore, the following constraints (4.51) to (4.54) synchronize the subsystem level with the function level:

$\forall f \in F\ \forall (i, j) \in A_F\ u \in U^i_f, \hat{u} \in W^j_f(u)\ :$

$$x^f_{iju\hat{u}} \leq x^{dp}_{iju\hat{u}}, \tag{4.51}$$

$\forall f \in F,\ \forall i \in N\ \forall u \in U^i_f$

$$t^f_{iu} \leq \sum_{m \in M^u} t^{dp}_{ium}, \tag{4.52}$$

$\forall (i,j) \in A\ u \in U^i, \hat{u} \in W^j(u)$:

$$x^{dp}_{iju\hat{u}} \leq \sum_{f \in F} x^f_{iju\hat{u}}, \tag{4.53}$$

$\forall i \in N\ \forall u \in U^i$:

$$\sum_{m \in M^u} t^{dp}_{ium} \leq \sum_{f \in F} t^f_{iu}. \tag{4.54}$$

Since we have different flows in our subsystem, we need the following flow constraints (4.55) to (4.58). Constraint (4.55) imposes the restriction that no more than the maximum number of connections E^{out}_{fu} can go out of unit u at location i for a function $f \in F$ in a subsystem, and it also synchronizes the arc and location variables of the subsystems:

$\forall f \in F,\ i \in N_f,\ \forall u \in U^i_f$:

$$\sum_{\substack{j \in V^+_f(i,u),\\ \hat{u} \in W^j_f(u)}} x^f_{iju\hat{u}} \leq E^{out}_{fu} t^f_{iu}. \tag{4.55}$$

The following constraint (4.56) imposes the same than (4.55) with the only difference that it is for connections arriving at a location.

$\forall f \in F,\ j \in N_f,\ \forall \hat{u} \in U^j_f$:

$$\sum_{\substack{i \in V^-_f(j,\hat{u}),\\ u \in U_f \ \mid\ \hat{u} \in C_f(u)}} x^f_{iju\hat{u}} \leq E^{in}_{fu} t^f_{j\hat{u}}. \tag{4.56}$$

Constraint (4.57) below imposes that if a unit u at location i is used for a subsystem, then at least $T^{f,in}_u$ connections must arrive at location i for the subsystem. Constraint (4.58) below imposes that a flow continues, which means that if a unit u at location i is used for a subsystem, at least $T^{f,out}_u$ connections must leave location i for the subsystem.

$\forall f \in F, \forall i \in N_f$:

$$\sum_{u \in (U^{\mathrm{b}}_f \cup \{e_f\}) \cap U^i_f} T^{f,\mathrm{in}}_u t^f_{iu} \leq \sum_{\substack{\ell \in V^-_f(i),\ u \in (U^{\mathrm{b}}_f \cup \{e_f\}) \cap U^i_f,\\ \hat{u} \in U_f \ \mid\ u \in C_f(\hat{u})}} x^f_{\ell i \hat{u} u}, \tag{4.57}$$

$$\sum_{u \in (U^{\mathrm{b}}_f \cup \{s_f\}) \cap U^i_f} T^{f,\mathrm{out}}_u t^f_{iu} \leq \sum_{\substack{\ell \in V^+_f(i),\ u \in (U^{\mathrm{b}}_f \cup \{s_f\}) \cap U^i_f,\\ \hat{u} \in W^\ell_f(u)}} x^f_{i\ell u\hat{u}}. \tag{4.58}$$

We need a final flow constraint which imposes that a flow starts and ends. This is achieved through constraints (4.57) and (4.58) as well as constraint (4.59), which imposes that a start and end unit are set for every flow.

$\forall f \in F,\ \forall u_f \in \{s_f, e_f\}$:

$$\sum_{i \in N^{u_f}_f} t^f_{iu_f} = 1. \tag{4.59}$$

After these flow constraints, we also need further architectural constraints for the subsystems and flows.

In the case of the DMS system, the functions must be connected. This means that, for example, the end unit of a function f_1 has to be also the start unit of another function f_2. Therefore, we need for some $f_1, f_2 \in F$ that

$\forall i \in N^{e_{f_1}}$ with $e_{f_1} = s_{f_2}$:

$$t^{df_1 p}_{ie_{f_1}} = t^{df_2 p}_{is_{f_2}}. \tag{4.60}$$

We also require the additional architectural constraints (4.61) and (4.62). Constraint (4.61) imposes the condition that if a unit u is installed, then it connects to all unit types it has to connect to:

$\forall f \in F,\ \forall i \in N_f,\ \forall u \in U^i_f,\ \forall \hat{u} \in C^+_f(u)$:

$$t^f_{iu} \leq \sum_{j \in V^+_f(i) \cap N^{\hat{u}}_f} x^f_{iju\hat{u}}. \tag{4.61}$$

Constraint (4.62) on the other hand imposes that if a unit u is installed, then it has a connection from all unit types it has to have a connection from:

$\forall f \in F,\ \forall j \in N_f,\ \forall u \in U^j_f,\ \forall \hat{u} \in C^-_f(u)$:

$$t^f_{ju} \leq \sum_{i \in V^-_f(j) \cap N^{\hat{u}}_f} x^f_{ij\hat{u}u}. \tag{4.62}$$

As shown in Chapter 3, constraints are needed to achieve a better linear relaxation for such a problem. Therefore, we also include the following constraints (4.63) and (4.64) into our subproblem formulation.

$\forall f \in F,\ \forall u \in U_f$:

$$\sum_{\substack{\hat{u} \in \{\overline{u} \ \mid \ u \in C^+_f(\overline{u})\}, \\ i \in N^{\hat{u}}_f}} t^f_{i\hat{u}} \leq E^{in}_{fu} \sum_{i \in N^u_f} t^f_{iu}, \tag{4.63}$$

$$\sum_{\substack{\hat{u} \in \{\overline{u} \ \mid \ u \in C^-_f(\overline{u})\}, \\ i \in N^{\hat{u}}_f}} t^f_{i\hat{u}} \leq E^{out}_{fu} \sum_{i \in N^u_f} t^f_{iu}. \tag{4.64}$$

Up until now, the constraints have been the same for every subproblem and are not based on the solution of any previously solved subproblem. Since our heuristic is based on solving the subproblems one after the other and use the solutions of the previously solved subproblems to improve the reliability, we introduce the following constraints. Let x^*, t^* be solutions of already solved subproblems.

First, we use these solutions to achieve redundancy through the following constraints. If $p \in \{1, 2\}$ for the current subproblem, then we add the following constraint to the formulation:

$\forall \hat{p} \in \{1, 2\} \backslash \{p\},\ \forall i \in N, \forall u \in U^i$:

$$t^{dp}_{iu} \leq 1 - t^{d\hat{p}^*}_{iu}. \tag{4.65}$$

We also need a constraint in the subproblem similar to (4.43) which limits the number of ports used. This constraint requires the solutions of the previously solved subproblems. Let

$$E_{um}^{dp} = E_{um} - \sum_{\hat{d}\in D\setminus\{d\},\hat{p}\in P\setminus\{p\}} \left(\sum_{\substack{\ell\in V^-(i,u),\\ \hat{u}\in U^\ell \ \mid \ u\in C(\hat{u})}} x_{\ell i\hat{u}u}^{\hat{d}\hat{p}^*} + \sum_{\substack{\ell\in V^+(i,u),\\ \hat{u}\in W^k(u)}} x_{i\ell u\hat{u}}^{\hat{d}\hat{p}^*} \right).$$

We then require:

$\forall i \in N,\ \forall u \in U^i$:

$$\sum_{\substack{\ell\in V^-(i,u),\\ \hat{u}\in U^\ell \ \mid \ u\in C(\hat{u}),\\ \sum_{\hat{d}\in D\setminus\{d\},\hat{p}\in P\setminus\{p\}} x_{\ell i\hat{u}u}^{\hat{d}\hat{p}^*}=0}} x_{\ell i\hat{u}u}^{dp} + \sum_{\substack{\ell\in V^+(i,u),\\ \hat{u}\in W^k(u),\\ \sum_{\hat{d}\in D\setminus\{d\},\hat{p}\in P\setminus\{p\}} x_{i\ell u\hat{u}}^{\hat{d}\hat{p}^*}=0}} x_{i\ell u\hat{u}}^{dp} \le \sum_{m\in M^u} t_{ium}^{dp} E_{um}^{dp}. \quad (4.66)$$

Furthermore, the following constraints synchronize the overall system variables with the solutions of the previously solved subproblems.

$$\forall i \in N\ \forall u \in U^i : \sum_{\hat{d}\in D,\ \hat{p}\in P} t_{iu}^{dp^*} \ge 1 \Rightarrow \sum_{m\in M^u} t_{ium} \ge 1. \quad (4.67)$$

The last group of constraints are related to the five characteristics of reliability. First, we propose the constraints (4.68) to (4.70) that are related to the characteristic (c). Constraint (4.68) gives us the sum of failure rates for the current subsystem:

$$\sum_{\substack{i\in N,u\in u,\\ m\in M^u}} f_{um} t_{ium}^{dp} + \sum_{\substack{\forall(i,j)\in A,\\ \forall u\in U^i \ \forall\hat{u}\in W^j(u)}} f_{cable} x_{iju\hat{u}}^{dp} = r^{dp}. \quad (4.68)$$

The following constraints (4.69) and (4.70) calculate the number of units and gives us the number of cables used in the subsystem, respectively.

$$\sum_{\substack{i\in N,u\in u,\\ m\in M^u}} t_{ium}^{dp} = t^{dp}, \quad (4.69)$$

$$\sum_{(i,j)\in A,\ u\in U^i,\hat{u}\in W^j(u)} x_{iju\hat{u}}^{dp} = x^{dp}. \quad (4.70)$$

We now define a number of constraints that are related to the characteristics (b) and (d). These are related to the overlap of units and connections of the different paths, and also to the sum of the failure rates of these units and connections that are used by different paths.

Let $\hat{t}_{\hat{p}}^{dp}$ and $\hat{x}_{\hat{p}}^{dp}$ be the number of overlapping units and connections of subsystem (d,p) and $(d,\hat{p})$, respectively. Furthermore, let $\hat{r}_{\hat{p}}^{dp}$ be the sum of failure rate of the overlapping units and connections of subsystem (d,p) and $(d,\hat{p})$. We need the following constraints which give us the values for these variables based on the previously solved subproblems and the current subproblem.

$\forall \hat{p} \in P\setminus\{p\}$:

$$\hat{t}_{\hat{p}}^{dp} = \sum_{i\in N,\ u\in U^i \mid \ \sum_{m\in M^u} t_{iu}^{d\hat{p}^*}=1,} t_{ium}^{dp}, \quad (4.71)$$

$$\hat{x}_{\hat{p}}^{dp} = \sum_{(i,j)\in A,\ u\in U^i,\hat{u}\in W^j(u)\mid \ x_{iju\hat{u}}^{d\hat{p}^*}=1} x_{iju\hat{u}}^{dp}, \quad (4.72)$$

$$\hat{r}_{\hat{p}}^{dp} = \sum_{(i,j)\in A,\ u\in U^i,\hat{u}\in W^j(u)|\ x_{iju\hat{u}}^{d\hat{p}^*}=1} f_{cable} x_{iju\hat{u}}^{dp} + \sum_{i\in N,\ u\in U^i|\ \sum_{m\in M^u} t_{ium}^{d\hat{p}^*}=1,} f_{um} t_{ium}^{dp}. \quad (4.73)$$

Let $\hat{t}_u^{dp}$ be the decision variable which represents the number of overlapping units of type $u \in U$ of subsystem (d, p) with all other previous solved subsystems of $d \in D$. For this we need the following constraints:

$\forall u \in U$:

$$\hat{t}_u^{dp} = \sum_{i\in N\ |\ \sum_{m\in M^u,\hat{p}\in P\setminus\{p\}} t_{iu}^{d\hat{p}^*}=1,} t_{ium}^{dp}. \quad (4.74)$$

The variables $\hat{r}_{\hat{p}}^{dp}$ can be used for the characteristic (b). However, we need further constraints and variables for characteristic (d). The following constraints and variables give us the difference of these variables for the different subsystem combinations.

$\forall(\hat{p}, \overline{p}) \in P\setminus\{p\} \times P\setminus\{p\}$:

$$\hat{t}_{\hat{p}}^{dp} - \hat{t}_{\overline{p}}^{dp} = \hat{t}_{(\hat{p},\overline{p})}^{dp,+} - \hat{t}_{(\hat{p},\overline{p})}^{dp,-}, \quad (4.75)$$

$$\hat{x}_{\hat{p}}^{dp} - \hat{x}_{\overline{p}}^{dp} = \hat{x}_{(\hat{p},\overline{p})}^{dp,+} - \hat{x}_{(\hat{p},\overline{p})}^{dp,-}, \quad (4.76)$$

$$\hat{r}_{\hat{p}}^{dp} - \hat{r}_{\overline{p}}^{dp} = \hat{r}_{(\hat{p},\overline{p})}^{dp,+} - \hat{r}_{(\hat{p},\overline{p})}^{dp,-}. \quad (4.77)$$

To consider characteristic (e), we need to know the number of units and the number of connections in the overall system. For this, we first calculate the value

$$X^{\text{before}} = \sum_{\substack{(i,j)\in A, u\in U^i,\hat{u}\in W^j(u),\\ \hat{d}\in D,\hat{p}\in P}} x_{iju\hat{u}}^{\hat{d}\hat{p}^*}, \quad (4.78)$$

which is based on previously solved subproblems. The following constraints (4.79) and (4.80) give us these numbers through the variables t^o and x^o:

$$\sum_{\substack{i\in N, u\in u,\\ m\in M^u}} t_{ium} = t^o, \quad (4.79)$$

$$\sum_{\substack{(i,j)\in A,\ u\in U^i,\hat{u}\in W^j(u):\\ \sum_{\hat{d}\in D}\sum_{\hat{p}\in P} x_{iju\hat{u}}^{\hat{d}\hat{p}^*}=0}} x_{iju\hat{u}}^{dp} + X^{\text{before}} = x^o. \quad (4.80)$$

Lastly, to use characteristic (a) in our heuristic, we define the following parameter s^{dp} and constraint (4.81). Let s^{dp} be the maximum number of overlapping units of subsystem (d, p) with all other paths of the same door, so that we have the constraint

$$\sum_{\hat{p}\in P\setminus\{p\}} \hat{t}_{\hat{p}}^{dp} \le s^{dp}. \quad (4.81)$$

With this, we have all the variables and constraints of the subproblem formulation and it only remains to define an objective function for the subproblem.

So far, we have only proposed constraints and variables associated to the characteristics. To effectively use these characteristics, we must incorporate these variables into the objective function of the subproblem. We need to make sure to create a feasible overall system, while solving the different subproblems. For that, we also need to consider the architectural constraint

(4.45) and incorporate them into the objective function. Therefore, let L_{iu} be the number of open ports of unit u at position u. Let $X \in \{0,1\}^{|N|\times|N|}$ with x_{ij} of X is 0 if arc (i,j) is not yet used in any subsystem and 1 otherwise. Also, let $T \in (U \cup \{0\})^{|N|}$ and $t_i = u$ if unit type $u \in U$ is set by a subsystem on position i and 0 otherwise. Let

$$\begin{aligned}
H_1 &:= \{i \in N \mid t_i = OVF \text{ and } |x_{*i}|_1 = 1\},\\
H_> &:= \{i \in N \mid t_i = OVF \text{ and } |x_{*i}|_1 > 1\},\\
B_j^1 &:= \{i \in N \mid (i,j) \in A, \sum_{j\in H_1} (X)_{ij}) > 0\} \text{ and}\\
B_j^> &:= \{i \in N \mid (i,j) \in A, \sum_{j\in H_>} (X)_{ij}) > 0\}.
\end{aligned}$$

Moreover, we need to choose appropriate coefficients $w_9, w_{10} \in \mathbb{R}_-$ with $w_9 < w_{10}$ for the objective function. These are based on the overall objective coefficients of the OVF and OCU units.

We then can define the objective function for a subproblem of $(d,p) \in D \times P$ with a coefficient vector $w \in \mathbb{R}_+^{10}$:

$$\begin{aligned}
v(dp) := \min\ & w_1 t^{total} + w_2 x^{total} + w_3 t^{dp} + w_4 x^{dp} + w_5 r^{dp} + w_6 \left(\hat{t}^{dp,+}_{(\hat{p},\overline{p})} + \hat{t}^{dp,-}_{(\hat{p},\overline{p})}\right)\\
& + w_7 \left(\hat{x}^{dp,+}_{(\hat{p},\overline{p})} + \hat{x}^{dp,-}_{(\hat{p},\overline{p})}\right) + w_8 \left(\hat{r}^{dp,+}_{(\hat{p},\overline{p})} + \hat{r}^{dp,-}_{(\hat{p},\overline{p})}\right)\\
& + \sum_{j\in H} \left(\sum_{i\in B_j^1, \hat{u}\in W_i(OVF)} w_9 x_{ij\hat{u}(OVF)} + \sum_{i\in B_j^>, \hat{u}\in W_i(OVF)} w_{10} x_{ij\hat{u}(OVF)} \right).
\end{aligned} \tag{4.82}$$

The coefficients are based on failure rate values of the unit types and cables and we will give an example coefficient vector which is used in the computational study.

4.2.2 Heuristic Algorithm

Now that we have presented the formulation of the master problem and the subproblems, we can introduce the algorithm of the heuristic. As mentioned before, it is built on the key idea of Dantzig-Wolfe decomposition with its master problem and subproblems, the characteristics of reliability, and branching on the variables t_{iu}.

The following points must be mentioned before we can show the algorithm of the heuristic in a flow chart.

- Let $V = \{v(d,p) \mid d \in D,\ p \in P\}$ be the subproblems. Furthermore, let $U_<$ be the ordered set of U regarding the objective coefficients of (4.47) with $U_< = \{u_1, \ldots, u_{|U|}\}$.
- When solving the subproblems, we solve the subsystems of the doors consecutively. Also, we first solve the subproblems associated with the disjoint paths and then solve the subproblem associated with the third path.
- Furthermore, when solving the subproblems for the first time, we alter the objective function of the subproblems associated with a third path and add $w_{new} \sum_{\forall \hat{p}\in P\setminus\{p\}} \hat{t}^{dp}_{\hat{p}}$. Therefore we have

$$\begin{aligned}
v(d3) := \min\ & w_1 t^{total} + w_2 x^{total} + w_3 t^{dp} + w_4 x^{dp} + w_5 \left(\hat{t}^{dp,+}_{(\hat{p},\overline{p})} + \hat{t}^{dp,-}_{(\hat{p},\overline{p})}\right)\\
& + w_6 \left(\hat{x}^{dp,+}_{(\hat{p},\overline{p})} + \hat{x}^{dp,-}_{(\hat{p},\overline{p})}\right) + w_7 \left(\hat{r}^{dp,+}_{(\hat{p},\overline{p})} + \hat{r}^{dp,-}_{(\hat{p},\overline{p})}\right) + w_{new} \sum_{\hat{p}\in P\setminus\{p\}} \hat{t}^{dp}_{\hat{p}}\\
& + \sum_{j\in H} \left(\sum_{i\in B_j^1, \hat{u}\in W_i(OVF)} w_1 x_{ij\hat{u}(OVF)} + \sum_{i\in B_j^>, \hat{u}\in W_i(OVF)} w_> x_{ij\hat{u}(OVF)} \right).
\end{aligned} \tag{4.83}$$

Through this, we achieve a solution of the overall problem where the third path overlaps as much as possible with the other two paths. We also have the highest possible s over which we can iterate and, therefore, we fix $s = \sum_{\hat{p} \in P \setminus \{p\}} \hat{t}_{\hat{p}}^{dp^*}$, after solving the subproblems for the first time.

- To use the solutions of the subproblems in the master problem, we fix the variables t_{iu}^{dp} and $x_{iju\hat{u}}^{dp}$ of the master problem as follows:
  $\forall (d,p) \in D \times P,\ \forall i \in N,\ \forall u \in U^i$:

$$t_{iu}^{dp} = \sum_{m \in M^u} t_{ium}^{dp^*}, \tag{4.84}$$

  $\forall (d,p) \in D \times P,\ \forall (i,j) \in A,\ \forall u \in U^i,\ \forall \hat{u} \in W_j(u)$:

$$x_{iju\hat{u}}^{dp} = x_{iju\hat{u}}^{dp^*}, \tag{4.85}$$

  where $x_{iju\hat{u}}^{dp^*}$ and $t_{ium}^{dp^*}$ are the solution values of the subproblems.

- Let $\overline{z}$ be the objective value o^{total^*} of the best reliability feasible solution found. For the following branching rules, we add the following constraint (4.86) to all subproblems:

$$\sum_{i \in N,\ u \in U^i,\ m \in M^u} w_{um} t_{ium} + w_c x^o \leq \overline{z}. \tag{4.86}$$

- As mentioned before, we use the idea of branching. We branch when we have either a reliability feasible solution from the master problem or when constraint (4.86) is infeasible for a subproblem. In the following, we show the two different ways of creating new sets of subproblems.

  1. First, we show how to branch when we have a reliability feasible solution. Let $\hat{t}_u^{d3^*},\ (u,d) \in U \times D$, be the values of the subproblems solutions. We construct a new set of subproblems V_u for each unit type $u \in U_<$. Let $u_l \in U_<$ with $l \in \{1, \ldots, |U|\}$. The new set of subproblems V_{u_l} has the following bounds on the variables $\hat{t}_u^{d3},\ (u,d) \in U \times D$:
     $\forall d \in D\ \forall u_k \in U_<$ with $k < l$:

$$\text{if } \max_{d \in D} \hat{t}_{u_k}^{d3^*} = 0 \text{ then } \hat{t}_{u_k}^{d3} = 0 \text{ else } \hat{t}_{u_k}^{d3} \geq \max_{d \in D} \hat{t}_{u_k}^{d3^*},$$

     and

$$\text{if } \max_{d \in D} \hat{t}_{u_l}^{d3^*} = 0 \text{ then } \hat{t}_{u_l}^{d3} = 1 \text{ else } \hat{t}_{u_l}^{d3} \leq \max_{d \in D} \hat{t}_{u_l}^{d3^*} - 1.$$

     Furthermore, for the set of subproblems we fix the parameter s to be the sum of the upper bounds of variables $\hat{t}_u^{d3},\ (u,d) \in U \times D$.

  2. We also branch if (4.86) is infeasible for a subproblem. If a subproblem is infeasible because of (4.86), we solve the subproblem again with an infinite right hand side of (4.86) and obtain a solution for every subproblem.
     Let $\hat{t}_u^{d3^*},\ (u,d) \in U \times D$, be the values of the subproblems solutions. We construct a new set of subproblems V_u for each unit type $u \in U_<$. Let $u_l \in U_<$ with $l \in \{1, \ldots, |U|\}$. The new set of subproblems V_{u_l} has the following new bounds on the variables $\hat{t}_u^{d3},\ (u,d) \in U \times D$:
     $\forall d \in D\ \forall u_k \in U_<$ with $k < l$:

$$\hat{t}_{u_k}^{d3} \leq \max_{d \in D} \hat{t}_{u_k}^{d3^*}, \text{ and } \hat{t}_{u_l}^{d3} \geq \max_{d \in D} \hat{t}_{u_l}^{d3^*} + 1.$$

Furthermore, for the new sets of subproblems we fix the parameter s to s^* which is the parameter value of the current set of subproblems.

- Lastly, having found a reliability feasible solution, we use an improvement step in the heuristic. If we find a reliability feasible solution, we still do not know if a system with more units or more overlapping units can have a better objective value. This is the case because we have different types of units and also each type has different models. Let s^* be the parameter value of the current reliability feasible solution and $\hat{t}_u^{d3^*},\ (u,d) \in U \times D$, be the values of the associated subproblems solutions. Let $\hat{t}_u = \max_{d \in D} \hat{t}_u^{d3^*},\ u \in U$, let V be the associated set of subproblems and let $\hat{z}$ be the objective value o^{total^*} of the reliability feasible solution. o^{total^*} is used for constraint (4.86). We create a new set of subproblems V_{new} with $s = s^* - 1$ which we use as the initial set of subproblems. Furthermore, we use the solution from the reliability feasible solution as the incumbent x^* of the algorithm. The following flow chart in Figure 4.2 shows the algorithm of the improvement step.

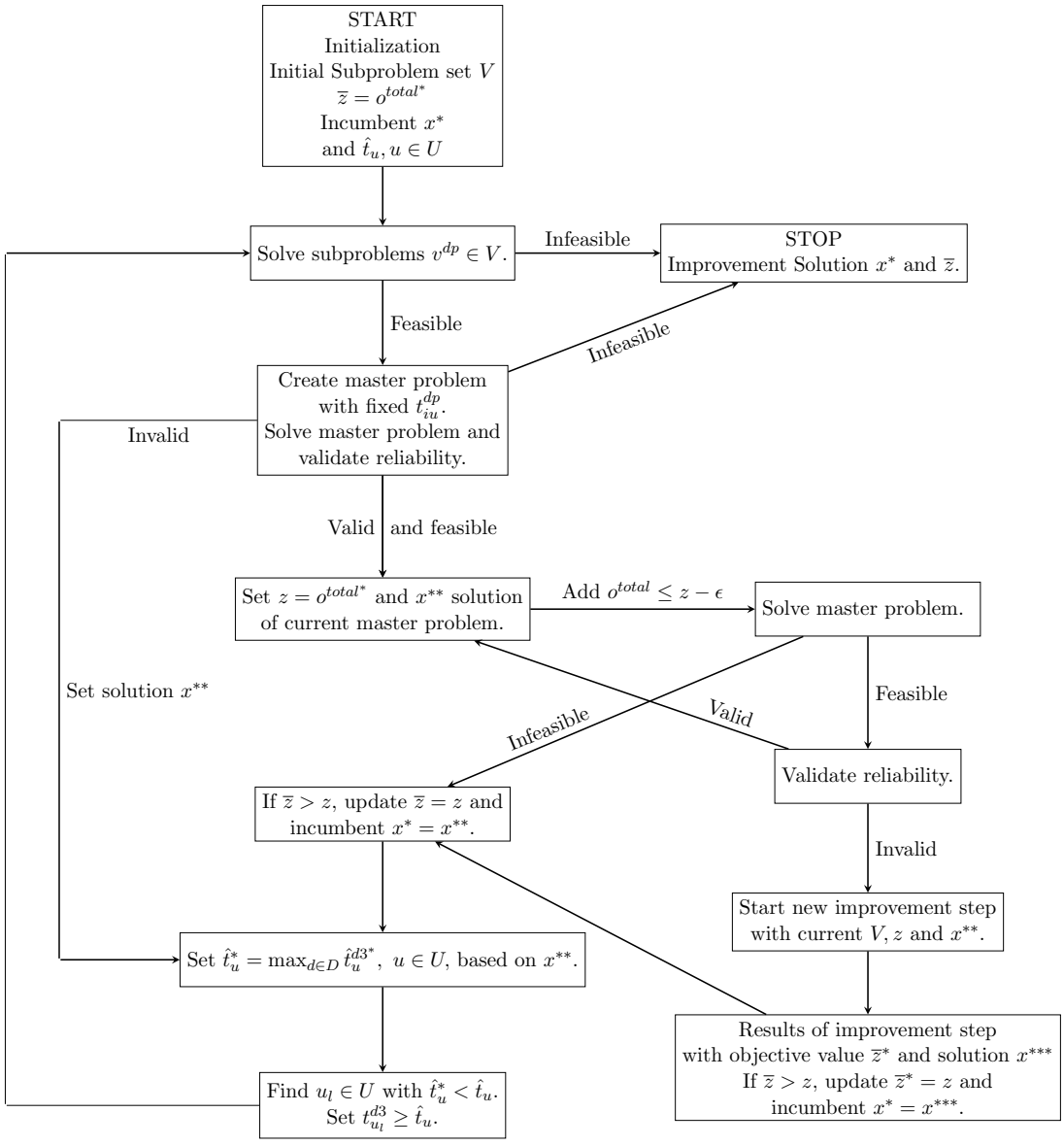

Figure 4.2: Improvement step

With these points, we now present the heuristic through a flow chart in Figure 4.3.

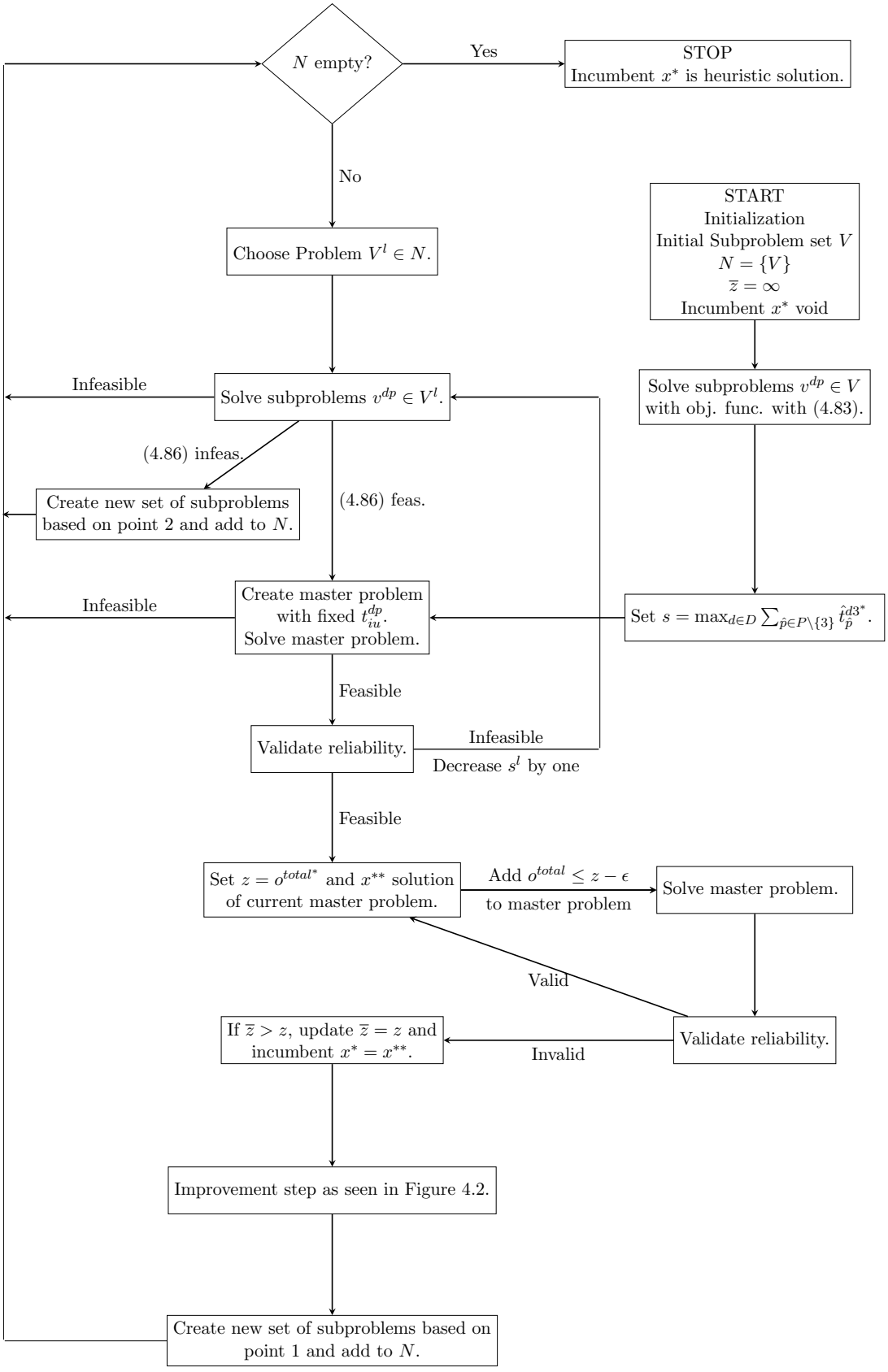

Figure 4.3: Heuristic for reliability feasibility

4.2.3 Computational Study for Reliability Heuristic

We carry out a computational study to analyze the performance of our heuristic. We cannot compare the new heuristic to other methods, because it is very specific to this problem and there exist no other fitting heuristics or exact methods to solve the problem. We will run problem instances that differ in the number of doors, thresholds for reliability, configurations of the locations, objective functions and unit type parameters. We will then present the best solution, computational time, number of nodes and range of solution values for each instance.

The instances are artificial and were built based on data provided by AIRBUS Group. As mentioned in the introduction, we have two functions in the DMS and therefore $F = \{F_1,\ F_2\}$. The first function is the information flow from two sensors at a door to a controller (CPIOM) and the second function is the information flow from that controller to an outflow valve. Also, the controller used in both functions has to be the same for a door and therefore constraint (4.60) is used.

The DMS has eight unit types: door (DO), latch & lock sensor (LLS), closed sensor (CS), outflow valve (OVF), outflow valve control unit (OCU), remote data concentrator (RDC), controller (CPIOM), switch (SWT). We have the following unit sets:

- $U^{s,e} = \{s_1 = DO, s_2 = CPIOM, e_1 = CPIOM, e_2 = OVF\}$,
- $U^{\mathrm{b}} = \{LLS, CS, OCU, RDC, SWT\}$,
- $U_1 = \{DO, LLS, CS, RDC, CPIOM, SWT\}$,
- $U_2 = \{OVG, OCU, RDC, CPIOM, SWT\}$.

Except for the door, all units are available in different models. For example, there can be various kinds of RDC models with different costs, weights or number of ports. In our instances, most units have two available models.

All computations were done with a Four Intel Xeon E5-2680 v3 2.5GHz, 192Gb RAM and CPLEX 12.5.1. The heuristic was coded in C++.

In our computations, we used failure rates from 10^{-5} to 10^{-2} and a reliability threshold from 0.998 to 0.9999. We ran computational tests for $2, 3, 4$ and 5 doors with 3 paths.

For the objective function (4.82), we used the coefficient set

$$w = (20, 1, 1, 1, 1000, 0.5, 0, 200, 10, 15).$$

These values were found after numerical experiments. For other failure rates, w_5 and w_8 would need to be adapted based on a few numerical experiments with small example models. For constraint (4.48), we use different randomly generated coefficients w_{um} and w_c which are the weight coefficients for the objective of the optimization. These intervals for the random value is based on data for real components used by AIRBUS Group. The parameters and coefficients used can be found in Appendix C.2. Table 4.1 shows the results of our computational tests. Next to the values of the best solution found, the table also shows the value range and reliability range. The value range is the range of the objective values of all feasible solutions found and the reliability range is the range of the reliabilities of all feasible solutions found. In Table 4.1, NA stands for "Not Achievable" which means there is no solution with three paths which reaches the necessary reliability threshold.

$\|D\|$	Reliability Threshold	Nodes	Time (s)	Optimal Value	Reliability	Value Range	Reliability Range
2	0.999	16	335	1292.85	0.99901	1292.85-1377.25	0.99900-0.99903
2	0.9991	21	407	1347.2	0.99910	1347.2-1378.15	0.99910-0.99915
2	0.9992	40	950	1404.95	0.99920	1404.95-1509.3	0.99920-0.99925
2	0.9993	36	811	1451.1	0.99931	1451.1-1553.3	0.99930-0.99932
2	0.9994	53	989	1479.3	0.99940	1479.3-1590.1	0.99940-0.99942

2	0.9995	64	1187	1569.3	0.99950	1569.3-1675.1	0.99950-0.99952
2	0.9996	58	1128	1599.35	0.99961	1599.35-1769.3	0.99960-0.99964
2	0.9997	39	949	1705.3	0.99970	1705.3-1755.1	0.99970-0.99970
2	0.9998	19	608	1786.3	0.99981	1786.3-1929.4	0.99980-0.99981
2	0.9999	9	460	1888.4	0.99990	1888.4-1888.4	0.99990-0.99990
3	0.999	93	4199	1485.9	0.99900	1485.9-1660.2	0.99900-0.99934
3	0.9991	88	3548	1525.9	0.99912	1525.9-1650.95	0.99912-0.99937
3	0.9992	77	4271	1545.9	0.99927	1545.9-1647.8	0.99920-0.99941
3	0.9993	66	2995	1575.9	0.99931	1575.9-1701.05	0.99930-0.99937
3	0.9994	53	2871	1625.9	0.99940	1625.9-1705.9	0.99940-0.99965
3	0.9995	44	2994	1703.9	0.99950	1703.9-1757.1	0.99950-0.99965
3	0.9996	27	1434	1705.9	0.99965	1705.9-1705.9	0.99965-0.99965
3	0.9997	13	1282	1747	0.99970	1747-1747	0.99970-0.99970
3	0.9998	14	1524	1854	0.99980	1854-1854.05	0.99980-0.99980
3	0.9999	1	294	NA	NA	NA	NA
4	0.999	81	3820	1606.6	0.99904	1606.6-1862.7	0.99900-0.99916
4	0.9991	65	3581	1616.55	0.99910	1616.55-1766.5	0.99910-0.99954
4	0.9992	51	3166	1686.55	0.99922	1686.55-1766.5	0.99920-0.99954
4	0.9993	40	3216	1756.55	0.99931	1756.55-1903.3	0.99930-0.99954
4	0.9994	35	2920	1766.5	0.99954	1766.5-2014.65	0.99940-0.99954
4	0.9995	22	1492	1766.5	0.99954	1766.5-1766.5	0.99954-0.99954
4	0.9996	17	1400	1806.5	0.99960	1806.5-1806.5	0.99960-0.99960
4	0.9997	14	1641	1896.5	0.99970	1896.5-1904.6	0.99970-0.99970
4	0.9998	9	1319	2004.6	0.99980	2004.6-2004.6	0.99980-0.99980
4	0.9999	1	308	NA	NA	NA	NA
5	0.999	77	8222	1728.8	0.99900	1728.8-1818.7	0.99900-0.99944
5	0.9991	58	7625	1788.8	0.99910	1788.8-1818.7	0.99910-0.99944
5	0.9992	45	7117	1818.7	0.99944	1818.7-1838.8	0.99920-0.99944
5	0.9993	49	6213	1818.1	0.99944	1818.1-1946.8	0.99930-0.99944
5	0.9994	42	4392	1818.1	0.99944	1818.1-1818.1	0.99944-0.99944
5	0.9995	31	4055	1858.1	0.99952	1858.1-1858.1	0.99952-0.99952
5	0.9996	13	3530	1928.1	0.99960	1928.1-1928.1	0.99960-0.99960
5	0.9997	13	3343	1998.85	0.99970	1998.85-1998.85	0.99970-0.99970
5	0.9998	9	3021	2168.2	0.99980	2168.2-2168.2	0.99980-0.99980
5	0.9999	1	685	NA	NA	NA	NA

Table 4.1: Results for the first set of w_{um} and w_c

Table 4.1 shows the results for the first set of w_{um} and w_c. We can see that in most cases the optimal value increases when we increase the threshold, except for cases where a solution has a very high reliability compared to the threshold and it is still valid for a higher reliability threshold. In these situations, the optimal solution can then be the same for multiple thresholds. Using this heuristic, we found multiple reliability feasible solutions with a large range of objective and reliability values. The heuristic is not built to just find a feasible solution, but to find the best feasible solution based on our reliability characteristics. Furthermore, for instances with $3, 4$ and 5 doors we see that the time decreases when the threshold increases. One reason for this is that the number of feasible solutions decreases and the branching tree becomes smaller, which can be seen by the number of nodes. However, this is not the case for instances with 2 doors. For these instances, the number of nodes first increases with the threshold and after a certain point decreases again.

We ran computational tests for a second set of w_{um} and w_c and the results show the same behavior. The results can be seen in Table C.2 in Appendix C.3. We ran further tests with a lower reliability threshold to observe if the behavior of the instances for 2 doors also occur for 3, 4 and 5 doors. These results can be seen Table C.3 in the Appendix C.3. Our results show that such behavior occurs for instances with 3 doors, but not for instances with 4 or 5 doors. It is probable that the same behavior occurs for 4 and 5 doors if we decrease the reliability threshold

even more.

4.3 Discussion

In this section, we discuss the practicability and limitations of the results in this chapter and the authenticity of the instances used in the computational study. We proposed MILP and MILNP formulation for the DMS problem with redundancy and reliability and presented the current problems of solving these models. Because these models are not solvable, we introduced the heuristic to obtain good reliability feasible solution. It is specialized to the model, but has a general structure so that it can also easily be fitted to similar network problems with multiple flows and paths.A limitation of our heuristic is that it only considers at most three paths. If more paths have to be used to reach the necessary level of reliability, the heuristic has to be extended and further iteration variations must be included. This makes the heuristic much more computationally expensive and possibly unusable.

For authenticity of the instances in the computational study, we must note that we only used artificial data due to confidentiality issues. However the instances are built in collaboration with AIRBUS Group and the cost, failure rates, weight and other parameters are based on real values provided by AIRBUS Group.

Lastly, the heuristic does not only provide one feasible solution but several. In practice, engineers do not only want the optimal solution for reviewing but a selection of good feasible solutions. Therefore, the heuristic can, as the redundancy method, be used in the preliminary design phase of a network in order to enable a quick exploration of the design space for a better initialization of the detailed design phases.

Chapter 5

Conclusion

This section summarises the main results of the thesis and explains our conclusions. Section 5.1 provides an overview of the contents of this dissertation, and in Section 5.2 we present the main results and possible future research directions.

5.1 Summary of the Contents

In Chapter 2 we first give a short overview of reliability evaluation and calculation techniques and discuss the problems of using them in optimization models. Afterwards we present Proposition 1, which is a result in probability and combinatorics that simplifies the classical probability principle of inclusion-exclusion formula for intersection of unions. Furthermore, we prove the time complexity of the formula in Proposition 1. We use this to introduce a new algorithm to calculate the reliability of a system network with multiple functions and implementations which is also usable in optimization models and compare the efficiency of this algorithm to a very known and often used algorithm KDH88. The contents of this chapter is published in [48] of which I am the main author.

Chapter 3 addresses the DMS problem with redundancy in the field of mixed integer programming problems. We first give an introduction into the problem of designing a door management system while considering redundancy and what as what kind of mixed integer programming problem we are going to formulate. We then introduce an MILP formulation and explore two solving approaches. The first solving approach is based on branch-and-bound and we introduce a new branching rule and heuristic which are tailored to this MILP formulation. The solving approach is successful and we present a computational study that compares it to the general MILP solver implemented in CPLEX. This approach and the MILP formulation are published in [49] of which I am the main author. The second approach is based on branch-and-price and also tailored to the formulation. The approach was not successful for the DMS problem with redundancy, but it gave us a good idea for a heuristic of the DMS problem with redundancy and reliability.

In Chapter 4 we first introduce MINLP and MILP formulations for the DMS problem with redundancy and reliability that is based on the formulation of Chapter 3 and discuss why these problems are not solvable. Afterwards, we introduce a new heuristic for the DMS problem with redundancy and reliability that gives good feasible solution. The heuristic is based on branch-and-price approach in Chapter 3, the reliability calculation algorithm from Chapter 2, and observed characteristic of reliability calculation.

Our main findings and future research possibilities on these topics are summarized in the following three sections.

5.2 Main Findings and Further Research

In the following we present the main results of the dissertation and what possible future research directions are possible.

5.2.1 Reliability Calculation Algorithm

Our main result presented in Chapter 2 is a new expression (Proposition 1) which considerably reduces the computational effort needed to calculate the probability principle of inclusion-exclusion when applied to intersections of unions of events. It has been shown that the formula obtained can be applied to the reliability calculation of a certain kind of complex network systems and it decreases the computational time significantly. It has also been shown that the computational complexity is reduced from doubly exponential to exponential with linear exponent. Moreover, we have provided a comparison with the general sum of disjoint products method KDH88 from [19] and showed that our method is more computationally efficient as KDH88 for our examples.

The complex network systems we considered in this thesis are systems with multiple functions, which means they have multiple start and end nodes in the system. If a system with only one function is considered, the simplification does not take effect and the number of terms of the probability principle of inclusion-exclusion does not decrease. We also considered that we have multiple implementations for each function. For a system with only one implementation per function, our proposed calculation method does not have any advantages. In all the other cases, that is, multiple functions and multiple implementations per function, the expression proposed in this thesis can be applied very effectively.

Another limitation of our method is the same which KDH88 and other SDP methods also have. The different paths in the system that represent the implementations for each function and the failure probability of all components in the system have to be known.

It must be noted that the result introduced in this paper does not only provide the option to calculate reliability more efficiently. It also allows to formulate optimization problems of complex network systems that include the exact reliability of the system without depending solely on heuristics to solve it which can be seen in Chapter 4.

A potential extension for future research is to generalize Proposition 1 and Lemma 2. For example, the implementation for a function i can also be used for function j with $i \neq j$ which results in $\mathcal{F}_i \cap \mathcal{F}_j \neq \emptyset$. This may result in a simplification of the probability principle of inclusion-exclusion with summand coefficients that are not -1 or 1, and still results in a decrease on the number of summands compared to Proposition 1.

Another potential extension for future research is to see if this simplification can be used with a multi-state system. In this case, the components of a system operate in any of several intermediate states with various effects on the entire system reliability.

5.2.2 Redundancy Network System Model for the DMS

In Chapter 3 we have proposed for the first time in the literature an MILP formulation to design the DMS of an aircraft while optimizing a certain criterion (e.g. cost or weight of the system) and considering redundancy. It is a network system with multiple functions and k-redundancy for the functions. Because the MILP formulation is not easily solvable with a standard solver like CPLEX, we introduced new branching rules and we proposed a heuristic to solve the problem. Both are specialized for the model proposed in this paper and through computational tests we were able to see that the DMS problem with redundancy can now be solved much more efficiently: Problem instances which could not be solved in several hours can now be solved in minutes.

As the next step for this research, we considered the reliability of the DMS and how to include it in the optimization model, which we addressed in Chapter 4. Another aspect that was not considered in this thesis is the speed of the data transportation, which is also important in safety systems. By adding this property, we would be closer to solve the most possible real problem and, thus, we would be able to provide aircraft engineers with a better decision tool in the design process.

5.2.3 Reliability Network System Model for the DMS

In Chapter 4 we have proposed for the first time MILP and MINLP formulations for the DMS problem with redundancy and reliability. Furthermore, we have proposed a heuristic for DMS problem with redundancy ad reliability. The heuristic is based on the key idea of Dantzig-Wolfe decomposition and branching of integer variables and differs greatly from a genetic heuristic, which, to the best of our knowledge, was so far the technique used by AIRBUS to solve this problem.

Because the problem is not exactly solvable as an MILP or MINLP problem, we introduced the heuristic to obtain good reliability feasible solution. It is specialized to the model, but has a general structure so that it can also easily be fitted to similar network problems with multiple flows and paths.

Regarding future research for the heuristic, it is possible to include other characteristics of a data network that are not easily incorporated into an MILP formulation. As mentioned before, the speed of the data transportation is not considered in the current model and it is also important in safety systems. Therefore, further research should include these requirements in the model. The corresponding constraints will be non-linear, which means that certain characteristics of data speed in a network have to be found and used in a heuristic. Another direction for future research is to find a solution to the linear relaxation problem of the MILP problem and, hence, being able to solve the MILP of the DMS problem with redundancy and reliability. Finally, it might also be interesting to see how we can extend the new heuristic to more than three paths.

Bibliography

[1] J. Abraham. An improved algorithm for network reliability. *IEEE Transactions on Reliability*, R-28(1):58 – 61, 1979.

[2] T. Achterberg and T. Berthold. Improving the feasibility pump. *Discrete Optimization*, 4(1):77–86, 2007.

[3] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[4] A. Balan and L. Traldi. Preprocessing minpaths for sum of disjoint products. *IEEE Transactions on Reliability*, 52(3):289 – 295, 2003.

[5] C. Bauer, L. Kristen, C. Bes, and M. Mongeau. Flight-control system architecture optimization for fly-by-wire airliners. *Journal of Guidance, Control, and Dynamics*, 30(4):1023–1043, 2007.

[6] M. Boyd. *Dynamic fault tree models: Techniques for analysis of advanced fault tolerant computer systems*. PhD thesis, Duke University, Durham, NC, USA, 1992. UMI Order No. GAX92-02503.

[7] K. Chung and P. Erdös. On the application of the Borel-Cantelli lemma. *Trans. Amer. Math. Soc.*, pages 179 – 186, 1952.

[8] D. Coit and A. E. Smith. Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach. *Computers & Operations Research*, 23(6):515–526, 1996.

[9] E. Datta and N. K. Goyal. Sum of disjoint product approach for reliability evaluation of stochastic flow networks. *International Journal of System Assurance Engineering and Management*, 8(2):1734–1749, 2017.

[10] D. Dawson and D. Sankoff. An inequality of probabilities. *Proceedings of the American Mathematical Society*, 18:504 – 507, 1967.

[11] J. Desrosiers and M. E. Lübbecke. Branch-price-and-cut algorithms. In *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010.

[12] W. Dotson and J. Gobien. A new analysis technique for probabilistic graphs. *IEEE Transactions on Circuits and Systems*, 26(10):855 – 865, 1979.

[13] T. Eilam-Tzoreff. The disjoint shortest paths problem. *Discrete Applied Mathematics*, 85(2):113–138, 1998.

[14] G. Gamrath and M. E. Lübbecke. Experiments with a generic dantzig-wolfe decomposition for integer programs. In P. Festa, editor, *Experimental Algorithms*, pages 239–252. Springer Berlin Heidelberg, 2010.

[15] M. Gen and Y. Yun. Soft computing approach for reliability optimization: State-of-the-art survey. *Reliability Engineering and System Safety*, 91:1008–1026, 2006.

[16] T. Gomes, J. Craveirinha, and L. Jorge. An effective algorithm for obtaining the minimal cost pair of disjoint paths with dual arc costs. *Computers & Operations Research*, 36(5):1670–1682, 2009.

[17] M. Grötschel, C. Monma, and M. Stoer. Design of survivable networks. In *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 617–672. Elsevier, 1995.

[18] L. Guo, H. Shen, and K. Liao. Improved approximation algorithms for computing k disjoint paths subject to two constraints. In *Computing and Combinatorics: 19th International Conference, COCOON 2013, Hangzhou, China, June 21-23, 2013. Proceedings*, pages 325–336. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[19] K. Heidtmann. Smaller sums of disjoint products by sub product inversion. *IEEE Transactions on Reliability*, 38:305 – 311, 1989.

[20] P. Helle, M. Masin, and L. Greenberg. Approximate reliability algebra for architecture optimization. In F. Ortmeier and P. Daniel, editors, *Computer Safety, Reliability, and Security*, pages 279–290. Springer Berlin Heidelberg, 2012.

[21] T.-J. Hsieh and W.-C. Yeh. Penalty guided bees search for redundancy allocation problems with a mix of components in seriesparallel systems. *Computers & Operations Research*, 39(11):2688 – 2704, 2012.

[22] C. Hwang and F. Tiliman. System-reliability evaluation techniques for complex/large systems-A review. *IEEE Transactions on Reliability*, 30(5):416 – 423, 1981.

[23] F. Iqbal and F. A. Kuipers. Disjoint paths in networks. *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 77–86, 2015.

[24] J. Kalvenes, J. Kennington, and E. Olinick. Base station location and service assignments in W–CDMA networks. *INFORMS Journal on Computing*, 18(3):366–376, 2006.

[25] F. Kuipers. An overview of algorithms for network survivability. *ISRN Communications and Networking*, 2012:1–19, 2012.

[26] W. Kuo and R. Prasad. System reliability optimization: An overview. In R. Soyer, T. A. Mazzuchi, and N. D. Singpurwalla, editors, *Mathematical Reliability: An Expository Perspective*, pages 31–54. Springer US, 2004.

[27] W. Kuo, V. Prasad, F. Tillman, and C. Hawang. Optimal reliability design fundamental and application. *Cambridge University Press*, 2001.

[28] W. Kuo and V. R. Prasad. An annotated overview of system-reliability optimization. *IEEE Transactions on Reliability*, 49(2):176–187, 2000.

[29] G. Levitin. Block diagram method for analyzing multi-state systems with uncovered failures. *Reliability Engineering and System Safety*, 92(6):727 – 734, 2007.

[30] X. Li, S. Zhou, X. Xu, L. Lin, and D. Wang. The reliability analysis based on subsystems of (n, k) -star graph. *IEEE Transactions on Reliability*, 65(4):1700–1709, 2016.

[31] Y. Li, H. Huang, Y. Liu, N. Xiao, and H. Li. A new fault tree analysis method: Fuzzy dynamic fault tree analysis. *Eksploatacja i Niezawodnosc-Maintenance and Reliability*, 14(3):208 – 214, 2012.

[32] Z. Liang, W. A. Chaovalitwongse, M. Cha, and S. B. Moon. Redundant multicast routing in multilayer networks with shared risk resource groups: Complexity, models and algorithms. *Computers & Operations Research*, 37(10):1731–1739, 2010.

[33] L. Lin, L. Xu, S. Zhou, and D. Wang. The reliability of subgraphs in the arrangement graph. *IEEE Transactions on Reliability*, 64(2):807–818, 2015.

[34] Y.-K. Lin and C.-F. Huang. Reliability of a multi-state computer network through k minimal paths within tolerable error rate and time threshold. *Quality and Reliability Engineering International*, 32:1393–1405, 2016.

[35] J. Linderoth and M. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS J. Comput.*, 11:171–187, 1999.

[36] C. Liu, N. Chen, and J. Yang. New method for multi-state system reliability analysis based on linear algebraic representation. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 229(5):469–482, 2015.

[37] A. Martin. *Integer programs with block structure*. habilitation, Technische Universität Berlin, 1998.

[38] Y. Mo, Y. Liu, and L. Cui. Performability analysis of multi-state series-parallel systems with heterogeneous components. *Reliability Engineering and System Safety*, 2017.

[39] Y.-F. Niu, Z.-Y. Gao, and W. H. Lam. A new efficient algorithm for finding all d-minimal cuts in multi-state networks. *Reliability Engineering and System Safety*, 166(Supplement C):151 – 163, 2017.

[40] H. Qi, Y. Fu, X. Qi, and Y. Lang. Architecture optimization of more electric aircraft actuation system. *Chinese Journal of Aeronautics*, 24:506–513, 2011.

[41] S. Rai, M. Veeraraghavan, and K. Trivedi. A survey of efficient reliability computation using disjoint products approach. *Networks*, 25(3):147 – 163, 1995.

[42] M. Ram and J. P. Davim. *Advances in Reliability and System Engineering*. Springer, 2017.

[43] J. E. Ramírez-Márquez and D. Coit. A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability. *Reliability Engineering and System Safety*, 87(2):253 – 264, 2005.

[44] M. Rausand. *Reliability of safety-critical systems: Theory and applications*. Wiley-IEEE, New York, 3rd edition, 2014.

[45] A. Rauzy, E. Chtelet, Y. Dutuit, and C. Brenguer. A practical comparison of methods to assess sum-of-products. *Reliability Engineering and System Safety*, 79(1):33 – 42, 2003.

[46] I. Rodríguez-Martín, J. Salazar-González, and H. Yaman. A branch-and-cut algorithm for two-level survivable network design problems. *Computers & Operations Research*, 67:102–112, 2016.

[47] F. S. Salman, R. Ravi, and J. N. Hooker. Solving the capacitated local access network design problem. *INFORMS Journal on Computing*, 20(2):243–254, 2008.

[48] L. Schäfer, S. García, and V. Srithammavanh. Simplification of inclusion-exclusion on intersections of unions with application to network systems reliability. *Reliability Engineering & System Safety*, 173:23 – 33, 2018.

[49] L. Schäfer, S. García, V. Srithammavanh, and A. Mitschke. Redundancy system design for an aircraft door management system. *Computers & Operations Research*, 94:11 – 22, 2018.

[50] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[51] S. Singh, A. Verma, S. Chatterjee, and V. Ramana. An efficient methodology to solve the k-terminal network reliability problem. In *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, pages 25 – 28, 2016.

[52] J. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.

[53] S. Twum and A. E. Models in design for reliability optimisation. *American Journal of Scientific and Industrial Research*, 4:95–110, 2013.

[54] D. Wojtaszek and J. Chinneck. Faster MIP solutions via new node selection rules. *Computers and Operations Research*, 37:1544–1556, 2010.

[55] L. A. Wolsey. *Integer Programming*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1998.

[56] N.-C. Xiao, L. Duan, and Z. Tang. Surrogate-model-based reliability method for structural systems with dependent truncated random variables. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 231(3):265–274, 2017.

[57] J. Xing, C. Feng, X. Qian, and P. Dai. A simple algorithm for sum of disjoint products. In *2012 Proceedings Annual Reliability and Maintainability Symposium*, pages 1 – 5, 2012.

[58] W.-C. Yeh. An improved sum-of-disjoint-products technique for the symbolic network reliability analysis with known minimal paths. *Reliability Engineering and System Safety*, 92(2):260 – 268, 2007.

[59] W.-C. Yeh. An improved sum-of-disjoint-products technique for symbolic multi-state flow network reliability. *IEEE Transactions on Reliability*, 64(4):1185 – 1193, 2015.

[60] J. Yuan and K. Ko. A factoring method to calculate reliability for systems of dependent components. *Reliability Engineering and System Safety*, 21(2):107 – 118, 1988.

[61] D. Zhong and Z. Qi. On the move to meaningful internet systems 2006: Otm 2006 workshops: Otm confederated international workshops and posters, awesome, cams, cominf, is, ksinbit, mios-ciao, monet, ontocontent, orm, persys, otm academy doctoral consortium, rdds, swws, and sebgis 2006, montpellier, france, october 29 - november 3, 2006. proceedings, part i. chapter A Petri Net Based Approach for Reliability Prediction of Web Services, pages 116 – 125. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

List of Figures

List of Tables

Appendices

Appendix A

Appendix for Chapter 2

A.1 Example for Lower Bound

In order to optimize the reliability of a system, lower bounds cannot be used. To use a lower bound, we need that the lower bound of the reliability of T_1 is lower than the lower bound of the reliability of T_2 if the reliability of a system T_1 is lower than the reliability of a system T_2. Otherwise, one cannot be sure that the optimal solution that it is obtained by using the lower bound is the optimal solution regarding the exact reliability. We did not find a lower bound that is suitable for optimization and fulfills that latter criterion.

We show an example of two systems where the reliability of one system is greater than the other, but the lower bound of the reliability of the former is smaller than the latter. This is the reason why it is not suitable to use in optimization. For the example, we use the following lower bound proposed by [10].

Let $A_1, \ldots, A_n$ be the considered events, $S_2 = \sum_{1 \leq i < j \leq n} P(A_i \cap A_j)$ and $S_1 = \sum_{k=1}^{n} P(A_k)$.

Theorem 5. *[10]*
Given a probability measure space $(\Omega, \mathbb{F}, P)$, let $A_k \in \mathbb{F}$, $k = 1, \ldots, n$.

$$P\left(\bigcup_{k=1}^{n} A_k\right) \geq \frac{\theta S_1^2}{2S_2 + (2-\theta)S_1} + \frac{(1-\theta)S_1^2}{2S_2 + (1-\theta)S_1} \tag{A.1}$$

with $\theta = 2S_2/S_1 - [2S_2/S_1]$.

Because it is not easy to define θ in an optimization constraint, we consider the minimum of the right-hand side which occurs with $\theta = 0$ and we obtain that

$$P\left(\bigcup_{k=1}^{n} A_k\right) \geq \frac{S_1^2}{2S_2 + S_1}. \tag{A.2}$$

Calculating the derivatives, it is easy to see that $\theta = 0$ gives the minimum of the right-hand side, if the lower bound given by the right-hand side is considered as function f of θ. It is easily seen that f is a concave function for $0 \leq \theta \leq 1$ and $f(0) = f(1)$. This gives you that $f(0) \leq f(\theta)$ for $0 \leq \theta < 1$.

For the example, let us assume that we have a system with one function and the function is implemented three times. Let A, B and C be the implementations. We can calculate for system T the exact reliability $P(T)$ as follows :

$$\begin{aligned} P(T) = P(A \cup B \cup C) &= P(A) + P(B) + P(C) \\ &\quad - P(A \cap B) - P(A \cap C) - P(B \cap C) \\ &\quad + P(A \cap B \cap C). \end{aligned}$$

We will also calculate a lower bound based on (A.2). Let T_1 and T_2 be as seen in Figures A.1 and A.2, respectively. Furthermore, let $P(1) = 0.5$, $P(2) = 0.7$, $P(3) = 0.2$, $P(4) = 0.6$ and $P(5) = 0.3$ be the probabilities of the unit not failing for T_1 and take $P(3) = 0.2521$ for T_2. We obtain that $P(T_1) = 0.2668$ and that $P(T_2) = 0.2668232$. Therefore, the reliability of T_2 is greater than the reliability of T_1. However, if we look at the lower bounds, we have that $P(T_1)_{lb} = 0.2260049$ and $P(T_2)_{lb} = 0.2257831$. Therefore, we can see that the lower bound of T_1 is greater than the lower bound of T_2. This shows us that the lower bounds are not monotone increasing with the reliability and therefore should not be used for exact optimization of reliability systems. Another example is system T_3 in Figure A.3. Let $1, 2, 4$ and 5 have the same probabilities of not failing as for T_1 and let $P(3) = 0.3$. We obtain that $P(T_3) = 0.261$ for T_3 which is smaller than $P(T_1)$ and $P(T_2)$. However, the lower bound of $P(T_3)$ is $P(T_3)_{lb} = 0.2278481$ which is greater than the lower bound of $P(T_1)$ and $P(T_2)$.

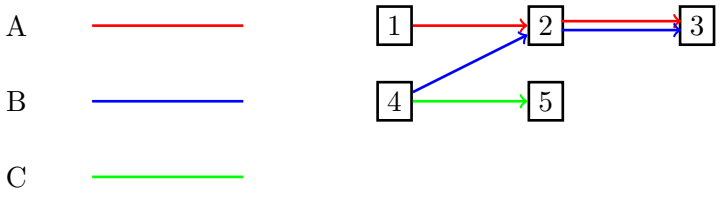

Figure A.1: System T_1

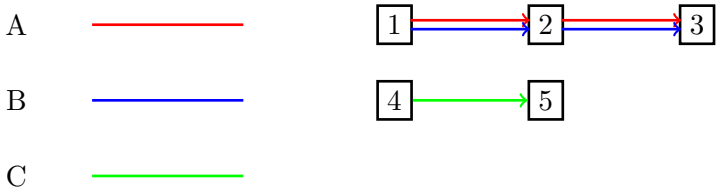

Figure A.2: System T_2

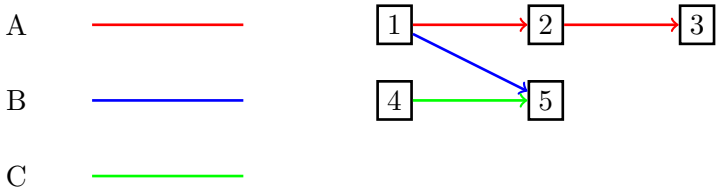

Figure A.3: System T_3

A.2 KDH88 Algorithm

Algorithm KDH88 was proposed [19]. The algorithm generates the sum of disjoint products recursively, using minsets (either minpaths or mincuts) as input data. In our computational tests we use minpaths and in the case of a system with multiple functions an implementations, a minpath is a set components $\bigcup_{i=1}^{n} T_{F_{iw_i}}$ for $w \in \{1, \ldots, t_1\} \times \ldots \times \{1, \ldots, t_n\}$ where $T_{F_{ij}}$ is defined as in Section 2.4.2. In KDH88, each minset contributes a set of disjoint terms, all of which are also disjoint with the prior minsets. The recursive buildup of the formula is accomplished with an inner loop within an outer loop. A recursion of the outer loop results in a portion of the system formula, representing the probability accounted for by a single minset. Each recursion includes a sequence of inner loop steps, one step for each prior minset, and each step possibly resulting in the modification of one or more terms. In the following, we will describe the algorithm and how we produced the minsets from our examples.

To create the minsets, we need the path for each implementation of every functions in the system. With this we also have the number of functions n and the number of implementations

for every function $t_1, \ldots, t_n$. Let $W = \{1, \ldots, t_1\} \times \ldots \times \{1, \ldots, t_n\}$. For every element $i \in W$, we have to create an array a_i whose length is the number of components o in the systems and is a minset/minpath. a_{ic} is 1 if component c is used in one of the implementations in i, otherwise takes value 0. Matrix A is the collection of all these arrays and has the dimension $|W| \times o$. The algorithm KDH88 creates all summands of the Sum of Disjoint Products formula. The algorithm is split into two functions which are *NextStep*, the main function with inner and outer loop, and *Mask*, a sub function called in *NextStep*. We will show both functions in Python code. Further details can be found in [19].

A.2.1 *NextStep* and *Mask*

The function *NextStep* needs the variables n, $j \in \{1, \ldots, i\}$, $i \in \{1, \ldots, |W|\}$, A and an array $v \in \mathbb{N}^o$. To create all summands, the function *NextStep* has to be called for every a_i, $i \in W$, with $NextStep(n, j, i, A, a_i)$. The functions in Python are

```
def NextStep(n,j,i,A,v): #NextStep function of KDH88 method
    dv=copy.deepcopy(v)
    if j<k:
        list_a=Mask(n,j,i,A,v)
        disjoint=list_a[2]
        if list_a[2]:
            NextStep(n,j+1,i,A,v)
        else:
            for s in xrange(1,list_a[0]+1):
                for i in xrange(0,n):
                    if v[k]==list_a[1][s-1] and A[j-1][k]<0:
                        dv[k]=-1
                NextStep(n,j+1,i,A,dv)
                for k in xrange(0,n):
                    if v[k]==list_a[1][s-1]:
                        if A[j-1][k]==0:
                            dv[k]=0
                        else:
                            dv[k]=list_a[1][s-1]
            for k in xrange(0,n):
                if(v[k]<0) and A[j-1][k]==0:
                    dv[k]=j
                    disjoint=True
            if disjoint:
                NextStep(n,j+1,i,A,dv)
    else:
        print("Summand is ", v)

def Mask(n,j,k,beg_matr,pro): #Mask function of KDH88 method
    disjoint=False
    r=0
    s=1
    b=False
    c=False
    mask=np.zeros((n,),dtype=np.int)
    while not disjoint and (s<j):
        for i in xrange(0,n):
            if pro[i]==s:
                if beg_matr[j-1][i]==0:
```

```
                b=True
            else:
                c=True
    if b:
        if not c:
            disjoint=True
        else:
            r+=1
            mask[r-1]=s
    b=False
    c=False
    s+=1
return([r,mask,disjoint])
```

A.3 Examples of Door Management Systems

In this section, we show an example input for DMS with two doors and three paths. Data for all other systems can be found at `https://sites.google.com/site/sergiogarciaquiles/`. They are given as CSV files where each row represents a path and each column represents a component or connection. The rows represent the paths are ordered by doors. For every row i, we have at column j a 1 if the component/connection represented by column j is used for the path that is represented by row i and 0 otherwise. The Python code we used to run the computational tests has also been uploaded to this repository.

The example system has 70 elements of which 27 are components and 43 are connections. Table A.1 shows what positions t_i and connections x_i are used in which path for every door. Failure probabilities of the components and connections are also provided.

$t\backslash x$	$(1,1)$	$(1,2)$	$(1,3)$	$(2,1)$	$(2,2)$	$(2,3)$	Prob.
t_1	1	1	1	0	0	0	0.020
t_2	0	0	0	1	1	1	0.020
t_3	0	0	0	0	0	1	0.020
t_4	0	0	0	0	1	0	0.050
t_5	0	0	1	0	0	0	0.030
t_6	1	0	0	0	0	0	0.010
t_7	0	0	1	0	0	0	0.030
t_8	0	0	0	0	1	0	0.050
t_9	0	0	0	1	0	0	0.090
t_{10}	0	1	0	0	0	0	0.090
t_{11}	0	0	0	1	0	0	0.060
t_{12}	1	0	0	0	0	0	0.020
t_{13}	0	1	0	0	0	0	0.090
t_{14}	0	0	0	0	0	1	0.010
t_{15}	1	0	0	1	0	1	0.090
t_{16}	0	1	1	0	1	0	0.070
t_{17}	0	0	1	0	1	0	0.090
t_{18}	0	1	1	1	0	0	0.020
t_{19}	1	0	1	0	1	1	0.010
t_{20}	1	0	0	0	0	1	0.060
t_{21}	0	1	0	1	0	1	0.080
t_{22}	0	1	0	1	0	0	0.050
t_{23}	1	0	0	0	1	0	0.060
t_{24}	0	1	0	1	0	0	0.090
t_{25}	1	0	1	0	1	1	0.080
t_{26}	1	0	1	0	1	1	0.020
t_{27}	0	1	0	1	0	0	0.050
x_1	0	0	1	0	0	0	0.004
x_2	1	0	0	0	0	0	0.004
x_3	0	0	1	0	0	0	0.004
x_4	0	1	0	0	0	0	0.004
x_5	1	0	0	0	0	0	0.004
x_6	0	1	0	0	0	0	0.004
x_7	0	0	0	0	0	1	0.004

x_8	0	0	0	0	1	0	0.004
x_9	0	0	0	0	1	0	0.004
x_{10}	0	0	0	1	0	0	0.004
x_{11}	0	0	0	1	0	0	0.004
x_{12}	0	0	0	0	0	1	0.004
x_{13}	0	0	0	0	0	1	0.004
x_{14}	0	0	0	0	1	0	0.004
x_{15}	0	0	1	0	0	0	0.004
x_{16}	1	0	0	0	0	0	0.004
x_{17}	0	0	1	0	0	0	0.004
x_{18}	0	0	0	0	1	0	0.004
x_{19}	0	0	0	1	0	0	0.004
x_{20}	0	1	0	0	0	0	0.004
x_{21}	0	0	0	1	0	0	0.004
x_{22}	1	0	0	0	0	0	0.004
x_{23}	0	1	0	0	0	0	0.004
x_{24}	0	0	0	0	0	1	0.004
x_{25}	0	0	1	0	1	0	0.004
x_{26}	0	1	0	1	0	0	0.004
x_{27}	0	0	1	0	0	0	0.004
x_{28}	0	1	0	1	0	0	0.004
x_{29}	0	0	1	0	1	0	0.004
x_{30}	1	0	0	0	0	1	0.004
x_{31}	1	0	1	0	1	1	0.004
x_{32}	1	0	0	0	0	1	0.004
x_{33}	0	0	0	0	0	1	0.004
x_{34}	0	1	0	1	0	0	0.004
x_{35}	0	0	0	1	0	0	0.004
x_{36}	0	1	0	0	0	0	0.004
x_{37}	1	0	0	0	1	0	0.004
x_{38}	0	1	0	1	0	0	0.004
x_{39}	1	0	1	0	1	1	0.004
x_{40}	1	0	1	0	1	1	0.004
x_{41}	1	0	1	0	1	1	0.004
x_{42}	0	1	0	1	0	0	0.004
x_{43}	0	1	0	1	0	0	0.004

Table A.1: Example systems data

Appendix B

Appendix for Chapter 3

B.1 MILP Solution Methods

In this section, we will present shortly two general MILP solving algorithms. The first one is the general branch-and-bound algorithm and the second one is the general branch-and-price. A general introduction to MIP can be found in [55].

B.1.1 Branch-and-bound Algorithm

First we introduce the basic branch-and-bound algorithm based on linear programming and give a few examples for improvements of the basic algorithm. The idea of branch-and-bound is a divide and conquer approach. For dividing a problem we use the following. Let $z = \max\{c^t x : x \in S\}$ be a MIP problem. The basic algorithm is the same for IP problems. We can break this problem up into smaller subproblems.

Proposition 6. *Let $S = S_1 \cup \ldots \cup S_K$ be a decomposition of S into smaller sets, and let $z^k = \max\{c^t x : x \in S_k\}$ for $k = 1, \ldots, K$. Then $z = \max_k z^k$.*

The branch-and-bound algorithm uses the decomposition, not to obtain easier problems, but to attain a feasible solution for the original problem by solving the LP relaxations.

Let z optimal value of the problem and $\underline{z}$ a lower bound to z that is set to $-\infty$ at the start. We see the original problem as the root and first node of an enumeration tree. Let $\mathcal{N}$ be the set of unsolved nodes in the tree. The first step of the algorithm is to choose a node from $\mathcal{N}$, which is at the beginning always the root, and solve its LP relaxation. After solving the LP relaxation, we can have four different results.

The first one is that the node problem is infeasible, in which case we discard the node from $\mathcal{N}$ and choose a new node. The second one is that the optimal solution is lower than the lower bound $\underline{z}$, in which case we also discard the node from $\mathcal{N}$ and choose a new node. The third type is that the optimal solution of the node problem is feasible for MIP and greater than the already found feasible solutions. We save the feasible solution as an incumbent optimal solution of the MIP x^*, set $\underline{z}$ equal to the optimal objective value, discard the node from $\mathcal{N}$ and choose a new node. These three types are called *prune by infeasibility*, *prune by bound* and *prune by optimality*, respectively. The last type is that we get an optimal solution for the LP relaxation that is infeasible for the MIP. We then give new bounds on a variable that decompositions the node problem and creates two new subproblems (nodes) which are added to $\mathcal{N}$. The solved node is discarded and a new node chosen. Figure B.1 shows the basic branch-and-bound algorithm in a flow chart.

There are two choices in the branch-and-bound algorithm: How to split the problem which is called *branching*, and which subproblem (node) to select. First we analyze the branching decision.

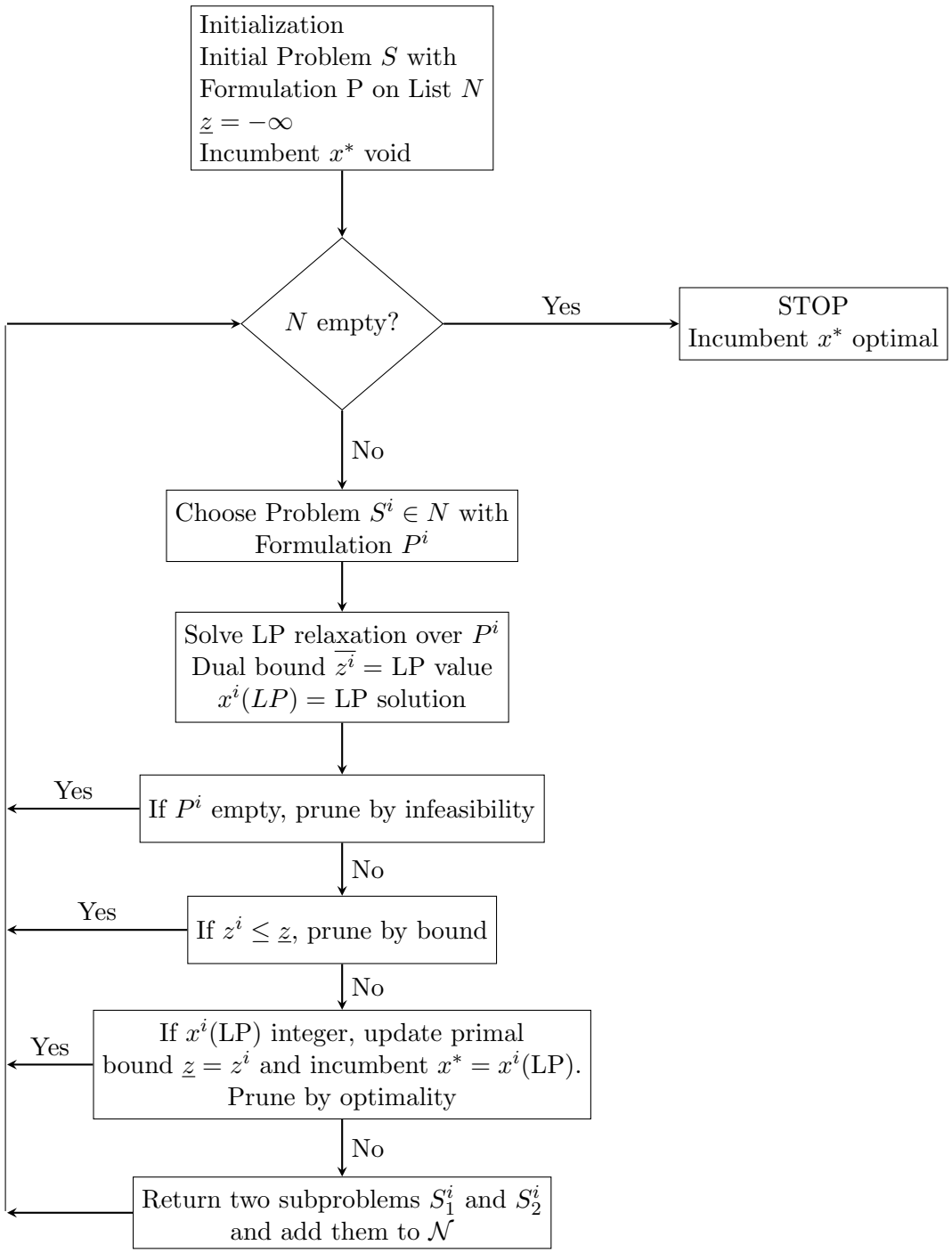

Figure B.1: Branch-and-bound flow chart

B.1.1.1 Branching

Branching is concerned with splitting the problem in a branch-and-bound algorithm. It is important to find efficient strategies for branching to reduce the number of nodes needed to get the optimal solution for the MIP. Therefore, efficient strategies become even more important for large scale optimization problems in MIP. In the following we will present the most popular rules that are used in today MIP solvers for branching such as CPLEX. In these strategies, the algorithm branches on a linear inequality that splits the feasible interval of a single variable.

Let j be some index so that $x_j^i \notin \mathbb{Z}$ in the current optimal LP solution of the problem S^i. We obtain the two subproblems $S_{j,-}^i$ and $S_{j,+}^i$ by adding the inequalities $x_j \leq \lfloor x_j^i \rfloor$ and $x_j \geq \lceil x_j^i \rceil$, respectively. The first subproblem is called left son and the second one right son. There are some branching strategies that use more complicated inequalities or they split the problem into more

than two subproblems. However, these are seldom used.

To decide on which such j we branch the problem, we first give a generic variable selection algorithm with a score function. The score function depends on the strategy used.

Algorithm 7. *Generic Variable Selection.*
Input: Current Subproblem S^i with an optimal LP solution x^i which is infeasible for the MIP.
Output: An index $j \in J$ of a fractional variable $x^i_j \notin \mathbb{Z}$.

1. *Let $C = \{j \in J |\ x^i_j \notin \mathbb{Z}\}$ be the set of branching candidates.*

2. *For all candidates $j \in C$, calculate a score value $t_j \in \mathbb{R}$.*

3. *Return an index $j \in C$ with $t_j = \max_{k \in C}\{t_k\}$.*

The following explanations are symmetric for the left S^i_- and right S^i_+ problems, which is why we explain it only with right sons.

1. Most infeasible branching: This is the most basic and still very common strategy. It has the score function $t_j = 0.5 - |x^i_j - \lfloor x^i_j \rfloor - 0.5|$, which means it chooses the variable with the fractional part closest to 0.5. The reason behind this choice it that this selects a variable where the least tendency can be recognized to which "side" (up or down) the variable should be rounded. Results in [3] indicate that this strategy is in general not better than selecting the variable randomly.

2. Pseudocost branching: This strategy works with the history of the success of variables on which we have already branched. We present the *pseudocost branching* from [37]. For alternatives, see [35]. We need some notation for the strategy. Let $\overline{c}_S$ be the optimal value of a problem S. We set $f^+_j = \lceil x^i_j \rceil - x^i_j$, $\Delta^+_j = \overline{c}_{S^i_{j,+}} - \overline{c}_{S^i}$ and let ς^+_j be the objective gain per unit change in variable j at node S^i with $\varsigma^+_j = \Delta^+_j / f^+_j$.
   Let σ^+_j denote the sum of ς^+_j over all problems (nodes) S^i, where j has been selected as branching variable and $S^i_{j,+}$ has been solved and was feasible. Let η^+_j be the number of these problems (nodes). The pseudocosts for the upward branching of variable j are

   (B.1) $$\Psi^+_j = \sigma^+_j / \eta^+_j.$$

   This provides us the scoring function

   (B.2) $$s_j = (1-\mu) * \min\{f^-_j \Psi^-_j, f^+_j \Psi^+_j\} + \mu * \max\{f^-_j \Psi^-_j, f^+_j \Psi^+_j\},$$

   which can be used in Algorithm 7 and results in the pseudocost branching. At the beginning of the branch-and-bound algorithm $\sigma^+_j = \eta^+_j = 0$ for all j. We call the pseudocost of a variable j *uninitialized* for the upward direction, if $\eta^+_j = 0$. Uninitialized upward pseudocosts are set to $\Psi^+_j = \Psi^+_{avg}$, where Ψ^+_{avg} is the average of the initialized upward pseudocosts over all variables. The average is set to 1, if all upward pseudocosts are uninitialized. The pseudocost is uninitialized if it is uninitialized in at least one direction.

   Because the strategy is based on the history of the successes, it has its weakness at the beginning of the branch-and-bound algorithm. Because of that it is often combined with another strategy that is strong at the beginning, see 4.

3. Strong branching: In *strong branching*, we test which of the fractional candidates provides the best progress before actually branching on any of them. This is done by giving a temporary lower bound $\lfloor x^i_j \rfloor$ and subsequently an upper bound $\lceil x^i_j \rceil$ for a variable j with

fractional LP value x_j^i and solving the linear relaxations for these. If we did that for all variables in C and solve the LPs to optimality, it is called *full strong branching*. Because the computational time is very high for full strong branching, we have two possibilities to speed it up. One is to restrict the possible variables to $C' \subset C$. Another way is to only use a few simplex iterations. This is possible, because the change of the objective function in the simplex algorithm usually decreases with the iterations. Full strong branching with restrictions is then called strong branching. In the following we will introduce a combination of strong branching and pseudocost branching.

4. Combinations of strong and pseudocost branching: The simplest combination is the hybrid strong/pseudocost branching. In this strategy, the strong branching is applied in the upper part of the branch-and-bound tree up to a given depth level d. Afterwards pseudocost branching is used. An extension on this strategy is the pseudocost branching with strong branching initialization. Because even in the hybrid strong/pseudocost branching strategy the decisions of pseudocost in the lower part of the tree are potentially based on uninitialized pseudocost values, the pseudocost branching with strong branching initialization uses strong branching for variables with uninitialized pseudocost and uses the resulting estimates to initialize the pseudocost. This results in a more dynamic use of strong branching.

A new strategy called reliability branching is proposed in [3]. It uses the idea of pseudocost branching with strong branching initialization, but does the strong branching not only on variables with uninitialized pseudocost values, but also on variables with unreliable pseudocosts. The pseudocost of a variable j is called unreliable, if $\min\{\eta_j^+, \eta_j^-\} < \eta_{rel}$, with η_{rel} being the "reliability" parameter. With this we get an even more dynamic way of using strong branching than in the previous strategies. The computational results of [3] give the indication that a more intensive dynamic use of strong branching, like the reliability branching, gives a significant improvement in both the nodes needed to solve the MIP and the time needed to solve the considered problem instances.

B.1.1.2 Node Selection

The second choice after branching you need to take in your branch-and-bound algorithm is the selection of the next node. This also affects how quickly the MIP is solved by branch-and-bound, because the number of evaluated nodes can be reduced. The two basic selection methods for choosing the next node are the Depth-First search and the Best-Bound search.

In the Depth-First search you choose the right or left son of the most recently explored node and there are different heuristics to choose between the sons. If the last solved node is either LP infeasible, MIP feasible, or worse than the incumbent solution, then the last created node which is not yet explored will be chosen.

In the Best-Bound method, the active node with the best upper bound, which is the LP relaxation optimal objective value of the parent node, will be chosen.

In practice, a compromise between these two methods is often adopted, i.e. initial depth-first method until at least one feasible solution has been found and then a mix between these methods. A small introduction into node selection is given in [54] and also new selection methods are proposed.

B.1.2 Branch-and-Price

Decomposition and reformulation of MIPs are classical approaches to obtaining stronger relaxations and reduce symmetry. One of these approaches is column generation that solves linear programming (LP) problems. If column generation is used for the general branch-and-bound algorithm, the algorithm is called branch-and-price. As we also have symmetry issues in our problem and will try to use branch-and-price to solve it.

We will give a short overview of column generation and the decomposition/reformulation that is needed for most problems to use column generation. For a more detailed introduction to column generation and branch-and-price, see [11].

- Column Generation: Let us consider the integer problem

$$\begin{aligned} &\min \sum_{j\in J} c_j\lambda_j \\ &\text{subject to} \sum_{j\in J} a_j\lambda_j \le b,\ \lambda \in \mathbb{Z}_+^{|J|} \end{aligned} \tag{B.3}$$

  If J is huge, many standard MILP solver have a problem, because the linear relaxation cannot be solved. Therefore the linear relaxation of (B.3), which we call *master problem*, by column generation. First a *restricted master problem* (RMP) which contains a subset of $J' \subseteq J$ of variables is solved and we obtain λ^* and π^*, the primal and dual optimal solution, respectively. Afterwords we solve the pricing problem $v := \min_{x\in X}\{c(x) - \pi a(x)\}$, where $c_j = c(x_j)$ and $a_j = a(x_j)$ reflect that each column $j \in J$ is associated with an element $x_j \in X$ from a domain X over which we can optimize, often a set of combinatorial objects like paths or other subgraphs. When $v < 0$ the variable λ_j and its coefficient column (c_j, a_j) corresponding to a minimizer x_j are added to the RMP and we start again with solving the RMP. If $v \ge 0$, it proves that there is no such improving variable and the current λ^* is an optimal solution to the master problem.

- Dantzig-Wolfe Decomposition: To exploit certain special structure in a linear problem, the Dantzig-Wolfe decomposition principle was devised. Let us consider the

$$\begin{aligned} z^* &:= \min cx \\ \text{subject to } Ax &\le b \\ x &\in X = \{x \in \mathbb{Z}_+^n \times \mathbb{Q}_+^q \,|\, Dx \le D\} \end{aligned} \tag{B.4}$$

  The decomposition build on the representation theorems by Minkowski and Weyl [50] and convexifies X. Each $x \in X$ can be expressed as a convex combination of finitely many extreme points $\{x_p\}_{p\in P}$ plus a non-negative combination of finitely many extreme rays $\{x_r\}_{r\in R}$ of $conv(X)$:

$$\forall x \in X\ \exists \lambda \in \mathbb{Q}_+^{|P|+|R|} :\ x = \sum_{p\in P} x_p\lambda_p + \sum_{r\in R} x_r\lambda_r,\ \sum_{p\in P}\lambda_p = 1. \tag{B.5}$$

  Using this and applying $c_j = cx_j$ and $a_j = Ax_j$, $j \in P \cup R$, we can rewrite problem B.4 to

$$\begin{aligned} \min &\sum_{p\in P} c_p\lambda_p + \sum_{r\in R} c_r\lambda_r \\ \text{subject to } &\sum_{p\in P} a_p\lambda_p + \sum_{r\in R} a_r\lambda_r \le b \\ &\sum_{p\in P}\lambda_p = 1 \\ &\lambda \ge 0 \\ &x = \sum_{p\in P} x_p\lambda_p + \sum_{r\in R} x_r\lambda_r \\ &x \in \mathbb{Z}_+^n \times \mathbb{Q}_+^q \end{aligned}$$

  With this formulation, we have a problem like (B.3) and can use column generation, where the constrains linking x and λ variables can be dropped. Thus, only the dual variables

π and π_0 remain relevant where π_0 corresponds to the convexity constraints. For column generation, the pricing problem is $\min\{cx - \pi Ax - \pi_o |\ x \in X\}$.

- Block diagonal structure: A block diagonal structure of D can be exploited by Dantzig-Wolfe decomposition. Let

$$D = \begin{pmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^{\mathfrak{k}} \end{pmatrix},\ d = \begin{pmatrix} d^1 \\ d^2 \\ \vdots \\ d^{\mathfrak{k}} \end{pmatrix}.$$

We can write $X = \bigtimes_{k \in K} X^k$, with $X^k = \{D^k x^k \geq d^k, x^k \geq 0\}$, $k \in K := \{1, \ldots, \mathfrak{k}\}$. We have a representation as in (B.5) for each $X^k,\ k \in K$, and have also $\mathfrak{k}$ pricing problems, each with its own convexity constraint constraint and associated dual variable $\min\{c^k x^k - \pi A^k x^k - \pi_o^k |\ x^k \in X^k\},\ k \in K$.

B.2 Negative Validity Example

In this example, we have $|D| = 3,\ |P| = 2$ and $|F| = 2$. There are four different unit types $U = \{U_1,\ U_2,\ U_3,\ U_4\}$. A unit of type U_1 cannot be used for different doors. All needed information of the units is given in Tables B.1 and B.2 for functions f_1 and f_2, respectively.

Unit types	Model	E_{um}	E_{fu}^{in}	E_{fu}^{out}	$T_u^{f,in}$	$T_u^{f,out}$	$C_f^+(u)$	$C_f^-(u)$
U_1	M_1	2	1	1	1	1	$\{U_2\}$	$\emptyset$
U_2	M_1	4	1	1	1	1	$\{U_3\}$	$\emptyset$
U_2	M_2	5	1	1	1	1	$\{U_3\}$	$\emptyset$
U_3	M_1	8	1	1	1	1	$\emptyset$	$\emptyset$

Table B.1: Unit parameters and corresponding sets for function f_1.

Unit types	Model	E_{um}	E_{fu}^{in}	E_{fu}^{out}	$T_u^{f,in}$	$T_u^{f,out}$	$C_f^+(u)$	$C_f^-(u)$
U_2	M_1	4	1	1	1	1	$\emptyset$	$\{U_3\}$
U_2	M_2	5	1	1	1	1	$\emptyset$	$\{U_3\}$
U_3	M_1	8	1	1	1	1	$\emptyset$	$\emptyset$
U_4	M_1	4	1	1	1	1	$\emptyset$	$\{U_2\}$

Table B.2: Unit parameters and corresponding sets for function f_2.

Tables B.3 to B.8 give a feasible linear relaxation solution for the small example and Figure B.2 illustrates it in a graph. To distinguish the different functions in Figure B.2 we have used thicker arrows for function f_2. As can be seen, 2-redundancy is not given and constraint (3.14) is fully satisfied for units of type U_2. Even though the system is feasible in the linear relaxation, for this combination of units there are no cable connections which are feasible in the integer formulation and unit type U_2. If such a case as here is discovered through the validity test, we can then branch on the corresponding t_{um} variables. In this example, we would branch with $t_{U_2,M_1} \geq 3$ and $t_{U_2,M_1} \leq 1$ or $t_{U_2,M_2} \geq 2$ and $t_{U_2,M_2} \leq 0$.

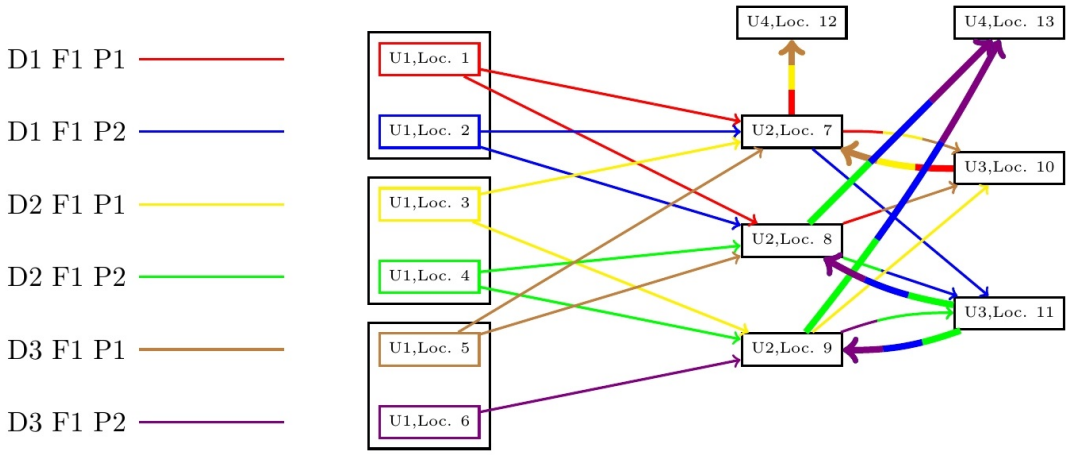

Figure B.2: Illustration of liner relaxation solution. Solution can be found in Tables B.7 and B.9.

i	1	2	3	4	5	6	7	8	9	10	11	12	13
u	U_1	U_1	U_1	U_1	U_1	U_1	U_2	U_2	U_2	U_3	U_3	U_4	U_4
t_{i,u,M_1}	1	1	1	1	1	1	0	1	1	1	1	1	1
t_{i,u,M_2}	0	0	0	0	0	0	1	0	0	0	0	0	0

Table B.3: Linear relaxation solution values for location variables of the overall system.

i	1	1	2	2	3	3	4	4	5	5	6	7	7	8	8	9	9
j	7	8	7	8	7	9	8	9	7	8	9	10	11	10	11	10	11
u	U_1	U_1	U_1	U_1	U_1	U_1	U_1	U_1	U_1	U_1	U_1	U_2	U_2	U_2	U_2	U_2	U_2
$\hat{u}$	U_2	U_2	U_2	U_2	U_2	U_2	U_2	U_2	U_2	U_2	U_2	U_3	U_3	U_3	U_3	U_3	U_3
$x_{iju\hat{u}}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1	0.5	0.5	0.5	0.5	0.5	1

Table B.4: Linear relaxation solution values for connection variables of the overall system.

i	7	7	8	8	9	9	10	10	10	11	11	11
j	12	13	12	13	12	13	7	8	9	7	8	9
u	U_2	U_2	U_2	U_2	U_2	U_2	U_3	U_3	U_3	U_3	U_3	U_3
$\hat{u}$	U_4	U_4	U_4	U_4	U_4	U_4	U_2	U_2	U_2	U_2	U_2	U_2
$x_{iju\hat{u}}$	1	0	0	0.5	0	0.5	1	0	0	0	0.5	0.5

Table B.5: Linear relaxation solution values for connection variables of the overall system.

i	u	t_{d_1,f_1,p_1}	t_{d_1,f_1,p_2}	t_{d_2,f_1,p_1}	t_{d_2,f_1,p_2}	t_{d_3,f_1,p_1}	t_{d_3,f_1,p_2}
7	U_2	0.5	0.5	0.5	0	0.5	0
8	U_2	0.5	0.5	0	0.5	0.5	0
9	U_2	0	0	0.5	0.5	0	1
10	U_3	1	0	1	0	1	0
11	U_3	0	1	0	1	0	1
12	U_4	1	0	1	0	1	0
13	U_4	0	1	0	1	0	1

Table B.6: Linear relaxation solution values for location variables and function f_2.

i	u	t_{d_1,f_1,p_1}	t_{d_1,f_1,p_2}	t_{d_2,f_1,p_1}	t_{d_2,f_1,p_2}	t_{d_3,f_1,p_1}	t_{d_3,f_1,p_2}
1	U_1	1	0	0	0	0	0
2	U_1	0	1	0	0	0	0
3	U_1	0	0	1	0	0	0
4	U_1	0	0	0	1	0	0
5	U_1	0	0	0	0	1	0
6	U_1	0	0	0	0	0	1
7	U_2	0.5	0.5	0.5	0	0.5	0
8	U_2	0.5	0.5	0	0.5	0.5	0
9	U_2	0	0	0.5	0.5	0	1
10	U_3	1	0	1	0	1	0
11	U_3	0	1	0	1	0	1

Table B.7: Linear relaxation solution values for location variables and function f_1.

i	j	u	$\hat{u}$	x_{d_1,f_1,p_1}	x_{d_1,f_1,p_2}	x_{d_2,f_1,p_1}	x_{d_2,f_1,p_2}	x_{d_3,f_1,p_1}	x_{d_3,f_1,p_2}
7	12	U_2	U_4	1	0	1	0	1	0
7	13	U_2	U_4	0	0	0	0	0	0
8	12	U_2	U_4	0	0	0	0	0	0
8	13	U_2	U_4	0	0.5	0	0.5	0	0.5
9	12	U_2	U_4	0	0	0	0	0	0
9	13	U_2	U_4	0	0.5	0	0.5	0	0.5
10	7	U_3	U_2	1	0	1	0	1	0
10	8	U_3	U_2	0	0	0	0	0	0
10	9	U_3	U_2	0	0	0	0	0	0
11	7	U_3	U_2	0	0	0	0	0	0
11	8	U_3	U_2	0	0.5	0	0.5	0	0.5
11	9	U_3	U_2	0	0.5	0	0.5	0	0.5

Table B.8: Linear relaxation solution values for connection variables for function f_2.

i	j	u	$\hat{u}$	$x_{iju\hat{u}}$	x_{d_1,f_1,p_1}	x_{d_1,f_1,p_2}	x_{d_2,f_1,p_1}	x_{d_2,f_1,p_2}	x_{d_3,f_1,p_1}	x_{d_3,f_1,p_2}
1	7	U_1	U_2	0.5	0.5	0	0	0	0	0
1	8	U_1	U_2	0.5	0.5	0	0	0	0	0
2	7	U_1	U_2	0.5	0	0	0	0	0	0
2	8	U_1	U_2	0.5	0	0.5	0	0	0	0
3	7	U_1	U_2	0.5	0	0	0.5	0	0	0
3	9	U_1	U_2	0.5	0	0	0.5	0	0	0
4	8	U_1	U_2	0.5	0	0	0	0.5	0	0
4	9	U_1	U_2	0.5	0	0	0	0.5	0	0
5	7	U_1	U_2	0.5	0	0	0	0	0.5	0
5	8	U_1	U_2	0.5	0	0	0	0	0.5	0
6	9	U_1	U_2	1	0	0	0	0	0	1
7	10	U_2	U_3	0.5	0.5	0	0.5	0	0.5	0
7	11	U_2	U_3	0.5	0	0.5	0	0	0	0
8	10	U_2	U_3	0.5	0.5	0	0	0	0.5	0
8	11	U_2	U_3	0.5	0	0.5	0	0.5	0	0
9	10	U_2	U_3	0.5	0	0	0.5	0	0	0
9	11	U_2	U_3	1	0	0	0	0.5	0	1

Table B.9: Linear relaxation solution values for connection variables.

Appendix C

Appendix for Chapter 4

C.1 Lower and Upper Bounds with the Number of Models and Possible Combinations

Units	LB	UB	#Models	Combinations
2	2	2	2	3
3	2	2	2	3
4	1	2	1	2
5	1	3	2	9
6	2	8	2	42
7	1	2	1	2
8	1	4	2	14
			#Comb. :	190512

Figure C.1: Combinations for k=2

Units	LB	UB	#Models	Combinations
2	3	3	2	4
3	3	3	2	4
4	2	3	1	2
5	2	3	2	7
6	4	10	2	56
7	2	3	1	2
8	2	4	2	12
			#Comb. :	301056

Figure C.2: Combinations for k=3

Units	LB	UB	#Models	Combinations
2	3	3	2	4
3	3	3	2	4
4	2	3	1	2
5	2	3	2	7
6	4	10	2	56
7	2	3	1	2
8	2	4	2	12
			#Comb. :	301056

Figure C.3: Combinations for k=4

Units	LB	UB	#Models	Combinations
2	4	5	2	11
3	4	5	2	11
4	2	3	1	2
5	2	3	2	7
6	4	10	2	56
7	2	3	1	2
8	2	4	2	12
			#Comb. :	2276736

Figure C.4: Combinations for k=5

Units	LB	UB	#Models	Combinations
2	4	5	2	11
3	4	5	2	11
4	2	3	1	2
5	2	3	2	7
6	4	10	2	56
7	2	3	1	2
8	2	4	2	12
			#Comb. :	2276736

Figure C.5: Combinations for k=6

C.2 Parameter and Coefficient Table for Computational Tests

Unit	Model	Ports	Coeff. weight set 1	Coeff. weight set 2	Failure Rate
DO	1	20	0	0	0
LLS	1	2	10	1.5	0.02
	2	2	25	2.5	0.01
CS	1	2	2	2	0.002
	2	2	10	10	0.001
OVF	1	4	80	100	0.001
	2	3	70	60	0.002
OCU	1	6	20	100	0.01
	2	6	30	120	0.005
RDC	1	6	100	70	0.002
	2	8	90	100	0.003
CPIOM	1	4	170	160	0.005
	2	4	150	155	0.007
SWT	1	24	170	120	0.005
	2	16	175	150	0.004
Cable	1	0	2	0.05	0.00001

Table C.1: Parameter and coefficients for units and cables used in reliability heuristic.

C.3 Computation Tables for Reliability Heuristic

$\|D\|$	Reliability Threshold	Nodes	Time (s)	Optimal Value	Reliability	Value Range	Reliability Range
2	0.999	51	1214	1466	0.99904	1466-1652	0.99900-0.99904
2	0.9991	37	999	1486	0.99910	1486-1624	0.99910-0.99914
2	0.9992	33	887	1528	0.99922	1528-1648	0.99920-0.99926
2	0.9993	102	1383	1551	0.99932	1551-1652	0.99930-0.99932
2	0.9994	71	1789	1646	0.99940	1646-1848	0.99937-0.99944
2	0.9995	55	1634	1725	0.99951	1725-1879	0.99950-0.99953
2	0.9996	52	1897	1810	0.99960	1810-1990	0.99960-0.99964
2	0.9997	29	1056	1901	0.99972	1901-2045	0.99970-0.99973
2	0.9998	22	869	2004	0.99980	2004-2172	0.99980-0.99980
2	0.9999	9	437	2118	0.99991	2118-2118	0.99991-0.99991
3	0.999	102	5352	1759	0.99901	1759-1962	0.99898-0.99921
3	0.9991	83	5466	1806	0.99914	1806-1903	0.99908-0.99921
3	0.9992	78	5544	1836	0.99921	1836-2042	0.99917-0.99923
3	0.9993	80	4946	1855	0.99932	1855-2024	0.99924-0.99937
3	0.9994	58	4610	1881	0.99940	1881-2088	0.99935-0.99960
3	0.9995	47	4069	1959	0.99950	1959-2108	0.99950-0.99960
3	0.9996	34	3018	2004	0.99960	2004-2208	0.99960-0.99965
3	0.9997	21	2307	2049	0.99970	2049-2182	0.99964-0.99970
3	0.9998	9	1161	2138	0.99980	2138-2138	0.99980-0.99980
3	0.9999	9	1136	NA	NA	NA	NA
4	0.999	72	3153	1830	0.99900	1830-2012	0.99900-0.99902
4	0.9991	48	4498	1897	0.99910	1897-2116	0.99910-0.99950
4	0.9992	54	4369	1922	0.99920	1922-2121	0.99920-0.99950
4	0.9993	54	4537	1974	0.99931	1974-2156	0.99930-0.99950

4	0.9994	27	2135	1998	0.99940	1998-2064	0.99940-0.99950
4	0.9995	26	2315	2058	0.99950	2058-2104	0.99950-0.99950
4	0.9996	31	2443	2099	0.99960	2099-2387	0.99960-0.99962
4	0.9997	9	1193	2174	0.99970	2174-2174	0.99970-0.99970
4	0.9998	9	1184	2258	0.99980	2258-2258	0.99980-0.99980
4	0.9999	9	1119	NA	NA	NA	NA
5	0.999	42	8085	2017	0.99906	2017-2148	0.99902-0.99939
5	0.9991	44	7922	2037	0.99909	2037-2148	0.99909-0.99939
5	0.9992	46	7702	2101	0.99920	2101-2159	0.99920-0.99939
5	0.9993	23	5191	2147	0.99932	2147-2148	0.99932-0.99939
5	0.9994	23	5208	2153	0.99940	2153-2172	0.99940-0.99940
5	0.9995	29	4977	2154	0.99951	2154-2279	0.99950-0.99951
5	0.9996	28	4958	2229	0.99960	2229-2431	0.99960-0.99960
5	0.9997	14	2991	2304	0.99970	2304-2304	0.99970-0.99970
5	0.9998	9	2616	2424	0.99980	2424-2424	0.99980-0.99980
5	0.9999	1	754	NA	NA	NA	NA

Table C.2: Results for second set of w_{um} and w_c

$\|D\|$	Reliability Threshold	Nodes	Time (s)	Optimal Value	Reliability	Value Range	Reliability Range
2	0.998	52	1050	1382	0.99806	1382-1420	0.99801-0.99834
2	0.9981	54	1140	1387	0.99813	1387-1420	0.99810-0.99839
3	0.998	68	4780	1461	0.99803	1461-1580	0.99800-0.99813
3	0.9981	77	5765	1495	0.99817	1495-1610	0.99809-0.99819
3	0.9982	85	6424	1498	0.99826	1498-1636	0.99819-0.99831
3	0.9983	91	6259	1513	0.99831	1513-1666	0.99830-0.99843
3	0.9984	86	4569	1532	0.99845	1532-1710	0.99840-0.99848
3	0.9985	128	4985	1556	0.99850	1556-1752	0.99848-0.99858
3	0.9986	106	5185	1563	0.99861	1563-1828	0.99855-0.99869
3	0.9987	106	4492	1603	0.99870	1603-1800	0.99866-0.99881
3	0.9988	112	4586	1623	0.99881	1623-1833	0.99877-0.99889
3	0.9989	104	5855	1655	0.99892	1655-1918	0.99884-0.99897
4	0.998	94	6096	1633	0.99809	1633-1830	0.99800-0.99837
4	0.9981	58	3370	1645	0.99811	1645-1820	0.99808-0.99818
4	0.9982	61	2582	1692	0.99824	1692-1875	0.99821-0.99827
4	0.9983	71	3218	1708	0.99831	1708-1898	0.99830-0.99838
4	0.9984	83	3133	1718	0.99840	1718-1912	0.99840-0.99846
4	0.9985	86	3359	1742	0.99850	1742-1980	0.99850-0.99853
4	0.9986	74	3334	1778	0.99864	1778-1938	0.99860-0.99869
4	0.9987	64	4379	1800	0.99870	1800-2029	0.99867-0.99879
4	0.9988	68	3411	1830	0.99880	1830-1944	0.99880-0.99902
4	0.9989	61	3665	1858	0.99890	1858-1982	0.99890-0.99901
5	0.998	54	14466	1984	0.99821	1984-2247	0.99800-0.99821
5	0.9981	56	13328	1984	0.99832	1984-2309	0.99810-0.99832
5	0.9982	51	12059	1984	0.99832	1984-2367	0.99820-0.99832
5	0.9983	54	11321	1968	0.99835	1968-2192	0.99835-0.99841

Table C.3: Results for second set of w_{um} and w_c and low thresholds

C.4 Lemmas for Reliability Characteristics

In this section, we will explore the reliability characteristics (b)-(d) which we used in the heuristic. These characteristic were observed in numerical experiments of our reliability calculation method. In the following, we are presenting lemmas that are related to these characteristics. Note that stronger assumptions are made for the lemmas than we can make based on the DMS. We are not able to prove these characteristics for reliability in general.

For the rest of the section, let F_1, F_2, F_3, F_4 be implementations of a function. Let T be the set of all components (units, cables, etc.) used in the system and T_i be the set of components used for F_i, $i \in \{1, \ldots, 4\}$. For $t \in T$, let $\mathfrak{f}_t$ be the event that t does not fail and f_t be the associated failure rate. Furthermore, let us use the following notation:

$$\forall l \in \{1, \ldots, 4\} : \prod_{t \in T_l} (1 - f_t) = a_l, \text{ and}$$
$$\forall l, k \in \{1, \ldots, 4\} : \prod_{t \in T_l \cap T_k} (1 - f_t) = a_{l \cap k}.$$

Lastly, we know that for $l \in \{3, 4\}$:

$$\begin{aligned} P\left(F_1 \cup F_2 \cup F_l\right) = &\prod_{t \in T_1} (1 - f_t) + \prod_{t \in T_2} (1 - f_t) + \prod_{t \in T_l} (1 - f_t) - \prod_{t \in T_1 \cup T_2} (1 - f_t) \\ &- \prod_{t \in T_1 \cup T_l} (1 - f_t) - \prod_{t \in T_2 \cup T_l} (1 - f_t) + \prod_{t \in T_1 \cup T_2 \cup T_l} (1 - f_t). \end{aligned} \tag{C.1}$$

We first look at characteristic (d). For this characteristic, we assumed that we have three paths and a redundant system with the first two paths being disjoint. The third path can share units and connections with the other two paths. The characteristic is that the closer sum of failure rates of the shared units and connections of the first and third path is to the sum of the second and third path, the higher is the reliability.

We were not able to prove this in general. We had to make the assumption that the reliability of the first path is equal to the reliability of the second path. Furthermore, we assume that the reliability of the third path and also the sum of failure rates the third path shares with other paths stays the same. With these assumptions, we can prove in Lemma 8 that the reliability is higher the smaller the difference between the sum of failures of shared units between the first and third path and between the second and third path is.

Lemma 8. *Let us assume a redundant system with $T_1 \cap T_2 = \emptyset$, $P(F_1) = P(F_2)$ and that $P(F_3) = P(F_4)$. Furthermore, let us assume that*

$$\begin{aligned} &\left| \sum_{t \in T_1 \cap T_3} \log(1 - f_t) - \sum_{t \in T_2 \cap T_3} \log(1 - f_t) \right| \\ \leq &\left| \sum_{t \in T_1 \cap T_4} \log(1 - f_t) - \sum_{t \in T_2 \cap T_4} \log(1 - f_t) \right| \end{aligned} \tag{C.2}$$

and that

$$\begin{aligned} &\sum_{t \in T_1 \cap T_3} \log(1 - f_t) + \sum_{t \in T_2 \cap T_3} \log(1 - f_t) = \sum_{t \in T_1 \cap T_4} \log(1 - f_t) + \sum_{t \in T_2 \cap T_4} \log(1 - f_t) \\ &\Leftrightarrow a_{1 \cap 3} a_{2 \cap 3} = a_{1 \cap 4} a_{2 \cap 4} \end{aligned} \tag{C.3}$$

Than it holds that

$$P\left(F_1 \cup F_2 \cup F_3\right) \geq P\left(F_1 \cup F_2 \cup F_4\right). \tag{C.4}$$

Before proving Lemma 8, we prove Lemma 9 which is necessary for Lemma 8.

Lemma 9. *Let $a, b, \hat{a}, \hat{b} \in \mathbb{R}^+$. Assume that $ab = \hat{a}\hat{b}$ and $|\log(a) - \log(b)| \leq |\log(\hat{a}) - \log(\hat{b})|$. Then, it holds that*

$$a + b \leq \hat{a} + \hat{b}$$

Proof. Let $a, b, \hat{a}, \hat{b} \in \mathbb{R}^+$. We assume that $ab = \hat{a}\hat{b}$ and $|\log(a) - \log(b)| \leq |\log(\hat{a}) - \log(\hat{b})|$. Without loss of generality, we can assume that $a \geq b$ and $\hat{a} \geq \hat{b}$. Therefore,

$$\begin{aligned}
& |\log(a) - \log(b)| \leq |\log(\hat{a}) - \log(\hat{b})| \\
\Leftrightarrow \quad & |\log\left(\frac{a}{b}\right)| \leq |\log\left(\frac{\hat{a}}{\hat{b}}\right)| \\
\Leftrightarrow \quad & \log\left(\frac{a}{b}\right) \leq \log\left(\frac{\hat{a}}{\hat{b}}\right) \\
\Leftrightarrow \quad & \frac{a}{b} \leq \frac{\hat{a}}{\hat{b}} \\
\Leftrightarrow \quad & a\hat{b} \leq \hat{a}b.
\end{aligned}$$

We first prove that $a \leq \hat{a}$ and $b \geq \hat{b}$. We prove these by contradiction. Let us assume that $a > \hat{a}$. It holds that

$$\begin{aligned}
& ab > \hat{a}b \geq a\hat{b} \\
\Rightarrow \quad & b > \hat{b} \\
\Rightarrow \quad & ab > a\hat{b} > \hat{a}\hat{b},
\end{aligned}$$

which is a contradiction to our assumption $ab = \hat{a}\hat{b}$. Hence, $a \leq \hat{a}$.
Let us now assume that $b < \hat{b}$. It holds that

$$\begin{aligned}
& \hat{a}\hat{b} > \hat{a}b \geq a\hat{b} \\
\Rightarrow \quad & a < \hat{a} \\
\Rightarrow \quad & ab < a\hat{b} < \hat{a}\hat{b},
\end{aligned}$$

which is a contradiction to our assumption $ab = \hat{a}\hat{b}$. Hence, $b \geq \hat{a}$.
As $a \leq \hat{a}$ and $b \geq \hat{b}$, we have that

$$\begin{aligned}
& \sqrt{a} - \sqrt{b} \leq \sqrt{\hat{a}} - \sqrt{\hat{b}} \\
\Leftrightarrow \quad & \left(\sqrt{a} - \sqrt{b})\right)^2 \leq \left(\sqrt{\hat{a}} - \sqrt{\hat{b}}\right)^2 \\
\Leftrightarrow & \left(\sqrt{a} - \sqrt{b})\right)^2 + 2\sqrt{ab} \leq \left(\sqrt{\hat{a}} - \sqrt{\hat{b}}\right)^2 + 2\sqrt{\hat{a}\hat{b}} \\
\Leftrightarrow \quad & a + b \leq \hat{a} + \hat{b}.
\end{aligned}$$

□

We can now prove Lemma 8.

Proof. We can reformulate (C.1) for $l \in \{3, 4\}$:

$$P\left(F_1 \cup F_2 \cup F_l\right) = a_1 + a_2 + a_2 - a_{1\cup 2} + a_l(a_{1\setminus l}a_{2\setminus l} - a_{1\setminus l} - a_{2\setminus l})$$

and

$$a_{1\setminus l}a_{2\setminus l} - a_{1\setminus l} - a_{2\setminus l} = \frac{a_1 a_2}{a_{1\cap l}a_{2\cap l}} - \frac{a_1}{a_{1\cap l}} - \frac{a_2}{a_{2\cap l}}$$

Because of (C.3), we know that

$$\frac{a_1 a_2}{a_{1\cap 3}a_{2\cap 3}} = \frac{a_1 a_2}{a_{1\cap 4}a_{2\cap 4}}$$

To prove that (C.4) holds, we only have to show that

$$-\frac{a_1}{a_{1\cap 3}} - \frac{a_2}{a_{2\cap 3}} \geq -\frac{a_1}{a_{1\cap 4}} - \frac{a_2}{a_{2\cap 4}}. \tag{C.5}$$

It holds that

$$\begin{aligned}(C.5) \Leftrightarrow & -\frac{a_1 a_{2\cap 3} - a_2 a_{1\cap 3}}{a_{1\cap 3}a_{2\cap 3}} \geq -\frac{a_1 a_{2\cap 4} - a_2 a_{1\cap 4}}{a_{1\cap 4}a_{2\cap 4}} \\ \Leftrightarrow \quad & a_{1\cap 3} + a_{2\cap 3} \leq a_{1\cap 4} + a_{2\cap 4} \end{aligned} \tag{C.6}$$

and inequality (C.6) holds by using Lemma 9 and assumption (C.2) and (C.3). Hence, we proved that (C.4) holds. □

The second characteristic we look at is (c). The characteristic states that if the sum of failure rates for a path is lower, the reliability of the overall system is higher.

We again consider a system with three paths. We assume that the first two paths are disjoint and that the third can share units and connections with other two paths.Furthermore, we assume that the disjoint paths are fixed. We prove for two cases of how the third path shares its units and connections with the other two paths in Lemma 10 and Lemma 11 that the lower the change sum of failure rates for the third path is, the higher is the reliability. There are more cases to consider, but we were not able to prove these yet.

Lemma 10. *Let us assume a redundant system with $T_1 \cap T_2 = \emptyset$, and that $P(F_3) \geq P(F_4)$. Furthermore, let us assume that $a_{1\cap 3} = a_{1\cap 4}$ and $a_{2\cap 3} = a_{2\cap 4}$. Then it holds that*

$$P\left(F_1 \cup F_2 \cup F_3\right) \geq P\left(F_1 \cup F_2 \cup F_4\right). \tag{C.7}$$

Proof. We know that for $l \in \{3, 4\}$:

$$P\left(F_1 \cup F_2 \cup F_l\right) = a_1 + a_2 + a_l - a_1 a_2 + a_l\left(a_{1\setminus l}a_{2\setminus l} - a_{1\setminus l} - a_{2\setminus l}\right).$$

For the function $f(a, b) = a + b - a \times b$, we have a maximum at $(1, 1)$ with $f(1, 1) = 1$. Therefore, we know that

$$\begin{aligned} & 1 \geq a_{1\setminus 3} + a_{2\setminus 3} - a_{1\setminus 3}a_{2\setminus 3} \\ \Leftrightarrow \quad & a_4 - a_3 \leq (a_4 - a_3)(a_{1\setminus 3} + a_{2\setminus 3} - a_{1\setminus 3}a_{2\setminus 3}) \\ \Leftrightarrow & a_4 + a_4(a_{1\setminus 4} + a_{2\setminus 4} - a_{1\setminus 4}a_{2\setminus 4}) \leq a_3 + a_3(a_{1\setminus 3} + a_{2\setminus 3} - a_{1\setminus 3}a_{2\setminus 3}) \\ \Leftrightarrow \quad & P\left(F_1 \cup F_2 \cup F_4\right) \leq P\left(F_1 \cup F_2 \cup F_3\right). \end{aligned}$$

Therefore, $(C.7)$ holds. □

Lemma 11. *Let us assume a redundant system with $T_1 \cap T_2 = \emptyset$, and that $P(F_3) \geq P(F_4)$. Furthermore, let us assume that $a_{1\cap 3} \geq a_{1\cap 4}$, $a_{2\cap 3} \geq a_{2\cap 4}$ and $a_{3\setminus(1\cup 2)} = a_{4\setminus(1\cup 2)}$. Than it holds that*

$$P\left(F_1 \cup F_2 \cup F_3\right) \geq P\left(F_1 \cup F_2 \cup F_4\right). \tag{C.8}$$

Proof. We can reformulate (C.1) and know that for $l \in \{3,4\}$:

$$P\left(F_1 \cup F_2 \cup F_l\right) = a_1 + a_2 + a_l - a_1a_2 - a_1a_{l\setminus(1\cup2)}a_{2\cap l} - a_2a_{l\setminus(1\cup2)}a_{1\cap l} + a_1a_2a_{l\setminus(1\cup2)}.$$

Let $\delta_1, \delta_2 \in \mathbb{R}^+$ with $a_{1\cap4} + \delta_1 = a_{1\cap3}$ and $a_{2\cap4} + \delta_2 = a_{2\cap3}$. We know that

$$\begin{aligned}
& P\left(F_1 \cup F_2 \cup F_3\right) \geq P\left(F_1 \cup F_2 \cup F_4\right) \\
\Leftrightarrow & a_1 + a_2 + a_3 - a_1a_2 - a_1a_{3\setminus(1\cup2)}a_{2\cap3} - a_2a_{3\setminus(1\cup2)}a_{1\cap3} + a_1a_2a_{3\setminus(1\cup2)} \\
\geq\ & a_1 + a_2 + a_4 - a_1a_2 - a_1a_{4\setminus(1\cup2)}a_{2\cap4} - a_2a_{4\setminus(1\cup2)}a_{1\cap4} + a_1a_2a_{4\setminus(1\cup2)} \\
\Leftrightarrow & a_{1\cap4}a_{2\cap4} - a_1a_{2\cap4} - a_2a_{1\cap4} \leq a_{1\cap3}a_{2\cap3} - a_1a_{2\cap3} - a_2a_{1\cap3} \\
\Leftrightarrow & a_{2\cap4}\left(a_{1\cap4} - a_1\right) - a_2a_{1\cap4} \leq a_{2\cap3}\left(a_{1\cap3} - a_1\right) - a_2a_{1\cap3} \\
= & \left(a_{2\cap4} + \delta_2\right)\left(a_{1\cap4} + \delta_1 - a_1\right) - a_2\left(a_{1\cap4} + \delta_1\right) \\
= & a_{2\cap4}\left(a_{1\cap4} - a_1\right) + a_{2\cap4}\delta_1 + \delta_2(a_{1\cap4}\delta_1 - a_1) - a_2a_{1\cap4} - \delta_1a_2 \\
\Leftrightarrow & 0 \leq a_{2\cap4}\delta_1 + \delta_2(a_{1\cap4} + \delta_1 - a_1) - \delta_1a_2 \\
\Leftrightarrow & 0 \leq \delta_1(a_{2\cap4} - a_2) + \delta_2(a_{1\cap4} + \delta_1 - a_1).
\end{aligned}$$

Therefore, $(C.9)$ holds. □

Lastly, we prove Lemma 12 that corresponds to characteristic (b). We assume again tat we have three paths and the third path does not share all units and connections with the other paths. Furthermore, we assume that the sum of failure rates of units and connections of all paths and the number of units and connections are fixed. We then prove in Lemma 12 for one case of how the third path shares its units and connections with the other two paths that the reliability of the overall system is higher if the sum of failure rates of the units and connections that the third path does not share with the other paths is higher. There are more cases to consider, but we were not able to prove these yet.

Lemma 12. *Let us assume a redundant system with $T_1 \cap T_2 = \emptyset$, and that $P(F_3) = P(F_4)$. Furthermore, let us assume that $a_{1\cap3} \geq a_{1\cap4}$ and $a_{2\cap3} \geq a_{2\cap4}$. Then it holds that*

$$P\left(F_1 \cup F_2 \cup F_3\right) \geq P\left(F_1 \cup F_2 \cup F_4\right). \tag{C.9}$$

Proof. We know that for $l \in \{3,4\}$:

$$P\left(F_1 \cup F_2 \cup F_l\right) = a_1 + a_2 + a_l - a_1a_2 + a_l\left(a_{1\setminus l}a_{2\setminus l} - a_{1\setminus l} - a_{2\setminus l}\right).$$

As $a_{1\cap3} \geq a_{1\cap4}$ and $a_{2\cap3} \geq a_{2\cap4}$, we know that $a_{1\setminus3} \leq a_{1\setminus4}$ and $a_{2\setminus3} \leq a_{2\setminus4}$. Let $\delta_1, \delta_2 \in \mathbb{R}^+$ be such that $a_{1\setminus3} + \delta_1 = a_{1\setminus4}$ and $a_{2\setminus3} + \delta_2 = a_{2\setminus4}$. We then know that

$$\begin{aligned}
& P\left(F_1 \cup F_2 \cup F_3\right) \geq P\left(F_1 \cup F_2 \cup F_4\right) \\
\Leftrightarrow & a_1 + a_2 + a_3 - a_1a_2 + a_3\left(a_{1\setminus3}a_{2\setminus3} - a_{1\setminus3} - a_{2\setminus3}\right) \\
\geq\ & a_1 + a_2 + a_4 - a_1a_2 + a_4\left(a_{1\setminus4}a_{2\setminus4} - a_{1\setminus4} - a_{2\setminus4}\right) \\
\Leftrightarrow & a_{1\setminus3}a_{1\setminus3} - a_{1\setminus3} - a_{1\setminus3} \geq a_{1\setminus4}a_{2\setminus4} - a_{1\setminus4} - a_{2\setminus4} \\
\Leftrightarrow & a_{2\setminus4}\left(a_{1\setminus4} - a_1\right) - a_2a_{1\setminus4} \\
= & (a_{1\setminus3} + \delta_1)(a_{2\setminus3} + \delta_2) - a_{1\setminus3} - \delta_1 - a_{2\setminus3} - \delta2 \\
\leq & a_{1\setminus3}a_{2\setminus3} - a_{1\setminus3} - a_{2\setminus3} \\
\Leftrightarrow & \delta_1a_{2\setminus3} + \delta_2a_{1\setminus3} + \delta_1\delta_2 - \delta_1 - \delta_2 \leq 0 \\
\Leftrightarrow & \delta_1(a_{2\setminus3} + \delta_2 - 1) + \delta_2(a_{1\setminus3} - 1) \leq 0 \\
\Leftrightarrow & \delta_1(a_{2\setminus4} - 1) + \delta_2(a_{1\setminus3} - 1) \leq 0.
\end{aligned}$$

Therefore, $(C.9)$ holds. □

With these lemmas, we have starting point to see in which cases the characteristics are true.