

Expert-based recommendation systems in the insurance industry—A complexity theoretical study

Vanessa Patricia Mindl

Vollständiger Abdruck der von der Fakultät für Wirtschafts- und Organisationswissenschaften der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors der Wirtschafts- und Sozialwissenschaften (Dr. rer. pol.)

genehmigten Dissertation.

Gutachter:

- 1.) Prof. Dr. Andreas Brieden
- 2.) Prof. Dr. Claudius Steinhardt

Die Dissertation wurde am 11.01.2023 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Wirtschafts- und Organisationswissenschaften am 05.04.2023 angenommen. Die mündliche Prüfung fand am 15.05.2023 statt.

To my family

Acknowledgements

Nobody said it was easy.
Nobody said it was so hard.

Coldplay—*Scientist*

As Coldplay writes so well, I was aware that writing a thesis would not be super easy, the proofs would not be trivial, and most importantly the last few weeks and months would not resemble a life on a pony farm, however, there were definitely times when you feel like you are climbing a very high mountain in Nepal. I am therefore very grateful to have been supported by several people whom I would like to thank at this point.

First and foremost, I would like to thank my doctoral advisor Prof. Dr. Andreas Brieden, without whom this work would not have been possible. I would especially like to thank him for his empowering guidance, the stimulating discussions, his trust and the always friendly and thus supportive working atmosphere.

I would also like to thank my colleagues and friends Saskia, Vienna, Martin and Matthias, and of course to all my other colleagues at university. They made the time at university a really joyful and unforgettable time—with lots of laughter and good conversations while drinking coffee. With you, even lengthy exam corrections were fun (and bizarre when you think of the weird texts about chicken pox). And without you, I would never have heard of the so-called bee swarm law (here a special thanks to Sebastian). In particular, I want to express my gratitude to Saskia and Martin for the very helpful input and the supportive words—both in terms of content and personally. I am so grateful for your time, your words and most of all your friendship.

Finally, I would like to thank my family, first of all my parents, my sister, and my grandfather—who taught me that you can achieve anything if you pursue a positive attitude towards life. I am so lucky to have a family so full of love and unlimited support. Knowing that I have people behind me who love me unconditionally not only gave me the strength to write this thesis, but also to be the person I am today.

Above all I would like to thank my husband Frank and my kids Emilia, Arthur, and Rebecca. You are my source of strength, my place of retreat and joy and I could never have done it without you. I love you.

Abstract

The insurance industry must constantly adapt to digital trends and technologies as well as ever-changing customer expectations. For instance, claims processing should be designed in such a way that it is not only as efficient as possible, but also customer-oriented and customer-satisfying. In this thesis, we propose an optimization problem—the expert-based recommendation system (EBRS problem)—that can improve the claims processing by providing appropriate recommendations to clerks. In a first step, we estimate the (yet unknown) quality of a claims processing by taking into account the implicit knowledge of experts via utilizing adapted methods of Conjoint Analysis. In a second step, we use this expert knowledge to find suitable recommendations for the EBRS problem. This approach can be utilized to generate automated recommendations for actions to be taken by clerks to assist them in their everyday work tasks. The thesis addresses in particular the proof that the EBRS problem in general belongs to the complexity class of NP -complete problems. We finally investigate the EBRS problem in the context of complexity theory to identify cases for which the problem is solvable in polynomial time.

Zusammenfassung

Versicherungsunternehmen müssen sich den veränderten Kundenerwartungen im digitalen Umfeld kontinuierlich anpassen. So gilt es beispielsweise, Schadenprozesse so zu gestalten, dass diese nicht nur möglichst effizient, sondern auch kundenorientiert abgewickelt werden. In dieser Arbeit stellen wir ein Optimierungsproblem vor, das den Schadenbearbeitungsprozess durch geeignete Empfehlungen verbessern kann. Dazu ermitteln wir in einem ersten Schritt die Qualität eines Schadenbearbeitungsprozesses unter Berücksichtigung des impliziten Wissens von Expert:innen. In einem zweiten Schritt nutzen wir dieses Wissen, um geeignete Empfehlungen für das Optimierungsproblem zu finden, das wir aufgrund dieser Vorgehensweise expertenbasiertes Empfehlungssystem, kurz EBRS, nennen. Dieser Ansatz kann verwendet werden, um automatisierte Handlungsempfehlungen für Personen in der Schadenbearbeitung zu generieren, die diese bei ihrer Arbeit unterstützen sollen. Die Dissertation widmet sich insbesondere dem Beweis, dass das EBRS-Problem im Allgemeinen zur Komplexitätsklasse der NP -vollständigen Probleme gehört. Anschließend wird das EBRS-Problem komplexitätstheoretisch untersucht, um Fälle zu identifizieren, in denen das Problem in polynomieller Zeit lösbar ist.

Contents

List of Figures	13
1 Overview of this thesis	15
2 Motivation	17
2.1 Expert-based improvement of the quality of claims processing	17
2.1.1 Claims processing in the digital age	18
2.1.2 Expert-based quality	21
2.1.3 Improving the expected quality of claims	28
2.2 Complexity of problems	32
3 Definitions and Preliminaries	37
3.1 Statistical methods to determine the quality claims processing	38
3.1.1 Conjoint analysis	38
3.1.1.1 Selection of attributes and their levels	39
3.1.1.2 Selection of data collection method	41
3.1.1.3 Selection of data collection design	42
3.1.1.4 Selection of data collection presentation	42
3.1.1.5 Evaluation of stimuli	43
3.1.1.6 Estimation of part-worth utilities	44
3.1.2 Clustering with k -medoids	44
3.1.2.1 Gower's coefficient	49
3.1.2.2 Validation of the clusters	50
3.2 Introduction to graph theory	52
3.2.1 Independent set	56
3.2.2 Matchings	57
3.3 Complexity theory	58
3.3.1 Running times of algorithms	59
3.3.2 The complexity class NP	62
3.3.3 NP -completeness	67
3.3.3.1 Cook's theorem	73
3.3.3.2 Methods for proving NP -completeness	75
3.3.4 Further NP -complete problems	78
3.3.4.1 The 3-SAT problem	78
3.3.4.2 The independent set problem	79
3.3.4.3 The perfect matching problem	80

3.3.4.4	The Knapsack problem	80
3.3.4.5	The Subset-Sum problem	82
3.3.4.6	Generalizations of the Knapsack problem	83
3.4	Integer linear programs	86
3.5	LP relaxation	90
3.6	Total unimodularity	92
4	Expert-based recommendation systems	101
4.1	Expert-based quality of claims processing and recommendations	101
4.1.1	Procedure to define the quality of claims processing	102
4.1.1.1	Selection of representative stimuli	104
4.1.1.2	Evaluation of stimuli and estimation of quality	110
4.1.2	Choice of recommendations	112
4.2	Modelling the EBRs problem and classification of its complexity	113
4.2.1	Formulation of the EBRs problem	113
4.2.1.1	Definition of the constraints	116
4.2.1.2	Defining the EBRs problem as an ILP	120
4.2.2	Decision version of the EBRs problem	124
4.2.3	Proof of NP-completeness	125
4.3	Complexity-theoretical investigations of the EBRs problem	129
4.3.1	Conflicts between recommendations but no dependencies	131
4.3.1.1	Case: Pairwise conflicts	131
4.3.1.2	Case: Stellar conflicts	138
4.3.1.3	Case: No solution	143
4.3.2	Dependent but no conflicting recommendations	146
4.3.2.1	Case: No solution	150
4.3.3	No conflicts and no dependencies	152
4.3.3.1	Case: Only bounds	153
4.3.3.2	Case: Limited weighted sum per model	161
5	Outlook and Conclusions	169
5.1	The effort—a generalization of the EBRs problem	169
5.2	Conclusion	172
	Bibliography	177

List of Figures

2.1	Improved claims processing	19
2.2	Overlapping interval scale	27
2.3	Schematic illustration of the EBRs procedure.	31
3.1	Steps of Conjoint Analysis	40
3.2	Geometric presentation of a graph	54
3.3	Geometric presentation of a path.	55
3.4	Geometric presentation of a cycle.	55
3.5	Example for an independent set.	57
3.6	Examples for graphs with and without perfect matching.	58
3.7	Running times of algorithms.	60
3.8	Example for a Karp's reduction	69
3.9	Set diagram of complexity classes	72
3.10	Example of a bipartite graph.	80
3.11	Example of an LP relaxation for a 0-1-ILP.	91
3.12	Example of a polyhedron (a) and a polytop (b).	93
3.13	Example for an intersection between hyperplane and polyhedron	94
3.14	Example of relation of a bounded polyhedron and its convex hull.	95
4.1	Sample excerpt of the first questionnaire.	108
4.2	Sample excerpt of the second questionnaire.	110
4.3	Example of an EBRs problem as ILP	123
4.4	Example to illustrate the need of setting $\gamma_i = 1$ for all $i \in [n]$.	129
4.5	Example to illustrate the need of $D = \emptyset$.	130
4.6	Example of the EBRs problem with a TU constraint matrix.	133
4.7	Assignment of the rows of the constraint matrix A^C to $I = (I_1, I_2)$	137
4.8	Different examples for stellar conflicts.	139
4.9	Example for stellar conflicts.	142
4.10	Example for "No solution" for conflicting recommendations	145
4.11	Example for "No solution" for dependent recommendations	151

List of Figures

1 Overview of this thesis

The aim of the thesis is to demonstrate the potential of mathematical optimization for improving claims processing in insurance companies, taking into account complexity-theoretical aspects.

In chapter 2 we motivate the goals and aims of our work by framing a very specific problem in the insurance industry—namely the processing of claims. We illustrate the key questions of this thesis in that specific context of claims processing, illustrating the relevance of a measurement of “quality” of such claims processing for improving current traditional best-practice handling via digitalization. We demonstrate how expert ratings, questionnaire design, conjoint analyses, cluster analyses, and optimization problems are aligned throughout the present thesis to improve the expected quality of claims by finding appropriate recommendations for the insurance industry.

In chapter 3 we give an overview of the key mathematical concepts underlying this thesis. We present our questionnaire design, which we use to assess experts on the quality of claims processing. We developed these questionnaires based on a k -medoids cluster analysis, which is outlined in this section. To evaluate the questionnaires, we used conjoint analysis, which is outlined. With an introduction to graph theory we lay the foundation for the optimization problems underlying the thesis. Finally, the mathematical concepts of optimization used in this work are introduced and the complexity class of NP-complete problems is defined and discussed, and examples of NP-complete problems are given.

In chapter 4—the main part of the thesis—we define and describe the *expert-based recommendation system* (EBRS problem) with respect to its complexity. A key step in this process is to estimate the *quality* of a claims processing, which does not exist in the dataset: By first looking at completed claims ex-post, we do this by drawing on the implicit knowledge of experts. Thereby, we not only estimate the quality itself, but also identify specific variables influencing this quality—which take the role of potential recommendations for claims processing. To improve the claims processing for future claims with the help of appropriate recommendations, we predict the expected quality ex ante in a first step with the help of prototypical claims. In a second step, we finally formulate the EBRS problem to optimize the expected quality during claims processing. We study the EBRS problem in complexity-theoretical terms to find cases that are solvable in polynomial time.

1 Overview of this thesis

Chapter 5 closes with reflections on the fact that not only the *number* of recommendations, but also the *effort* for the claim handler (or even for the insurance company itself) to actually implement specific recommendations may play a role in claims processing. This leads to the idea of considering the *effort* in the EBRs problem in addition to the number of recommendations in future work. The thesis concludes with a summary and this outlook.

2 Motivation

In chapter 2 of this thesis, we explain and justify what we mean by the quality of claims processing in the insurance industry and why it is especially important in the digital age to take a closer look at it. Afterwards, we motivate the expert-based recommendation system with which the quality of the claims processing can be improved. Finally, we explain the importance of considering the complexity of the so defined integer linear program.

Section 2.1 begins with an overview of current ideas on claims processing in the digital age. From this, the necessity of an expert-based quality of the processing of claims is motivated. That means, based on the assessments of various experts, it is necessary to develop quality characteristics that can be used to improve the claims processing. We further motivate with which methods we utilize these quality features to add the information of quality to the data set. Subsequently, we motivate how we generate a recommendation system based on these quality features, with which we can optimize the previously defined expected quality.

The improvement of the expected quality of a claims processing can be formulated as an integer linear program. In section 2.2 we motivate the need to consider the complexity-theoretic investigation of different cases of the expert-based recommendation system to see how *efficiently* the problem can be solved. To better grasp the notion of efficiency, we first give an initial definition of an efficient algorithm. We also briefly explain the complexity classes \mathbb{P} and \mathbb{NP} , and their relation. Furthermore, we motivate the complexity class of \mathbb{NP} -complete problems.

2.1 Expert-based improvement of the quality of claims processing in the insurance industry

In the digital age, a variety of processes are undergoing change. Insurance companies, whose business processes are already based on the handling of data, have always been part of this change. One of the key areas that can undergo a transformation in order to optimize processes is claims processing.

2.1.1 Claims processing in the digital age

Due to the possibilities offered by the ever-advancing digitalization, the insurance industry has been in a state of upheaval for several years and is under pressure to undergo a digital transformation [BKM21]. This pressure is exacerbated by the fact that the possibilities offered by digitalization are creating a competitive landscape in which so-called FinTechs are also pushing their way in [BCC⁺18, BKM21]. McKinsey even formulates this pressure very radically as “attackers are transforming the competitive landscape and elevating customer expectations” [BCC⁺18].

For traditional insurers in particular, this represents a major challenge, because—according to a survey by Bitkom—claims are still primarily processed in paper form [Ver20]. In the survey, only six percent of the customers stated that they were able to process the claim purely digitally until payment was made. Yet, especially in the Corona pandemic, customers have become accustomed to using more and more digital channels and technologies [BKM21], which is exactly what many FinTech companies offer [BCC⁺18].

Especially in order to attract new customers, but also to retain existing customers, the possibilities of digitalization are manifold. A particular focus is on claims settlement, because claims are a kind of *moment of truth* and offer the opportunity to create positive experiences in the relationship between customer and insurer, thus laying the foundation for improved customer loyalty [KMPF15, BP04]. In addition, there is great potential for improvement, as too little optimization effort has been made in this area in recent years [BP04, BCC⁺18]. In claims processing, the methods of digitalization such as data analytics or artificial intelligence offer innovative possibilities for interaction between the customer, the insurance company, and third parties—and can also make the settlement process more customer-specific in particular [BKM21].

In addition to customer satisfaction, improved claims processing can also lead to higher employee satisfaction and efficiency [BP04, BCC⁺18, DAV19]. Increasing the efficiency of claims processing is particularly interesting because it can significantly reduce costs [BCC⁺18, Gen22]. Furthermore, both McKinsey and DAV see the opportunity to improve effectiveness in addition to customer satisfaction and efficiency through a well-implemented and customer-oriented digitalization of the claims processing [BCC⁺18, DAV19]. For example, the processing accuracy associated with effectiveness can reduce the risk of claims disputes and litigation, as well as fraud, which in turn also reduces the costs of claims processing [BCC⁺18, Gen22].

Furthermore, effectiveness also has an impact in terms of customer loyalty. As already mentioned, the quality of an insurer becomes apparent in the event of a claim. If the effectiveness of the claims processing is not satisfactory, 87 % of customers see this as a reason to change their insurance company [Gen22]. In

2.1 Expert-based improvement of the quality of claims processing

summary, an improved claims processing can increase three aspects in particular—customer satisfaction, efficiency and effectiveness, as illustrated in the following figure 2.1

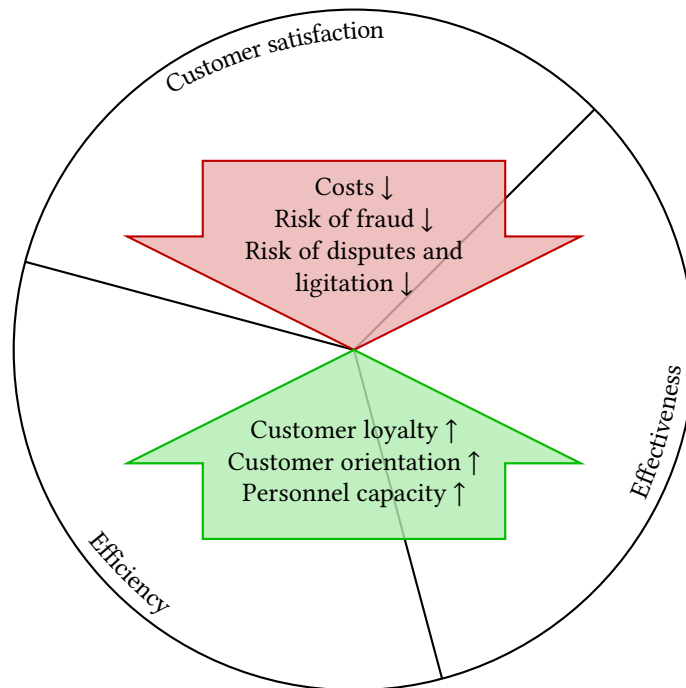


Figure 2.1: Improved claims processing—adapted from [BCC⁺18].

The importance of improved claims handling is illustrated by the fact that, according to Deloitte, nearly 70 % of an insurance company’s expenses are attributable to claims settlement [Gen22]. However, efforts to increase the efficiency of claims processes are still in their infancy, even though the potential for optimization in the area of claims settlement, e.g., through situation-based claims settlement or process optimization, can be easily realized [BP04, BKM21]. Furthermore, if efficiency is increased, this also allows personnel capacities to be used more sensibly [DAV19]. This is especially important if a serious accident occurs: in the event that a customer is involved in such a serious accident, it is essential that the claims handler has the capacity to take care of this customer on a human and personal level. The importance of such free capacity is evidenced by the fact that “90 % of a claims handling is about solving the problem of a customer who has experienced a tragic incident” [Gen22].

Summarizing all these aspects, it can be stated that digitalization offers the opportunity to implement an efficient and effective claims processing that also provides the personnel capacities, depending on the case constellation and customer requirements, so that human and personal care is made possible [BKM21]. Considering this, one goal for insurers may be to fully or partially automate the processing of

2 Motivation

claims with the help of digital technologies, but also the use of artificial intelligence and data analysis methods. It is therefore not surprising that McKinsey predicts a 50 % increase in automated claims processings by 2030 [Gen21]. Collaboration with InsurTech, some of which already have or can provide the necessary technologies, is a good way to achieve this goal. They are therefore not only competitive opponents, but can also act as suitable digital transformation partners [Cag21].

There is a wide range of digitalization options to make the claims processing more efficient—from implementing an app, using the already mentioned artificial intelligence, processing the dataset using Big Data technologies, to sophisticated telematics tariffs. However, not all of these approaches are effective. A very well-known example of such an *aberration* are the telematics tariffs in motor vehicle insurance. For more information we refer the interested reader to Morawetz [Mor16] and Walthes [Wal17b].

Instead, it may make sense to modernize existing procedural structures and improve interaction with customers and employees [Wal17b].

The improvement of existing procedural structures is precisely the focus of this thesis. Instead of fundamentally changing the claims processing with new digital technologies, we rather aim at improving existing claims processing by automatically giving suitable recommendations. A special focus is on customer satisfaction—which does not exclude efficiency and effectiveness. This depends on the processing of claims under consideration.

To achieve an improvement of the claims processing, however, we do not focus on the purely process-related structures. For example, we do not analyze a given dataset with Big Data technologies as they can be found in [DAV19] in the usual sense. For successful digitalization of a claims processing, we believe it is imperative to first better understand and formalize the knowledge and experience of experts [BKM21]. This means that we aim at surveying all parties involved in a claims processing and transfer their knowledge (derived from their answers) to the given data set for digital use. The inclusion of expert knowledge on the one hand bears potential for improving the claims processing, while on the other hand it is a challenge that has not been sufficiently addressed in the discussion to date.

By incorporating the experience and knowledge of the process participants (i.e., customers, claim handlers, and managers) from different perspectives, we generate a “personalized and expert-based digital claims processing” [BKM21]. Our focus is therefore on increasing *expert-based quality* of the claims processing. In the next section, we will clarify what we understand by this term.

2.1.2 Expert-based quality

As we discussed in section 2.1.1, a better settlement of claims may increase the employees and customers satisfaction. As described in [BKM21], customer satisfaction can be a differentiating factor in the increasingly competitive environment if implemented appropriately. Digitalization now offers the opportunity to achieve true customer satisfaction by, e.g., developing a tailored claims processing, as discussed further in our recent article [BKM21]. At best, these newly-developed digital processes help to improve customer orientation and enable human as well as personalized care in the event of an accident. Thus, the challenge we face in the claims processing is both (1) to exploit the possibilities of digitalization and (2) to take personal interests—in addition to interests of the management and the company—into account, raising the question:

How can personal interests be taken into account?

To better understand personal interests, we should keep in mind that there are several internal and external parties involved in the processing of claims. Each party involved pursues different interests depending on their position. For example, a manager may have a different focus on processing a claim than a clerk or customer. If all personal interests are met to a certain degree, satisfaction with the overall handling of the claim is higher. Yet, meeting all personal interests to the same extent is not possible. Varying views of the different parties may result in certain actions during claims processing being perceived positive by one party, but negative by another. Therefore, another challenge we face is to consider the interests of all parties as good as possible. However, before we can engage in this challenge, we need to find an answer for how to take personal interest into account at all. As we aim to increase satisfaction of all parties by addressing their interests we reformulate the question to:

How can satisfaction with the processing of claims be measured?

One plausible answer is to survey the parties involved in the claims settlement process. In other words, we interview the *experts*. By experts we mean both the internal parties, e.g., the manager or the clerk, and the external parties, e.g., the customers or the car workshops. By interviewing experts we obtain their opinions, perspectives and expertise which is why we refer to the resulting model as *expert-based*.

The difficulty is to ask questions in a way that they cover both the conscious and the unconscious evaluation criteria, but also to ensure that the answers are appropriately incorporated into the digital claims processing.

We assume for the rest of this thesis that the more conscious and/or unconscious criteria are met, the more satisfied the expert—and the higher their rating of the

2 Motivation

quality of the specific claims processing. This introduces the concept of *rating the quality*—which we operationalize as a valid measure of satisfaction.

To a certain extent, this concept can be compared to “rating a product” that one wants to purchase as a customer: For example, if we have to choose between different yoghurts while shopping at a supermarket, we will include both some attributes of the yoghurts we are consciously aware of in the evaluation, and possibly also attributes that we are not consciously aware of. In this context, conscious attributes may be taste, fat content or whether the yoghurt is organic—while external packaging may serve as a potential unconscious attribute (which the customer, or expert may only become aware of when discussing the evaluation afterwards).

Reasonably, the unconscious criteria are not easy to assess, especially in contexts, where many attributes influence the evaluation and the influencing attributes themselves are not easy to uncover. In fact, we assume both to be the case when rating claims processing.

One common statistical method to reveal unconscious criteria is the *conjoint analysis*. In a conjoint analysis experts do not value every single attribute or attribute level as it is the case for *compositional* approaches, such as the *self-explicated method*. Instead the whole product—here, the whole claims processing—is evaluated. Hence, the name conjoint, which stands for *CONsidered JOINTly* [BEPW18]. Thus, the experts do not evaluate each individual attribute level, but look at the claim submitted for evaluation—the so-called *stimuli*—as a whole. Based on the overall assessment of the claims processing, the *part-worth utilities* of each attribute level are then estimated. The conjoint analysis is therefore a *decompositional* approach. We describe conjoint analysis in more detail in section 3.1.1.

Conjoint analysis is used as a generic term for various decompositional procedures that estimate the structure of experts’ preferences. We will explain the methods we have used in terms of conjoint analysis in more detail below, and also address the differences to compositional approaches—especially the self-explicated approach.

One advantage of conjoint analysis is that it represents whole claims processings in the questionnaire and therefore has a greater similarity to a real choice or, in our case, the real evaluation situation [GHH07]. In our opinion, the importance of presenting realistic and whole damage cases reveals when considering the goal of the questionnaire—to measure satisfaction with claims processing, which is a so far rather unknown and innovative approach [BKM21]. We assume that a decompositional questionnaire design, by considering holistic damage cases, will allow experts to better adapt to an approach that may have been entirely new to them so far. This means they are not forced to evaluate each attribute for itself, which may not be possible to them not only because of the new approach, but also because their expert view might rather be holistic than isolated. We therefore assume that they can assess each process unbiased and according to their expert intuition when evaluating cases rather than attributes. Another advantage of conjoint analysis

is that it prevents us from getting only socially accepted answers [GHH07], and therefore, getting the real opinion of all expert levels—from the clerk to the manager. That means, conjoint analysis has a greater chance to detect the real importance weights for each expert compared to self-explicated approach [GHH07].

To represent holistic and real claims, we apply the *profile method*. It takes into account all attributes and not only a selection of these, as would be the case, for example, with the *two-factor method*, in which respondents evaluate pairs of attribute. Unlike to the evaluation of products, where choices between a pair of attributes can be evaluated validly, in claims processing more attributes and corresponding levels have to be evaluated. It is therefore not possible to submit all case constellations resulting from the combination of all attributes and corresponding levels to an expert for evaluation. Otherwise, the experts would have to assess several hundreds of theoretical claims, which is beyond human comprehension. That is one reason for why Green and Srinivasan suggest 30 stimuli as an upper limit [GS78].

In our case, the self-explicated approach would indeed be advantageous, as it entails a lower cognitive load for respondents in the evaluation process and enables respondents to evaluate a higher number of attributes and levels [SM18, GHH07]. Especially in the insurance context this is of great importance, as insurance datasets usually consist of many attributes [Gru18, Wal17a]. Nevertheless, we judge the advantages of the conjoint analysis to be higher and therefore, we have to guarantee that this disadvantage is taken into account when creating the questionnaire.

That is why we decided to use a so-called *hybrid approach*, which is an improvement of (especially traditional) conjoint analysis. It combines both the compositional and decompositional approaches and is mainly used in cases where researchers need to consider more attributes and levels than traditional approaches can handle [SM18]—which is the challenge we face.

Our questionnaire is inspired by the *adaptive conjoint analysis* described in more detail by Steiner and Meißner [SM18]. In our questionnaire design, each expert directly evaluates the importance of each attribute in a first step (i.e., compositional part) and the entire claims processing with selected combinations of attributes in a second step (i.e., decompositional part). For the second step, we only consider the important attributes per expert according to the first questionnaire. In this way, we reduce the spectrum of possible combinations.

Nevertheless, the so-called *full design* of conjoint analysis, in which an expert is presented with all combinations of the attributes and levels, is still not possible. To account for that, we reduce the cases by selecting a subset of stimuli for the questionnaire in such a way that it represents the full design as well as possible—a so-called *reduced design*). In section 4.1.1.1, we discuss in more detail how we used this improvement to reduce the number of attributes per expert and the method we used to select appropriate stimuli.

2 Motivation

In defining the stimuli, another challenges—besides the need to reduce the number of stimuli—is that the attributes and their levels are not independent and some constellations do not exist in reality. For example, in the case of mobile insurance, the constellation of glass damage and total loss is rather unusual (if realistic at all). An example of dependent attributes would be attorney fees, and a lawyer was involved in the claims processing, as there should be no attorney’s fees per se unless an attorney was involved in the claim settlement.

Standard approaches to defining the stimuli, such as the profile or two-factor method, have in common that the stimuli are fictional cases generated by combinations of the attributes and their levels. Therefore, it is very likely that unrealistic combinations are presented to the experts. In such cases, the experts cannot meaningfully evaluate the stimulus, which is why we have to ensure that the presented stimuli exist in reality. Therefore, we decided to use real damage cases as stimuli. In combination with the choice of the profile method, this has the advantage that we can simulate a real claims processing and thus imitate realistic decisions of the experts. We aim to improve the quality of the claims processing precisely by evaluating such realistic decisions.

Of course, it is not always possible to represent the entire spectrum in terms of a full design, i.e., we cannot consider all possible combinations of attribute levels, depending on the number of attributes and their levels. Instead, we need to ensure that all (or almost all) attribute levels are considered in the questionnaire so that we can estimate at least a part-worth utility per attribute level. The more frequently an attribute level can be considered in the questionnaire, the better we can obtain a stable estimate of the associated part-worth utility. These part-worth utilities are needed to subsequently calculate the total utility—the quality of the process—in the compositional part [BEPW18, SM18, GHH07].

Another advantage of conjoint analysis is the possibility to reveal interactions between attributes [GHH07]. This is important because it may provide viable insights to improve claims processing.

By discussing the results with the experts afterwards, we aim at getting relevant insights into the evaluation scheme for assessing the quality of a claims process per expert, which does not yet exist in this way.

In summary, we have chosen to use conjoint analysis—even though a very large number of attributes have to be considered in our context of claims processing in the insurance industry, which is why the use of self-explanatory approaches might be recommended—due to the following reasons: Since conjoint analysis considers the processing of a claim as a whole, we see a great advantage in the fact that a real appraisal situation can be simulated and thus hidden drivers and interaction effects, among other things, can be identified [GHH07]. We also assume that the decompositional approach allows experts to follow their expert intuition, which facilitates the completely new approach to evaluating the quality of a claims

processing. Conventional conjoint analysis approaches provide fictitious stimuli to the respondents, but in the insurance context this can lead to unrealistic claims that could not be evaluated by the experts. An advantage of the conjoint analysis is the possibility to take real claims as stimuli. Among other things, we want to use precisely this advantage of the conjoint analysis to better understand the evaluation of the quality and to draw insights from it in discussion with the respective expert.

A good overview and introduction to the methods of the conjoint analysis is given by Steiner and Meißner [SM18], Gustafsson and colleagues [GHH07], or Backhaus and colleagues [BEPW18]. For a detailed description of the conjoint analysis compared to self-explicated approaches see Gustafsson and colleagues [GHH07], or Steiner and Meißner [SM18].

As described above, we decided to select real claims from an insurance dataset to generate stimuli. However, a simple random sample may not yield a good representation where each attribute level in the questionnaire is considered [BEPW18]. Therefore, the next question we ask is as follows:

How can representative and real stimuli for each expert be selected from a given insurance dataset?

The goal we pursue with the questionnaire is to measure the satisfaction per expert with the quality of the specific processing of claims, taking into account all the different perspectives of the parties involved. In order to gain insights into different perspectives of the involved parties, we therefore create *individualized questionnaires*. This gives us the opportunity to see *per expert* which aspects of the claims processing are considered important and are weighted positively or negatively in terms of quality. For example, to consider the customer satisfaction, as described in section 2.1.1, it is necessary to ask customers—remember that we also consider them “experts”—what they consider positive about a claims processing.

Asking the experts and individualizing the questionnaires further motivates us to frame our recommendation system *expert-based*.

As discussed above, we apply the hybrid approach and therefore have to create two questionnaires. As stated, the experts are asked to evaluate the importance of each attribute contained in the dataset in a first questionnaire (i.e., the compositional part of the applied conjoint analysis), which we will discuss in more detail in section 4.1.1.1.

The goal of the second questionnaire is to present stimuli to the experts for evaluation so that we can estimate part-worth utilities for each attribute and the corresponding level. Each stimulus is a whole claim, and to represent the processing of whole claims, we consider only *closed* processes. Thus, we look at the processing of claims *ex post*.

2 Motivation

To select real claims as stimuli, we proceed as follows. Based on the attributes that seemed important to the respective expert according to the first questionnaire, we aim at selecting those cases that best represent these attributes—in a sense, span the space of these attributes.

To find good representatives of the dataset that cover the entire space of selected attributes, we use *cluster analysis*. More precisely, we apply the *k-medoids* method. Similar to *k-means*, it splits the dataset of n observations into k clusters. But instead of *k-means*, it uses actual data points as the centers of each cluster—called *medoids*. We use these medoids as our representatives. Another reason for applying *k-medoids* is that it can be classified as a *mixed clustering* method, which also allows to process variables of mixed data type that are usually present in insurance datasets.

The *k-medoids* methods measures how similar or dissimilar an observation is to other observations and assigns the observation to the closest cluster based on this *dissimilarity measure*. For real-valued attributes, we could apply a distance measure such as the Euclidean distance, which is not possible for mixed attributes. Therefore, we need an appropriate dissimilarity measure that can handle these different scales. We have chosen to use the *Gower's coefficient* of Kaufman and Rousseeuw [KR05]. For more details to Gower's distance we refer to section 3.1.2.1

Based on Gower's distance *k-medoids* forms cluster where all observations within a cluster are rather similar, while the observations of different cluster are rather dissimilar. That is, we look for observations whose average dissimilarity is minimal to other observations *within* a cluster and maximal to observations of *other* cluster. Or, to frame it in our context, the claims within a cluster are most similar based on their attributes and therefore can be expected (in a best case) to be most similar in their quality, whereas claims from different clusters are most dissimilar in both attributes and quality. For more detail on *k-medoids* we refer to section 3.1.2.

In summary, with the *k-medoids* clustering we identify k medoids and thus have k representatives of our dataset that ideally span the entire attribute space and are thus well suited as real stimuli. However, insurance datasets often contain a large variety of attributes and corresponding attribute levels—even after reducing the number of the attributes with the first questionnaire. Moreover, there may be asymmetric attributes in the dataset with levels that are sparsely filled.

Therefore, there may exist attribute levels that do not appear in the second questionnaire. In order to have enough data to estimate part-worth utilities for all attribute levels, we therefore implemented a so-called *conditional sampling from k clusters*—i.e., we sample another claim from each cluster (besides the medoid), under the condition that the missing or sparsely populated attribute level is included. In order not to destroy the “structure” of the clusters, the observation with the missing attribute level is taken from the cluster that most frequently contains the missing attribute level. Furthermore, to avoid causing imbalance in the selection of stimuli

2.1 Expert-based improvement of the quality of claims processing

by drawing a claim only from some clusters and not from all clusters, we generate half of the stimuli via k -medoids and the other half by conditional sampling from the clusters.

We describe the methods for selecting appropriate and real stimuli applying our conditional sampling approach in more detail in section [4.1.1.1](#).

The above mentioned sampling procedure is our answer for how to find representative and real stimuli for the questionnaire. To appropriately account for the different levels of expertise of each expert group considered, we created personalized questionnaires for each group of experts—in this thesis later on called *models*. However, the question of how to measure the experts *satisfaction* with the claims processing—the *quality*—still needs to be answered. For that, we presented each expert with the personalized stimuli—real closed claims—in the (second) survey. For each stimulus, they were asked to indicate their satisfaction with the claims processing on an *interval scale* with overlapping intervals, i.e., each expert was asked to indicate a percentage range of his or her satisfaction with the claims processing. The overlapping interval scale was presented as shown in figure [2.2](#).

0%– 20%	10%– 30%	20%– 40%	30%– 50%	40%– 60%	50%– 70%	60%– 80%	70%– 90%	80%– 100%
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 2.2: Overlapping interval scale to indicate the experts' satisfaction with the claims processing.

On the left side we see the worst case, i. e. an expert rates a claims processing in a range of 0 % to 20 % or, the expert is satisfied with the claims processing only to 0 % to 20 %. If the expert is still not satisfied with the claims processing, but from the personal point of view more criteria are met, the claims can be assessed in satisfaction interval 10 % to 30 %. In a best case, the claims processing is rated in the range 80 % to 100 %, which means that the expert is completely satisfied with the claims processing.

With the interval scale, especially the overlapping interval scale, we enable the experts to give a tendentious rather than a direct evaluation of the process. Thus, we give each expert the possibility to rate a claims processing spontaneously and based on their expert intuition. We consider this important as the explicit evaluation of the claims processing is new to experts—as already mentioned above.

After all experts have evaluated their stimuli, we can estimate the part-worth utilities for each attribute level per expert or model. For this, we use the standard conjoint analysis methods discussed in section [3.1.1](#) and [4.1.1.2](#).

2.1.3 Improving the expected quality of claims by finding appropriate recommendations

As motivated above, we identified the criteria and their corresponding part-worth utilities that lead to the quality rating of a given claim per expert. With that, we can estimate a measure of *quality* for each claim in the whole data set—which is one way to sophisticatedly generate this missing variable in the dataset. After these steps, we finally have a dataset that includes the quality of claims processing. In addition, we obtained personalized criteria per model, which, if met, lead to higher satisfaction with the claims processing. Yet, another relevant question is still unanswered.

How can the the quality of the claims process be improved?

In this thesis, we aim to answer this question by deriving appropriate recommendations that can be used to improve the claims processing.

This improvement of the quality of a claims processing can be formulated as an *optimization problem*, i.e., the problem is to optimize the quality of the claims processing by making appropriate recommendations. We call it *expert-based recommendation system* or EBRs problem. We motivate the EBRs problem below and refer to section [4.2](#) for more information, in particular on the mathematical model.

So far we have worked with completed claims processes or, in other words, we have considered the claims processes *ex post*. However, in order to improve the claims processing, it is necessary to consider them *ex ante*, as otherwise, it would not be possible to intervene positively during the claims processing. It is reasonable that, in a best case, we intervene already at the beginning of the claims processing, i.e., directly after the claim notification.

At the beginning of claims processing not all attributes of the dataset are available. In addition, and perhaps more importantly, there are still a variety of options which may or may not improve the quality of the process. As we have seen in section [2.1.2](#), the criteria for assessing quality are individual and depend on the particular expert. These criteria, which have not yet occurred, are those that we can influence to ensure the most satisfactory processing of the claim. Thus, these criteria correspond to the *recommendations* R_i , $i \in [n]$, where n is the number of all recommendations (independent of the model).

Since not all variables of the claims processing are available, we cannot calculate the quality, but estimate the *expected quality* of the claim. For that, we identify *prototypical classes of claims* based on the given information at the beginning of the process using innovative predictive algorithms. These algorithms, which are based on algorithms of convex optimization, are not subject of this thesis. Briefly summarized, we used methods similarly to the prototypical bookings by Brieden and Gritzmam when generating the prototypical classes of claims [\[BG20\]](#).

2.1 Expert-based improvement of the quality of claims processing

The variables available at the beginning of the processing of a claim are used to predict the expected quality and to identify prototypical claim classes. In addition and in interaction with the prototypical claim classes they can also be used to adapt the recommendations to the respective claim type. For example, it might be useful to distinguish between glass damage and collision damage when making recommendations (provided the information about the type of damage is available). Moreover, which recommendations make sense in the respective situation depends to a large extent on the claims processing of the respective insurance company and should therefore be clarified in cooperation with the experts. Since it depends on the given situation of the respective insurance company, we will only address this aspect to a very limited extent in this thesis.

To give an example of the usefulness of adapting the recommendations to the specific situation, let us consider on the one hand a glass damage and on the other hand a collision damage. In the case of a glass damage, an appropriate recommendation package may be to handle the claim as quickly as possible and without direct customer contact (e.g., without a phone call or e-mail contact). This would probably not be a desirable recommendation package in the case of a collision damage claim where there are also casualties. Here, it may well make sense—and also be desired by the customer—to contact the customer personally.

As already mentioned at the beginning of this section, we improve the quality of the claims processing by formulating an optimization problem called the EBRs problem. Furthermore, we have illustrated that the criteria of the second questionnaire of the conjoint analysis for calculating the quality can be considered as recommendations for improving the quality. As each expert received a personalized questionnaire for this purpose, there are also different recommendations per expert. This of course also affects the expected quality.

In order to formulate an optimization problem, we define a so-called *processing index* PI_j —representing the expected quality—per model $j \in [m]$, where m is the number of experts interviewed. The goal of the optimization problem is to maximize the processing index. Here, the processing index aggregates all recommendations and their associated weights per model—the weights of the recommendations will be discussed below. In general, there exist different options to aggregate the recommendations, such as, *additive* or *multiplicative* methods. In terms of optimizing the expected quality, we chose additive aggregation of the recommendations.

Adding up all processing indices PI_j , $j \in [m]$, of the models provides a so-called *aggregated processing index* PI , which is given as follows.

$$PI = PI_1 + \dots + PI_m$$

Of course, the quality does not depend directly on the criteria, but much more on the associated weighting when the corresponding recommendation is *activated*.

2 Motivation

Before we explain what we mean by *activating*, we explain the weights of the recommendations in more detail. The weighting γ_i per recommendation R_i , $i \in [n]$, is largely composed of the part-worth utilities from the conjoint analysis (see section 2.1.2). This part-worth utility corresponds precisely to the influence on the expected quality when the recommendation is made. In addition to the part-worth utilities also other informations can be considered in the weighting. For example, the prioritization of customer satisfaction could be included. We will describe the weights in more detail in section 4.2.1.

As mentioned above we aim at improving the quality of a claims processing by making or *activating* a recommendation. We formulate this (mathematically) by a so-called *activation variable* given by the following formula.

$$x_i = \begin{cases} 1 & \text{if the recommendation is activated} \\ 0 & \text{if the recommendation is not activated,} \end{cases}$$

for all $i \in [n]$, with n is the number of recommendations in total.

In summary, with $x_i \in \{0, 1\}$ for all $i \in [n]$ and by additive aggregation of the recommendations per model, the objective function of the EBRs problem is given by

$$\text{PI} = \sum_{i=1}^n \gamma_i x_i$$

The EBRs problem is therefore a so-called *integer linear program* or ILP, which we explain in detail in section 4.2.1. Furthermore, we refer to [BKM21]. For more information about ILP's we refer the interested reader to section 3.4.

The objective of the EBRs problem is to make an appropriate or optimized set of recommendations so that the quality of the claims processing improves—depending on the given situation, as discussed above.

In terms of application, the EBRs problem can be formulated as follows.

Remark 2.1.1. The (expected) quality of the claims processing of a potential claim in the future should be optimized by suitable selection of recommendations. Thus, the goal is to activate from all possible combinations of recommendations those that maximize the expected aggregate *processing index* (PI).

With the EBRs problem, we thus generate an automated recommendation system. This digital process can support a clerk and thus create new freedom, for example, to respond more closely to customer needs when necessary (see also section 2.1.1).

In the following figure 2.3 we will illustrate the EBRs procedure for better understanding.

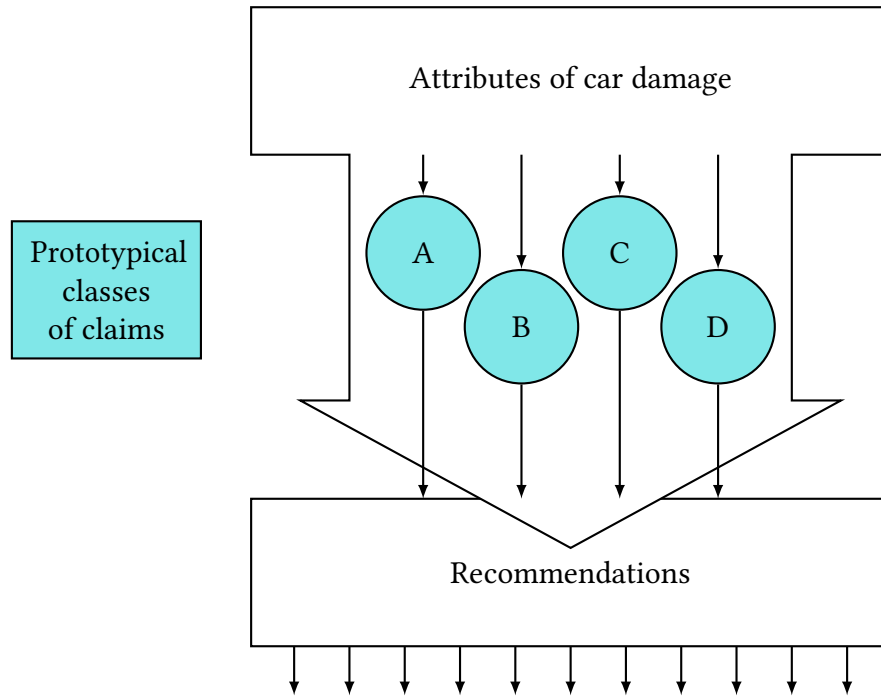


Figure 2.3: Schematic illustration of the EBRS procedure.

Let us assume that a car damage is reported to the insurance company. With the information available at this point, we can predict the expected quality of the claims processing per expert by assigning the claim to a prototypical class of claims. This makes it possible to assess at the outset whether this claim requires special attention. Using the attributes available at the beginning of the process and the prototypical classes of claims, we identify those recommendations that are suitable for improving the quality. Therefore, we identify per model PI_j , $j \in [m]$, a set of appropriate recommendations for improving the model based expected quality. Based on this sets of appropriate recommendations (per model), and given certain constraints, a final set of appropriate recommendations can be activated to maximize the expected aggregated PI. An example of a constraint is an upper limit on the pronunciation of recommendations. We will discuss constraints in section [4.2.1.1](#)

In section [4.2](#) the search for a suitable set of activated recommendations is described and the model is presented as an optimization problem. Furthermore, a so-called *decision version* of the problem is formulated. That is, instead of pronouncing an activated set of recommendations, the decision problem gives a yes or no answer if such a set exists. In section [4.2.2](#) we show that we can transform the decision version of the EBRS problem to the optimization version and vice versa.

One goal of this thesis is to study the *complexity* of the EBRS problem for different cases and restrictions. We address this question in section [4.3](#). In the following section, the terms *complexity* and *efficient algorithm* are introduced and their relevance to the problem is motivated.

2.2 Complexity of problems

In practical applications, an *efficient* algorithm that solves a problem, i.e., an algorithm that runs in acceptable time and consumes an acceptable amount of computer memory, is obviously desirable. It depends on the particular situation when one can speak of an algorithm that performs acceptably. If we apply the EBRS problem in the insurance industry to find appropriate recommendations to improve the claims processing, such an algorithm should solve the given problem within seconds or minutes. With that a claim handler can take appropriate steps to process the given claim as soon as it is reported.

In this thesis, we consider the performance of the EBRS problem more generally. More specifically, we consider the *worst-case running time* of the problem. The amount of space it requires to compute the problem is not topic of this thesis. The question we ask is as follows:

Can the EBRS problem be solved by an efficient algorithm?

To answer this question, we must first define what is meant by an efficient algorithm.

In computer science we distinct between *polynomial time algorithms* and *exponential time algorithm* [GJ79]. An exponential time algorithm is thereby defined as follows.

Definition 2.2.1. Let $n \geq 0$ be the length of the input for a given problem and $c > 0$ be a constant. Furthermore, let $p(n)$ be a polynomial of n . If the computation time $f : \mathbb{N} \rightarrow \mathbb{R}^+$ of an algorithm A terminates within $c \cdot |p(n)|$, i. e., $|f(n)| \leq c \cdot |p(n)|$, then we speak of a *polynomial time algorithm* [GJ79, KV06, DPV06]. The *complexity* of A is denoted by $O(p(n))$ [GJ79].

We discuss this in more detail in section 3.3.1

Any algorithm for which the time complexity cannot be bounded by a polynomial $p(n)$ is called an exponential time algorithm, although it would be more accurate to say that it is a non-polynomial time algorithm. An example for a non-polynomial time algorithm is given by an algorithm with time complexity $O(n^{\log n})$ [GJ79].

Finally, we speak of an efficient algorithm, if it solves a problem in polynomial time [KV06]. We group the problems which can be solved by an efficient algorithm in the complexity class \mathbb{P} . It is important to note that for the classification we are not looking at the optimization version of the problem, but the decision version, i. e. the complexity class \mathbb{P} consists of *decision problems* that can be solved (deterministically) in polynomial time. To remember, a decision problem is defined as a problem that can be posed in such a way that one can answer *yes* or *no* to it. For more details we refer to section 3.3.2.

More generally, it is common to compare the complexity of different problems based on their decision version. This is also the case in this thesis. However, in case of the EBRs problem there is no difference between the optimization and the decision version of the problem in terms of polynomial time solvability. We prove this statement in section 4.2.2. In this thesis, we therefore often speak of the EBRs problem without specifying more precisely whether this is the decision or optimization version.

Finally, to answer the latter guiding question, the EBRs problem cannot be solved by an efficient algorithm and therefore, it is not contained in \mathbb{P} . We prove this in section 4.2.3.

Instead, the EBRs problem is *intractible*, i. e., “it is so hard that no polynomial time algorithm can possibly solve it” [GJ79, p. 8]. Or in other words, there exists no efficient algorithm to solve the problem. It should be mentioned that in this thesis we consider intractability only in terms of running time and not whether a solution can be represented polynomially. Furthermore as already stated, we observe the worst-case running time to classify a problem according to its complexity. Thus, regardless of the particular encoding scheme or computer model, an intractable problem cannot be solved efficiently [GJ79].

In section 4.2.3 we show that the EBRs problem belongs to the class of decision problems which can be solved in *nondeterministic polynomial time*, i. e. we cannot solve the problem efficiently, but we can verify in polynomial time whether a given solution solves the problem. This class of problems is denoted by NP .

The complexity classes \mathbb{P} and NP are known by the problem

$$\mathbb{P} \stackrel{?}{=} \text{NP},$$

which is unsolved so far. It is one of the famous millennium problems¹. The so-called \mathbb{P} vs. NP *problem* was first formulated by Stephen Cook and Leonid Levin². However, it has already been shown that $\mathbb{P} \subseteq \text{NP}$ [GLS93]. In this thesis, we assume that $\mathbb{P} \neq \text{NP}$ as it is widely assumed.

In section 4.2.3, we do not only show that $\text{EBRS} \in \text{NP}$, but also that the EBRs problem is NP-hard . The class of NP-hard problems consists of intractible problems that can be transformed polynomially to any problem of the class NP . And therefore, if one could find a polynomial solver for any problem of this class, one could solve all problems. In summary, the EBRs problem is not only included in the complexity class NP , but also in the complexity class of NP-hard problems. Therefore, we know that it is an NP-complete problem. This class of problems was first introduced by

¹For the millennium problems, refer to <https://www.claymath.org/millennium-problems>

²For the official problem description, see <https://www.claymath.org/millennium-problems/p-vs-np-problem>

2 Motivation

Cook [Coo71]. In his work, he proved the NP-completeness for the first problem—the so-called SAT problem. Based on this result, Karp found a set of NP-complete problems like the CLIQUE problem or the Knapsack problem [Kar72]. For more information about the class of NP-complete problems we refer to section 3.3.3.

NP-complete problems standardize all problems of the class NP, because each problem is a special case of the other. This is due to the polynomial transformation. A very important representative of this class are *integer linear programs*—ILPs for short. Various combinatorial problems (e.g., matching, CLIQUE, etc.) can thus be reduced to integer programming problems [GLS93]. For more information about ILPs we refer to section 3.4. In sections 4.2.1.1 and 4.2.1.2 we define the EBRS problem as an ILP.

In summary, we know that the EBRS problem is NP-complete and therefore cannot be solved in polynomial time. This would, of course, be advantageous for practical applications. For example, in our context within the insurance industry, it would be beneficial to have a suitable set of recommendations for the processing of one specific claim available within a very short time as soon as a damage is reported.

But, depending on the situation and the inputs and parameters needed to formulate the EBRS-situation problem, NP-completeness does not preclude the existence of an efficient algorithm to solve the problem [DPV06]. As we recall, in complexity theory we consider the problem more generally and classify its complexity according to its worst-case running time.

Therefore, a main part of this thesis deals with the search for different cases of the EBRS-problem which are solvable in polynomial time. In doing so, however, we find that the problem nevertheless remains NP-complete under certain restrictions on the constraints. This complexity-theoretical study is carried out in section 4.3.

For the following cases we can prove that the EBRS problem is solvable in polynomial time.

First, there exists two special cases of the EBRS problem with *totally unimodular* matrix of constraints. This special cases are considered in sections 4.3.1.1 and 4.3.1.2. Total unimodularity guarantees that we find an integral optimal solution of the problem in polynomial time by ignoring the integrality constraint—this is a so-called *LP relaxation*. We discuss the LP relaxation and the total unimodularity in sections 3.5 and 3.6.

Second, we find a special case of the EBRS problem in which a so-called *Greedy algorithm* solves the problem optimally in polynomial time. We discuss this case in section 4.3.3.1.

Another interesting investigation would be to what extent the EBRS problem can be solved approximately. That is, whether there exists an algorithm that finds the best possible approximation—rather than an optimal solution—of the given problem in polynomial time. Such an approximation is given, for example, by an LP relaxation

mentioned earlier (see section 3.5). This is because, in general, we cannot find an optimal solution for an ILP if we ignore the integrality constraint—except for totally unimodular constraint matrices.

In this thesis, one focus is on the complexity-theoretic study of the problem, but we give an outlook on possible approximate algorithms in the sections 5.1 and 5.

2 Motivation

3 Definitions and Preliminaries

In chapter 3 we define, explain, and describe important methods, terms, statements, and theorems from statistics, graph theory, and (linear) optimization that we use in the remainder of this thesis.

In section 3.1 we present the statistical methods to detect attributes or characteristics with which an expert (consciously or unconsciously) rates the quality of the claims processing. In subsection 3.1.1 we introduce the conjoint analysis which we use to add the variable *quality* to an insurance dataset—a variable that was not previously included. Furthermore, we will use this method to determine the part-worth utilities, which indicate the level of influence of individual attributes on quality. To construct the questionnaire for the conjoint analysis, we apply k -medoids clustering. This allows us to find suitable stimuli that are evaluated by the experts. We describe the k -medoids algorithm in detail in subsection 3.1.2.

In section 3.2 we give a brief introduction to graph theory. We define the terms used and introduce some special sets of graphs that we need for the complexity-theoretic investigations of the EBRS problem. Among other things the graph-theoretic terms allow us to formulate combinatorial optimization problems in a more descriptive way.

The main part of this thesis focuses on the complexity-theoretic investigation of the EBRS problem which is NP -complete (see section 4.2.13). In section 3.3 we therefore present in detail the complexity theory mainly used in this thesis. We start with the definition of the worst-case running time of algorithms in subsection 3.3.1 in order to classify the EBRS problem according to its efficiency and to compare it to other optimization problems.

In subchapters 3.3.2 and 3.3.3 we describe the complexity classes NP as well as NP -completeness and explain what is meant by intractability. In addition, we discuss methods to prove that a problem is NP -complete. Finally, in subsection 3.3.4 we describe other NP -complete problems used in this thesis.

We will formulate the EBRS problem as an integer linear program (ILP) in the remainder of this thesis. In section 3.4 we therefore give an introduction to ILPs and furthermore discuss LP relaxations.

For the complexity-theoretic study, we discuss the special feature for ILPs with a totally unimodular constraint matrix in section 3.6. In such cases, we know that every solution of the corresponding LP relaxation has an integer solution. To this

end, as a first step, we define totally unimodular matrices and introduce lemmas and theorems later used to prove the total unimodularity of a matrix.

3.1 Statistical methods to determine the quality claims processing

Before we can optimize the quality of the claims processing, we have to work with experts to define what is meant by quality. Since the consideration of a claims processing according to its quality is new [BKM21], we need to facilitate the evaluation of the processing to the experts by (1) a suitable questionnaire design as well as (2) a method to reveal the conscious and unconscious assessment criteria of the quality of a claim. In this section, we describe the methods used for that.

Conjoint analysis is a statistical method that consists of several steps that are designed differently according to the given purpose [GHH07]. In the following section we describe the conjoint analysis as we use it to implement the quality of the claims processing into an insurance dataset. In order to justify the method used in each step of the conjoint analysis, we already partially address the difficulties and advantages that arise in assessing the quality of the claims processing. The description of the conjoint analysis is mainly based on Gustafsson and colleagues [GHH07], and Steiner and Meißner [SM18]. In the second part of this section we describe the k -medoids algorithm, which we use to select suitable stimuli for the conjoint analysis.

3.1.1 Conjoint analysis

In subsection 2.1.2 we presented the objective of the EBRs problem—to improve the quality of an insurance company’s claims processing by optimizing the expert-based recommendation system. As the name of the optimization problem implies, the recommendations we need to improve the quality of the process must first be obtained through appropriate surveys of the experts. Yet, not only the recommendations have to be assessed, but also the weighting of the individual recommendations.

A suitable framework for such a survey is provided by the *conjoint analysis*. It is a suitable tool to measure the *preference* of each expert with respect to the processing of a claim [SM18, GHH07]. Here, we use the term “expert” to refer to different people involved in the claims processing. In order to improve the quality in its entirety, all opinions of the involved parties have to be asked. Of course, in the claims processing of an insurance company there exists different perspectives—e.g., that of the customer, the claims handler, or the manager. Our goal is to capture all these different perspectives individually with the help of a survey.

A claims handler requires to meet the different needs of the parties involved so that all parties are satisfied at the end of a claims processing. This can be a challenging task, even more if the parties involved evaluate the processing of the claim differently.

Another challenging task is that each party involved—or each expert—evaluates the claims processing according to conscious, but also unconscious criteria, some of which a claims processor cannot know—especially as the criteria may differ between experts. Therefore, in order to improve the quality of the claims processing and to support a claims handler, we need to ensure that both conscious and unconscious decisions are taken into account and that the different perspectives are considered. In particular, the uncovering of unconscious decisions is the strength of conjoint analysis [GLK10, CRB09].

In this section, we define the methods of conjoint analysis and show, how it can help to improve the quality of the claims processing. We proceed in a similar way to Gustafsson and colleagues [GHH07], and Steiner and Meißner [SM18], as follows: By presenting various claims processings to each expert for evaluation in their entirety, we aim to uncover both unconscious and conscious decisions that led to the assessment of the quality of the claims processing. Through the conjoint analysis questionnaire design, we know the decision criteria (per expert) from which we derive the recommendations for the claims processing (see subsection 4.1.2). For the weighting of the individual recommendation, we decompose the overall evaluation of the claims processing into individual estimated *utility* per decision criterion—the so-called *part-worth utilities*. Conjoint analysis is therefore also referred to as a *decompositional approach* [SM18, BEPW18].

In general, conjoint analysis can be divided into the following steps, which are handled differently depending on the application [GHH07]. In the course of this section, we will explain which conjoint analysis methods we have used in each of these steps and explain them at the respective point. For more information on the other methods not explained, we refer the interested reader to [GHH07, SM18, BEPW18, GLK10]. The steps are illustrated in figure 3.1 which is adapted from Gustafsson and colleagues [GHH07].

3.1.1.1 Selection of attributes and their levels

Unlike in [GHH07], we do not start with the selection of a *preference function*, but search for suitable *attributes* and their *levels*, as recommended by Vriens [Vri95] and Backhaus and colleagues [BEPW18]. This seems important for the specific application of conjoint analyses in our context of claims processing because of two reasons. First, unlike methods used, e.g., in the launch of a new product, we use the variables of the dataset as attributes for the survey. Secondly, many attributes exist for a claims processing of an insurance company. Selecting all attributes would

3 Definitions and Preliminaries

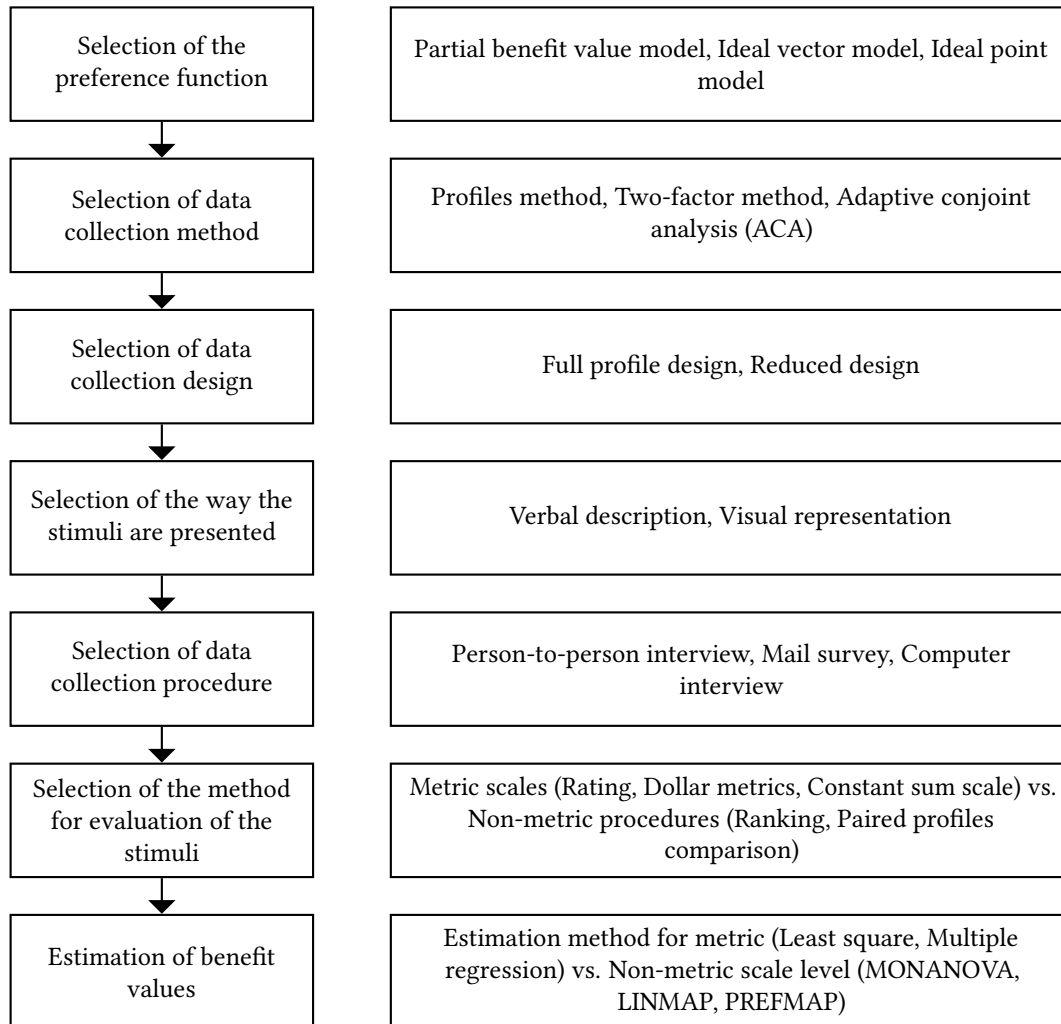


Figure 3.1: Steps of Conjoint Analysis—adapted from Gustafsson and colleagues [GHH07 p. 5].

result in too large a set of attributes for a survey, and moreover, it is very likely that not all variables in the data set are important to the expert. Thus, we need to find an appropriate selection of attributes.

For that, we applied a preceding questionnaire to reduce the number of attributes. This can be compared to the *self-explicated approach* that also asks for the relative importance of each attribute in a first step [GHH07]. By applying this self-explicated approach it is possible for the experts to process a greater number of attributes and their levels [GHH07, GS90].

The goal of this preceding questionnaire is to assess the experts perceived importance of each attribute of the dataset. The experts rate each variable of the dataset, and based on this rating we can reduce the number of attributes for stimuli selection. We discuss the resulting preceding questionnaire in detail in subsection [4.1.1.1.1](#).

3.1.1.2 Selection of data collection method

The major challenge in evaluating a claims processing is that the *quality* of the process has not had to be evaluated before. The experts are therefore in a somehow new situation, which needs to be made as simple as possible. On the one hand, the applied method should ensure that the experts themselves uncover the characteristics of a good claims processing without being influenced by external opinions. On the other hand, it is also intended to ensure that the data collection process identifies the subconscious decisions so that these can be incorporated into the estimation of the part-worth utilities [GLK10].

Another challenge is that the attributes and their levels are usually considered independent. Yet, the independence usually is not given in the evaluation of a damage processing. This is particularly important when estimating the part-worth utilities. However, when collecting data, we should make sure that these dependencies are also reflected in the stimuli so that they can be evaluated accordingly. It can be seen that these dependencies cannot be captured when using the *one-factor method*. Another classic data collection method is given by the *two-factor method*, in which respondents are presented with two attributes to choose from. It is therefore also called *trade-off analysis*. In the case of the insurance data set—where the attributes are interdependent—a trade-off is not possible.

Finally, the *profile method* provides the best conditions for taking into account all these aspects to be considered. In the profile method all attributes are considered simultaneously for evaluation. By taking a holistic view, it is possible that the dependencies between the attributes are mapped and also unconscious decisions are more likely to be revealed. Furthermore, it is assumed that the holistic view also reduces the risk that the experts will be influenced in their decisions by external influences (i.e., demonstrating socially desirable response tendencies) [GHH07, MK08].

3.1.1.3 Selection of data collection design

In this step, we have to choose between a *full profile design* and a *reduced design* [GHH07, BEPW18, SM18]. In the first case, all combinations of attribute manifestations are included in the questionnaire. However, this bears the risk that data collection becomes very expensive and overwhelms the respondents [GHH07, BEPW18]. Since especially our insurance dataset contains a lot of attributes (even after reduction of the attributes by the preceding questionnaire), only a reduced design is suitable in our case. A very frequently used reduced design is Addelman’s *orthogonal design* [Add62]. It is an established method to be particularly useful for constructing asymmetric and symmetric main effect designs [GHH07]. The orthogonality is particularly helpful for maintaining attribute independence. But this independence is not often given in insurance datasets.

Instead, we developed a reduced design method that does not resolve the dependent structure and represents all combinations of attribute manifestations as well as possible. We do this by applying the *k-medoids algorithm*—a *clustering method*—which we will discuss in more detail in subsection 3.1.2. In this clustering algorithm, the requirement of orthogonality is replaced by a search for *k* clusters that represent the dataset as well as possible, thus ensuring to some degree that all relevant combinations are represented. Each cluster center—the *medoid*—is then selected as one stimulus. It is important to note that each medoid is a *real* claim processing with which we assure the preservation of the dependency structure of the dataset.

3.1.1.4 Selection of data collection presentation and procedure

Deviating from the procedure in [GHH07], we summarized the steps *Selection of the way the stimuli are presented* and *Selection of data collection procedure* in this section. In this step, the questionnaire designer must choose between how the stimuli are presented to the respondents, i.e., to choose between a verbal description, or a visual representation, and how the data will be collected. This can be done, e.g., through a face-to-face interview or a paper-based survey.

As mentioned above, the experts face the challenge of evaluating the claims processing according to its quality. In addition, they face the challenge of having to evaluate a large number of stimuli in order to do justice to an overall picture of all possible claim combinations (see also subsection 4.1.1.2), which exceeds the recommended number of at most 30 stimuli given in the literature [GS78]. This is one reason why—to avoid overwhelming the experts—we chose an interface they are familiar with for presenting the stimuli, i.e., the interface they usually use to process a claim in real-live contexts.

Due to the number of stimuli, the system chosen to present the stimuli, as well as the challenge of evaluating quality already mentioned, we believe it is plausible to

let the experts assess the stimuli at their own pace (and on their own). The experts could indicate the respective assessment result in a questionnaire.

3.1.1.5 Evaluation of stimuli

There are two different ways to rate the stimuli, using a metric scale or a non-metric scale. The usual approach is to rate the stimuli using a *ranking* [BEPW18], where the outcome is an order of preferences [GHH07]. If a metric scale is preferred, a *rating method* should be applied, where each stimulus is evaluated by a numbered scale. Beside ranking and rating method, there also exists *paired profil comparisons* like the *dollar metric* [GHH07, SM18].

The ranking method is not appropriate in our insurance context because there are different types of claims contained in an insurance dataset—for example, glass damage and collision damage. In case of a glass damage, we may focus on different attributes when evaluating quality than in case of a collision damage. Therefore, the two types of damage cannot be compared in a direct sense of ranking.

For this reason, the experts were not asked to rank the claims in order of satisfaction, but rather asked to indicate their satisfaction for *each* of the claims processings on a percentage scale, e.g., 100 % satisfied with the processing of a claim or only 0 %. By using a rating scale the expert can express the intensity of satisfaction more directly.

To facilitate the evaluation process, we used an *interval scale*, i.e., the experts were not instructed to evaluate the processing of the claims directly, but to indicate the interval in which they would locate the claim processing. Furthermore, the intervals were overlapping. This overlapping interval scale can be justified by the following reasons. We recall that the evaluation of a claims processing according to its quality is new for the experts. Therefore, it will be difficult for (some) experts to directly indicate the percentage that expresses the satisfaction with the process. Instead, an expert may be undecided whether to select the option 80 % or 90 %, for example. In an interval [80 %, 100 %] both options are included. In the next overlapping interval [70 %, 90 %], the expert can make cutbacks, but isn't forced to commit to the option of a rather very high or yet not so high satisfaction rating.

Overall, with overlapping intervals, an expert can make decisions based on rules of thumb and, more importantly, based on expert intuition. And with the ability to decide from the gut, we want to favor the evaluation by subconscious decisions. We describe the evaluation of stimuli in more detail in subsection 4.1.1.2.

3.1.1.6 Estimation of part-worth utilities

From the previous step, we know that we have chosen a rating method to evaluate the stimuli and therefore, our data has metric scale. For metric scale, a regular dummy regression method is used most frequently. Within that, the most important estimations techniques are monotonic analysis of variance (*MONANOVA*), linear programming technique for multidimensional analysis of preference (*LINMAP*), and ordinary least square regression (*OLS*), [GHH07]. For more information on *MONANOVA* we refer to Backhaus and colleagues [BEPW18]. The *LINMAP* method is explained by Srinivasan and Shocker [SS73], and Thakkar [Tha21].

We decided to use an *OLS* regression or *multivariate regression*, respectively, to estimate the part-worth utilities considering the dependence of the attributes. When estimating the part-worth utilities using multivariate regression analysis, we need to take into account the dependent attributes that may be given. Among other things, it may be further necessary to consider interaction effects given in the insurance dataset¹

For more information about multivariate regression, multicollinearity and interaction effects we refer to Hastie and colleagues [HTF09], Fahrmeir and colleagues [FKLM13], and Field and colleagues [FMF12]. As estimation techniques for non-metric scale are not applied in this thesis, we therefore refer to Gustafsson and colleagues [GHH07], Steiner and Meißner [SM18], and Backhaus and colleagues [BEPW18].

3.1.2 Clustering with k -medoids

In the last section, we described the method we use to measure the quality of the claims processing and, based on this, find suitable recommendations for improving the claims processing. We also discussed above that standard orthogonal designs cannot be applied to measure the satisfaction with a claims processing as the attributes are not independent. Therefore, we decided to present the participant real claims, i.e., we selected real damage events from a given insurance dataset.

There are several methods for selecting real claims from a given dataset. The simplest method would be a random sample, yet, a random sample has the disadvantage that it does not guarantee the representation of the entire spectrum of possible combinations of attribute manifestations [GHH07, BEPW18].

Instead, we applied a *clustering method* with which we cover the entire space of all possible combinations of attribute manifestations. Or, in other words, find k

¹The exact description of the estimation method is not part of this thesis, since on the one hand the focus is on the complexity-theoretical investigation, and on the other hand the exact implementation of the estimation cannot be mentioned due to data protection and business secret reasons.

representative objects among the observations of the dataset that represents various aspects of the structure of the dataset—the *k-medoids method* [KR05].

In this subsection, we introduce cluster analysis in general in a first step and explain the *k-medoids* algorithm in more detail in a second step. We proceed as described by Hastie and colleagues [HTF09], Backhaus and colleagues [BEPW18], Hennig and colleagues [HMMR15], and Kaufman and colleagues [KR05].

Utilizing cluster analysis, we group a collection of observations from a dataset into subsets or *clusters* such that the observations within each cluster are more closely related than the observations associated with different clusters [HTF09]. In other words, observations within a cluster are very similar to each other, while observations in different clusters are very different from each other. The clustering method considered here attempts to group observations based on their definition of *dissimilarity* and not on their similarity. Hence, the degree of dissimilarity is crucial.

Basically, there are different methods of cluster analysis, which differ mainly in the following two aspects [BEPW18].

- a) The choice of proximity measure to quantify the similarity or dissimilarity between the observations, and
- b) the choice of classification method, i.e., *top-down* or *bottom-up* procedure.

Based on these two aspects, many clustering methods have been developed in the last years [HMMR15]. For our purpose, we focus on clustering methods that partition a dataset into $k \in \mathbb{N}$ different clusters by considering that the sum of squared errors within a cluster is minimal. The number of clusters k is predefined. Therefore, a common approach is *k-means clustering*, which is described in detail by Hastie and colleagues [HTF09], and Hennig and colleagues [HMMR15].

Furthermore, we are interested in *combinatorial algorithms*, i.e., algorithms that work directly with the observed data without direct reference to an underlying probability model [HTF09]. In general, a combinatorial algorithm works as follows: Let $i \in \{1, \dots, n\}$ be observations of a given dataset. In addition, let each observation i be uniquely assigned to a cluster $k < n$, which is characterized by an *encoder* C , i.e., $k = C(i)$. The goal of the combinatorial algorithm is to find a particular encoder $C^*(i)$ for which the dissimilarities between each pair of observations within each cluster k are minimized and the dissimilarities between each pair of observations of two different clusters are maximized. How an algorithm satisfies this objective depends, among other things, on the choice of a dissimilarity measure.

In fact, different dissimilarity measures can lead to different segmentation when using the same algorithm [HTF09]. In a first step, we therefore define the *dissimilarity measure* rather general and then describe the measure we use in more detail in subsection [3.1.2.1](#).

3 Definitions and Preliminaries

We define the dissimilarity measure following Hastie and colleagues [HTF09] as follows.

Definition 3.1.1. Let $i = 1, \dots, n$ be the observations of a dataset and $j = 1, \dots, p$ be the attributes of the dataset. With x_{ij} the measurements for observation i on attribute j are denoted. With

$$d_{ii'}^{(j)} := d_j(x_{ij}, x_{i'j})$$

the dissimilarity between values of the j 'th attribute for observations i and i' is denoted. Then, by

$$d_{ii'} = \sum_{j=1}^p d_{ii'}^{(j)}$$

the dissimilarity between observations i and i' is defined.

For example, a commonly used measure is the squared distance [HTF09]

$$d_{ii'}^{(j)} = (x_{ij} - x_{i'j})^2,$$

which is also used by the k -means algorithm. However, in the case of categorical data, squared distances are not appropriate [HTF09, MRS⁺22]. Categorical data is very likely included in an insurance dataset as, e.g., an insurance dataset may contain the variables *fraud suspicion (yes/no)* or *person reporting the damage*.

As mentioned earlier, the goal of cluster analysis is to divide a dataset into k appropriate clusters. In addition, to find suitable stimuli for the questionnaire, we look for k (real) representatives of the dataset that best explain the full range of attribute manifestations. Therefore, a suitable clustering method is the *k -medoids clustering*.

We clarify this decision by considering the main differences of k -means and k -medoid clustering. The description of the differences follows Hennig and colleagues [HMMR15].

- a) Both algorithms use cluster centers and group all other observations around this centers. In the case of k -means, the cluster centers are the mean values of the clusters, i.e., they are usually not real observations but calculated centers. k -medoids uses real observations of a dataset as cluster centers—the so-called *medoids*. We described that we prefer real claim processings to ensure the preservation of the dependency structure of the attributes and their levels in subsection 3.1.1. Moreover, by using real claims it is possible to present the stimuli in the interface used by the insurance company for evaluation.
- b) An advantage of the k -medoid over k -means algorithm is that the former is more robust to outliers [KR05].

- c) Unlike the k -means algorithm, k -medoid can be used for any dissimilarity measure derived from the dataset. This is especially important, as an insurance dataset very likely consists of variables of mixed type. Moreover, it can handle asymmetric binary attributes like they can occur in insurance datasets. To give an example, we consider fraud detection again. The case where this variable indicates fraud is rather rare and has a special meaning in its significance. We discuss this in more detail in subsection [4.1.1.1.2](#).
- d) A disadvantage of k -medoids is that it is more computationally intensive than k -means.

Following the idea of a combinatorial algorithm described above, we can describe a k -medoids clustering by the following algorithm based on Hastie and colleagues [\[HTF09\]](#), and Hennig and colleagues [\[HMMR15\]](#). Given the number of clusters k the algorithm proceeds as follows.

k -Medoids Algorithm.

Algorithm 3.1.2. a) Identification of a set of k medoids: We identify for a given cluster assignment C the observation that minimizes the total dissimilarity from other observations in each cluster, i.e.,

$$i_k^* = \arg \min_{\{i: C(i)=k\}} \sum_{C(i')=k} d_{ii'}$$

The current estimates of the medoids are then defined as $m_k = x_{i_k^*}$ for all k .

- b) *Assignment of the remaining $n - k$ observations:* For the given current set of medoids $\{x_{i_1}, \dots, x_{i_k}\}$ we assign each remaining observation to the closest cluster center such that the total sum of dissimilarities is minimized, i.e.,

$$C(i) = \arg \min_{1 \leq l \leq k} d(x_i, x_{i_l}) = \arg \min_{1 \leq l \leq k} d_{ii_l}$$

- c) Iterate steps 1 and 2 until the assignments do not change.

To find an optimal clustering, the choice of the number of clusters, $k \in \mathbb{N}$, also plays an important role. We determine the number k based on a validation criterion, which we describe in more detail in subsection [3.1.2.2](#). As the k medoids represent our stimuli for the questionnaire, we also have to consider the number of stimuli an expert can process in the evaluation without exceeding the cognitive load (see subsections [2.1.2](#) and [3.1.1](#))—in addition to the validation criterion.

Finding a solution is generally *hard* (see subsections [3.3.2](#) and [3.3.3](#)). More precisely, it belongs to the class of *combinatorial optimization problems* which cannot be

solved *efficiently* (see subsection 3.3.1) by complete enumeration [HTF09]. We do not discuss the computational complexity of the k -medoids algorithm in this thesis, but refer the interested reader to [HTF09, HMMR15, KR05]. There are several procedures described in the literature to solve the problem of minimizing the sum of overall dissimilarities. A good overview is given by Hennig and colleagues [HMMR15].

Kaufman and Rousseeuw invented an algorithm to solve the *hard* problem of finding k representative cluster centers or medoids to minimize the average dissimilarity. They called this algorithm *partitioning around medoids* or short *PAM algorithm* [KR05, HTF09]. It is a proven and robust algorithm. However, it should be noted that the PAM algorithm is an approximation algorithm that only finds a local minimum [HTF09].

To start the PAM algorithm for a dataset with attributes that are not (all) interval-scaled, we first have to define a dissimilarity matrix. As mentioned above, it is very likely that an insurance dataset consists of mixed data types, such as binary, nominal, and metric scale. For example, it may contain information about the amount of payments, the cost estimate, the person reporting the claim, or, as mentioned earlier, a reasonable suspicion of fraud. Therefore, we need a dissimilarity measure that can handle the mixed type attributes. How the dissimilarity matrix is calculated can be seen in subsection 3.1.2.1. With that a *mixed clustering* is applied, where mixed refers to the different scales contained in the dataset.

Finally, we can describe the PAM algorithm by the following two steps, which are similar to the steps of the k -medoids algorithm. In a first step, k representatives are selected and the remaining observations are assigned to the closest medoid according to the dissimilarity matrix used. In a second step, each preliminary medoid is swapped with an observation that is not a center in this step. The observation selected for the swap is the one that reduces the average dissimilarity the most. These two steps are repeated until the algorithm converges, i.e., no new medoid can be found in each cluster.

The peculiarity of the PAM algorithm is that it tries all the observations in the cluster as a new medoid with respect to the largest reduction of the value of the total criterion [HTF09],

$$\min_{C, \{i\}_1^k} \sum_{l=1}^k \sum_{C(i)=l} d_{ii_l}.$$

With that the PAM algorithm optimizes more holistically than the k -medoids algorithm. To find an optimal number of clusters k as input to the PAM algorithm, Rousseeuw introduced a validation criterion—a so-called *silhouette width*. The silhouette width is explained in more detail in subsection 3.1.2.2 while in the following subsection, we define the dissimilarity measure used—which can also handle mixed types of variables.

3.1.2.1 Gower's coefficient

As mentioned above, insurance datasets often contain attributes of mixed types, and thus we need a dissimilarity measure that can handle these different scales. Kaufman and Rousseeuw developed a program that computes pairwise dissimilarities between observations in the dataset. They called this program *DAISY* [MRS⁺22, KR05]. It can be considered a preprocessing step that prepares the dataset for the actual cluster analysis. The *DAISY* procedure is implemented in the package *cluster* in R [MRS⁺22].

DAISY can also handle data consisting of interval scales only, but the focus of this thesis is on mixed type data. To handle data of mixed type, such as nominal, ordinal or (a)symmetric binary scale, it uses the *general dissimilarity coefficient of Gower*, which was originally invented by Gower [Gow71]. In this subsection, a definition of the *Gower coefficient* is given, which is also used in the R as described in [MRS⁺22]. The following explanations essentially follow Maechler and colleagues [MRS⁺22], and Gower [Gow71].

For each attribute $j \in [p]$, we define a *dissimilarity score* $d_{ii'}^{(j)} \in [0, 1]$, where $d_{ii'}^{(j)} = 1$ means that observations i and i' are far apart along attribute j , and $d_{ii'}^{(j)} = 0$ if both observations are close. Here, the score depends on the type of the considered attribute.

a) *Metric type*: Let R_j be the *range* of the attribute j . Then,

$$d_{ii'}^{(j)} = \frac{|x_i^{(j)} - x_{i'}^{(j)}|}{R_j},$$

with $x_i^{(j)}$ is the value of observation i in attribute j .

b) *Ordinal type*: Assign a rank to the entries of the ordinal attribute j , $j \in [p]$, with $r^{(j)} = 1, \dots, R^{(j)}$. Then, the entry of attribute j of observation i is normalized by

$$z_i^{(j)} = \frac{r_i^{(j)} - 1}{R^{(j)} - 1}.$$

The ordinal attribute can then be treated like an interval-scaled attribute.

c) *Nominal type*: The dissimilarity score $d_{ii'}^{(j)}$ simply shows, if the attribute levels are identical or not, i.e.,

$$d_{ii'}^{(j)} = \mathbb{1}_{\{x_i^{(j)} \neq x_{i'}^{(j)}\}}.$$

d) *(A)symmetric binary type*: The difference between the symmetric and asymmetric binary attributes is that in the latter case the result is not equally important (e.g., for fraud detection). We define the dissimilarity score equally to the nominal type as described in [MRS⁺22].

3 Definitions and Preliminaries

Furthermore, we define a 0-1 weight $\delta_{ii'}^{(j)}$ for each attribute $j, j \in [p]$ by

$$\delta_{ii'}^{(j)} = \begin{cases} 0, & \text{if one or both observations have a missing value in attribute } j \text{ or,} \\ 0, & \text{if the attribute is asymmetric binary and both values are zero;} \\ 1, & \text{otherwise.} \end{cases}$$

In contrast to Gower's original formula, in [MRS⁺22] a weight $w_j \in \mathbb{R}^+$ is defined for each attribute j . In our case, we do not weight the attributes and set all weights $w_j = 1$ for all $j \in [p]$ as in the original form.

With that we can define finally the dissimilarity score or *Gower's distance*, respectively, between observation i and i' by

$$d_{ii'} := d(x_i, x_{i'}) = \frac{\sum_{j=1}^p w_j \delta_{ii'}^{(j)} d_{ii'}^{(j)}}{\sum_{j=1}^p w_j \delta_{ii'}^{(j)}}.$$

In [Gow71] it is shown that $d_{ii'} \in [0, 1]$ is truly a distance measure.

By computing Gower's distance $d_{ii'}$ for each pair of observations i and $i', i, i' \in [n]$, the resulting dissimilarity matrix is given by

$$D = \begin{pmatrix} 1 & d_{12} & \dots & d_{1n} \\ \vdots & \ddots & & \vdots \\ d_{n1} & \dots & d_{nn-1} & 1 \end{pmatrix}.$$

The dissimilarity score between $d_{ii'}$ and $d_{i'i}$ is of course the same.

3.1.2.2 Validation of the clusters

In order to have a criterion with which we can check the cohesion and separation quality of a partitioning technique (as given by the k -medoids algorithm), Rousseeuw introduced the *silhouette width*. With the silhouette width, we have a measure for each observation $i \in [n]$ of the dataset that shows whether an observation "lies well within its cluster" or whether it merely lies somewhere between the clusters [Rou87]. The definition of the silhouette width is based on [MRS⁺22, Rou87, KR05].

To compute the silhouette width, we suppose a given clustering in K clusters of the observations of a dataset, i.e., $C(i) = k$ with $k \in [K]$ and $i \in [n]$. We denote with C_k the set of observations with assignment to cluster k , i.e., all observations with $C(i) = k$. Furthermore, we denote by $|C_k|$ the number of observations i within C_k .

We also assume a dissimilarity measure d , which in our case is defined by Gower's distance (see subsection 3.1.2.1). It is also possible to compute the silhouette width

for a similarity measure, but this is not the subject of this paper and we refer the interested reader to [Rou87].

The silhouette width $s(i)$ is calculated per observation $i \in [n]$.

Definition 3.1.3. Let $a(i)$ with

$$a(i) := \frac{1}{|C_k| - 1} \sum_{\substack{i' \in C_k \\ i \neq i'}} d_{ii'}$$

be the *average dissimilarity* between observation i and all other observations of the cluster k to which observation i belongs. As it is not meaningful to consider the dissimilarity of an observation with itself, we set $i \neq i'$ and therefore, divide by $|C_k| - 1$.

We define with $d_{iC_{\bar{k}}}$ the *average dissimilarity* of observation i to all observations of $C_{\bar{k}}$, where $C_{\bar{k}}$ are the remaining clusters to which i does not belong to, i.e., $i \notin C_{\bar{k}}$. Specifically, $d_{iC_{\bar{k}}}$ is defined by

$$d_{iC_{\bar{k}}} := \frac{1}{|C_{\bar{k}}|} \sum_{i' \in C_{\bar{k}}} d_{ii'}$$

With

$$b(i) := \min_{C_{\bar{k}} \neq C_k} d_{iC_{\bar{k}}}$$

we define the *minimum dissimilarity* between i and all observations i' of i 's *nearest* cluster to which i doesn't belong.

Finally, we define by

$$s(i) := \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

the *silhouette width* for observation i .

The silhouette width is the standardized distance between the average dissimilarity of observation i to other observations within its cluster and the average dissimilarity of i to the observations of its *nearest* cluster. That is, $s(i)$ indicates how similar i is to other observations of the current cluster compared to observations in the nearest cluster—or in other words, how well the current cluster represents observation i .

The range of the silhouette width is $-1 \leq s(i) \leq 1$ with $s(i)$ almost 1 shows that the observation is very well clustered. Indeed, it means that the average dissimilarity within the cluster $a(i)$ is much smaller than the average dissimilarity to the nearest cluster $b(i)$. In case of $s(i)$ is about 0, $a(i)$ and $b(i)$ are nearly equal and hence, the

3 Definitions and Preliminaries

observation lies between two clusters. Observations with $s(i)$ near -1 are most likely misclassified. This means that $a(i)$ is much larger than $b(i)$ and therefore, the observation i is closer to the *nearest* cluster than to C_k . We set $s(i) = 0$, if i is the only observation of the considered cluster C_k .

To evaluate the clustering validity we look beside the silhouette plot to the *total average silhouette width* over all n observations as well as the *average silhouette width within each cluster*. For both, the closer the average silhouette width is to 1, the better the respective clustering C . Rousseeuw states that the average silhouette width “might be used to select an appropriate number of clusters” [Rou87]. We define the total average silhouette width by

$$\bar{s} = \frac{1}{n} \sum_{i=1}^n s(i)$$

and the clusterwise average of silhouette width by

$$\bar{s}_k = \frac{1}{|C_k|} \sum_{i \in C_k} s(i)$$

for all $k \in [K]$ and with n_k is the cardinality of the k 'th cluster.

3.2 Introduction to graph theory

In combinatorial optimization many computational problems are based on graphs [CLRS09], since graphs allow that problems can be “expressed with clarity and precision in the concise pictorial language” [DPV06, p. 91]. Briefly, a graph can be described as a construct of so-called *vertices* that are connected by so-called *edges*.

We also illustrate and explain the EBRS-problem and its complexity-theoretic investigation partly in terms of graphs in this thesis—and we describe other problems that we use for our studies in the context of graph theory.

In this section, we first introduce some basic terms of graph theory, which we will encounter in the further course of this thesis. Moreover, we define two special types of graph, which we need to show \mathbb{NP} -completeness of the EBRS problem. The introduction is based on Dietel [Die17] Gritzmann [Gri13], Kleinberg and Tardos [KT13], and West [Wes00]. For more detailed information about graph theory we also refer the interested reader to mentioned literature.

Before we explain and define graphs, we first define the set of k -elemental sets.

Definition 3.2.1. Let X be a set and 2^X the corresponding power set. Then, by

$$\binom{X}{k} := \{S \in 2^X : |S| = k\}$$

for $k \in \mathbb{N}_0$ the set of k -elemental sets of X is given.

This allows us to define a graph.

Definition 3.2.2. Let V and E be finite sets with $V \cap E = \emptyset$ and let

$$\nu : E \rightarrow \binom{V}{1} \cup \binom{V}{2} \cup V^2$$

be a map.

- a) Let $G := (V, E, \nu)$. Then G is called *common graph*. The elements of V are called *vertices or nodes* and the elements of E are called *edges* of G .
- b) Let $e \in E$. For $\nu(e) \in \binom{V}{1} \cup \binom{V}{2}$ the edge e is *undirected*, for $\nu(e) \in V^2$ the edge e is *directed*.
- c) Let $v, w \in V$, $e \in E$. Furthermore, let $\nu(e) \in \{\{v\}, \{(v, v)\}\}$ or $\nu(e) \in \{\{v, w\}, \{(v, w), (w, v)\}\}$, then v and e are *incident*. It is also denoted by $v \in e$.
- d) Let $v, w \in V$ be incident to an edge $e \in E$, i.e., $v, w \in e$. Then, the vertices are called *ends* of the edge.
- e) Let $e \in E$. If $\nu(e) \in \{\{v, w\}, \{(v, w), (w, v)\}\}$, then the two vertices $v, w \in V$ are called *adjacent*,
- f) Let $e, f \in E$. Two edges $e \neq f$ are *adjacent*, if they have an end in common.
- g) Let $e \in E$. If there exists a vertex $v \in V$ with $\nu(e) \in \{\{v\}, (v, v)\}$, then e is called *loop*.

We illustrate the definition of a graph by the following example.

Example 3.2.3. In this example, we consider a graph $G = (V, E, \nu)$ with

$$V = \{v_1, v_2, v_3, v_4, v_5\} \text{ and } E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}.$$

Furthermore, we define a map $\nu : E \rightarrow \binom{V}{1} \cup \binom{V}{2} \cup V^2$ for each edge as follows

$$\begin{aligned} \nu(e_1) &:= (v_1, v_2), & \nu(e_5) &:= (v_5, v_3), \\ \nu(e_2) &:= (v_1, v_3), & \nu(e_6) &:= \{v_5, v_4\}, \\ \nu(e_3) &:= (v_3, v_1), & \nu(e_7) &:= \{v_3, v_4\}, \\ \nu(e_4) &:= (v_2, v_5), & \nu(e_8) &:= \{v_4\}. \end{aligned}$$

3 Definitions and Preliminaries

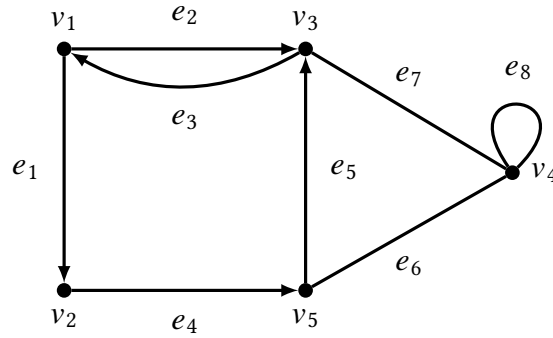


Figure 3.2: Geometric presentation of the graph given in example [3.2.3](#).

The resulting graph is illustrated in figure [3.2](#).

Based on this example we will explain the graph-theoretic terms defined above.

The above illustrated graph consists of directed as well as undirected edges and a loop. The edges e_1, e_2, e_3, e_4 and e_5 are directed edges. Whereas the edges e_6 and e_7 are undirected. The edge e_8 is an undirected loop. Clearly, a graph consisting just of directed edges is called *directed graph* and a graph consisting of undirected edges is called *undirected graph*.

To name an example for an incident vertex, let's consider the edge e_6 . The vertex v_5 is incident to this edge. Furthermore, v_4 and v_5 are the ends of edge e_6 . We denote this by $v_4, v_5 \in e_6$. And therefore, v_4 and v_5 are adjacent vertices. v_4 is incident to e_7 and with $v_4 \in e_6$ and $v_4 \in e_7$ we know that e_6 and e_7 are adjacent edges.

According to the definition and illustrated by the example, the map ν assigns each edge $e \in E$ to the corresponding vertices (uniquely). In this thesis, we omit the definition of the map in the remainder of this thesis and denote the assignment by (v) , $\{v\}$, (v, w) or $\{v, w\}$, respectively. And with that, we denote a graph by $G = (V, E)$ —instead of $G = (V, E, \nu)$.

In the next paragraph, we define a special type of graph—a so-called *path*. We denote a path by $P = (V, E)$. It consists of vertices and edges of the form

$$V = \{v_0, v_1, \dots, v_k\} \text{ and } E = \{\{v_0, v_1\}, \{v_1, v_2\}, \dots, \{v_{k-1}, v_k\}\}.$$

If we look more closely at the set of edges, we can see that a path connects the vertices v_0 and v_k over several edges. To better understand the concept of a path, we illustrate it in the following figure [3.3](#).

Note, that by the index $k \in \mathbb{N}$ we obtain an ordering of the vertices $v_0, \dots, v_k \in V$.

A path P with additional edge $\{v_k, v_0\}$ is called a *cycle* $C := P + \{v_k, v_0\}$, i.e., the “last” vertex of path P is connected with the startvertex v_0 . For example, if we consider the path P with $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}\}$

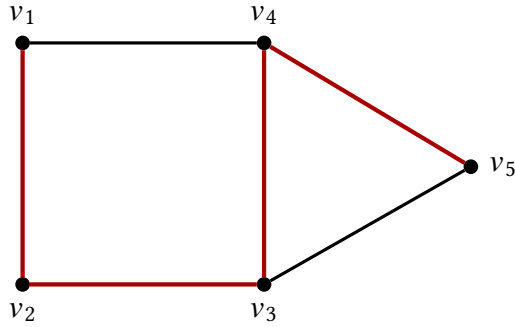


Figure 3.3: Geometric presentation of a path.

in figure 3.3 and “close” the path with edge $\{v_4, v_1\}$, then we get a cycle C . We illustrate this cycle in the following figure 3.4.

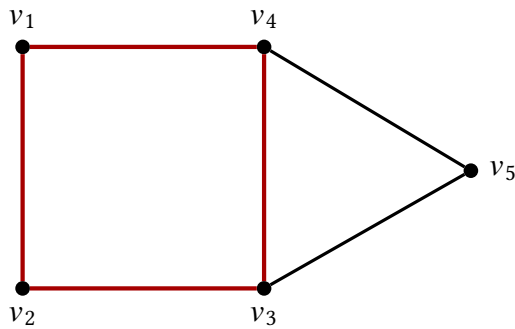


Figure 3.4: Geometric presentation of a cycle.

A graph is called *cyclic*, if it contains at least one cycle. In this thesis, we are especially interested in *acyclic* graphs, i.e., graphs in which no cycle is contained.

For the definition of a path and a cycle, we focused on undirected graphs. But of course, we could define a path and a cycle for directed graphs similarly. However, we leave this to the reader.

Finally, we define the so-called (*node-arc*) *incidence matrix* of a graph $G = (V, E)$ with the following definition.

Definition 3.2.4. Let $G = (V, E)$ be a graph with ordered vertices and edges, i.e., $V = \{v_1, \dots, v_n\}$ and $E = \{e_1, \dots, e_m\}$. We call the matrix $S_G := (\sigma_{i,j})_{\substack{i \in [n] \\ j \in [m]}}$ with

$$\sigma_{i,j} := \begin{cases} 1, & \text{if } v_i \in e_j = \{v_i, v_{i'}\}, \\ 2, & \text{if } v_i \in e_j = \{v_i\}, \\ -1, & \text{if } e_j \text{ is directed and } v_i \in e_j = (v_i, v_{i'}), \\ 1, & \text{if } e_j \text{ is directed and } v_i \in e_j = (v_{i'}, v_i), \\ 0, & \text{otherwise,} \end{cases}$$

incidence matrix of G .

To clarify the definition of the incidence matrix, we again look at the graph from example 3.2.3 and set up the corresponding incidence matrix. Let the set of vertices $V = \{v_1, \dots, v_5\}$ and the set of edges $E = \{e_1, \dots, e_8\}$ be given. The rows of the matrix corresponds to the vertices of the graph and the columns pertain to the edges. The incidence matrix is given by

$$\sigma_{i,j} = \begin{pmatrix} -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & -1 & 1 & 0 & 0 \end{pmatrix}.$$

Besides directed and undirected, as well as cyclic and acyclic graphs, there are many other different types of graphs depending on their structure. To mention some, there exist complete graphs, planar graphs, bipartite graphs, and Hamilton cycles [Die17, Wes00]. We will not discuss all such graphs in this thesis and refer the interested reader to the mentioned literature.

In the next subsections, we focus on special sets of a graph $G = (V, E)$, which we use in this thesis to prove \mathbb{NP} -completeness. In the first part, we describe the *independent set* of vertices and in the second part we define *matching* of edges. Briefly, both definitions can be summarized as a set of independent vertices and a set of independent edges of a graph G , respectively. The first is a subset U of V of non-adjacent vertices and the latter is the set M of edges that are not adjacent to each other. We introduce this in more detail and given examples in the following. These subsections are based on Gritzmann [Gri13], Korte and Vygeb [KV06], and Diestel [Die17].

3.2.1 Independent set

We start with the independent set, which we need to show that the EBRS problem is \mathbb{NP} -complete (see subsection 4.2.3). An independent set of a graph G is defined as follows.

Definition 3.2.5. Given an undirected graph $G = (V, E)$, an *independent set* is a subset of vertices $U \subseteq V$, such that no two vertices in U are adjacent. An independent set is *maximal* if no vertex can be added without violating independence, and *maximum* if it maximizes $|U|$.

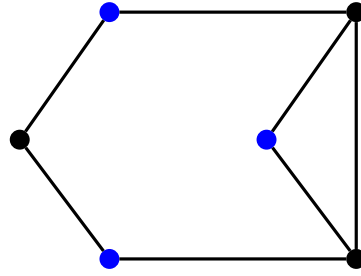


Figure 3.5: Example for an independent set.

We illustrate this in figure [3.5](#).

The maximum independent set of the illustrated graph is given by the blue vertices. The search for a maximum independent set can be formulated as an optimization problem. We introduce this problem and show that it is NP -complete in subsection [3.3.4.2](#).

3.2.2 Matchings

We now define what is meant by a matching as follows.

Definition 3.2.6. Given a graph $G = (V, E)$ and $M \subset E$. M is called *matching*, if

$$\begin{aligned} & \{v_1, w_1\} \in M \wedge \{v_2, w_2\} \in M \wedge \{v_1, w_1\} \cap \{v_2, w_2\} \neq \emptyset \\ \Rightarrow & \{v_1, w_1\} = \{v_2, w_2\}. \end{aligned} \quad (3.1)$$

By equation [\(3.1\)](#) we define a matching as a set of pairwise edges, which are non-adjacent to each other. i.e., if a vertex is the endpoint of two edges, not both edges are contained in the set M (or both edges are the same).

If a vertex is an endpoint of an edge which is contained in the matching M , i.e., $v \in e$ with $e \in M$, we say that the vertex is *covered* by the matching M . This is an important definition for the optimization problem we formulate below based on matching.

In subsection [3.3.4.3](#) we introduce the so-called *Perfect Matching* problem. It is a well-known NP -complete problem from graph theory, which we use to show that a special case of the EBRs problem is NP -complete (see subsection [4.3.2](#)). To describe this problem we define *perfect matchings* as follows.

Definition 3.2.7. Given a graph $G = (V, E)$ and $M \subset E$. M is called *perfect matching*, if all vertices of G are covered by the set M , i.e.,

$$v \in \bigcup_{e \in M} e \text{ for all } v \in V.$$

3 Definitions and Preliminaries

A perfect matching M is therefore a set of edges, for which the edges are non-adjacent and the vertices are covered by M . We illustrate this by the following figure [3.6](#).

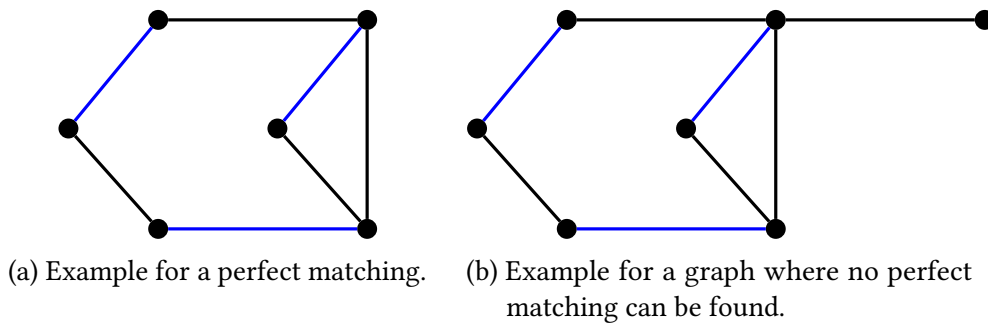


Figure 3.6: Examples for graphs with and without perfect matching.

Left, in subfigure [3.6a](#), we see a perfect matching, while to the right, in subfigure [3.6b](#), no perfect matching can be found.

As mentioned before, it can be shown that both graph-theoretic problems belong to the complexity class of NP -complete problems. Moreover, we use them to prove that the EBRs-problem and a special form of it are NP -complete. In the next section we give an introduction to complexity theory, with a particular focus on the complexity class of NP -complete problems.

3.3 Complexity theory

With complexity theory, we classify problems based how difficult they are to solve. The main part of this thesis focuses on the classification of the EBRs problem and its cases based on its complexity, where the cases differ by the constraints. We do not consider all existing complexity classes in this chapter, but focus on the complexity class NP and the class of NP -complete problems. In subsection [4.2.13](#), we prove that the EBRs problem is NP -complete. Furthermore, we only consider the complexity of problems according to their running time. More information about the complexity of problems depending on their memory requirements we refer the interested reader to Keller, Pferschy, and Pisinger [\[KPP04\]](#).

We start with the definition of running time in subsection [3.3.1](#) and further discuss what is meant by an efficient algorithm. In subsection [3.3.2](#) we describe the complexity class NP and, therefore, define all necessary terms in a first step. In subsection [3.3.3](#) we define the class of NP -complete problems. Furthermore, we discuss methods to prove NP -completeness, which we also apply in this thesis. At the end of this section, we present further NP -complete problems used in this thesis.

3.3.1 Running times of algorithms

The performance of an algorithm is of interest when comparing different algorithms to solve a given problem. The performance of an algorithm is defined by its running time and the memory required to solve the problem [KPP04]. In this thesis, we focus on the time complexity of an algorithm and are therefore interested in the running time of the algorithms.

The running time of an algorithm depends on different factors. Yet, complexity theory is also about comparing and classifying problems according to their difficulty. Therefore, we need a uniform definition of the runtime to be able to compare it for different algorithms and problems. In this chapter, we give such a definition of the running time, show how we can measure it, and what we understand by an efficient algorithm.

Of course, the “faster” an algorithm solves a problem, the better. There exist several methods to measure the performance of an algorithm. For example, we can directly measure the running time. But this is dependent of many parameters, the used computer, the operating system, etc. Dasgupta and colleagues describe the complexity of characterizing algorithms in such a way as follows: “Accounting for these architecture-specific minutiae is a nightmarishly complex task and yields a result that does not generalize from one computer to the next” [DPV06, p. 15].

In order to compare the complexity of problems in a more general way, it has proven useful not to directly consider the required elementary computer steps for the given scenario, but to use a more general consideration of the running time of problems. A viable and widely used simplification for comparing different running times is *worst-case analysis* [KPP04]. That means, instead of comparing directly the used elemental computer steps for each instance of a considered problem, we give an upper bound on the number of elemental computer steps required to solve any input of the problem [KPP04].

To define the upper bound of the number of elementary computer steps, we have to count the exact number of necessary operations of an algorithm and therefore need to consider the implementational details of the algorithm [KPP04]. Depending on the problem, this can be a challenging task. That is why we define an *asymptotic upper bound* for the running time instead. The asymptotic upper bound illustrates “the order of magnitude of the increase in running time” [KPP04, p. 12] for a given size of input. This can be illustrated as in figure 3.7.

In a first step, we give a detailed explanation of elementary computer steps [Gri13].

Remark 3.3.1. Elementary computer steps are defined as value assignments, *elementary comparison operations* and *elementary arithmetic operations*. \leq , \geq , $=$, and

3 Definitions and Preliminaries

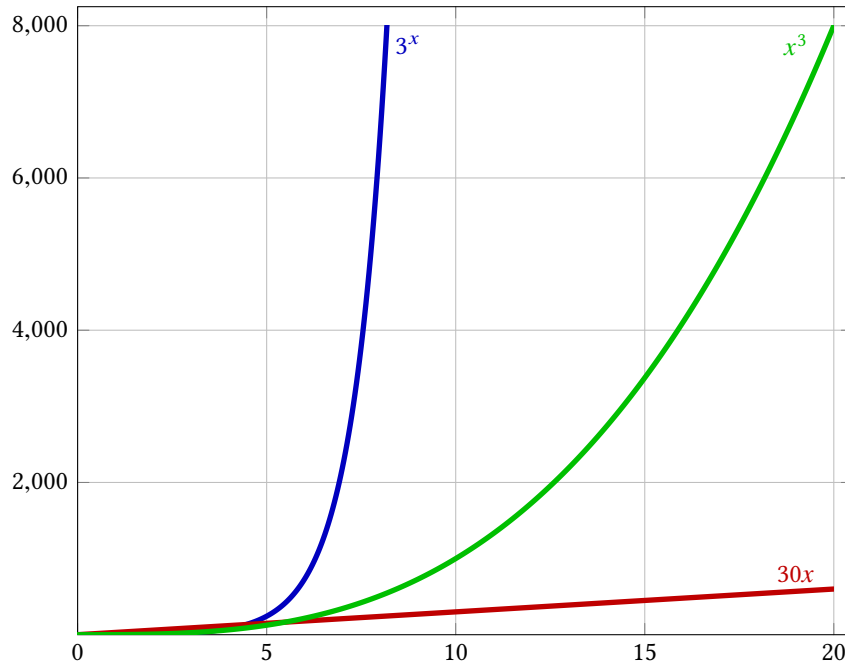


Figure 3.7: Running times of algorithms.

\neq are the elementary comparison operations and addition (+), subtraction (-), multiplication (\cdot) and division (\div) are the elementary arithmetic operations.

Furthermore, there exist *elementary program commands*. They are assignments like (\leftarrow), BEGIN, END, WHILE, DO, and HALT.

As defined by Keller and colleagues the asymptotic upper bound is dependent on the size of input, $\text{size}(I)$, with I is the accepted input of a problem [KPP04]. We denote the size of the input by n , i.e., $n := \text{size}(I)$. Formally, the size of the input is measured by the number of bits required to represent the input number, i.e., we consider the coding length. Since in most cases the coding length is proportional to the input, e.g., the length of an array, we informally use the input size and interpret n as the *input length* of the given instance of a problem. The size or input length n depends on the problem considered. For example, if we are searching an element in a list, then I is a list with n elements and the input length equals the number of elements in the list. Or, in case we are multiplying two matrices, the input length equals the dimensions of the matrices.

The above interpretation is appropriate since in this thesis we consider integer problems in particular. For more details on the coding length we refer the interested reader to Gritzmam [Gri13].

Finally, we can define the asymptotic upper bound of the running time. It is usually described by the so-called *O -notation* and defined as follows [KV06, DPV06].

Definition 3.3.2. Let A be an algorithm and l be an input with input length n that is accepted by the algorithm A . Let $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$. It is $f = O(g)$, if there is a constant $c > 0$ such that $f(n) \leq c \cdot g(n)$.

We pronounce $f = O(g)$ by “ f grows no faster than g ”, with f the running time of the algorithm at input length n and g the asymptotic upper bound. Since this is a worst-case analysis, A terminates after at most $c \cdot g(n)$ elementary computer steps for each input l of the algorithm, and we say “ A runs in $O(g)$ time”. Another way to pronounce $f(n) \leq c \cdot g(n)$ is that the “*running time* of A is $O(g)$ ”. This pronunciation is based on Korte and Vygen [KV06].

For a better understanding of the worst-case running time of an algorithm A , we consider the following example adapted from Keller and colleagues [KPP04].

Example 3.3.3. Given an input length n and a function $g(n) := n^2 + n + 5$. Then, the following functions are included in $O(g(n))$:

$$f(n) = \begin{cases} 10n, \\ n \log n, \\ 10\,000n^2 \end{cases}$$

Instead of considering the whole function $g(n)$ for the asymptotic upper bound, it is a permissible simplification to ignore the lower-order terms n and 5 of the function g . We then say that the above defined functions $f(n)$ are included in $O(n^2)$. The permissible simplification is acceptable, since these terms get insignificant by growing input length n [DPV06].

Such permissible simplifications are generally used for the asymptotic upper bounds. There exist further simplification rules, which are summarized by Dasgupta and colleagues [DPV06].

In the last part of this subsection, we are interested in the definition and explanation of the term *efficiency*. In the main part of this thesis we ask whether there exists an efficient algorithm for the EBRs problem and its cases that solves the problem in reasonable time. In this thesis, we speak of an *efficient algorithm* if it runs in polynomial time, i.e., its running time is asymptotically no slower than $O(n^k)$ with $k \geq 2$. We define a polynomial time algorithm similar to Korte and Vygen [KV06].

Definition 3.3.4. Let n be the input length of an algorithm A . An algorithm A runs in *polynomial time*, if there is an integer k such that it has a running time of $O(n^k)$.

In the definition of an efficient algorithm, we have focused on a specific given algorithm. In the next chapter, we do not consider the complexity of a single algorithm, but the complexity of decision problems. That is, we consider the

algorithmic difficulty of a problem. Or, in other words, we consider the question of whether there is an algorithm for this problem that can solve it in efficient time.

We already motivated in section 2.2 that the complexity class \mathbb{P} of decision problems exist. This complexity class consists of problems, for which a polynomial time algorithm is known. But of course, there exists other problems which are *intractible*, i.e., they are so “hard that no polynomial time algorithm can possibly solve it” [GJ79, p. 8].

In the next two subsections we discuss such decision problems, for which no known polynomial time algorithm is known, but for which we can verify in polynomial time, whether a given *certificate* solves the problem. We summarize these decision problems in the class \mathbb{NP} (see subsection 3.3.2). Furthermore, in subsection 3.3.3 we discuss a class of problems which can be polynomially transformed into a known problem of class \mathbb{NP} , and thus, the problem is as hard as the hardest problem of class \mathbb{NP} . If the considered problem also belongs to the class \mathbb{NP} itself, we call this problems \mathbb{NP} -complete.

3.3.2 The complexity class \mathbb{NP}

Complexity theory studies computational problems according to their resource usage, which could be time or storage. In this thesis we focus on the time complexity of problems. i.e., we are interested in the question whether there exists a polynomial time algorithm to solve this problem. For more details on the definition of an algorithm and its running time we refer to the previous subsection 3.3.1. Of course, the acceptability of the computation time depends on the particular application [Gri13]. Yet, in general, we are looking for an *efficient* algorithm to solve the problems in operations research applications.

We talk about an efficient algorithm, if it can solve a problem in polynomial time. That means, its solution algorithm has a worst-case running time of $\mathcal{O}(n^k)$ on input length n and for some constant $k \in \mathbb{N}$ [CLRS09]. But of course not all problems can be solved efficiently. For example, in subsection 4.2.3 we show that the EBRS problem cannot be solved in polynomial time—but we can show that $\text{EBRS} \in \mathbb{NP}$.

That leads us to the question of what is special about the class \mathbb{NP} .

We know that we cannot solve the problems of the class \mathbb{NP} in polynomial time in general. However, we can *verify* in polynomial time in the size of the input of the problem if a given *certificate* is truly the solution of the considered problem.

In the first part of this subsection we introduce and define the class \mathbb{NP} informally as well as formally and explain all related terms, such as, language, certificate, string, and input. In the latter part of this subsection we show how to prove that a problem belongs to the class \mathbb{NP} exemplarily. This is based mainly on the work by [Gri13], and Garey and Johnson [GJ79].

Let us start with an explanation of the term “problem”, as given by Gritzmann [Gri13, p. 173]. Formally a problem Π can be seen as a relation, which assigns to an input all accepted outputs. For a given input I we are searching for an output O , so that the pair (I, O) is in relation, or (I, O) belongs to Π .

The following definition explains the basic terms needed for complexity theory and gives a mathematical definition of a problem.

Definition 3.3.5. a) Let Σ be a finite and nonempty set. Then, Σ is called *alphabet*.

b) A *string* describes each finite sequence $I := \{\sigma_1, \dots, \sigma_r\}$ with $r \in \mathbb{N}_0$ consisting of elements of Σ . r is the *length* or *size* of the string I and is denoted with $|I|$.

c) Let Σ^* be the set of all finite strings over Σ . Each subset $L \subset \Sigma^*$ is called *language* over the alphabet Σ .

d) Let Σ_1 and Σ_2 be alphabets. Each subset Π of $\Sigma_1^* \times \Sigma_2^*$ defines a *problem* on $\Sigma_1^* \times \Sigma_2^*$. Each string I of Σ_1^* is called (*theoretical*) *input* of Π .

e) Let Π be a problem of $\Sigma_1^* \times \Sigma_2^*$. Then

$$L := L(\Pi) := \{I : \exists(O \in \Sigma_2^*) : (I, O) \in \Pi\}$$

is called *input language* of Π . The elements of L are also called *instances* of Π . We also speak of Π *accepts* the input $I \in L$. For $I \in L$ each string $O \in \Sigma_2^*$ denotes with $(I, O) \in \Pi$ a *solution* or output of problem Π .

Informally, we can define an instance of a problem as all inputs needed to compute an output to the problem. With that, we want to guarantee that the input is not inadmissible.

For the complexity-theoretical investigations, the so-called *decision problems* are usually considered. Unlike optimization problems, which asks for a deterministic solution to the problem, decision problems answers only “yes” and “no” questions. By considering decision problems we guarantee a certain standardization of the problems that allows us to compare the complexity of the problems [Gri13]. The standardization is given by, first, considering only problems with a trivial output, i.e., $O \in \Sigma_2^* = \{0, 1\}$, and second, the input should be appropriate and effective—a so-called *instance*. The latter requirement and its importance gets clearer, if we get aware of the difference between input and instance.

A problem is defined as an arbitrary subset of $\Sigma_1^* \times \Sigma_2^*$, for which is not yet guaranteed that the given input string $I \in \Sigma_1^*$ is also an input language $I \in L$. So the input I is unlike an instance a theoretical input, which probably contains not all necessary informations to assign an output.

By the following definition we define a decision problem more formally.

3 Definitions and Preliminaries

Definition 3.3.6 (Decision Problem). Let Σ be an alphabet, $\Pi \subset \Sigma^* \times \{0, 1\}$, $L := L(\Pi)$ and for all $l \in L$ the following condition holds

$$(l, 0) \in \Pi \Leftrightarrow (l, 1) \notin \Pi.$$

If there exists a polynomial time algorithm, which decides for each $l \in \Sigma^*$ whether $l \in L$, Π is called *decision problem* over Σ . Each input l with $(l, 1) \in \Pi$ is called *yes-instance* and all inputs with $(l, 0) \in \Pi$ are called *no-instance* of Π .

In the following, we give an example of a well-known decision problem, the CLIQUE problem, which is a problem known from graph theory.

Example 3.3.7 (CLIQUE problem). Let $G = (V, E)$ be an undirected graph with a set of vertices V and a set of edges E . With a clique we are searching for a subset of vertices $V' \subseteq V$ in G , in which every pair of vertices is connected by an edge. The size of the clique is the number of vertices contained in the subset. The decision problem is defined as follows.

Problem 3.3.8 (CLIQUE). Given a pair (G, k) with G is an undirected graph and $k \in \mathbb{N}$. Does there exist a clique of size at least k in the graph?

The instance of the problem is any pair (G, k) . The problem assigns “yes” or 1 to the instance, if G contains a clique of size k . Otherwise, it assigns “no” or 0. For more information about clique, we refer to Garey and Johnson [GJ79], Dasgupta and colleagues [DPV06], and Cormen and colleagues [CLRS09].

With that we have defined the basic terms to explain complexity.

To define the class \mathbb{NP} we are focusing on *nondeterministic algorithms* for decision problems. The term nondeterministic is related to the so-called *Turing machine*, more precisely, a *nondeterministic Turing machine*. We do not want to give a detailed introduction to Turing machines, but we want to motivate the idea and give an informal definition of a Turing machine based on the work by Garey and Johnson [GJ79].

We begin with the idea of a *deterministic Turing machine*. With it, we have a theoretical and mathematical model for computation with which we can “formalize the notion of an algorithm” [GJ79, p. 23]. With that, we can capture the concept of an efficient computable algorithm, not depending on technical characteristics, limits, and new achievements of real computers [Gri13]. With that, we can measure the computability of problems in a more general sense.

In short, a (deterministic one-tape) Turing machine can be seen as a strip of tape split in infinite sections and a finite state control. Furthermore, it has a read-write head with which it can read or write symbols from the alphabet Γ of or on the tape. This is done according to a table of rules, which can be found, e.g., by Gritzmam

[Gri13], or Korte and Vygen [KV06]. Beside some input symbols of a subset $\Sigma \subset \Gamma$ there exists some blank symbols $b \in \Gamma \setminus \Sigma$. Furthermore, there exists a finite set Q of states, including a start-state q_0 and two halt-states q_Y and q_N . Finally, there exists a transition function

$$\delta : (Q \setminus \{q_Y, q_N\}) \times \Gamma \rightarrow Q \times \Gamma \times D$$

with $D = \{-1, 1\}$. D is the set of direction of movement of the head. With the transition function the step by step process of the machine is defined.

Now that we have informally defined a Turing machine, we describe informally a nondeterministic Turing machine, which we need to define the class \mathbb{NP} . It also fulfills the idea of a theoretical and mathematical model for computation.

The difference to a deterministic Turing machine is mainly due to the fact that there are “two distinct stages” [GJ79, p. 30] during computation—a so-called *guessing stage* and a *checking stage*. Summarized, in the first stage the algorithm guesses some certificate C for a given instance I . Afterwards the algorithm provides I and C to the second stage, in which the answer “yes” or “no” is delivered. The algorithm proceeds deterministically in the checking stage, but for the first stage it provides the certificate in a nondeterministic manner. In more detail we can describe the nondeterministic Turing machine as follows.

It consists of two heads, with which it scans, reads, and writes on the tape. In the guessing stage the guessing module goes step by step from section to section of the tape and either write some symbol from Γ on the tape section being scanned, and move one section to the left, or stops. Thereby, the guessing module writes on the tape any string of Γ^* in a totally arbitrary manner—or in a completely nondeterministic way.

After the guessing module finishes at state q_0 , the start state of the finite state control—the checking stage—starts. Thereby, it follows the rules of a deterministic Turing machine. The finite state control terminates, if it reaches one of the two halt states, q_Y or q_N . In case of q_Y the computation is accepted. If the computation is not accepted or does not halt, the finite state control stops in q_N .

With this description of the functionality of a nondeterministic Turing machine we can define the class \mathbb{NP} informally as follows.

Definition 3.3.9. The class \mathbb{NP} is the class of problems, which can be solved by a *nondeterministic Turing machine*.

We call problems of the class \mathbb{NP} *nondeterministic polynomial problems*. We refer the interested reader for more details to Garey and Johnson [GJ79], Gritzmann [Gri13], or Korte and Vygen [KV06].

3 Definitions and Preliminaries

In the remainder of this chapter we give a more formal definition of this class of problems, so that we can prove that a problem is in \mathbb{NP} . We use this in chapter 4. For this definition we, again, follow Gritzmann [Gri13].

Before we can define the class \mathbb{NP} we first need to introduce the so-called *concatenation*, used to join strings together.

Definition 3.3.10. Let Σ and Γ be alphabets and $I := (\sigma_1, \dots, \sigma_p)$ as well as $T := (\tau_1, \dots, \tau_q)$ finite strings over Σ^* and Γ^* , respectively. The operation \odot with

$$I \odot T := (\sigma_1, \dots, \sigma_p, \tau_1, \dots, \tau_q)$$

is called *concatenation*.

Definition 3.3.11 (Complexity class \mathbb{NP}). Let Σ be an alphabet. The class \mathbb{NP} consists of all decision problems Π , for which a decision problem $\Pi' \in \mathbb{P}$ and a polynomial π exists with the following properties.

- a) Π' accepts all inputs $I \odot C$, for which I is an instance of Π and $C \in \Sigma^*$ with $|C| \leq \pi(|I|)$.
- b) There exists a $C \in \Sigma^*$ with $|C| \leq \pi(|I|)$ and $(I \odot C, 1) \in \Pi'$ if and only if $(I, 1) \in \Pi$.

Each string $C \in \Sigma^*$ with $|C| \leq \pi(|I|)$ is called *potential certificate*. Each potential certificate C with $(I \odot C, 1) \in \Pi'$ is called *certificate*. Π' is called the *checking or verification problem* belonging to Π .

In the following we explain the meaning of the definition.

We already mentioned that the class \mathbb{NP} consist of all problems, which can be verified in polynomial time. To verify a problem we need a solution. For a given “solution”—the certificat C —we do not ask, if there exists a clique of size k —as it would be the case for problem Π . Instead we ask in the checking problem Π' , if the instance I concatenated with the certificat C is a clique of size k . See therefore also the explanation of the nondeterministic Turing machine.

But as \mathbb{NP} is a class of nondeterministic polynomial algorithms also the certificat C should be of polynomial input size. This is guaranteed by $|C| \leq \pi(|I|)$, i.e., the size of C is at most a polynomial of input size $|I|$.

Furthermore, the checking algorithm should run in polynomial time as already mentioned above, when describing the checking stage.

We exemplify the concept of the certificate the checking problem by the following example.

Example 3.3.12. Let us consider again the CLIQUE problem. Let I be an instance consisting of an arbitrary undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}$. V is the set of vertices and E is the set of edges of the graph G . We already formulate the decision problem in problem [3.3.8](#).

The certificate C is a subset $V' \subseteq V$ of size at least k , i.e., $|V'| \geq k$. The checking algorithm verifies for the input $(I \oplus C)$, if the subset of vertices V' is truly a clique of size k .

With that we can show that $\text{CLIQUE} \in \text{NP}$ [\[CLRS09\]](#).

Let us therefore assume that there exists a yes instance $(I, 1) \in \text{CLIQUE}$. Furthermore, a certificate C is given with $(I \oplus C, 1) \in \text{CLIQUE}'$. I and C are defined as above. We only need to show that there exists a polynomial π such that $|C| \leq \pi(|I|)$ and that the checking algorithm runs in polynomial time.

As the certificate C is a subset of vertices of size at least k we can find a polynomial π , such that $|C| \leq \pi(|I|)$. The checking algorithm verifies for each pair $u, v \in V'$, if the corresponding edge (u, v) belongs to E . This can be done in polynomial time.

In the next chapter, we define and discuss the class of NP-complete problems. It is a class of problems, which can be polynomially transformed into each other. Exactly this property is of interest for the consideration of the millennium problem $\mathbb{P} \stackrel{?}{=} \text{NP}$.

3.3.3 NP-completeness

As we already described in the motivation of the complexity theory in section [2.2](#) one of the famous millennium problems is whether

$$\mathbb{P} \stackrel{?}{=} \text{NP}.$$

It is widely believed that $\mathbb{P} \neq \text{NP}$ [\[CLRS09, Gri13, GJ79\]](#), but so far it has not been proven. However, it can be shown that $\mathbb{P} \subseteq \text{NP}$. This can be justified by the fact that if we can solve a problem with a deterministic algorithm in polynomial time, we can also solve it with a nondeterministic algorithm in polynomial time. This is because—since there is a deterministic polynomial-time algorithm to solve the problem—we can ignore the guessing stage of the nondeterministic algorithm and use this algorithm as the checking stage algorithm instead. For more details see Garey and Johnson [\[GJ79\]](#), Kleinberg and Tardos [\[KT13\]](#), or Cormen and colleagues [\[CLRS09\]](#).

In terms of solvability in polynomial time, the class of NP-complete decision problems is very interesting, as for this class of problems holds: if one of its problems is solvable in polynomial time, then all problems contained in NP are solvable in polynomial time [\[CLRS09\]](#).

This leads us to the definition of the class of NP -complete problems, which can be informally described as a set of problems contained in NP , and are as *hard* as any problem in NP [CLRS09]. Here, the term “hard” does not refer directly to the difficulty of solving the problem, but rather can be understood as the ability to compare problems of class NP according to their complexity [Gri13]. We focus in a first step on the term *hard*.

In order to compare problems of the class NP —as just mentioned—we must introduce the terms *reducibility* and *transformation* of problems, respectively. Both terms are used in literature, and their applied definition is not always obvious. Therefore, we define both terms as used in this thesis to avoid misunderstandings.

In short and very simply speaking, we reduce a problem to another by transforming its instances so that the other problem accepts them as instances as well. Thus, reducibility aims at transforming a language $L_1 \subseteq \Sigma_1^*$ to a language $L_2 \subseteq \Sigma_2^*$. And in case we find a polynomial-time subroutine for the corresponding decision problem of L_1 , we also have a polynomial-time subroutine for L_2 .

We will leave this statement as it is for now and discuss it in more detail later. Instead, we first give a short comparison of terms *Cook’s reduction* and *Karp’s reduction*. Both reduction methods are important methods to compare problems of the class NP . The description of both methods is based on the work by Garey and Johnson [GJ79].

In this thesis, we apply Karp’s reduction and give only a brief introduction of Cook’s reduction, following the works of Garey and Johnson [GJ79], and Gritzmann [Gri13]. Briefly, Cook’s reduction can be summarized as follows. Cook mentions the term reducibility for the first time in the paper [Coo71]. There he introduced a method for comparing problems, and called it \mathbb{P} -*reducibility*. In this case, reducibility of languages is achieved by applying a polynomial-time *Turing reduction*. That is, there exists an algorithm that reduces a problem Π_1 to a problem Π_2 performing polynomially many elementary operations on numbers of polynomial size and polynomially many calls of a so-called *oracle* [Gri13]. Cook’s reduction can also be viewed as *Turing reductions* running in polynomial time.

Karp invented a less cumbersome polynomial transformability and showed that Cook’s theorem (Theorem 3.3.26) remains true by applying Karp’s reduction instead of Cook’s reduction [Kar72]. Further, the Karp’s reduction shapes the current definition of NP -completeness [GJ79]. For a more detailed description of the differences of Karp and Cook reduction we refer to Garey and Johnson [GJ79], and Ruiz-Vanoye and colleagues [RVPOR⁺11].

The Karp’s reduction can finally be defined as follows [Gri13].

Definition 3.3.13 (Karp’s reduction). Let Σ be an alphabet, Π_1 and Π_2 be decision problems, and $\chi : L(\Pi_1) \rightarrow L(\Pi_2)$ be computable by a polynomial-time algorithm.

If it holds that

$$(l, 1) \in \Pi_1 \Leftrightarrow (\chi(l), 1) \in \Pi_2, \quad (3.2)$$

then χ is a *polynomial transformation* or *Karp's reduction* from Π_1 to Π_2 . We write $\Pi_1 \leq_P \Pi_2$.

Thus, we have defined Karp's reduction at the level of decision problems. However, it is also possible to define it at the level of input languages, as defined below.

Definition 3.3.14. Let $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ be languages and let $\chi : \Sigma_1^* \rightarrow \Sigma_2^*$. If χ is a function computable in polynomial time, such that for all l with $l \in L_1$ if and only if $\chi(l) \in L_2$, then χ is a polynomial transformation. We write $L_1 \leq_P L_2$.

Remark 3.3.15. We use definition 3.3.13 of Karp's reduction to show that the EBRs problem is NP-complete in subsection 4.2.3. But to show the properties of polynomial transformation we consider it sufficient to use the definition based on input languages.

The sufficiency of the definition 3.3.14 is justified by the fact that if there exists a polynomial-time computable function $\chi : \Sigma_1^* \rightarrow \Sigma_2^*$ with which the input language $L_1 = L_1(\Pi_1)$ is reducible to another input language $L_2 = L_2(\Pi_2)$ in polynomial time, then there also exists a polynomial-time computable function $\chi : L_1(\Pi_1) \rightarrow L_2(\Pi_2)$ with which a problem Π_1 is reducible to another problem Π_2 in polynomial time. We just do not differ between a yes or no instance.

The definition of Karp's reduction is illustrated in figure 3.8.

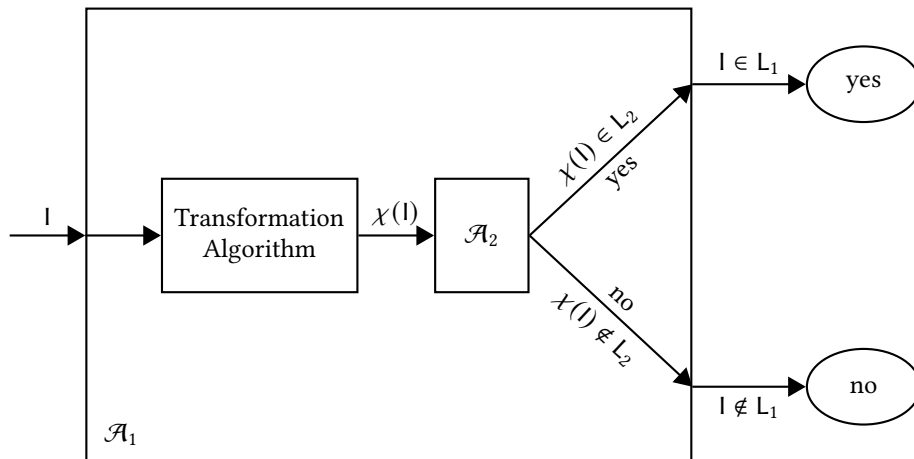


Figure 3.8: Example for a Karp's reduction—adapted from Cormen and colleagues [CLRS09, p. 1069].

The following lemma illustrates a property of polynomial transformations.

3 Definitions and Preliminaries

Lemma 3.3.16. *If $\Pi_1 \leq_P \Pi_2$, then $\Pi_2 \in \mathbb{P}$ implies $\Pi_1 \in \mathbb{P}$.*

We follow Garey and Johnson [GJ79], as well as Cormen and colleagues [CLRS09] to prove lemma 3.3.16

Proof. Let Σ_1 and Σ_2 be alphabets of L_1 and L_2 , respectively. Let χ be a polynomial transformation from L_1 to L_2 and by \mathcal{A}_χ we denote the polynomial time algorithm that computes χ . Furthermore, with \mathcal{A}_2 we define a polynomial algorithm that decides if an instance is in L_2 . To prove the lemma we show, that there exists a polynomial time algorithm \mathcal{A}_1 that decides affiliation to L_1 by composing the algorithms \mathcal{A}_χ and \mathcal{A}_2 .

For this, we first transform an input $l \in \Sigma_1^*$ to an input $\chi(l) \in \Sigma_2^*$ by applying the algorithm \mathcal{A}_χ . Afterwards we apply the algorithm \mathcal{A}_2 to determine if $\chi(l) \in L_2$. The transformation procedure is illustrated in figure 3.8. There it also can be seen that algorithm \mathcal{A}_1 uses the output of \mathcal{A}_2 and \mathcal{A}_χ to generate its own output. And this holds because of the assumption $L_1 \leq_P L_2$ and therefore, $l \in L_1 \Leftrightarrow \chi(l) \in L_2$.

As both applied algorithms \mathcal{A}_χ and \mathcal{A}_2 run in polynomial time, also \mathcal{A}_1 runs in polynomial time. This can be easily seen by defining with p_χ and p_2 polynomial functions bounding the running time of \mathcal{A}_χ and \mathcal{A}_2 , respectively. I.e., it is $|\chi(l)| \leq p_\chi(|l|)$ and $|\chi_2(l)| \leq p_2(|l|)$.

With that and based on the description of the algorithm above, its running time is $\mathcal{O}(p_\chi(|l|) + p_2(p_\chi(|l|)))$ and therefore, the algorithm is bounded by a polynomial of the input size $|l|$. \square

Remark 3.3.17. Equivalently it can be shown, that $\Pi_2 \notin \mathbb{P}$ implies $\Pi_1 \notin \mathbb{P}$, if $\Pi_1 \leq_P \Pi_2$ [GJ79].

With the definition of polynomial transformation, we can define the class of NP-hard problems—the class of problems, which are as “hard” as all problems in NP according to their complexity.

Definition 3.3.18 (NP-hard). A decision problem Π_1 is NP-hard if $\Pi_2 \leq_P \Pi_1$ for all $\Pi_2 \in \text{NP}$.

Notice, NP-hard problems do not need to be contained in NP itself.

As we mentioned at the beginning of this subsection, the difference with NP-complete problems is that NP-complete problems are as “hard” as any problem of NP, in addition to being in NP. This finally leads us to the definition of NP-completeness.

Definition 3.3.19 (NP-complete). A decision problem Π is called NP-complete if

- a) $\Pi \in \text{NP}$, and
- b) Π is NP -hard.

It would be a great effort to prove NP -completeness for a decision problem Π by showing $\Pi' \leq_P \Pi$, for all $\Pi' \in \text{NP}$. By the following lemma we show that the polynomial transformation is transitive. Thus, it is sufficient to prove the NP -completeness of a problem by reducing a single known NP -complete problem to it. The following lemma is based on Garey and Johnson [GJ79], and Kleinberg and Tardos [KT13].

Lemma 3.3.20. *If $\Pi_1 \leq_P \Pi_2$ and $\Pi_2 \leq_P \Pi_3$, then $\Pi_1 \leq_P \Pi_3$.*

Proof. Let Σ_1, Σ_2 , and Σ_3 be alphabets and $\Pi_1 \subset \Sigma_1 \times \{0, 1\}$, $\Pi_2 \subset \Sigma_2 \times \{0, 1\}$ and $\Pi_3 \subset \Sigma_3 \times \{0, 1\}$ corresponding decision problems. Furthermore, let $L_1 = L_1(\Pi_1)$, $L_2 = L_2(\Pi_2)$ and $L_3 = L_3(\Pi_3)$ be input languages. To prove that $\Pi_1 \leq_P \Pi_2$ it is sufficient to show $L_1 \leq_P L_2$ (see remark 3.3.15).

Let $\chi_1 : \Sigma_1^* \rightarrow \Sigma_2^*$ and $\chi_2 : \Sigma_2^* \rightarrow \Sigma_3^*$ be polynomial transformations. χ_1 transforms L_1 to L_2 and χ_2 transforms L_2 to L_3 . We define by $\chi_3 : \Sigma_1^* \rightarrow \Sigma_3^*$ with $\chi_3(l) = \chi_2(\chi_1(l))$ for all $l \in \Sigma_1^*$ a transformation from L_1 to L_3 . To prove $L_1 \leq_P L_3$, we need to show the following.

- a) χ_3 truly transforms an instance $l \in L_1$ to an instance $\chi_3(l) \in L_3$.
- b) χ_3 is computable by a polynomial time algorithm.

To prove a) we show with the definition of χ_3 that

$$\chi_3(l) = \chi_2(\chi_1(l)) \in L_3 \Leftrightarrow l \in L_1. \quad (3.3)$$

Therefore, we assume $\Pi_1 \leq_P \Pi_2$ and $\Pi_2 \leq_P \Pi_3$. With that, we know that all instances l with $l \in L_1$ are polynomially transformed to $\chi_1(l) \in L_2$ and all input languages $\chi_1(l) \in L_2$ are polynomially transformed to $\chi_2(\chi_1(l)) \in L_3$. And therefore, equation (3.3) follows.

With that, there also exists polynomial time algorithms \mathcal{A}_2 and \mathcal{A}_3 that recognize if $\chi_1(l) \in L_2$ and $\chi_2(\chi_1(l)) \in L_3$, respectively. Moreover, \mathcal{A}_{χ_1} and \mathcal{A}_{χ_2} are polynomial time algorithms that computes χ_2 and χ_1 .

Due to the definition of χ_3 the algorithm for χ_3 is a composition of \mathcal{A}_2 , \mathcal{A}_{χ_1} , \mathcal{A}_3 , and \mathcal{A}_{χ_2} . A composition of a polynomial time algorithm is still a polynomial time algorithm. Let p_{χ_1} , p_{χ_2} , and p_3 be polynomial functions bounding the running times of \mathcal{A}_{χ_1} , \mathcal{A}_{χ_2} , and \mathcal{A}_3 based on the input size $|l|$. Then, $|\chi_3(l)| \leq p_{\chi_2}(p_{\chi_1}(|l|))$. And the running time of the algorithm composed like \mathcal{A}_3 is

$$\mathcal{O}(p_{\chi_1}(|l|) + p_{\chi_2}(|l|) + p_3(p_{\chi_2}(p_{\chi_1}(|l|))))$$

and with that an algorithm that runs in polynomial time. \square

3 Definitions and Preliminaries

The transitivity of polynomial transformation leads us to the following lemma, which “simplifies” the definition of NP-completeness.

Lemma 3.3.21. Π_1 is NP-complete if

- a) $\Pi_1 \in \text{NP}$ and
- b) $\Pi_2 \leq_P \Pi_1$ for a (known) NP-complete language Π_2 .

Proof. With Π_2 is a NP-complete problem, we know that all $\Pi_3 \in \text{NP}$ are polynomial transformable to Π_2 , i.e., $\Pi_3 \leq_P \Pi_2$ for all $\Pi_3 \in \text{NP}$. With the transitivity of polynomial transformation (see lemma 3.3.20) any Π_3 is reducible to Π_1 , which shows the NP-hardness of Π_1 . \square

In a next step, we will briefly illustrate why the class of NP-complete problems is relevant. To this end, let us mention, as we did at the beginning of this subsection, that the class of NP-complete problems is of interest in the discussion of $\mathbb{P} = \text{NP}$. This is because this class of problems holds the following property.

The class of NP-complete problems consists of a set of decision problems (a subset of the class of NP) that no one knows how to solve efficiently, but if there were a polynomial time solution for even a single NP-complete problem, then every problem in NP would be solvable in polynomial time. But also, if any problem in NP is demonstrably intractible, then so are all NP-complete problems. This is due to lemma 3.3.16 and the transitivity of polynomial transformations (see lemma 3.3.20). We illustrate this in figure 3.9, which also maps the role of the class of NP-hard problems.

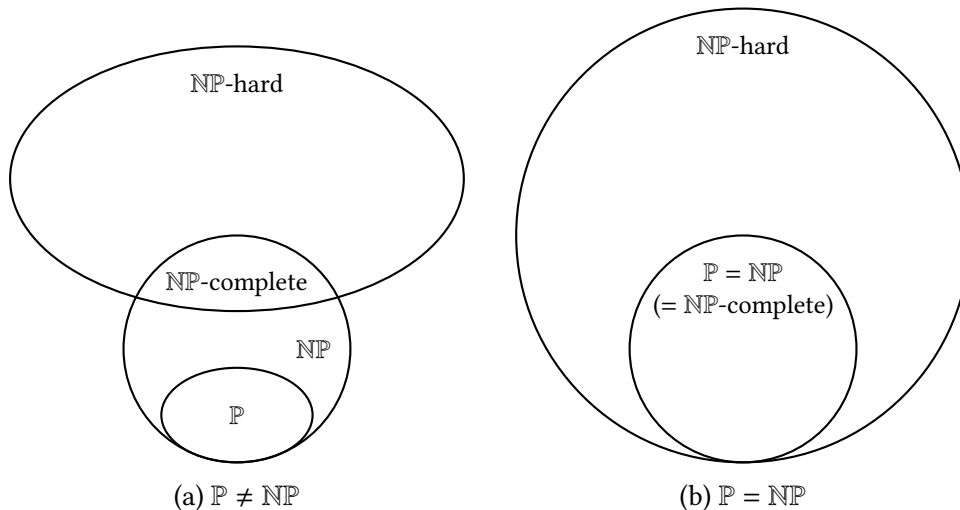


Figure 3.9: Set diagram of (possible) relations between the complexity classes \mathbb{P} , NP , and the sets of NP-hard and NP-complete problems.

We summarize this special characteristic of NP-complete problems with the following theorem, which is adapted from Cormen and colleagues [CLRS09].

Theorem 3.3.22. *If any NP-complete problem is solvable in polynomial time, then $\mathbb{P} = \text{NP}$. Equivalently, if any problem in NP is not polynomial-time solvable, then no NP-complete problem is solvable in polynomial time.*

As already mentioned the theorem can be proven by applying lemma 3.3.20 and lemma 3.3.16.

Proof. We prove the first statement of the theorem, first. We assume that $\Pi \in \mathbb{P}$ and also that Π is NP-complete. With Π is NP-complete, it is also NP-hard and, therefore, $\Pi' \leq_P \Pi$ for any $\Pi' \in \text{NP}$. As we assume that also $\Pi \in \mathbb{P}$, it holds by applying lemma 3.3.16 that $\Pi' \in \mathbb{P}$. And if this holds for any problem $\Pi' \in \text{NP}$, then it holds for all $\Pi' \in \text{NP}$ due to lemma 3.3.20.

The second statement is the contrapositive of the first statement and, therefore, none of the NP-complete problems can be solvable in polynomial time, because, if any problem $\Pi' \in \text{NP}$ is polynomial-time solvable, we can apply the first statement. \square

The first statement of the theorem can be assigned to the right side of figure 3.9. The second statement of the theorem corresponds to the left side and we can see that the sets of NP-complete problems and problems in the class \mathbb{P} are disjoint.

3.3.3.1 Cook's theorem

In the last subsection we defined and discussed the class of NP-complete problems and learned the relevance of the NP-complete problems. Moreover, to prove that a problem belongs to the class of NP-complete problems, we showed that it is sufficient to reduce an already known NP-complete problem to it. This naturally raises the question of an initial problem for which NP-completeness has been shown.

Cook showed that there is one problem called (*Boolean*) *Satisfiability problem*, or short SAT, that is NP-complete. It is a decision problem of Boolean logic and uses the following terms with which it can be described.

Definition 3.3.23. Let $X = \{x_1, \dots, x_n\}$ be a set of *Boolean variables* and let $t : X \rightarrow \{\text{true}, \text{false}\}$ be the corresponding *truth assignment* for X .

The *literals* over X are defined by extending t to the set $L := X \cup \{\bar{x} : x \in X\}$, i.e., it contains not only x , but also \bar{x} —the so-called *negation* of x . The interpretation of the literals is given in the following way:

$$t(\bar{x}) = \text{true if } t(x) = \text{false, and } t(\bar{x}) = \text{false if } t(x) = \text{true.}$$

3 Definitions and Preliminaries

We briefly denote the corresponding literals of x by x and \bar{x} . Furthermore, a literal x is true if and only if the variable x is true.

A *clause* C is a set of literals over X , which is connected by a disjunction of those literals. A clause is *satisfied* by a truth assignment if and only if at least one of the contained literals is true. And with that the clause evaluates to true.

A *collection* $C = \{C_1, \dots, C_m\}$ of clauses is a conjunction of clauses, i.e., $C_1 \wedge C_2 \wedge \dots \wedge C_m$. It is satisfied if and only if there exists a truth assignment for X that satisfies all clauses $C_j \in C$ for all $j = 1, \dots, m$ and therefore, the conjunction of clauses evaluates to true.

An example for a clause C is given by $(x_1 \vee x_2 \vee \bar{x}_3)$. Finally, the SAT problem can be formulated as follows.

Problem 3.3.24 (SAT). Let $n, m \in \mathbb{N}$. For a set $X := \{x_1, \dots, x_n\}$ of boolean variables and a collection $C := \{C_1, \dots, C_m\}$ of clauses over X , is there a satisfying truth assignment for C , so that it evaluates to true?

We demonstrate the SAT problem by the following example.

Example 3.3.25. Let $X = \{x_1, x_2, x_3, x_4\}$ be a set of Boolean variables, and let $(x_1 \vee \bar{x}_2)$, $(\bar{x}_1 \vee x_3)$ and $(\bar{x}_2 \vee x_3)$ be three clauses. For the collection

$$C = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_3) \wedge (\bar{x}_2 \vee x_3)$$

the truth assignment t that sets all variables as “true” satisfies the collection.

Another example is given by the collection

$$C' = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4).$$

The truth assignment t' that sets all variables equal to “false” satisfies the collection.

Finally, we state the theorem of Cook, which now gives a first known NP-complete problem.

Theorem 3.3.26 (Cook’s theorem). SAT is NP-complete.

In this thesis, we do not prove Cook’s theorem and instead refer the interested reader to the work of Cook himself [Coo71], Garey and Johnson [GJ79], and Korte and Vygen [KV06]. A good description of the basic idea of the proof can also be found in the work of Kleinberg and Tardos [KT13].

3.3.3.2 Methods for proving NP-completeness

There exists more than one method to prove NP-completeness [GJ79]. In this thesis, we focus on two widely used methods—reduction and restriction. In a first step, we show how NP-completeness is proven by reductions. More precisely, we prove NP-completeness by Karp’s reductions. As we mostly use Karp’s reductions in this thesis, we illustrate the technique of Karp’s reduction by proving that the CLIQUE problem is NP-complete. In a second step, we describe the technique of restrictions.

3.3.3.2.1 Reduction. The basic idea of showing NP-hardness by applying Karp’s reduction and the lemma 3.3.21 is to prove by contradiction that the problem must be NP-hard. Otherwise, every problem in NP would be solvable in polynomial time (see theorem 3.3.22). This can be summarized as follows.

Let Π be a problem whose complexity is unknown and for which we want to show that it is NP-hard. Furthermore, there exists a known NP-complete problem Π' . By applying Karp’s reduction we show that $\Pi' \leq_P \Pi$ and therefore, Π' is not “harder” than Π . So if we assume that there exists a polynomial-time algorithm which solves Π , then we would also have a polynomial-time algorithm that solves Π' (see lemma 3.3.16). Since there is a polynomial transformation between the two problems (denoted by \leq_P) and Π' is a known NP-complete problem, we can prove by contradiction that Π must also be a NP-hard problem. By lemma 3.3.20 we know that it is sufficient to show $\Pi' \leq_P \Pi$ for any known NP-complete problem Π .

Moreover, if we can prove that $\Pi \in \text{NP}$, then Π is NP-complete. The strategy of proving NP-completeness is also shown in the work of Kleinberg and Tardos [KT13].

To exemplarily prove NP-completeness, we use the already discussed CLIQUE problem. To show that CLIQUE is NP-complete, we use SAT—the first NP-complete problem proven (see subsection 3.3.3.1).

Theorem 3.3.27. CLIQUE is NP-complete.

Proof. In subsection 3.3.2 we have already shown that $\text{CLIQUE} \in \text{NP}$. Therefore, it is sufficient if we prove the NP-hardness of CLIQUE. We apply Karp’s reduction (see definition 3.3.13) and lemma 3.3.21

We assume that there exists a polynomial-time subroutine for CLIQUE. We now prove that

$$\text{SAT} \leq_P \text{CLIQUE}.$$

3 Definitions and Preliminaries

Let a collection C of size $m \in \mathbb{N}_0$ be given, which is an instance of the SAT problem. Thus, the collection is given by $C = \{C_1, \dots, C_m\}$ over a set of boolean variables $X = \{x_1, \dots, x_n\}$. We define by z_{ji} the literal of the boolean variable x_i , $i \in [n]$, contained in clause C_j . Each clause $C_j = \{z_{j1}, \dots, z_{jt}\}$, with $[t] \subseteq [n]$, consists of literals z_{ji} connected by a disjunction.

We transform this given input to an instance for the CLIQUE problem by the following transformation.

For every clause C_j we interpret all t literals as vertices of a graph $G = (V, E)$. The edges $\{v_{ji}, v_{j'i'}\}$ of the graph are constructed between each pair of vertices v_{ji} and $v_{j'i'}$ of G except one of the following two conditions is satisfied.

- a) Vertices belonging to the same clause are not connected, $\{v_{ji}, v_{j'i'}\} \notin E \Leftrightarrow j = j'$.
- b) Two literals are complementary to each other and, therefore, cannot be satisfied simultaneously, $\{v_{ji}, v_{j'i'}\} \notin E \Leftrightarrow z_{j'i'} = \bar{z}_{ji}$.

Here, the integer $k \in \mathbb{N}_0$ of the instance of CLIQUE is given by $m \in \mathbb{N}_0$.

The considered transformation is computable in polynomial time. This is because the transformation of each literal of each clause to a vertex can be performed in $O(t \cdot m)$ time. Further, the construction of the edges taking into account the two conditions takes $O(m^2)$ time.

In a last step we show the correctness of the reduction. For this we prove the following claim.

Claim: Let $m \in \mathbb{N}_0$ and C be a collection of SAT. A graph $G = (V, E)$ has a clique of size at least $k = m$ if and only if C is satisfiable.

“ \Leftarrow ”: Let C be satisfiable. Then there exists an assignment $t(x_1), \dots, t(x_n)$ for X so that all clauses $C_j \in C$ are satisfied. As each clause is a disjunction of its literals, it is sufficient if one of the contained literals $\{z_{j1}, \dots, z_{jt}\}$ is satisfied. Therefore, we choose one satisfied literal per clause and denote the selection with z_1^*, \dots, z_m^* . v_1, \dots, v_m are the corresponding vertices in G to the satisfied literals we selected according to our transformation.

We show that this is a clique of size $k = m$.

By construction of G , every pair of v_1, \dots, v_m has a connecting edge. As only one literal per clause is selected, and if all truth assignments of the selected literals are fulfilled, no complement can be contained in the selection. Otherwise, this would be a contradiction to our selection of satisfied literals z_1^*, \dots, z_m^* .

“ \Rightarrow ”: Let G be a clique of size at least $k = m$. Furthermore, we assume that v_1, \dots, v_q be a clique of size $q \geq m$ in G . With that also v_1, \dots, v_m form a clique of G . By transformation there exists no edge, which connects two vertices within a clause. Therefore, every vertex v_j of the considered clique corresponds to a literal z_j^* from exactly one clause C_j . Moreover, we know per assumption that v_1, \dots, v_m is a clique. And as there exists no edge between complementary literals according to the transformation, the corresponding literals z_j^* and $z_{j'}^*$ of any pair v_j and $v_{j'} \in \{v_1, \dots, v_m\}$ can be satisfied simultaneously.

To show that a collection C of SAT is satisfiable, we need a satisfying assignment for X . It is sufficient that the literals z_1^*, \dots, z_m^* are satisfied, and the remaining Boolean variables are assigned arbitrarily. As per construction every clause C_j for all $j = 1, \dots, m$ contains a literal z_j , which is satisfied. And with that, also C is satisfied.

With this, we proven that CLIQUE is NP-hard, and since CLIQUE \in NP, we have also shown the NP-completeness of CLIQUE. \square

By this reduction method, Karp proven NP-completeness for a lot of problems [Kar72]. Therefore he reduced the SAT problem to 21 problems by applying his invented Karp’s reduction—among others the Knapsack problem, the CLIQUE problem and 0-1 integer problem. For a description of the Knapsack problem we refer to subsection 3.3.4. A description of the 0-1 integer problem can be found in section 3.4.

As we mentioned at the beginning of this subsection there exists a second method to prove NP-hardness—*restriction*.

3.3.3.2.2 Restriction. In the case of restriction, it is sufficient to show that a given problem $\Pi \in$ NP contains a problem Π' as a special case, which is a known NP-complete problem. To show that a problem can be restricted to a known NP-complete problem Π' , we need to specify the restrictions that must be applied to Π . By this given restrictions, the resulting instance must be identical to an instance of Π' . However, it is not necessary that “the restricted problem and the known NP-complete problem [are] *exact* duplicates of one another, but rather that there [is] an obvious one-to-one correspondence between their instances that preserves yes and no answers” [GJ79, p. 63].

For example, another way to show NP-completeness of SAT is to find a known NP-complete problem that is a restricted special case of SAT.

A famous NP-complete special case of SAT is the 3-SAT problem. The 3-SAT problem is defined similarly to SAT, but each clause consists of exactly three literals.

3 Definitions and Preliminaries

Hence, the restriction applied to SAT is simply a SAT problem where each clause is restricted to exactly three literals.

The following corollary summarizes the method of restriction, but from a different perspective.

Corollary 3.3.28. *A problem $\Pi \in \text{NP}$, which is a generalization of an NP -complete problem Π' , is NP -complete.*

Proof. We know that $\Pi \in \text{NP}$ and with that it can be solved in nondeterministic polynomial time. Furthermore, we know that for restricted constraints the problem is NP -complete. We denote the restricted version of the problem by Π' .

Let us assume that Π can be solved in polynomial time. We can transform an instance of Π' into an instance of Π by restricting the constraint accordingly. This can be done in polynomial time since it is a one-to-one transformation. And with that, we would have also a polynomial time algorithm to solve Π' . However, this is contrary to Π' is NP -complete. Therefore, it is at least as hard as Π' to solve, in terms of Karp's reduction. And with that, we have shown that Π is NP -complete. \square

3.3.4 Further NP -complete problems

In this subsection, we present other NP -complete problems that we will use in the rest of this thesis. The goal is not to explain these problems in detail, but rather to present the basic ideas of these problems. Therefore, we will not prove the NP -completeness of these problems, but refer to appropriate literature.

3.3.4.1 The 3-SAT problem

In subsection [3.3.3](#) we have already introduced the SAT problem. The SAT problem searches for a satisfaction of a truth assignment for a collection of clauses. It is the first problem for which NP -completeness was proven by Cook's theorem (see theorem [3.3.26](#)). In recent years, it has been studied whether the SAT-problem can be solved in polynomial time if the literals per clause are constrained. And indeed it was proven that by restricting the literals per clause to two, the so-called 2-SAT $\in \mathbb{P}$ [\[GJ79\]](#).

In this subsection, we consider the 3-SAT problem. Similar to the 2-SAT problem, the literals per clause are restricted, but in this case there exist three literals per clause. More precisely, for a given set of Boolean variables $X := \{x_1, \dots, x_n\}$, we define a collection $\mathcal{C} := \{C_1, \dots, C_m\}$ in which each clause C_j is a set of exactly three literals, i.e., $|C_j| = 3$ for all $j \in [m]$. Then, the 3-SAT problem is as follows.

Problem 3.3.29 (3-SAT). Let $n, m \in \mathbb{N}$. For a set $X := \{x_1, \dots, x_n\}$ of Boolean variables and a collection $C := \{C_1, \dots, C_m\}$ of clauses over X with $|C_j| = 3$ for all $j \in [m]$, is there a satisfying truth assignment for C , so that it evaluates to true?

The collection C' in example 3.3.25 is a 3-SAT problem.

By reducing the SAT problem to 3-SAT, we can show that it is NP -complete. The proof is given by Geary and Johnson [GJ79, p. 48]. Therefore, without proof, we state the following proposition.

Proposition 3.3.30. *3-SAT is NP -complete.*

For more information about the 3-SAT problem, we refer the interested reader to Geary and Johnson [GJ79], and Kleinberg and Tardos [KT13].

3.3.4.2 The independent set problem

In this subsection we introduce the *independent set problem* or IS problem, which we use to prove that the EBRs problem is NP -complete (see subsection 4.2.3).

Before formulating the IS problem, we briefly review the definition of independent set as defined in subsection 3.2.1. An independent set is a term from graph theory and describes a set of vertices $U \subseteq V$ in which no two vertices are adjacent. An independent set is maximal, if no further vertex can be added to U . If furthermore, $|U|$ is maximal, the independent set is maximum.

The decision problem is formulated as follows, based on the work by Kleinberg and Tardos [KT13].

Problem 3.3.31. Given an undirected graph G and an integer k , does G contain an independent set of size at least k ?

The IS problem is NP -complete, too. Kleinberg and Tardos give a prove by reducing 3-SAT to the IS problem, i.e., $3\text{-SAT} \leq_P \text{IS}$ [KT13, p. 460].

Proposition 3.3.32. *IS is NP -complete.*

We find more informations about the IS problem in the work of Geary and Johnson [GJ79], Kleinberg and Tardos [KT13] and Dasgupta and colleagues [DPV06].

3.3.4.3 The perfect matching problem

We introduced the *perfect matching* in subsection [3.2.1](#). Given a graph $G = (V, E)$, it is a set $M \subset E$ of edges that covers all vertices $v \in V$. Finding a perfect matching in a graph is an NP -complete problem called *perfect matching problem* or PM problem.

The perfect matching problem is given as follows.

Problem 3.3.33 (PMP). Given an undirected graph $G = (V, E)$ and an integer $k \in \mathbb{N}_0$, does there exist a subset $M \subseteq E$ with cardinality of M is at least k , such that M is a perfect matching?

As already mentioned the PM problem is NP -complete.

Proposition 3.3.34. PM is NP -complete.

As in previous subsections, we will not prove the proposition, but refer to the work of Plaisted and Zaks [\[PZ80\]](#). They show that the perfect matching problem restricted to *bipartite graphs* is NP -complete by reducing 3-SAT to it. Here, the set of vertices of a bipartite graph can be partitioned into two sets of vertices $U \subset V$ and $U' \subset V$ such that $U \cap U' = \emptyset$ and both sets are independent, i.e., vertices in the same partition must not be adjacent [\[Die17\]](#). Moreover, each edge has its end in different classes and, thus, for each edge $e \in E$ one vertex of the edge is in U and the other is in U' . Figure [3.10](#) shows such a bipartite graph.

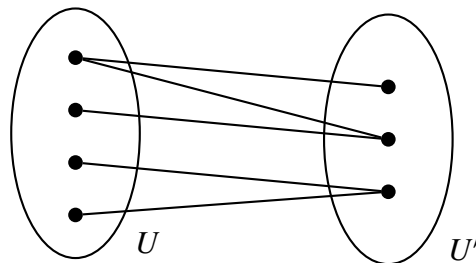


Figure 3.10: Example of a bipartite graph.

Since the restricted PM problem is NP -complete, also the generalization of PM is NP -complete.

3.3.4.4 The Knapsack problem

A very well studied problem is the *Knapsack problem* or KP problem. Briefly, the objective of the problem is to pack a Knapsack with items $i = 1, \dots, n$ of value p_i without exceeding a Knapsack capacity c . It is a maximization problem and, therefore, the higher the total value of the packed items, the better.

Of course, in practical applications the problem is not restricted to packing a Knapsack. Kellerer and colleagues introduce the Knapsack problem as a general binary decision problem [KPP04], i.e., there exists a binary variable x_i for $i \in [n]$ with $x_i = 1$ denotes the choice of a first alternative, and $x_i = 0$ the choice of a second alternative. Each decision is accompanied by a value p_i for all items. Yet, if we choose the first alternative, i.e., $x_i = 1$, we also have to take into account a weight or resource w_i . If we choose the second alternative instead, we do not have to consider any weight. Thus, we have to check the feasibility of each particular selection of alternatives. The feasibility is expressed by the capacity c and can be calculated as “a linear combination of coefficients for binary decision” [KPP04]. We write the feasibility check as

$$\sum_{i=1}^n w_i x_i \leq c.$$

For example, real-world decision making processes that can be formulated as a Knapsack problem are the selection of investments and portfolios, or the least wasteful way to cut raw materials.

In this thesis we always refer to the 0-1 Knapsack problem when talking about the Knapsack problem. i.e., we consider the binary variable

$$x_i = \begin{cases} 1, & \text{if item } i \text{ is packed,} \\ 0, & \text{otherwise.} \end{cases}$$

We already mentioned that the feasibility check of the Knapsack problem is a linear combination. In total, the optimization version of the Knapsack problem is given by:

Program 3.3.35 (KP program).

$$\begin{aligned} & \max \sum_{i=1}^n p_i x_i \\ \text{s.t.} & \sum_{i=1}^n w_i x_i \leq c \\ & x_i \in \{0, 1\} \quad i \in [n]. \end{aligned}$$

We assume w.l.o.g. that p_i , w_i , and c are positive integers. Furthermore, we assume that

$$\sum_{i=1}^n w_i > c \text{ and } w_i \geq c \quad \forall i \in [n].$$

3 Definitions and Preliminaries

Otherwise, we could solve the problem trivially. For if $\sum_{i=1}^n w_i x_i \leq c$, we solve the KP problem by setting $x_i = 1$ for all $i \in [n]$. And if $w_i > c$, we simply set the corresponding binary variable equal to zero, i.e., $x_i = 0, i \in [n]$. The cases where p_i, w_i , or c are no positive integers are discussed by Martello and Toth [MT07].

Finally, the decision version of the Knapsack problem is formulated as follows.

Problem 3.3.36. Given an integer $k \in \mathbb{N}_0$, can the value of the packed items sum up to a value of at least k without exceeding the Knapsack capacity?

We show by the following proposition that the KP problem is NP-complete [DPV06, KPP04, MT07].

Proposition 3.3.37. KP is NP-complete.

For the proof we refer to Karp [Kar72].

3.3.4.5 The Subset-Sum problem

A special case of the Knapsack problem is the SUBSET-SUM problem. In itself, it is very similar to the KP problem, but instead of value p_i , we are also observing the weights w_i in the maximization, i.e., $p_i = w_i$ for all $i \in [n]$, or in other words, when profit and weight are strongly correlated. It is formulated as follows.

Program 3.3.38.

$$\begin{array}{ll} \max & \sum_{i=1}^n w_i x_i \\ \text{s.t.} & \sum_{i=1}^n w_i x_i \leq c \\ & x_i \in \{0, 1\} \quad i \in [n] \end{array}$$

Similar to the KP problem we assume w.l.o.g. that w_i and c are positive integers and

$$\sum_{i=1}^n w_i > c \text{ and } w_i < c \quad \forall i \in [n].$$

In practical applications the SUBSET-SUM is applied in situations when a quantitative target should be reached under the constraint that upward deviations are avoided and downward deviations are minimized [MT07]. Therefore, we reformulate the SUBSET-SUM problem by considering a set of positive integers S and an

integer target $t > 0$. The problem is then given by finding a subset $S' \subset S$ for which the contained positive integers sum up to exactly t .

We therefore define the problem as follows [CLRS09].

Problem 3.3.39 (SUBSET-SUM). Given an instance (S, t) . Does there exist a subset $S' \subseteq S$ such that $\sum_{s \in S'} s = t$?

The reformulation can be justified by the fact that we interpret the product of a binary variable x_i with its weight w_i for all $i \in [n]$ as a positive number, and all positive numbers yield the set S . The set of all accepted items i , i.e., all items with $x_i = 1$, is defined as the set S' . Illustratively, we can consider the set S' as our Knapsack. Since the KP problem is a maximization problem with an upper bound, we would pack as much items into the Knapsack, such that it is as close to the Knapsack capacity c as possible. For the SUBSET-SUM problem we restrict the set of feasible solutions of the KP problem to those which fit the upper bound exactly, i.e., $\sum_{s \in S'} s = t$.

To explain the SUBSET-SUM problem more clearly, we give the following example.

Example 3.3.40. Let $S = \{3, 4, 10, 12, 14, 28, 30, 37, 45\}$ and $t = 50$. Then, $S' = \{3, 10, 37\}$.

Both versions of the SUBSET-SUM problem are NP-complete. For the proof of the following proposition, we refer to Cormen and colleagues [CLRS09], who prove NP-completeness based on the second version of the SUBSET-SUM problem as described in problem 3.3.39. Since this is a restriction of the first, one has shown NP-completeness for both problems.

Proposition 3.3.41. SUBSET-SUM is NP-complete.

More details on the SUBSET-SUM problem can be found in the work of Cormen and colleagues [CLRS09], and Martello and Toth [MT07].

3.3.4.6 Generalizations of the Knapsack problem

In this subsection we present two generalizations of the already discussed KP problem. One is the *Precedence-constrained Knapsack problem* or PCKP problem and the other is the *Knapsack problem with conflict graph* or KCG problem.

We start with the PCKP problem and follow You and Yamada [YY07], as well as Samphaiboon and Yamada [SY00] for the description. In the PCKP problem the

3 Definitions and Preliminaries

Knapsack problem is generalized by partially ordered items through a set of precedence relations. This can be illustrated by the requirement that we cannot pack an item until all preceding items have been included in the Knapsack.

In short, the PCKP problem is motivated graph theoretically by expressing the dependencies in terms of edges connecting two vertices. In more detail this can be described as follows.

Let $G = (V, E)$ be an undirected graph with vertex set $V = \{1, \dots, n\}$ and edge set $E \subseteq V \times V$, where $n := |V|$ and $m := |E|$. We interpret the vertex set as the set of items that can be packed into a Knapsack until a capacity c is reached. Similar to the Knapsack problem, each item has a value p_i and a weight w_i , $i \in V$. And, w.l.o.g. we assume that c , p_i , and w_i are positive integers for all $i \in [n]$.

As we mentioned the edge set E represents the dependencies between the items or the *precedence relation* [YY07], respectively. Therefore, we define the set as

$$\begin{aligned} E &= \{(i, j) \in E : \text{item } j \text{ can be accepted only if item } i \text{ is already packed}\} \\ &= \{(i, j) \in E : \text{item } i \text{ precedes item } j\}. \end{aligned} \quad (3.4)$$

To guarantee well-defined precedence relations, we suppose that there exists no cyclic graph G (see section 3.2). Furthermore, we assume that

$$w_i \leq c \quad \forall i \in V \quad \text{and} \quad \sum_{i=1}^n w_i \geq c$$

to exclude trivial solutions (see also the KP problem).

Finally, the PCKP problem is given by:

Program 3.3.42 (PCKP).

$$\begin{aligned} & \max \sum_{i=1}^n p_i x_i \\ \text{s. t.} \quad & \sum_{i=1}^n w_i x_i \leq c \\ & x_i \geq x_j \quad \forall (i, j) \in E \\ & x_i \in \{0, 1\} \quad \forall i \in V \end{aligned}$$

with

$$x_i = \begin{cases} 1, & \text{if item } i \text{ is accepted or packed, respectively,} \\ 0, & \text{otherwise,} \end{cases}$$

for all $i \in [n]$.

By the next problem we give the decision version of the PCKP problem.

Problem 3.3.43. Given an integer $k \in N_0$, can the value of the packed item sum up to a value of at least k , without exceeding the Knapsack capacity and under consideration of the precedence relations between the items?

As the PCKP problem is a generalization of the KP problem, it is NP -hard. Otherwise, we would also have a polynomial time solver for the Knapsack problem. Rather, it can be shown that PCKP is NP -complete.

Proposition 3.3.44. PCKP is NP -complete.

A proof of this proposition is given by Garey and Johnson [GJ79].

As the name suggests, the KCG problem is also illustrated and explained in graph theoretic terms. In the KCG problem, we consider a packing problem with items that may not be packed together in a Knapsack.

Similar to the PCKP problem we consider the items as vertices (which are uniquely assigned). The conflicts between the items are represented by the edges of the corresponding undirected graph $G = (V, E)$, i.e., the edge $e_j = \{i, i'\} \in E$ with $i, i' \in V$ indicates that items i and i' cannot be packed together. The graph is defined as above. Furthermore, each item has a profit p_i and a weight w_i for all $i \in [n]$.

Finally, the KCG problem can be formulated as follows.

Program 3.3.45.

$$\begin{array}{ll} \max & \sum_{i=1}^n p_i x_i \\ \text{s. t.} & \sum_{i=1}^n w_i x_i \leq c \\ & x_i + x_{i'} \leq 1 \quad \forall (i, i') \in E \\ & x_i \in \{0, 1\} \quad i \in [n] \end{array}$$

Similarly to the PCKP problem, the decision version of the KCG problem can be formulated. Instead of taking into account the precedence relation between the items we must consider the conflicts between the items. Therefore, the decision version is as follows.

Problem 3.3.46. Given an integer $k \in N_0$, can the value of the packed item sum up to a value of at least k without exceeding the Knapsack capacity, and under consideration of the conflicts between the items?

In addition, KCG is a generalization of the KP problem and, therefore, it is NP -hard. Furthermore, we state the NP -completeness by the following proposition.

Proposition 3.3.47. *KCG is NP-complete.*

For more information about the KCG problem we refer to the work of Pferschy and Schauer [PS16], and Yamada and colleagues [YKW02].

3.4 Integer linear programs

In the last subsections, we have focused on the decision version of problems. The standardized decision problems whose solution requires a “yes” or “no” answer (see definition 3.3.6), allowed us to determine their complexity according to their running time. In this section, we change the perspective on the problems considered in, for example, subsection 3.3.4. Instead of the decision version, we now consider their optimization version, i.e., we search for an efficient algorithm to solve the problem deterministically and obtain a specific solution $x^* \in \mathbb{Z}^n$ of the problem. In this thesis, we consider a special form of optimization problems—the *integer linear optimization problems* or *integer linear programming (ILP)*. In our case, they are even restricted to a binary solution, i.e., $x^* \in \{0, 1\}^n$.

Since the focus of this thesis is rather on the complexity-theoretic investigation, we only give a short overview and definition of ILPs and refer the interested reader to the work of Schrijver [Sch98], Dantzig and Thapa [DT97], Korte and Vygen [KV06], or Gritzmann [Gri13]. The descriptions in this section are based on those given by Korte and Vygen [KV06].

In a first step, we define linear problems and introduce ILPs based on the following definition.

Definition 3.4.1. Let $m, n \in \mathbb{N}$. For an instance of the linear problem—the so-called *linear program (LP)*—the task is to find a column vector $x \in \mathbb{R}^n$ such that $c^T x$ is maximum under the constraint $Ax \leq b$. The instance of such a problem is given by a matrix of constraints $A \in \mathbb{R}^{m \times n}$, column vectors $b \in \mathbb{R}^m$, and a row vector $c \in \mathbb{R}^n$.

A vector $x \in \mathbb{R}^n$ is called *feasible solution*, if it satisfies the constraint $Ax \leq b$. The *set of all feasible solutions* of the LP is given by

$$P := \{x : Ax \leq b, x \in \mathbb{R}^n\}.$$

If no (bounded) set of feasible solutions can be found, either $P = \emptyset$ holds for the LP, or there exists a $x \in \mathbb{R}^n$ with $Ax \leq b$ and $c^T x > \alpha$ for all $\alpha \in \mathbb{R}$. The problem is *infeasible* in case of $P = \emptyset$ and *unbounded* for the latter case. If a feasible solution is maximal it is called *optimum solution*.

Remark 3.4.2. We write the linear program as

$$\max\{c^T x : Ax \leq b\}.$$

In this thesis, we focus exclusively on maximization problems. At this point, however, we want to briefly mention that a maximization problem can easily be transformed into a minimization problem by multiplying it by -1 . This leads to the following remark.

Remark 3.4.3. By defining the maximization problem we also defined the minimization problem as

$$\max_{x \in P} c^T x = - \min_{x \in P} (-c)^T x.$$

Linear programs can be solved in polynomial time. This was shown in 1979 by Leonid Khachiyan who discovered the so-called *ellipsoid algorithm* which runs in polynomial time and solves linear programs [Kha80, KV06, GLS93]. Grötschel and colleagues stated that Khachiyan’s result caused “great excitement in the world of mathematical programming” [GLS93, p. 64], which is understandable since it shows that LPs are in the complexity class \mathbb{P} . Specifically, Khachiyan formulated the following theorem.

Theorem 3.4.4 (Khachiyan, 1979). *There exists a polynomial-time algorithm for LPs (with rational input), and this algorithm finds an optimum solution if there exists one.*

We will not discuss the ellipsoid algorithm and the proof of this theorem in this thesis and instead refer the interested reader to the works of Korte and Vygen [KV06], Grötschel and colleagues [GLS93], and Khachiyan [Kha80].

In the second part of this section, we now want to define integer linear programs. In subsection 3.3.4 we have already discussed ILPs, such as the Independent Set or Knapsack Problem. However, we have not yet identified them as ILPs. For the definition of ILPs, we follow Gritzmann [Gri13].

Definition 3.4.5 (ILP). Let a linear program $\max\{c^T x : Ax \leq b\}$ as defined in definition 3.4.1 be given. If $x \in \mathbb{Z}^n$, then the program is an *integer linear program (ILP)*. An ILP with $x \in \{0, 1\}^n$ is called a *0-1 integer linear problem (0-1-ILP)*.

Remark 3.4.6. In preparation for the EBRS problem discussed later, both the matrix A and the vector b are integer, i.e., $A \in \mathbb{Z}^{n \times n}$ and $b \in \mathbb{Z}^m$. Therefore, an ILP or a 0-1-ILP with integer A and b is defined in the remainder of this thesis.

3 Definitions and Preliminaries

Similar as for the LP, we define the set of feasible solutions of an ILP by

$$P_I := \{x : Ax \leq b, x \in \mathbb{Z}^n\}.$$

Unlike LPs, ILPs are now, in general, no longer solvable in polynomial time. That is, by the restriction of $x \in \mathbb{Z}^n$, no solution can be found in polynomial time. Even more, an ILP is \mathbb{NP} -complete.

Theorem 3.4.7. *An ILP is (in general) \mathbb{NP} -complete.*

A proof of the theorem is given by Karp [Kar72]. A 0-1 integer linear program is still \mathbb{NP} -complete, which we prove by the following lemma.

Lemma 3.4.8. *0-1-ILP is \mathbb{NP} -complete.*

Proof. We prove this lemma by showing that the 3-SAT problem defined in problem 3.3.29 can be reduced to 0-1-ILP in the sense of Karp. To show that the ILP is \mathbb{NP} -complete, we prove that it is contained in the complexity class \mathbb{NP} and that $3\text{-SAT} \leq_P 0\text{-1-ILP}$.

a) “0-1-ILP $\in \mathbb{NP}$ ”:

Let $I = (m, n, A, b, c)$ be an instance of 0-1-ILP, for which a certificate C —a vector $x \in \{0, 1\}^n$ —is given. Furthermore, let the instance concatenated with the certificate C be a solution to the checking problem 0-1-ILP' so that $(I \oplus C, 1) \in 0\text{-1-ILP}$. In other words, the certificate C is a vector $x \in \{0, 1\}^n$ such that $Ax \leq b$ is satisfied and $c^T x$ is maximal under this condition. The verification of the solution $(I \oplus C, 1)$ can be done in polynomial time. Ax can be computed in $\mathcal{O}(nm)$ and the satisfaction of the constraint $Ax \leq b$ can be checked in $\mathcal{O}(m)$ by element-wise comparison. Thus, 0-1-ILP can be verified in polynomial time and hence $0\text{-1-ILP} \in \mathbb{NP}$.

b) “3-SAT \leq_P 0-1-ILP”:

This statement is proven similarly as in the work by Karp [Kar72]. Let $C := \{C_1, \dots, C_m\}$ be a collection of clauses with $|C_j| = 3$ for all $j \in [m]$ over a set $X := \{x_1, \dots, x_n\}$ of Boolean variables. A collection C is transformed into an instance of 0-1-ILP by the following transformation rules.

In a first step each true variable ($x_i = \text{true}$) is transformed to $z_i = 1$ and each false variable ($x_i = \text{false}$) is transformed to $z_i = 0$ for all $i \in [n]$. This transformation can be done in $\mathcal{O}(n)$. We construct the matrix A by defining each entry a_{ji} of the matrix by

$$a_{ji} := \begin{cases} -1 & \text{if } z_i \in C_j, \\ 1 & \text{if } \bar{z}_i \in C_j, \\ 0 & \text{otherwise,} \end{cases}$$

with $i \in [n]$ and $j \in [m]$. Therefore, $A \in \{-1, 0, 1\}^{n \times m}$.

For each clause C_j we define the constraint b_j as follows.

$$b_j := \sum_{i=1}^n \max(0, a_{ji}) - 1 \quad (3.5)$$

Thus b_j can be described as the sum of all negated literals in C_j minus one. This transformation runs in $O(nm)$ and is therefore a polynomial time transformation.

To show that the transformation is correct we prove the following claim.

Claim: There exists a vector $z \in \{0, 1\}^n$ that satisfies the constraint $Az \leq b$ if and only if there exists a truth assignment of the collection C .

We denote by $(Az)_j, j \in [m]$, the j 'th row of Az .

To prove the claim, we consider the inequality $Az \leq b$ row by row and discuss it in comparison with a truth assignment of a collection C .

In this proof we consider the 3-SAT problem, i.e., there exists exact three literals per clause. To assign "yes" to a clause at least one literal must be true.

By our transformation rules discussed above we defined b_j for all rows $j \in [m]$ as the sum of all negated literals in C_j minus one. Moreover, if there exists a negated literal \bar{z}_i in a clause C_j , then we denote it with 1 according to the transformation rule. Non-negated literals are not counted.

The -1 in the definition of b_j determines that at least one literal must be true ($z_i = 1$) or false ($z_i = 0$) for the inequality $(Az)_j \leq b_j$ to be satisfied. Whether the literal must be true or false depends on the number of negated literals in a clause.

Then, it holds for all $j \in [m]$

$$(Az)_j \leq b_j \Leftrightarrow \text{at least one literal is satisfied per clause.}$$

And if at least one literal is satisfied per clause, then by definition clause C_j is also satisfied. And since all clauses $C_j, j \in [m]$, are satisfied, the collection C is also satisfied. Therefore,

$$Az \leq b \Leftrightarrow x \text{ is an assignment satisfying } C.$$

□

To illustrate the transformation rule in the proof, in particular the definition of $b_j, j \in [m]$, (see formula (3.5)), we give the following example.

3 Definitions and Preliminaries

Example 3.4.9. Let a collection of clauses $C = \{C_1, \dots, C_4\}$ over a set of Boolean variables $X := \{x_1, \dots, x_5\}$ be given. We are searching for a truth assignment for

$$(x_1 \vee x_2 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_5) \wedge (\bar{x}_2 \vee x_4 \vee x_5).$$

We transform the given instance into an instance of a 0-1-ILP with constraint $Az \leq b$ by transformation rules defined in the proof of theorem [3.4.7](#) as follows.

$$\begin{pmatrix} -1 & -1 & 0 & -1 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 & -1 \end{pmatrix} \cdot \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{pmatrix} \leq \begin{pmatrix} -1 \\ 1 \\ 2 \\ 0 \end{pmatrix}$$

$x_1 = \text{true}$, $x_2 = \text{true}$, $x_3 = \text{false}$, $x_4 = \text{true}$, and $x_5 = \text{false}$ is a solution to C . Applying the transformation rules, we obtain $z_1 = 1$, $z_2 = 1$, $z_3 = 0$, $z_4 = 1$, and $z_5 = 0$, which is a solution of the corresponding 0-1-ILP.

As mentioned, we will give now a brief and exemplary overview of the functionality of the definition of b .

For this purpose, we consider the clause C_1 and the corresponding linear inequality $-z_1 - z_2 - z_4 \leq -1$. For $b_1 = -1$ to be satisfied, z_1 , z_2 , or z_4 must be equal to 1 because $z_i \in \{0, 1\}$ for all $i \in [n]$, and, thus, one of the Boolean variables x_1 , x_2 , or x_4 must also be true. Thus, both C_1 and the associated inequality are satisfied.

Similarly, we can argue for clause $C_3 = (\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_5)$. It is satisfied if at least one $x_i \in C_3$ is false, which is also reflected in the corresponding inequality, which is satisfied if one $z_i = 0$, $i \in \{2, 3, 5\}$.

We leave the consideration of the remaining two clauses to the reader at this point. In general, it can be said that by the definition of b the clauses are fulfilled, if and only if the associated inequalities are fulfilled.

3.5 LP relaxation

As proven in the previous chapter ILPs are in general \mathbb{NP} -complete, and hence there exists, in general, no efficient algorithm that solves a problem of this type under the assumption $\mathbb{P} \neq \mathbb{NP}$. In this subsection, we briefly introduce a way to solve an ILP approximately using the knowledge that an LP can be solved in polynomial time. For a more detailed introduction to the so-called *LP relaxations* we refer the interested reader to the works of Cormen and colleagues [\[CLRS09\]](#), and Matousek and Gärtner [\[MG07\]](#).

As the name LP relaxation implies, we simplify an ILP by neglecting the integrality constraint and solving an LP instead. This method gives us more information that

we can use to approximate a solution for the ILP. In case of 0-1-ILP the approach is similar. We remove the $x_i \in \{0, 1\}$ constraint and replace it with $0 \leq x_i \leq 1$. We, thus, transform the integer constraints into linear constraints for all $i \in [n]$ and convert a NP-hard problem into a related problem that is solvable in polynomial time by this relaxation technique.

More formally, the LP relaxation of an ILP

$$\max \{c^T x : x \in \mathbb{Z}^n \wedge Ax \leq b\}$$

is given by the program

$$\max \{c^T x : x \in \mathbb{R}^n \wedge Ax \leq b\}.$$

Here, an instance of the program is given by a constraint matrix $A \in \mathbb{Z}^{m \times n}$, vectors $b \in \mathbb{Z}^m$, and $c \in \mathbb{Z}^n$, as well as constants $m, n \in \mathbb{N}$.

Figure 3.11 illustrates the concept of an LP relaxation for a 0-1-ILP.

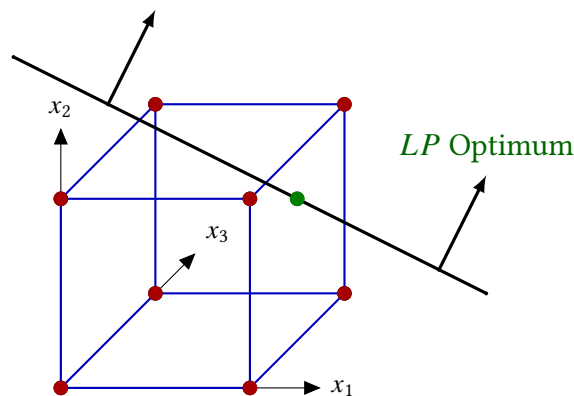


Figure 3.11: Example of an LP relaxation for a 0-1-ILP.

The 0-1-ILP is represented by the red dots in the figure. The LP relaxation searches for an optimal solution within the cube defined by the constraints $0 \leq x_i \leq 1$ for all $i \in [n]$ —represented by the blue lines. In the example given, an optimal solution of the LP relaxation is found at the point $x^* = (1, 0,75, 0,5)$. It is not an integer solution, but it gives some indication of a good integer solution (except x_3^*).

LP relaxations will also be important for the next section. There we consider 0-1-ILPs with *totally unimodular* matrices A . The peculiarity of problems with totally unimodular matrices is that an optimal solution of the LP relaxation is integer and therefore also an optimal solution of the original ILP.

3.6 Total unimodularity

In subsection [3.4](#) we discussed and proved that integer linear programs (ILPs) are in general not solvable in polynomial time. But as mentioned in subsection [3.5](#), in the case of a *totally unimodular* constraint matrix $A \in \mathbb{Z}^{m \times n}$, an ILP can be considered equivalent to its *LP relaxation*. More precisely, in case of a totally unimodular matrix $A \in \mathbb{Z}^{n \times m}$ and an integer $b \in \mathbb{Z}^m$ an ILP can be solved in polynomial time. This statement is based on a theorem of Hoffman and Kruskal (see theorem [3.6.14](#)). They showed that an ILP over a totally unimodular constraint matrix $A \in \mathbb{Z}^{m \times n}$ guarantees an integer solution, if $b \in \mathbb{Z}^m$.

Briefly and simplistically, an LP relaxation can be defined as the omission of the integrality condition $x \in \mathbb{Z}^n$ (or $x \in \{0, 1\}^n$). For more details on LP relaxation we refer to subsection [3.5](#) above.

In subsection [4.3.1.1](#) we will show for a special case of the EBRs problem that its constraint matrix is totally unimodular. And by Hoffman and Kruskal's theorem, we thus know that this special case of the EBRs problem can be solved in polynomial time.

In this section we define total unimodularity and discuss methods to prove that a matrix is totally unimodular. To define total unimodularity, we explain in a first step (integer) polyhedra as well as convex hulls. At the end of this section we will prove the theorem of Hoffman and Kruskal. In this chapter we mainly follow the explanations of Korte and Vygen [\[KV06\]](#).

We start with the definition of a polyhedron, a special geometric structure of the set of feasible solutions $P = \{x : Ax \leq b, x \in \mathbb{R}^n\}$ of an LP.

Definition 3.6.1. Let $P \subset \mathbb{R}^n$. P is called *polyhedron* if and only if there exists

$$n, m \in \mathbb{N}_0 \wedge A \in \mathbb{R}^{m \times n} \wedge b \in \mathbb{R}^m$$

with

$$P = \{x : Ax \leq b, x \in \mathbb{R}^n\}. \quad (3.6)$$

P is called *polytop* if and only if P is a polyhedron and P is bounded.

From the definition of the polyhedron in formula [\(3.6\)](#) we can see that finitely many inequalities—the constraints—are defined by $Ax \leq b$. A set of feasible solutions for $\max\{c^T x : x \in \mathbb{R}^n \wedge Ax \leq b\}$ can therefore be interpreted as the intersection of finitely many halfspaces [\[KV06\]](#). We illustrate the definition in figure [3.12](#). On the left, we can see an example for a polyhedron, whereas on the right a polytop is mapped.

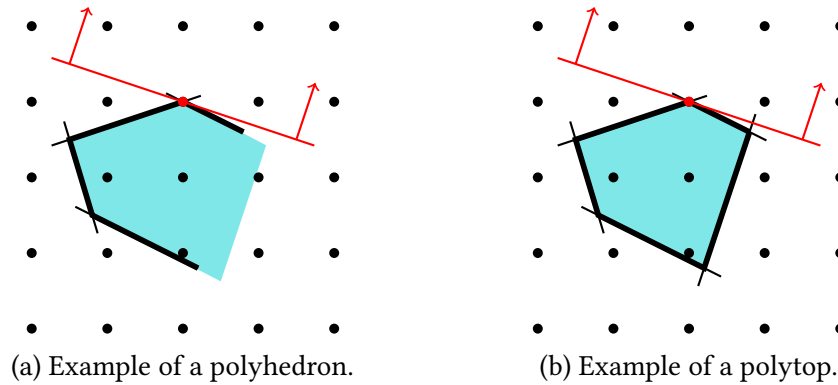


Figure 3.12: Example of a polyhedron (a) and a polytope (b).

Looking at figure [3.12](#), we see that a unique optimal solution to the LP problem $\max\{c^T x : Ax \leq b\}$ is given if x is a vertex of the corresponding polyhedron or polytope P —depending on the optimization direction. In this example, it is given by the red dots.

We can also find the solution of an LP of the form $\max\{c^T x : Ax \leq b\}$ using a different approach. This different approach may help us better understand the concept of totally unimodular matrices and the resulting possibility of neglecting the integrality constraint when solving an ILP. We will discuss this approach rather informally and refer for more details to the work of Schrijver [\[Sch03\]](#). For this different approach, we first informally define *hyperplanes*. In Euclidean space of dimension n , a hyperplane can be described as a subspace of dimension $n - 1$. For example, a hyperplane of a 3-dimensional Euclidean space is a 2-dimensional plane. We can describe a hyperplane with a linear equation of the form $c^T x = \lambda$ with $\lambda \in \mathbb{R}$ and non-trivial vector $c \in \mathbb{R}^n$. That is,

$$H := \{x : c^T x = \lambda\}.$$

In general, the goal of an LP is to maximize the linear program $c^T x$ under consideration of the constraints, which are given by the polyhedron P . For this other approach, we consider the linear program $c^T x$ as a hyperplane.

In descriptive terms, we can think of finding a solution to the ILP by shifting the hyperplane until the constraints are satisfied and at the same time $c^T x$ has taken a value as high as possible. This is the case when we find an intersection between hyperplane and polyhedron.

The following example, shown in figure [3.13](#), will serve to better understand this approach.

Example 3.6.2. Let $\lambda \in \mathbb{R}$ be a constant. Informally, let us think of a hyperplane H which is moved until it touches the polyhedron P of feasible solutions at a vertex x^* . And suppose further that the vertex satisfies $c^T x^* \leq \lambda$. Then we know

3 Definitions and Preliminaries

that the hyperplane H intersects with the polyhedron P at vertex x^* and it holds $P \cap H = \{x^*\}$. If x^* maximizes $c^T x^*$, then this would be a unique optimal solution. These considerations are represented in figure [3.13](#).

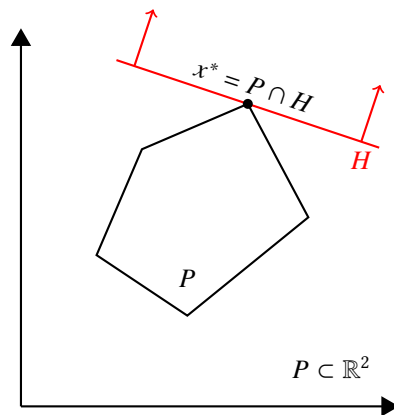


Figure 3.13: Example for a solution x^* found by an intersection between hyperplane and polyhedron.

Depending on the constant $\lambda \in \mathbb{R}$, a unique vertex cannot always be found. If the hyperplane is shifted according to the optimization direction and it “touches” the polytope not at a single vertex but at an edge of the polytope, then the intersection $P \cap H$ contains infinitely many x . In such situations, there are infinitely many solutions to the problem.

The following lemma summarizes the observations just outlined. We refer for the proof to the work of Schrijver [\[Sch03\]](#).

Lemma 3.6.3. *Let $P \subset \mathbb{R}^n$ be a polyhedron and let $x \in P$ be a vertex of the polyhedron. Then the following properties of the vertex are equivalent.*

- a) *There exists a hyperplane $H = \{x : c^T x = \lambda\}$ such that $c^T x \leq \lambda$ for all $x \in P$ and $P \cap H = \{x\}$.*
- b) *There are n constraints $a_i^T x \leq b_j$ valid for P , which are tight at v , i.e., $a_i^T v = b_j$ for $i = 1, \dots, n$, and a_1, \dots, a_n are linear independent.*

We then know that the hyperplane H affects the polyhedron P at vertex x^* and $P \cap H = \{x^*\}$. Instead, we refer the interested reader to the works of Matousek Gärtner [\[MG07\]](#), Kleinberg and Tardos [\[KT13\]](#), Korte and Vygen [\[KV06\]](#), Gritzmann [\[Gri13\]](#), and Dasgupta and colleagues [\[DPV06\]](#), who all give a good introduction to this topic.

As we have now seen, a solution of an LP can be described as a boundary point between a hyperplane and a polyhedron. If it were now true for the polyhedron that it has a so-called *integer hull*, i.e., that an edge or vertex is integer, then x^* with

$P \cap H = \{x^*\}$ would automatically be integer. So we could neglect the integrality constraint of an ILP. This is exactly the advantage of totally unimodular constraint matrices of an ILP. We explain this concept in more detail in the following.

A feasible solution of an ILP or a 0-1-ILP would be an integer vector $x \in \mathbb{Z}^n$ or $x \in \{0, 1\}$, respectively. With that the set of feasible solutions would be given by

$$P_I = \{x : Ax \leq b, x \in \mathbb{Z}^n\} \quad \text{or} \quad P_I = \{x : Ax \leq b, x \in \{0, 1\}^n\}.$$

To connect with LP relaxation of $\max\{c^T : x \in \mathbb{Z}^n \wedge Ax \leq b\}$ with $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}^n$, we define the set of feasible solutions of an ILP as an *integer hull* of a polyhedron P .

Definition 3.6.4. Let $P \subset \mathbb{R}^n$. A *convex hull* $\text{conv}(P)$ of P is given by

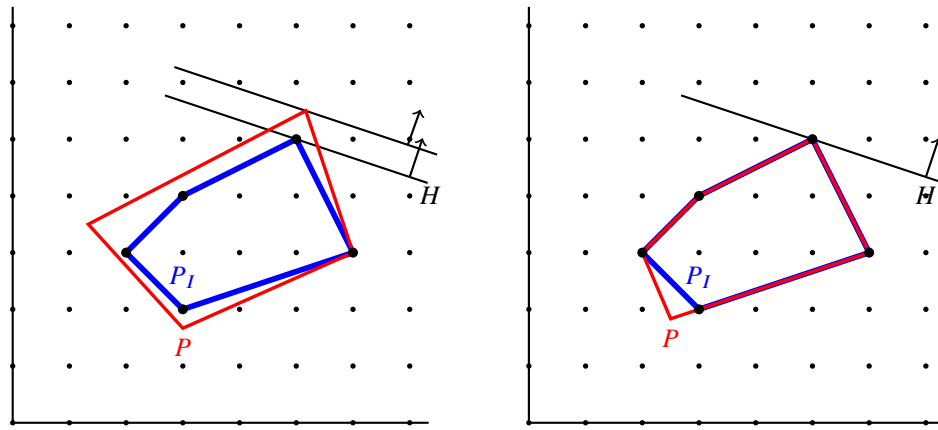
$$\text{conv}(P) := \bigcap \{C : P \subset C \subset \mathbb{R}^n \wedge C \text{ is convex}\}.$$

A convex hull

$$P_I := \{x : Ax \leq b\}_I = \text{conv}(\mathbb{Z}^n \cap P)$$

of integer vectors $A \in \mathbb{Z}^{n \times m}$, $b \in \mathbb{Z}^m$, and $x \in \mathbb{Z}^n$ of the polyhedron P is called *integer hull*.

The following figure 3.14 shows the relation of a bounded polyhedron and its convex hull, and is adopted from the work of Korte and Vygen [KV06].



(a) Example of a polyhedron different from the integer hull. (b) Example of an integer polyhedron.

Figure 3.14: Example of relation of a bounded polyhedron and its convex hull.

In subfigure 3.14a the polyhedron P and its integer hull P_I are different from each other. And, thus, the boundary point of the hyperplane with P and of the

3 Definitions and Preliminaries

hyperplane with P_I are not the same ($P \cap H \neq P_I \cap H$). But as we can see in subfigure [3.14b](#), if the polyhedron is *integer*, that is, the polyhedron is equal the convex hull, $P = P_I$, then the boundary point of P and P_I with the hyperplane is the same, i.e. $P \cap H = P_I \cap H = \{x^*\}$.

As we focus in this thesis on the EBRs problem and its complexity theoretical study, we do not discuss the relation of P and P_I in detail, but refer the interested reader to the work of Korte and Vygen [\[KV06\]](#), Schrijver [\[Sch03\]](#), or Gritzmann [\[Gri13\]](#).

We determine when a polyhedron is integer by the following definition, which is adapted from Korte and Vygen [\[KV06\]](#).

Definition 3.6.5. A polyhedron P is integer, if $P = P_I$.

As we have already discussed, if $P = P_I$ holds, each vertex of the polyhedron P is an integer vector and therefore the solution of an LP problem $\{c^T x : Ax \leq b\}$ is integer if $c \in \mathbb{Z}^n$. This was first proven by a theorem of Hoffman [\[Hof74\]](#), as well as Edmonds and Giles [\[EG77\]](#).

We want to use this property of integer polyhedra and see in subsection [4.3.1.1](#) that there is a special case of the EBRs problem for which $P = P_I$, where P is the polyhedron of the LP relaxation of the EBRs problem. Therefore, we use a theorem of Hoffman and Kruskal (see theorem [3.6.14](#)) which states that in case of integer b the solution of the polyhedron $\{x : Ax \leq b, x \geq 0\}$ is integral if and only if the constraint matrix $A \in \mathbb{Z}^{n \times m}$ is *total unimodularity*.

In the next part of this section, we define totally unimodular matrices and describe methods for proving total unimodularity. A matrix A is total unimodular if it satisfies the following definition.

Definition 3.6.6. A matrix A is *totally unimodular (TU)* if for each quadratic submatrix $U \subseteq A$ the determinant $\det(U) \in \{-1, 0, 1\}$.

Showing that a matrix is totally unimodular by checking for each submatrix $U \subseteq A$ whether its determinant is equal to -1 , 0 , or 1 is quite computationally intensive. Therefore, we introduce theorems, lemmas, and remarks with which the proof of total unimodularity is easier. They give an overview of the conditions under which the property of total unimodularity is preserved and how we can show that matrix $A \in \mathbb{Z}^{m \times n}$ is totally unimodular. Moreover, they support the proof of theorem [3.6.14](#).

The following remark results from the calculation rules for determinants [\[MPS14, Rad66\]](#).

Remark 3.6.7. Let $A \in \{-1, 0, 1\}^{m \times n}$ be TU. Then every matrix \hat{A} arising out of the the following transformations of A remains TU.

- a) Deleting or adding a row or column.
- b) Multiplying a row or column with -1 .
- c) Permuting the rows and columns.

The following matrices remain TU under the following transformations.

Lemma 3.6.8. *Let $A \in \mathbb{Z}^{m \times n}$. With \hat{A} one of the following matrices is denoted*

$$A^T, -A, (A, E_m), \begin{pmatrix} A \\ -A \end{pmatrix}, \begin{pmatrix} A \\ E_n \end{pmatrix}.$$

Then A is TU if and only if \hat{A} is TU.

We define E_m and E_n as unit matrices of $\text{rank}(E_m) = m$ and $\text{rank}(E_n) = n$, respectively.

Proof. a) $\hat{A} = A^T$:

This results from the calculation rules for determinants. If for each quadratic submatrix $U \subseteq A$ it holds $\det(U) \in \{-1, 0, 1\}$ than also for each quadratic submatrix $U^T \subseteq A^T$, as $\det(U) = \det(U^T)$. And therefore, $\det(A) = \det(A^T)$.

b) $\hat{A} = -A$:

This results also directly from the calculations rules for determinants.

c) $\hat{A} = (A, E_m)$

From the calculation rules for determinants we know further that each permutation of two rows or columns of a matrix multiplies the determinant with -1 . So by only changing the sign of any quadratic submatrix of $C \subseteq (A, E_m)$ we can permute the rows of C such that it takes the form

$$C = \begin{pmatrix} B & 0 \\ D & E_k \end{pmatrix}$$

with $k \leq m$ and B is any quadratic submatrix of A . And with that $\det(C) = \det(B)$.

d) $\hat{A} = \begin{pmatrix} A \\ -A \end{pmatrix}$:

By the calculation rules of determinants we know that by multiplying any row (or column) with -1 also the determinant is multiplied with -1 . And furthermore we know, that duplicating rows (or columns) results in a determinant equal to zero. And by applying both rules it is already shown that $\begin{pmatrix} A \\ -A \end{pmatrix}$ is TU.

e) $\hat{A} = \begin{pmatrix} A \\ E_n \end{pmatrix}$:

In this case, we can compute any square matrix $C \subseteq \begin{pmatrix} A \\ E_n \end{pmatrix}$ by developing the determinant according to the rows of the unit matrix and get $\det(C) \in \{-1, 0, 1\}$.

□

3 Definitions and Preliminaries

The following theorem proven by Heller and Tompkins [HT57] gives a sufficient condition to show that a matrix is totally unimodular. We refer for the proof also to the work of Heller and Tompkins [HT57].

Lemma 3.6.9. *Let A be an $m \times n$ matrix whose rows can be partitioned into two disjoint sets I_1 and I_2 . Then the following four conditions together are sufficient for A to be TU.*

- a) $A \in \{-1, 0, 1\}^{m \times n}$.
- b) Every column of A contains at most two non-zero entries.
- c) If two non-zero entries in a column of A have the same sign, then the rows of one is in I_1 , and the other in I_2 .
- d) If two non-zero entries in a column of A have opposite signs, then the rows of both are in I_1 , or both in I_2 .

We conclude from this lemma the following remark.

Remark 3.6.10. If $A \in \mathbb{Z}^{m \times n}$ is TU, then $A \in \{-1, 0, 1\}^{m \times n}$.

The following theorem gives a very useful criterion with which the total unimodularity of a matrix can be shown.

Theorem 3.6.11. *Let $A = (a_1, \dots, a_m)^T \in \mathbb{Z}^{m \times n}$. A is TU if and only if for each nonempty index set $I \subset [m]$ there exists a partition (I_1, I_2) of I with*

$$\sum_{i \in I_1} a_i - \sum_{i \in I_2} a_i \in \{-1, 0, 1\}^n. \quad (3.7)$$

We refer for the proof of this theorem to the work of Korte and Vygen [KV06], and Ghouila-Houri [GH62].

Finally, before stating and partially proving the theorem of Hoffman and Kruskal, we first define *unimodular* matrices as follows.

Definition 3.6.12. Let $A \in \mathbb{Z}^{n \times n}$ with $|\det(A)| = 1$. Then we call A *unimodular*.

The following remark follows directly from the definition of totally unimodular and unimodular matrices [Gri13].

Remark 3.6.13. Let $A \in \mathbb{Z}^{m \times n}$ be TU and $\text{rank}(A) = n$. Then, A is also unimodular.

Finally, the theorem of Hoffman and Kruskal is as follows.

Theorem 3.6.14 (Hoffman & Kruskal, 1956). *An integer matrix A is TU if and only if the polyhedron $\{x : Ax \leq b, x \geq 0\}$ is integer for each integer vector b .*

For our purposes, “ \Rightarrow ” of the theorem is of relevance. Therefore, we don’t prove “ \Leftarrow ” and instead refer to the work of Korte and Vygen [KV06].

Proof. The proof is adapted from the work of Korte and Vygen [KV06].

“ \Rightarrow ”: Assume that the matrix A is TU. Let $b \in \mathbb{Z}^m$ and let x be any vertex of P .

With A TU we know that $\begin{pmatrix} A \\ E_n \end{pmatrix}$ is TU according to lemma 3.6.8.

With x is a vertex of P , we know with lemma 3.6.3 that x is the solution of $A'x = b'$ for some subsystem $A'x \leq b'$ of $\begin{pmatrix} A \\ -E_n \end{pmatrix} x \leq \begin{pmatrix} b \\ 0 \end{pmatrix}$. Thereby, A' is a non-singular $n \times n$ matrix, i.e., $\text{rank}(A') = n$. With A' being nonsingular and TU, $|\det(A')| = 1$.

By Cramer’s rule, the solution is given by

$$x = (A')^{-1}b' = \frac{(A')^{\text{ad}}}{\det(A')}b'.$$

In a last step, we need to show that $x \in \mathbb{Z}^n$.

Let $(a'_{i,j})_{i,j \in [n]} = A'$. We define the adjoint of A' as follows

$$(A')^{\text{ad}} = (a'_{i,j})_{i,j \in [n]}^{\text{ad}} \quad \text{with}$$

$$(a'_{i,j})^{\text{ad}} := (-1)^{i+j} \det(a'_{k,l})_{\substack{k \in [n] \setminus \{i\} \\ l \in [n] \setminus \{j\}}} \quad \text{for } i, j \in \{1, \dots, n\}.$$

With $A' \subset \begin{pmatrix} A \\ -E_n \end{pmatrix}$ is TU and by the definition of the adjoint of A' , it is $(A')^{\text{ad}} \in \{-1, 0, 1\}^{n \times n}$. And, therefore, with $|\det(A')| = 1$ and $b' \in \mathbb{Z}^n$ it follows that

$$x = \frac{(A')^{\text{ad}}}{\det(A')}b' \in \mathbb{Z}^n.$$

Thus x is integer. □

The following theorem of Schrijver gives us the important insight that the total unimodularity of a matrix can be shown in polynomial time. And thus, in case of a totally unimodular constraint matrix, we can indeed find a solution to the associated problem in polynomial time. We refer for the proof to Schrijver [Sch03].

Theorem 3.6.15. *Let $A \in \mathbb{Z}^{n \times m}$. Showing that A is totally unimodular can be done in polynomial time.*

3 Definitions and Preliminaries

4 Expert-based recommendation systems

In chapter 4 of this thesis, we define the *expert-based recommendation system* (EBRS problem) and discuss its complexity according to its running time both in general and for different cases.

The EBRS problem is an optimization problem that optimizes a claims processing of an insurance company. We have already discussed the importance of improving the claims processing in section 2.1. To optimize a claims processing, we must first define the *quality of the process*. We address this issue in section 4.1 and explain the approach we used to select appropriate recommendations.

In section 4.2, we formulate the EBRS problem as an optimization problem, but also give the decision version of the problem. Finally, we prove that the EBRS problem is NP-complete.

In section 4.3 we discuss different cases of the EBRS-problem in terms of their complexity. We define the different cases by changing the constraints of the problem, neglecting some of them or restricting them further. Depending on the change of the constraint, the complexity of the case changes. One main goal of this thesis is to find cases of the EBRS-problem for which a solution can be found in polynomial time.

4.1 Expert-based quality of claims processing and identification of recommendations

In order to improve the quality of claims processing, we first need to clarify what we mean by *quality* in this context—and how we assess it. That is, in order to be able to set up an optimization problem that improves quality, we must first of all find very clear characteristics with which we can deterministically measure the quality of a claims processing.

Furthermore, we want to optimize the quality by incorporating the different perspectives of people—from now on called *experts* regardless of their specific role—involved in the claims processing. The inclusion of different perspectives create the term

expert-based by asking the experts about their satisfaction with specifically selected claims processing.

Each participant in such a claims processing naturally evaluates it differently—and based on different criteria. Therefore, different characteristics of the processing of claims are important to each participant—depending on their point of view—which makes this consideration important to improve quality. Our goal is to derive recommendations for an insurance company from these characteristics.

Of course, it is also possible that two or more respondents make the same recommendation. Yet, the *weighting* of their recommendation may be different for each expert. Recognizing such different weighting of recommendations will be discussed in subsection [4.2.1.1](#).

In a first step, we define the quality of a specific processing of claims taking into account the different perspectives, opinions and experiences of the experts involved. In this thesis, we assume that a priori there exists no given variable in a dataset containing information about such *quality*. Instead, we need to apply a statistical method designed to “measure how respondents trade-off various alternatives and their respective attribute levels” [SM18, p. 3]—the so-called *conjoint analysis* (as described in detail in subsection [3.1.1](#)). The basic idea of the conjoint analysis is that an expert evaluates the whole claim in terms of her or his satisfaction with it, rather than evaluating the attributes of a claim separately.

To specify the quality of the claims processing the experts evaluate prototypical claims. Based on this evaluation it is possible to calculate the so-called part-worth utilities of the attributes and attributes levels, respectively. Furthermore, as the conjoint analysis is a decompositional approach, we use this part-worth utilities to estimate the quality of each claims processing in a dataset.

We describe the procedure of the conjoint analysis to add the information about the quality of the claims processing to the dataset in subsection [4.1.1](#). Furthermore, we discuss the corresponding experimental design to create the so-called *stimuli* for the questionnaire as well as the evaluation of the stimuli.

Thus, we know that the quality depends on the evaluation of the attributes and the corresponding levels. One plausible way to improve quality is to change the levels of the attributes with regard to the experts’ estimations. So by evaluating the prototypical claims, the experts also indirectly created a repertoire for finding recommendations. We describe this in detail in the section [4.1.2](#).

4.1.1 Procedure to define the quality of claims processing

Assessing the quality of a claims processing or satisfaction with the processing of a claim is very individual—and often unconscious criteria are the drivers for experts’ satisfaction. To detect such unconscious drivers the conjoint analysis

is a valid method due to the holistic assessment of the claims processing (see subsection 2.1.2).

In general, the procedure of the conjoint analysis can be divided in seven steps—as described in subsection 3.1.1, as well as by Gustafsson and colleagues [GHH07].

- a) Selection of the preference function,
- b) Selection of representative stimuli,
 - ba) Selection of data collection method,
 - bb) Selection of data collection design,
- c) Presentation of the stimuli,
- d) Selection of data collection procedure,
- e) Evaluation of the stimuli,
- f) Calculation of the part-worth utilities.

We do not focus on a detailed description of each step of the conjoint analysis procedure in this thesis. Instead, we focus on the problem that typical experimental designs are not suitable for finding appropriate stimuli for evaluation in our context because of the large number of attributes in insurance datasets. It is therefore a challenge to map the entire space of attributes and their levels through the stimuli.

Thus, an appropriate selection of attributes and their levels can be challenging, and may result in a further step of conjoint analysis [BEPW18, Vri95]—relevant to reduce the number of attributes that should be considered for generating the stimuli per expert.

Another challenge is that typical experimental designs such as Addelman’s orthogonal design [Add62], which maps the entire space of attributes, generate stimuli that do not exist in reality. Even more, by applying such standard methods there would exist stimuli which are inconsistent. This is due to the fact, that typical experimental designs assume the independence of the attributes and their levels [BEPW18]. Yet, in case of insurance datasets some variables are highly dependent. Thus, we developed a method with which real (and not fictional) damage cases are selected for evaluation. More specifically, we are looking for prototypical claims that best represent the entire space of damage cases. Therefore, we applied the so-called *k-medoids clustering* (see subsection 3.1.2), which allows us to find representative observations of the dataset that we can utilize as stimuli.

Before describing the individual steps for creating suitable stimuli, we would like to recall the *ex post* consideration of the claims. Of course, an expert can only evaluate the whole claims processing once it has been completed. Therefore, we only consider closed claims in the following.

We describe the implementation of quality into the dataset step by step in the following subsections. First, we describe the selection of the representative stimuli by applying the k -medoids method and all related intermediate steps. In a next step, we briefly describe the method by which the stimuli are evaluated by the experts. In the last subsection, we finally describe how the quality of the claims processing is estimated.

4.1.1.1 Selection of representative stimuli

Considering typical data collection methods (e.g., the profile or two-factor method), the number of stimuli to be rated by the experts fastly increases. For example, in case of the profile method and a complete design, seven attributes with two attribute levels each would generate 128 stimuli. The evaluation of such a large amount of stimuli is difficult—even for an expert—and usually introduces enormous cognitive load that may extend his or her cognitive capacity. In case of insurance datasets even more variables must be considered, leading to the necessity of a reduced design, i.e., a subset of all possible stimuli.

Another point to consider is reducing information overload by presenting too many attributes in the questionnaire. In the literature different amounts of attributes are recommended [GHH07]. Green and Shrinivasan suggest that when using the profile method, no more than six attributes should be used simultaneously in a questionnaire [GS78], while Malhotra recommends that respondents be able to process ten attributes without excessive strain [Mal82].

We further need to consider that there exist interaction effects between different variables of the dataset, and some constellations do not represent real claims as already described in subsection 2.1.3. However, standard experimental design approaches assume that the variables in the dataset are independent, and, furthermore, do not take into account that not all constellations do exist.

We choose to face these difficulties by using *prototypical claims* as stimuli, which we derive from the dataset by applying k -medoids clusterings. Furthermore, to reduce the information overload we decided that the experts evaluate not only real damage cases but also—in the best case—evaluate this prototypical cases in the interface they are familiar with. With the help of the interface we generate an *authentic assessment situation*. By using the familiar interface we assume that the experts know the positions of the attributes that are important to them—on an intuitive and completely automated level of implicit action schemata [Swe10]. To further strengthen the focus on the important attributes, we implemented a preceding questionnaire asking the experts to rate each variable in the dataset according to its importance. This approach is consistent with Green and Srinivasan’s recommendation to focus on attributes that are important to the respective person, as this allows them to process

a greater number of attributes and their manifestations [GHH07]. We will describe this questionnaire in detail later in this section.

Presenting the stimuli on the interface familiar to the experts has another advantage. It allows us to avoid that the experts are influenced by other factors of the survey design. A representation that is unfamiliar to the experts may, in some circumstances, cause the experts to focus on features that they might otherwise *not* consider. A possible reason could be, for example, a more prominent position than in the interface used.

In addition, using the interface that most of the experts work with on a daily basis can help the experts to indicate satisfaction with the claims processing based more on expert intuition feeling.

As a last advantage to mention, we assume that the familiar interface helps the experts to evaluate more stimuli without cognitive overload, since they do not have to get used to a different interface, i.e., we utilize the company's original interface to *reduce extraneous cognitive load* during the assessment situation [Swe10].

A disadvantage of presenting the stimuli in the familiar interface is that other attributes may be included in the evaluation of the damage cases than those that were rated as important in the first questionnaire. However, the result of the conjoint analysis—the calculation of the part-worth utilities for each considered attribute and its level to estimate the quality of a future claim—is not influenced, because all relevant attributes are considered according to the decompositional approach. However, it affects the selection of stimuli that are based on the previous questionnaire. If the attributes do not match between the first questionnaire and those considered in the familiar interface, this may have an impact on the second questionnaire. Namely, the expert may not be presented with all the attributes that are actually important to her or him. For example, if an expert rates glass damage and corresponding processing speed as important in the previous questionnaire, but not workshop binding, he or she will need to rate more stimuli focusing on glass and speed attributes. If, during the evaluation process of the second questionnaire in the familiar interface, she or he recognizes that workshop binding is important to her or him, then there may be too few or no such cases in the stimuli presented.

To find prototypical claims we apply the *k-medoids* clustering, a special clustering method that searches for representative observations of the dataset—the so called *medoids*. In our case the medoids are real damage cases that represent “the structure of the data” [MRS⁺22, p. 47].

As with all clustering methods, the goal of *k-medoids* is to find *subgroups*, or *clusters*, in a dataset [JWHT13]. In case of *k-medoids* the cluster algorithm works by searching *k* observations, for which the distances or dissimilarities within a cluster are as small or similar as possible and the distances or dissimilarities to other clusters are as different as possible. The “center” of each cluster is in case of this clustering method the so-called *medoid*—a real and non-fictional claim. For

more details we refer to chapter [3.1.2](#). The observations are assigned to the cluster that is closest or most similar to them. One question we want to address now is as follows.

How does this help us find representative stimuli?

With the clustering method we want to identify subgroups of typical damage cases. With the medoid, we have a typical claim representing each of the k clusters—and, therefore, each medoid is used as a stimulus.

One challenge is to choose an appropriate number of $k \in \mathbb{N}$ so that the stimuli represent the full range of claims in the dataset as well as possible. In doing so, we need to consider the following facts.

First, we need to check the validation criteria for the k -medoids method. The quality with which the medoids reproduce the damage cases that arise in reality depends, among other things, on the choice of k . We choose to consider the average silhouette width for different k to select an appropriate $k \in \mathbb{N}$.

Since there are many attributes in insurance datasets, it would normally be a good choice to select $k > 100$ stimuli to represent the entire dataset. But as discussed in the beginning of this chapter, it is not purposeful to present such a large number of stimuli to the experts for evaluation, as it would simply overwhelm them. Therefore, we need to balance between a good representation of the dataset and the limits based on the cognitive capacity of the working memory of the experts [[Swe10](#)].

Another point to consider refers to the potential existence of sparsely populated and binary asymmetric variables in the dataset that would not be present in the questionnaire if k is chosen too small. Since new claims may appear that have these missing attribute levels (i.e., those attribute levels not considered in the questionnaire), we should ensure that these attribute levels are also valued, as otherwise, we will not be able to estimate a part-worth utility for these attribute levels.

To take into account all facts for selecting an appropriate number of medoids, we have on the one hand the already mentioned preceding questionnaire, with which we can reduce the attributes to a subset of attributes, which are important based on the rating of respective expert. We describe this questionnaire in detail in subsection [4.1.1.1.1](#). On the other hand, to account for sparse levels of the attributes that are important to the experts according to the preceding questionnaire, we decided to select only half of the stimuli using the k -medoids method: We selected k medoids as stimuli—and another k stimuli were sampled from each cluster under the condition that the missing attribute level is included in the sample (assuming that we need $2k$ of stimuli). More specifically, we sample another k stimuli from the k clusters, but not completely at random, but subject to the condition that a missing attribute level is included. To increase the chances of drawing a missing attribute

level and to account for cluster structure, we draw a sample from the clusters where the missing attribute level occurs more frequent.

In summary, we select two observations per cluster. The first, by applying the k -medoids method and the second by selecting another sample from each cluster under the condition that sparse attribute levels are included. We call these method *conditional sampling from k clusters*. We discuss the selection of medoids in more detail later in this chapter.

In the next paragraphs we describe each step for selecting representative stimuli in more detail.

4.1.1.1.1 Expert-based selection of important attributes The goal is to find a representative sample of the insurance dataset for which the experts have to rate the quality of the claims processing. As already discussed usual questionnaire design algorithms cannot be applied to our dataset so that we decided to reduce the dataset for evaluation by finding representative medoids. But also with the k -medoid method we face the problem, that the number of attributes is too large, and the number of medoids of an adequate and validated k -medoids clustering is higher than an expert could possibly process. In our case the number of medoids would be higher than 100, i.e., $k > 100$. Therefore, we further reduce the number of variables by simply asking each expert, which variables have the most impact on the claims processing based on her or his opinion.

In contrast to decompositional approaches, we evaluate the attributes directly with the preceding questionnaire. The direct measurement of the importance of each variable is comparable to asking experts for *importance weights* as used in *compositional preference measurement* [SM18]. Overall, we apply a so-called *hybrid approach* combining the advantages of both—decompositional and compositional—methods [GHH07, SM18], utilizing specifically the advantage of the compositional approach to handle a large number of attributes [GHH07].

In contrast to the evaluation of the importance weights in a range of $[0, 1]$, we use the preceding questionnaire more as a selection criteria to choose those attributes that are important to the expert. Furthermore, with this questionnaire we gain first insights into the possible different perspectives of the experts. That is, with this method we personalize the second questionnaire based on this first selection of important attributes.

To find the most important variables, we ask each expert to rate each variable according to its impact on the quality of the claims processing in the dataset. For this purpose, for each variable in the dataset, each expert indicates whether the impact is *very low*, *low*, *high* or *very high*. Here, we deliberately chose an even number of choices. Thus, the experts are forced to choose between a (rather) high or a (rather) low influence of the variable on the quality. Depending on the number

of attributes in the insurance dataset under consideration, we then consider for the second questionnaire those variables that have a high and/or very high influence on quality of the claims processing (see fig. 4.1).

der Bundeswehr
Universität München

Expertenbefragung: Schadenabwicklung in Kraft-Haft und Kraft-Kasko

Information zum Schadenprozess	Kraft-Haft				Kraft-Kasko			
	irrelevant	eher geringe Relevanz	eher größere Relevanz	große Relevanz	irrelevant	eher geringe Relevanz	eher größere Relevanz	große Relevanz
Dauer der Abwicklung (für den Fall eines bereits abgeschlossenen oder stornierten Schadenfalles)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Status der Bearbeitung (z.B. geschlossen, offen, storniert, ...)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Ist es ein Geschäftsjahres- oder ein Vorjahresschaden? (GJ/VJ).	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Summe der Zahlungen des letzten Stichtags (Stand zum letzten Monatsende, Vorgängereintrag; 0, wenn kein Vorgängereintrag existent)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 4.1: Sample excerpt of the first questionnaire.

4.1.1.1.2 Conditional sampling from k clusters Based on the chosen-to-be-important variables of the preceding questionnaire, we will find k representative observations in the dataset per expert using the k -medoids algorithm. As discussed at the beginning of this chapter, we decided to select only half of the medoids as stimuli and select the other half from the clusters taking into account the sparse attribute levels that may not yet be included in the second questionnaire.

To apply k -medoids, we have to calculate a metric in a first step. In most insurance datasets there exist mixed types of variables. Therefore, typical clustering methods such as the k -means algorithm, which uses Euclidean distance as a metric, cannot be applied to this type of datasets. This is another reason why we need a *mixed clustering* method for such datasets besides the arguments given in subsection 2.1.2

The k -medoids algorithm can use a metric that allows us to measure whether observations are very similar or different from each other based on dissimilarities between them. We use, therefore, a generalization of *Gower's formula*. The metric is also called *Gower's coefficient* [Gow71] and is described in more detail in subsection 3.1.2.1. The range of Gower's coefficient is defined by $d_{ii'} := d(x_i, x_{i'}) \in [0, 1]$ for all observations x_i and $x_{i'}$ with $i, i' \in [n]$. Gower's coefficient identifies with $d_{ii'} = 0$ identical observations x_i and $x_{i'}$ and with $d_{ii'} = 1$ maximal dissimilar observations x_i and $x_{i'}$.

With Gower's coefficient we can now apply the k -medoids algorithm to partition the data into k clusters around medoids. To do this, we use a *partitioning around medoids* or the *PAM-algorithm* for short, which is described in detail in subsection 3.1.2 [MRS⁺22, KR05, HTF09]. In summary, the PAM-algorithm consists of two phases.

In the first phase—the so called *build phase*— k medoids are selected and each observation is assigned to its nearest medoid due to the calculated metric—here Gower’s coefficient. In the second phase—the so called *swap phase*—a new medoid is searched for each cluster, for which the average dissimilarity coefficient decreases the most. This two phases are repeated until the algorithm converges, i.e., no medoid is exchanged in any cluster.

To find an appropriate number $k \in \mathbb{N}$ and thus k final medoids that represent the dataset as well as possible, we have to trade off between a good segregation of the clusters, the right number of stimuli to match the cognitive capacity of the experts (i.e., preventing cognitive overload), and a representation of all attribute levels that are sparsely filled or asymmetric binary. Therefore, we considered the following validation criteria.

In a first step, we looked at some cluster informations. We checked among other things the cardinality of the k clusters, to see if there exists outliers or very small clusters compared with other cluster sizes. Furthermore, we looked at the maximal and average dissimilarity between the observations and the medoids within each cluster. Since $d_{ii'} \in [0, 1]$ for all $i, i' \in [n]$, where $d_{ii'} = 0$ represents identical observations x_i and $x_{i'}$, lower average dissimilarity indicates better cluster fit. The same is true for the so-called *diameter*, a coefficient that measures the maximal dissimilarity between two observations of the cluster. A last check of the cluster information is the so-called *separation*, i.e., the minimal dissimilarity between an observation of the cluster and an observation of another cluster. Here, of course, a high value is preferred.

Another validation check is done by looking at the so-called *silhouette width*. For a detailed description we refer to subsection [3.1.2.2](#). Briefly, considering the silhouette width for each observation $x_i, i \in [n]$, of the dataset, we measure how similar it is to its own cluster (*cohesion*) compared to other clusters (*separation*). Thus, it shows how well the individual observations were classified. A high value indicates good clustering, while a low value indicates that the clustering may have too many or too few clusters. To evaluate the goodness of clustering based on the silhouette width, we considered the *average silhouette width* and the silhouette plot.

In our case, we obtain the best average silhouette width for $2k \geq 100$ medoids, even for the dataset that was restricted and personalized based on the first questionnaire. But rating more than 100 stimuli is a very difficult task and would simply overwhelm the experts. If we remember the goal of this cluster analysis, we do not want to find a perfect clustering, but a good representation of the dataset, so that we can estimate the part-worth utility for each attribute and corresponding attribute level. Thus, it is not required, that the valuation criteria are “best”. They should just give a directive for the goodness of the segregation of the clusters. To comply with this directives we decided to set the number of stimuli to a maximum in the sense of meeting the cognitive load of the experts and set $2k = 50$.

Because of this decision we face the problem that not all attribute levels of the sparse attributes are included in the second questionnaire. For our case, we decided to select $k = 25$ medoids and take additional stimuli from each corresponding $k = 25$ cluster. In this sample, we must ensure that the claims submitted in the second questionnaire are as diverse as possible. We call this method *conditional sampling from clusters*.


Our applied approach clearly has advantages, which we have already described. But it also has disadvantages, which are discussed below.

To reduce the number of attributes of large insurance datasets, we implemented a preceding questionnaire and personalized the stimuli selection. However, the advantage of reducing the number of attributes also has the disadvantage that this restriction means that we no longer map all possible claims and, in the worst case, focus only on a small subset of claims. Furthermore, in general, due to the size of the dataset and the fact that the *full profile* method is not used, there is always the risk that some case constellations are not recorded and thus an interaction effect cannot be estimated.

4.1.1.2 Evaluation of stimuli and estimation of quality

Above, we presented arguments that led to the decision that 50 prototypical claims had to be evaluated by the experts. In this subsection, we now describe the assessment process in detail, i.e., how these prototypical claims are presented to the experts, and how they are asked about their satisfaction with the processing of the claim. Based on this assessment, we then estimate the quality of the claims processing. In order to get an assessment of the experts that is as realistic as possible, we presented these prototypical claims in the familiar interface the experts are used to, or in other words, we simulated a real claims processing. We discuss the advantages of the utilization of the interface in subsection [4.1.1.1](#).

For the evaluation of the stimuli, we present each expert with her or his personalized selection of claims. In addition, we only consider closed claims processings. An excerpt of the questionnaire used is given in figure [4.2](#).

der Bundeswehr
Universität  München

Expertenbefragung II: Experte 8

Kasko (Expertengruppe 2)
Schadenfälle 1–25

Zu wie viel Prozent ist Ihrer persönlichen Meinung nach der Schadenverlauf zufriedenstellend?

Nr.	Geschäftsjahr	Schadensnr.	Vertrag	M.Datum	S.Datum	0%–20%	10%–30%	20%–40%	30%–50%	40%–60%	50%–70%	60%–80%	70%–90%	80%–100%
1						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 4.2: Sample excerpt of the second questionnaire.

4.1 Expert-based quality of claims processing and recommendations

It is challenging to rate claims processing by directly indicating the percentage of satisfaction on a scale of 0 % to 100 %, so surveyors are asked to state their satisfaction in an interval. That is, we ask experts to indicate their satisfaction on an interval scale, with overlapping intervals. For example, suppose that an expert is very satisfied with a claims processing. Instead of rating satisfaction at 95 % (which the expert may not be able to specify as accurately), the expert may indicate a trend by rating satisfaction at 80-100 %. The overlapping interval scales and the accompanying indication of a tendency instead of a direct rating can be helpful for the expert especially if she or he is undecided between two intervals. The intervals are defined as in the above right corner of figure [4.2](#).

To better understand the advantage of overlapping intervals, suppose that an expert is undecided whether the submitted claim has been processed to her or his full satisfaction, or whether she or he is only very satisfied. In this case the expert could choose the interval 80 % to 100 %. In another case, an expert again rates the processing of a claim as very good, but tends to say that not everything went quite optimal. With the overlapping intervals, the expert now has the opportunity to weigh up. In doing so, the expert's thoughts might be as follows. If she or he chooses the interval 80 % to 100 %, it *contains* the best possible score—the 100 %. If this does not feel intuitively right to the expert, he or she would take the interval 70 % to 90 %, which contains at least a satisfaction of 90 %.

We apply multivariate regression to estimate a part-worth utility for each attribute level within a model defined by the expert. Therefore, we transform each interval of the overlapping interval scale to its mean value. For example, we define the interval 0 % to 20 % as 10 %—or 0.1, respectively.

Since this work mainly focuses on the complexity-theoretic investigation of the expert-based recommendation system defined later, we refer the interested reader to the work of Backhaus and colleagues [\[BEPW18\]](#), Gustafsson and colleagues [\[GHH07\]](#), and Steiner and Meißner [\[SM18\]](#) for more information on part-worth utilities. For more details on multivariate regression, we recommend Hastie and colleagues [\[HTF09\]](#), Fahrmeier and colleagues [\[FKLM13\]](#), or again Backhaus and colleagues [\[BEPW18\]](#).

Finally, by applying the part-worth utilities to the rest of the dataset, we can estimate the corresponding quality of the claims processing for each completed claim. Since we used multivariate regression to estimate the part-worth utilities, the calculation of quality is simply done by substituting the attribute values of an unknown claim into the regression equation. With that we have added the missing target variable of the optimization in the whole dataset—the quality of the claims processing.

In the further course of this thesis, we now turn our attention to the question of which recommendations can have a positive influence on the claims processing. And with that we change the perspective—from *ex post* to *ex ante*.

4.1.2 Choice of recommendations

In the last subsection, we focused on implementing the quality of the claims processing in the dataset. We therefore considered the processing *ex post* and used a customized conjoint analysis to add a variable with quality information to the insurance dataset.

Experts were surveyed as part of the conjoint analysis. They were asked to indicate attributes of claims processing that were important to them and to rate each attribute according to its (positive or negative) influence on claims processing. Based on these evaluated attributes, we were able to calculate the quality of the claims processing. For the calculation we used multivariate regression analysis. The recommendations we intend to use to optimize future claims processings are now more or less derived from these attributes we asked about in the *ex post* observation.

Yet, not all attributes are suitable recommendations. Or they should only be pronounced in an adapted form. One reason for this is that some recommendations are not implementable in a particular situation. An example of this would be that we can only refer to a workshop affiliation if this is contractually agreed.

In order to make meaningful recommendations, the recommendation system should take such conditions into account. Therefore, we need to identify all those recommendations that should only be made under certain conditions.

In the following, we list other possible reasons for making a recommendation conditionally: One reason could be economic restrictions on making a recommendation that must be taken into account. For example, consider recommending a fraud audit. This may involve high costs, and therefore may only be advisable in certain situations. Making a recommendation also may mean consuming human resources, which are limited. Thus, it may make sense to identify those recommendations that cannot be considered (to full extent) due to the current staffing situation.

Therefore, in general, it is highly recommended to discuss the recommendations and their conditions in close consultation with the experts. This ensures that the recommendation system makes sensible and implementable recommendations.

In summary, the selection of recommendations is highly dependent on the given situation and, therefore, the involvement of experts in the selection process is of great importance. During such discussions, we not only gain insights into which attributes are suitable as recommendations, but also whether or to what extent they can be realized. It is important to note that the goal is not to reduce the number of recommendations, but to identify all potential and realizable recommendations.

4.2 Modelling the EBRs problem and classification of its complexity

In the latter section, we defined the quality of the claims processing with the help of appropriate expert interviews and expanded the dataset to include this information. We also identified recommendations to improve a future claims processing with respect to the defined quality.

Up to now, we have considered each expert and the corresponding model separately. However, the goal is of course to optimize a claims processing considering all expert opinions with regard to the quality of processing. Therefore, in a first step, we aggregate all individual expert models. In this way, we obtain a pool of recommendations that take into account all the different perspectives, and that we use to provide optimal expert-based recommendations to improve the expected quality of claims processing. Here, expert-based actually means that all expert opinions are included. We call the resulting optimization problem an expert-based recommendation system (EBRS), which is formulated in subsection [4.2.1](#).

The main goal of this thesis is to study the time complexity of the EBRs-problem, i.e., we are interested in the time taken by an algorithm in general to solve the EBRs-problem. For this reason, we consider the decision version of the EBRs-problem, which we formulate in subsection [4.2.2](#).

Finally, in chapter [4.2.3](#) we prove that the EBRs problem is NP-complete. This means that the problem cannot be solved in polynomial time, but given a solution, we can verify in polynomial time if it is a solution to the EBRs problem.

4.2.1 Formulation of the EBRs problem

We have discussed the motivation, the advantages, and the positive effects of the expert-based recommendation system in detail in subsection [2.1.3](#). In addition, we gave an application-oriented formulation (see remark [2.1.1](#)). Now we will specify this formulation, give an overview of all required parameters, and define the optimization problem including the definition of the constraints.

Therefore, we briefly summarize the results of the last chapters.

Several experts are interviewed and the interview of each expert generates a separate model. In each model, the expected quality is predicted by finding prototypical classes of claims based on the attributes available at the beginning of the claims processing. Therefore, we used innovative algorithms of convex optimization—see subsection [2.1.3](#) and the work of Brieden and Gritzmann [\[BG20\]](#). Furthermore we identified those attributes, which define the quality of claims processing per model by asking the experts. These attributes are interpreted as recommendations

to improve the claims processing (see subsection 4.1.2). To make meaningful recommendations, it might be necessary to restrict the set of recommendations per expert and per prototypical class of claim. For example, a glass claim can be processed much faster than a collision claim. This should be taken into account when making a recommendation on processing speed.

Every single model estimates the expected quality of the corresponding expert. And therefore, for every model we define a *processing index* PI_j , $j = 1, \dots, m$, where $m \in \mathbb{N}$ is the total number of models considered. We use the processing index to identify the recommendations and parameters of each model to generate the quality of the claims processing. Therefore, PI_j denotes the model of expert j in the following. Before defining the EBRs problem, we introduce the input variables and parameters of the problem.

By R_i , $i = 1, \dots, n$, we denote the i -th recommendation and $n \in \mathbb{N}$ is the total number of overall recommendations. $S := \{R_1, \dots, R_n\}$ defines the set of recommendations in total independent of the considered model and thus, $n := |S| \in \mathbb{N}$. Each model PI_j consists of a set of recommendations $S_j \subseteq S \forall j \in [m]$. Of course, it is possible that a recommendation is contained in several models. This is the case, when some experts value the claims processing with the same attribute(s). Therefore, it holds that $S = S_1 \cup \dots \cup S_m$.

With γ_i , $i \in [n]$, we denote the parameters of each recommendation R_i . The parameters γ_i , $i \in [n]$ are the weighting of each recommendation and measure the impact of the recommendation R_i on the expected quality, when it is recommended. The weighting of each recommendation is derived from the part-worth utilities, with which we assign quality to the dataset per expert. The weighting can have a positive as well as a negative impact.

Finally, we define a so-called *activation variable* $x_i \in \{0, 1\}$ for every recommendation R_i , $i \in [n]$, by

$$x_i := \begin{cases} 1 & \text{if recommendation } R_i \text{ is activated, and} \\ 0 & \text{if recommendation } R_i \text{ is not activated.} \end{cases}$$

As mentioned earlier, for the optimization we aggregate the different models PI_j so that the *aggregated expected quality* PI is improved. To aggregate the single models to an overall model, different methods exist, e.g., the *additive* or *multiplicative method*. In this thesis we choose the additive method for two reasons.

- a) Our recommendation system starts with the reporting of the claim. Therefore, we consider the claims processing ex ante and estimate the expected quality for each model PI_j , $j \in [m]$, by the expected value

$$\mathbb{E}[PI_j] = \sum_{R_i \in S_j} \gamma_i x_i$$

for all $i \in [n]$ and $j \in [m]$. With $x_i \in [0, 1]$ this equals the definition of the expected value.

In this case, $x_i \in [0, 1]$ can be interpreted as the probabilities of occurrence for each recommendation R_i in each prototypical class of claim. It can be simply calculated by looking at the frequency of activation of this recommendation.

- b) In the main part of this thesis, we will study the complexity of the EBRS problem. The goal of the EBRS problem is to activate a selection of recommendations from all possible combinations of recommendations that maximizes the expected aggregated quality. As the activation variable is binary, i.e., $x_i \in \{0, 1\}$ for all $i \in [n]$, we formulate the EBRS problem as an integer linear program (ILP) that is generally NP -complete (see subsection [4.2.13](#)).

It should be further noted that the additive model can be transformed into the multiplicative model by, e. g., logarithmizing. However, we will not discuss this further.

As mentioned earlier, it is possible for a recommendation to be contained in two or more models. This also has an impact on the weighting of the corresponding recommendation, because it is not only dependent on the respective rating and the resulting part-worth utility of a single model (or expert). If we activate a recommendation that is included in two or more models, it is obvious that it is also recommended in each of the other models. In case of the additive method, this results in

$$\gamma_i := \sum_{j \in \{j | R_i \in S_j\}} \gamma_{ij}$$

for all $R_i \in S$.

In the remainder of the subsection, we define the EBRS model step by step.

If we consider the last stated arguments for the additive model, we already pointed out the basic idea of the EBRS problem: Maximizing the *aggregated expected quality* by activating an appropriate set of recommendations. The aggregated expected quality is defined by

$$\mathbb{E}[\text{PI}] := \sum_{i=1}^n \gamma_i x_i.$$

The EBRS problem without consideration of further constraints is given by

Program 4.2.1.

$$\begin{array}{ll} \max & \sum_{i=1}^n \gamma_i x_i \\ \text{s.t.} & x_i \in \{0, 1\} \quad \forall i \in [n] \end{array}$$

Before discussing the constraints of the model in detail, we first take a closer look at the weights of the recommendations $\gamma_i, i \in [n]$.

We defined the weighting $\gamma_i, i \in [n]$, as the weighting of each attribute level based on the part-worth utilities. Yet—as mentioned in subsection 2.1.3—another relevant weighting that should be considered is the relevance of the expert models. This allows us to weight a single model or group of models higher than other models, e.g., we could emphasize the customer satisfaction. Thus, the weighting of relevance does not belong to an individual recommendation, but to the model PI_j for all $j \in [m]$. To include such weighting in the parameters γ_i of each model PI_j , we proceed as follows.

We denote with $w_j \in [0, 1]$ the weighting of relevance of each model $PI_j, j \in [m]$, where $w_j = 0$ means that expert j should not be considered in the optimization and $w_j = 1$ stands for an exclusive consideration of PI_j . By including a relevance weighting the definition of the parameters changes to

$$\gamma_i := \sum_{j \in \{j | R_i \in S_j\}} w_j \gamma_{ij} \quad \forall i \in [n].$$

Before defining the constraints in the following subsection, we establish two assumptions for the optimization problem.

Remark 4.2.2. Without loss of generality there exist no conflicts and dependencies between the recommendations within a model $PI_j, j \in [m]$.

This is not a restriction of the EBRs problem, because in case of conflicts or dependencies within a model we have the possibility to split the model into further models without influencing the result of the optimization problem. However, excluding these cases from the beginning has the advantage that we do not need to consider them in the further process of the thesis.

Remark 4.2.3. Without loss of generality, there exists no model PI_j with $S_j = \emptyset$ for all $j \in [m]$.

With that we exclude all models $PI_j, j = 1, \dots, m$, for which no recommendation exists, i.e., $S_j = \emptyset$. Again, this is not a restriction on the EBRs problem, because w.l.o.g. we can simply delete empty models.

4.2.1.1 Definition of the constraints

The goal of the EBRs problem is to maximize the aggregated expected quality PI of the claims processing by making appropriate recommendations. Depending on the situation in which the optimization algorithm is to be applied, various constraints must be taken into account. In this subsection, we will discuss these

various constraints in general and illustrate them with appropriate examples. Since these are general constraints, we make no claim to completeness.

For the first constraint, we recognize that a clerk can only process a certain number of recommendations. Therefore, there are cases where the clerk cannot process all n recommendations of the recommendation system. Apart from that, processing all recommendations would not be in the spirit of an efficient claims processing (see subsection 2.1.1). Instead, an efficient and streamlined claims processing can free up a claims handler to focus more on claims and customers that require more attention. For example, a glass damage should require little (or no) attention and effort compared to a collision damage with casualties, as 90 % of the claims handlers time is to look after tragic damage cases and the capacity of the clerk is limited [Gen22].

By limiting the number of activated recommendations (especially for less serious claims) in consultation with the insurer's expert, we reduce the workload per claim of the claims handler while still ensuring that the parties involved are satisfied with the claims processing. Therefore, we define with $u \in \mathbb{N}_0$ the *maximum number of activated recommendations* per claim.

In some cases it might be meaningful to implement a lower bound to the activation of the recommendations, e.g., to guarantee a predefined quality target. Therefore, we define with $l \in \mathbb{N}_0$ the *minimum number of activated recommendations* per claim.

Defining the lower and upper bounds, we formulate the first constraint of the EBRS problem by

Constraint 4.2.4.

$$l \leq \sum_{i=1}^n x_i \leq u$$

Besides the lower and upper bound for the number of all activated recommendations, we also consider lower and upper bounds for the activation of the recommendations per model PI_j , $j \in [m]$. With $u_j \in \mathbb{N}_0$ and $l_j \in \mathbb{N}_0$ we define the *maximum* and *minimum number of activated recommendations per PI_j* for all $j \in [m]$, respectively. u_j and l_j allow us to control which expert opinions regarding quality should be considered more.

One example of this is the many reasons to focus more on customer satisfaction, as described in detail in subsection 2.1.1. Let PI_j be the model of customers. By defining a lower bound l_j , the optimization problem must activate at least l_j of customer recommendations. Depending on the settings of the other models, this may lead to a stronger consideration of customer preferences in the claims processing.

Another example for defining lower bounds per model is the requirement that at least one recommendation per model should be considered in the optimization. In this case we define $l_j := 1$ for all $j \in [m]$.

Likewise, it is possible—if necessary—to limit the number of activated recommendations to a maximum of one recommendation per model by setting $u_j := 1$ for all $j \in [m]$. In combination with the overall lower bound we can, e.g., force the optimization algorithm to choose exactly one recommendation per model, so that the individual satisfaction with the process is considered evenly.

Thus, the second constraint we want to consider in the EBRs problem is the upper and lower bound on the number of activations per model PI_j , which is given by

Constraint 4.2.5.

$$l_j \leq \sum_{R_i \in S_j} x_i \leq u_j \quad \forall j \in [m].$$

There can be conflicts and dependencies between the recommendations, which we have to take into account by means of the constraint. Therefore, we define the following additional constraints.

We start with conflicts. Two recommendations are in conflict with each other if processing the one recommendation would mean that processing the other is not possible. An illustrative example is given by the recommendations “*Call the customer.*” and “*Contact the customer digitally during the whole process.*”, which cannot be processed simultaneously in a single claims processing. Therefore, by defining an appropriate constraint we guarantee, that only one of these recommendations is activated in the optimization problem, and the recommendation system is consistent in the given recommendations.

Note, that there could exist recommendations, which are contained in different models and have the same meaning, but impact the quality of the claims processing differently (which can be seen from the sign of the corresponding weighting). We treat such recommendations as a single recommendation. For example, $R_{ij} =$ “*Call the customer.*” and $R_{ij'} =$ “*Do not call the customer.*” are treated as one recommendation $R_i =$ “*Call the customer.*”. The difference is, that $R_{ij'}$ has a negative impact, if R_i is activated. In this thesis we consider such cases not as conflicting recommendations, but define the weighting so that this conflict is taken into account $\gamma_i = \gamma_{ij} + \gamma_{ij'}$ with $\gamma_{ij} > 0$ and $\gamma_{ij'} < 0$.

Finally, we define the *set of conflicts between all considered recommendations* by

$$C := \left\{ \{R_i, R_{i'}\} \in \binom{S}{2} \mid R_i \text{ and } R_{i'} \text{ in conflict, } i, i' \in [n] \right\} \quad (4.1)$$

with $R_i \in S$ and $c := |C| \in \mathbb{N}_0$ is the *number of overall conflicts*. By

Constraint 4.2.6.

$$x_i + x_{i'} \leq 1 \quad \forall \{R_i, R_{i'}\} \in C$$

the optimization algorithm can activate only one recommendation of each conflicting pair of recommendations.

Besides conflicts, there may also be dependencies between the recommendations. We speak of a dependency, if the activation of one recommendation implies the activation of another recommendation.

For example, let us assume that the insurance company works with a mobil app for damage announcements, which has the opportunity to upload pictures from the accident. Furthermore, we assume that the mobil app is newly invented by the insurance company and, therefore, the managers want to praise it to the customers. The recommendation system can pick up this new policy by adding the recommendation $R_i = \text{“Point out the usage of the mobile app to the customer.”}$ to the system. The quality management claims that the customers should use also the functionality to taking pictures from the accident scene directly after the accident happens as it helps to prevent frauds. Therefore, the second recommendation $R_j = \text{“Ask the customer to upload pictures from the accident.”}$ is added to the system. Of course, the activation of R_j requires that R_i is already activated. Thus, these recommendations are dependent.

As we can see, the activation of dependent recommendations has a direction, i.e., it is possible that the activation of recommendation R_i implicates the activation of $R_{i'}$, but not the other way round. Therefore, the *set of positive dependencies between all considered recommendations* $R_i \in S$ is defined by

$$D := \{(R_i, R_{i'}) \in S \times S \mid R_{i'} \text{ requires } R_i, i, i' \in [n]\}. \quad (4.2)$$

With this definition we represent mutual dependent recommendations by defining two tuples $(R_i, R_{i'}) \in D$ and $(R_{i'}, R_i) \in D$, i.e., R_i automatically activates $R_{i'}$ and the other way round.

Defining dependencies via a tuple (i.e., with direction) leads to the following constraint.

Constraint 4.2.7.

$$x_i \leq x_{i'} \quad \forall (R_i, R_{i'}) \in D.$$

Since the conflicts and dependencies between the recommendations depend on the companies and the actuarial framework or situation, we want to reiterate the importance of involving the experts in the process.

Another point that should be noted and discussed with the experts are conflicts of interest between customers and insurance company stakeholders during claims processing. If detected and implemented accordingly in the recommendation system, they can be handled fairly [MF20, Tho19]. The involvement of experts is again important here. Conflicts of interest depend heavily on the goals and framework conditions of the insurance company as well as the situation at hand. If implemented well, the recommendation system can even support the clerk in managing such conflicts of interest in the best possible way.

Finally, another advantage of including expert knowledge is that corporate policy can be taken into account within the recommendation system.

In summary, the EBRs problem can be formulated mathematically by the following *linear integer programming formulation* or ILP.

Program 4.2.8 (EBRS).

$$\begin{array}{ll}
 \max & \sum_{i=1}^n \gamma_i x_i, \\
 \text{s.t.} & l \leq \sum_{i=1}^n x_i \leq u \\
 & l_j \leq \sum_{R_i \in S_j} x_i \leq u_j \quad \forall j \in [m] \\
 & x_i + x_{i'} \leq 1 \quad \forall \{R_i, R_{i'}\} \in C \\
 & x_i \leq x_{i'} \quad \forall (R_i, R_{i'}) \in D \\
 & x_i \in \{0, 1\} \quad \forall i \in [n]
 \end{array}$$

In a last step, we formulate the EBRs problem in terms of an *optimization version* as follows.

Problem 4.2.9. Maximize the weighted sum over all activation variables x_1, \dots, x_n by activating a suitable set of recommendations under consideration of the constraints described above.

4.2.1.2 Defining the EBRs problem as an ILP

In the last subsection we formulated the optimization version of the EBRs problem and defined the ILP by program 4.2.8. With $x_i \in \{0, 1\}$ for all $i \in [n]$, it is more precisely even a 0-1-ILP (see definition 3.4.5). We will reformulate this ILP in this subsection and redefine it with the use of matrices. For this purpose, we apply the definitions of ILP's described in section 3.4.

4.2 Modelling the EBRs problem and classification of its complexity

In case a solution exists, we can find an optimal solution of the EBRs problem, if we activate the recommendations accordingly. Activating a recommendation is done by setting $x_i = 1$ for $i = 1, \dots, n$, and otherwise, it is $x_i = 0$. Therefore, an *optimal solution vector* is given by

$$x^* = \begin{pmatrix} x_1^* \\ \vdots \\ x_n^* \end{pmatrix} \in \{0, 1\}^n.$$

In matrix form, the EBRs problem is defined as

$$\max\{c^T x : Ax \leq b\},$$

where the instances are given as follows. We begin with the weights belonging to the recommendations and define them by

$$c^T := (\gamma_1, \dots, \gamma_n)^T \in \mathbb{R}^n. \quad (4.3)$$

So we search the optimal solution value by maximizing $c^T x$ under consideration of the constraints $Ax \leq b$, which is given by

$$\begin{pmatrix} 1 & \dots & 1 \\ -1 & \dots & -1 \\ & a_1^u & \\ & \vdots & \\ & a_m^u & \\ & a_1^l & \\ & \vdots & \\ & a_m^l & \\ & a_1^C & \\ & \vdots & \\ & a_{|C|}^C & \\ & a_1^D & \\ & \vdots & \\ & a_{|D|}^D & \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \leq \begin{pmatrix} u \\ l \\ u_1 \\ \vdots \\ u_m \\ l_1 \\ \vdots \\ l_m \\ 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad (4.4)$$

with $A \in \{-1, 0, 1\}^{(2+2m+|C|+|D|) \times n}$ and $b \in \mathbb{Z}^{2+2m+|C|+|D|}$ as $u, l, u_1, \dots, u_m, l_1, \dots, l_m \in \mathbb{N}_0$. The matrix A is structured as follows.

The first and second row of A result in the inequalities for the lower and upper bound for all recommendations of the system

$$\sum_{i=1}^n x_i \leq u \text{ and } \sum_{i=1}^n x_i \geq l,$$

respectively. $a_1^u, \dots, a_m^u \in \{0, 1\}^n$ and $a_1^l, \dots, a_m^l \in \{0, 1\}^n$ are the column vectors, which assign those activations variables $x_i, i \in [n]$ to the corresponding model PI_j , for which $R_i \in S_j$. The activation of these column vectors is restricted by the corresponding upper bound u_j and lower bound $l_j, j \in [m]$, respectively. It holds

$$a_{ji}^u := \begin{cases} 1, & \text{if } x_i \text{ is contained in model } PI_j, \text{ i.e., } R_i \in S_j, \\ 0, & \text{otherwise,} \end{cases}$$

for all $i \in [n]$ and $j \in [m]$. The same holds for $a_1^l, \dots, a_m^l \in \{0, 1\}^n$, but instead of $a_{ji}^u = 1$ it is $a_{ji}^l = -1$.

Each column vectors $a_1^C, \dots, a_{|C|}^C \in \{0, 1\}^n$ represents a pair of recommendations, which are in conflict to each other. In total there exists $|C|$ pairs of conflicting recommendations with C defined as the set of conflicts—see equation (4.1). For all $i \in [n]$ and $k = 1, \dots, |C|$ the column vectors are given by

$$a_{ki}^C := \begin{cases} 1, & \text{if } \{R_i, R_{i'}\} \in C, \\ 0, & \text{otherwise.} \end{cases}$$

With this definition exactly two entries of each column vector a_{ki}^C are equal to 1. Similarly, the column vectors $a_1^D, \dots, a_{|D|}^D \in \{-1, 0, 1\}^n$ are defined. We therefore reformulate the inequalities corresponding to the dependent recommendations from $x_i \leq x_{i'}$ to $x_i - x_{i'} \leq 0$ for all $(R_i, R_{i'}) \in D$. Naturally, each pair of dependent recommendations $(R_i, R_{i'}) \in D$ is represented by a column vector a_{di} and

$$a_{di}^D := \begin{cases} 1, & \text{if recommendation } R_{i'} \text{ requires } R_i, \\ -1, & \text{if recommendation } R_i \text{ requires } R_{i'}, \\ 0, & \text{otherwise,} \end{cases}$$

for all $i \in [n]$ and $d = 1, \dots, |D|$. This definition is based on the set of dependent recommendations D —see equation (4.2).

The set of feasible solution is given by the polytope

$$P := \{x : x \in \{0, 1\}^n \wedge Ax \leq b\}$$

where A, b , and c^T as in the equations (4.4) and (4.3).

For a better understanding of the matrix A we give the following example.

Example 4.2.10. Let three models PI_1, PI_2 , and PI_3 be given and, thus, $m = 3$. Furthermore, nine recommendations x_1, \dots, x_9 are given, which are distributed to the models by $S_1 = \{x_1, x_2, x_3\}$, $S_2 = \{x_4, x_5\}$ and $S_3 = \{x_6, x_7, x_8, x_9\}$. This distribution is shown in figure 4.3

4.2 Modelling the EBRs problem and classification of its complexity

Each x_i has a weight $\gamma_i \geq 0$ for all $i = 1, \dots, 9$ and the overall lower bound and upper bound are given by $l = 3$ and $u = 4$, respectively. There exist two pairs of conflicting recommendations given by the set of conflicting recommendations $C = \{(R_3, R_8), (R_5, R_7)\}$ with $|C| = 2$. And additionally, there exist two pairs of dependent recommendations given by the set of dependent recommendations $D = \{(R_6, R_4), (R_4, R_2)\}$ with $|D| = 2$.

Then the set of feasible solutions of the 0-1 integer linear program $\max\{c^T x : Ax \leq b\}$ with $c^T = (\gamma_1, \dots, \gamma_9)$ is given by all $x \in \{0, 1\}^9$ satisfying the following inequalities (with zeros omitted).

$$\underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & & & & & & \\ & & & 1 & 1 & & & & \\ & & & & & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & & & & & & \\ & & & -1 & -1 & & & & \\ & & & & & -1 & -1 & -1 & -1 \\ & & 1 & & & & & & 1 \\ & & & & 1 & & 1 & & \\ & & & -1 & & 1 & & & \\ -1 & & 1 & & & & & & \end{pmatrix}}_{A \in \{-1, 0, 1\}^{(2+2 \cdot 3+|C|+|D|) \times n}} \cdot \underbrace{\begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \\ 3 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}}_{b \in \mathbb{Z}^{2+2 \cdot 3+|C|+|D|}} \leq \begin{pmatrix} x_1 \\ \vdots \\ x_9 \end{pmatrix}.$$

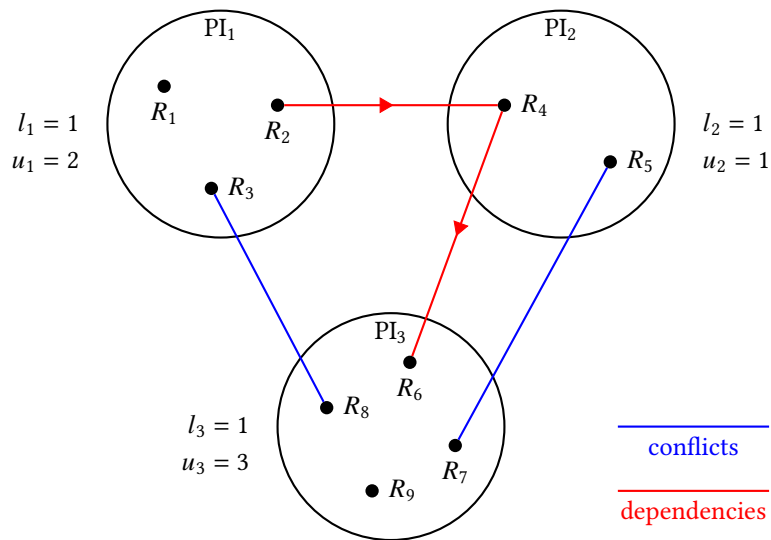


Figure 4.3: Example of an EBRs problem as ILP as given in example 4.2.10.

4.2.2 Decision version of the EBRs problem

We have formulated the optimization version of the EBRs-problem in the last subsections. In the remainder of this thesis, instead of dealing with the algorithmic computability of the problem, we will be concerned with the time complexity of the *decision version* of the EBRs-problem. To this end, we will show in the next chapter that the EBRs problem is \mathbb{NP} -complete.

To prove the \mathbb{NP} -completeness we first formulate the decision version of the problem. However, as we will see at the end of this chapter, we can use the decision and optimization versions of the problem synonymously when considering solvability in polynomial time.

The decision version of the EBRs problem is given as follows.

Problem 4.2.11. Given an integer $k \in \mathbb{N}_0$, can the weighted sum over all activation variables x_1, \dots, x_n sum up to a value of at least k by activating a set of appropriate recommendations under consideration of the constraints?

The way the decision version is formulated, the problem searches for the answer yes or no. That is, a solution of the decision problem is an assignment of a truth value to each corresponding instance of the problem.

An instance of the EBRs problem is given by an integer $k \in \mathbb{N}_0$, a set of activation variables $\{x_1, \dots, x_n\}$, where $n \in \mathbb{N}$ is the number of all recommendations of the system, the corresponding weights $\gamma_1 \dots, \gamma_n$ and all inputs needed to define the constraints. The problem assigns “true” to the instance if it can activate an appropriate set of recommendations under consideration of the constraints, so that the weighted sum over all activation variables sum up to a value of at least k . Otherwise it assigns “false”. For more details we refer to section [3.3](#).

With the following corollary, we show that there is no significant difference between formulating the EBRs problem in its decision version and in its optimization version in terms of solvability in polynomial time.

Corollary 4.2.12. *In terms of polynomial-time solvability, there is no difference between the optimization version (problem [4.2.9](#)) and the decision version (problem [4.2.11](#)) of the EBRs problem.*

Proof. We show the statement of the corollary by proving that given a solution of the optimization version, we automatically have a solution of the decision version and vice versa.

In a first step, we assume that we have a method for solving the optimization version and the weighted sum over all activation variables is maximal when a given set of recommendations is activated. Moreover, $\sum_{i=1}^n \gamma_i x_i = k$ by activating this given set

of recommendations. And thus, for any k , we then automatically have a solution to the decision version of the problem. This is because the decision version of the problem seeks for a yes or no answer to the question of whether the weighted sum over all activation variables yields a value of at least k when activating a set of appropriate recommendations.

For the reverse direction, we assume that we can solve the decision version for any k . If we now take the largest k for which we get the answer “yes”, we, thus, also know the maximum number to which the weighted sum adds up. More precisely, for a given set of recommendations $S = \{R_1, \dots, R_n\}$ with corresponding activation variables x_1, \dots, x_n we solve the decision version of the problem for each k until we find the maximal number of k for which the answer is yes. This can be done in $O(\log n)$ using binary search [DPV06, KT13].

We would like to mention that the solution for the optimization version generated in this way may not be unique, since the weighted sum may add up to k by activating different sets of activation variables. However, this does not affect the correctness of the proof. \square

In the remainder of this thesis, we speak synonymously of the optimization version and the decision version of the problem and simply call them problem. However, for the complexity proofs we mostly use the decision version and explicitly state when we consider the optimization version.

4.2.3 Proof of NP-completeness

Now that we have formulated the EBRS problem, we are interested in whether and how “easy” it is to solve. Here, “easy to solve” implies the following question.

Is the EBRS problem solvable in polynomial time?

In this subsection we will prove that the EBRS problem is NP-complete, i.e., the problem is in the complexity class NP and NP-hard.

Problems Π in the complexity class NP, i.e., $\Pi \in \text{NP}$, describe a class of problems, for which a solution can be *verified* “easily”. That implies that we can verify in polynomial time whether an existing certificate is the solution to our problem. Yet, this class of problems typically cannot be solved in polynomial time.

Since NP-complete problems are also NP-hard, any problem $\Pi' \in \text{NP}$ is polynomially reducible to the considered NP-complete problem, i.e., $\Pi' \leq_P \Pi$ for all $\Pi' \in \text{NP}$ —see definition 3.3.18.

In summary, NP-complete problems have the special property that in the case of an existing polynomial time solution for a single NP-complete problem, there exists a

polynomial time solution for all problems in \mathbb{NP} . We motivated this in section [2.2](#). For more details on the complexity of a problem, we refer to section [3.3](#).

To prove the \mathbb{NP} -hardness of the EBRs problem, we do not show $\Pi' \leq_P \Pi$ for all $\Pi' \in \mathbb{NP}$, but apply lemma [3.3.21](#). According to this lemma, it suffices to prove that a problem Π is \mathbb{NP} -hard if there exists a known \mathbb{NP} -complete problem Π' with $\Pi' \leq_P \Pi$. We describe this in detail in subsection [3.3.3](#).

With the proof of the following theorem we show the \mathbb{NP} -completeness of the EBRs problem.

Theorem 4.2.13. *The decision problem EBRs is \mathbb{NP} -complete.*

Proof. To prove that the EBRs problem is \mathbb{NP} -complete, we need to show that (a) $\text{EBRS} \in \mathbb{NP}$, and (b) EBRs is \mathbb{NP} -hard, according to lemma [3.3.21](#).

a) “EBRS $\in \mathbb{NP}$ ”:

Let I be a yes instance of the decision problem EBRs for which a certificate C is given, such that the input I concatenated with certificate C is a solution of the associated checking problem EBRS' , i.e., $(I \circledast C, 1) \in \text{EBRS}'$. C consists of a natural number $w \in \mathbb{N}$ of activated recommendations R_{i_1}, \dots, R_{i_w} with $x_{i_1} = \dots = x_{i_w} = 1$ and their corresponding weights $\gamma_{i_1}, \dots, \gamma_{i_w} \in \mathbb{R}_0^+$ with $i \in [n]$ and $\{i_1, \dots, i_w\} \subseteq [n]$ such that the weighted sum over all activation variables sum up to a target value k at least, i.e.,

$$\sum_{i=1}^w \gamma_{i_i} x_{i_i} \geq k.$$

We write the checking problem EBRS' in matrix notation. Then C is a solution vector $(x_1, \dots, x_n)^T \in \{0, 1\}$ of the EBRs problem, for which for an appropriate subset $\{i_1, \dots, i_w\} \subseteq [n]$ the corresponding activation variables x_{i_1}, \dots, x_{i_w} are set to one.

We can find a polynomial π such that $|C| \leq \pi(|I|)$ as C is a vector with binary entries. Furthermore, the algorithm to check whether C is a solution to the given EBRs problem needs to perform simple matrix operations, and therefore run in polynomial time. Thus, one can find a polynomial time verifier, and therefore $\text{EBRS} \in \mathbb{NP}$.

b) “EBRS is \mathbb{NP} -hard”:

It suffices to show that there exists an \mathbb{NP} -complete problem Π with $\Pi \leq_P \text{EBRS}$, i.e., by applying a *Karp reduction*. More specifically, we transform a given instance of Π into a single instance of EBRs with the same answer [\[KT13\]](#).

As an NP-complete problem we choose the independent set (IS) problem. For more informations about the IS problem we refer to definition [3.2.5](#) and problem [3.3.31](#). We need to show that

$$\text{IS} \leq_P \text{EBRS}.$$

Therefore, we assume that a polynomial time subroutine exists for EBRS. In order to solve the IS problem (using the EBRS subroutine), any input of the IS problem must be transformed first.

Let an arbitrary undirected graph $G = (V, E)$ with $n = |V|$ nodes and $c = |E|$ edges be given as input of the IS problem. Furthermore, let an integer $k \in \mathbb{N}_0$ be given (see problem [3.3.31](#)).

The transformation interprets every node $v_i \in V$ as a recommendation $R_i \in S$ for all $i \in [n]$. Each edge $e_j \in E$, $j \in [c]$, with $v(e_j) = \{v_i, v_{i'}\} \in E$ is transformed to a conflict between two recommendations R_i and $R_{i'}$, and with that $\{R_i, R_{i'}\} \in C$. Thus $|C| = c$.

Let I be any set of nodes of G with $k = |I| \in \mathbb{N}_0$. The transformation interprets $v_i \in I$ as an activation of the recommendation R_i by setting $x_i = 1$. If $v_i \notin I$ than the corresponding recommendation R_i is not activated and $x_i = 0$.

Furthermore, any model of the EBRS problem PI_j , $j \in [m]$, consists of exactly one node and recommendation, respectively, and so $|S_1|, \dots, |S_n| = 1$ with $m = n$. To ensure that an IS is found by the subroutine of EBRS, the weight of each recommendation is set to one and hence $\gamma_i = 1$ for all $i \in [n]$. In addition, there exist no dependencies between the recommendations and hence $D = \emptyset$. The lower bounds are set to zero, i.e., $l = 0$ and $l_i = 0$ for all $j \in [i]$, respectively. Thus it is allowed to recommend any action. The upper bound is set to $u = n$ for all recommendations and $u_i = 1$ for all $i \in [n]$, since there exists exactly one recommendation per model PI_j , $j \in [m]$.

With the following claim we prove the correctness of the reduction.

Claim: Let $k \in \mathbb{N}_0$ (with $k \leq n$). A graph $G = (V, E)$ has an independent set of size at least k if and only if the sum over all recommendations is at least k under consideration of the constraints.

Note: By proving this claim, we show that if the EBRS problem yields an output “yes” for the transformed input by activating k recommendations, then there exists an IS of size k , and vice versa.

“ \Rightarrow ”: Let I be an independent set of graph G with size at least k , i.e., $|I| \geq k$. Under this assumption, we need to show that, taking into account the constraints, the sum over all activation variables is also at least k .

When $v_i \in I$, then $R_i \in I$ and with $R_i \in I$ the recommendation is activated, i.e., $x_i = 1$ for all $R_i \in I$ (see transformation). Since there

are no adjacent nodes in the independent set I , there is also no conflict between the activated recommendations. Due to $D = \emptyset$ no contradiction can occur in the subroutine. i.e., it is ensured that the subroutine is not forced to activate a recommendation that would lead to a violation of the constraints. With $\gamma_i = 1$ the sum over all activation variables is truly k or higher.

And thus a solution for the subroutine is found, since the sum over all activation variables is indeed at least k considering all given constraints.

“ \Leftarrow ”: Assume that the sum over all activation variables is at least k under consideration of the given constraints. We show that under this assumption there exists an independent set I of at least k vertices for an undirected graph $G = (V, E)$ with (in total) $n = |V|$ nodes $v_i, i \in [n]$ and $c = |C| = |E|$ edges $e_j, j \in [c]$.

Let I consist of all activated recommendations R_i . I is an independent set as due to the constraints no conflict exists between two activated recommendations. This implies that no edge exists between any two nodes contained in the set I .

As the sum over all activation variables is at least k , set I has at least cardinality $|I| \geq k$. This is true because with $\gamma_i = 1$ for all $i \in [n]$ the recommendations are equally weighted and with $D = \emptyset$ no dependencies have to be considered. Thus, all activated variables sum up to at least k only if at least k recommendations are activated. Therefore, I is an independent set of size at least k .

It remains to show that the transformation is computable in polynomial time.

Every node is interpreted as a recommendation and, thus, all nodes $v_i \in V$ are transformed to $R_i \in S$ and it is $|V| = |S| = n$. This is a one-to-one translation, which is computable in polynomial time. Every edge $e_j \in E$ with $j \in [c]$ of the graph G is transformed to a conflict between two recommendations. That means, every edge e_j with $v(e_j) = \{v_i, v_{i'}\}$ is a conflict between the recommendations R_i and $R_{i'}$. And so, the number of edges is identical to the number of conflicts ($|E| = |C|$), which is also a one-to-one translation that can be computed in polynomial time. Therefore, the total transformation is computable in polynomial time.

Thus, using Karp's reduction, we have proved that EBRS is an NP -hard decision problem.

With $\text{EBRS} \in \text{NP}$ and EBRS being an NP -hard problem we finally proved that the decision problem EBRS is NP -complete. \square

To better understand the setting of the input parameters $D = \emptyset$ and $\gamma_i = 1$ for all $i \in [n]$ in the proof, we discuss the following two figures. For this purpose, we choose two rather simplified examples.

The first figure 4.4 is intended to illustrate that the choice of $\gamma_i = 1$ for all $i \in [n]$ is necessary. Without setting the weighting equal to one, in this example the solution algorithm chooses to recommend R_1 instead of R_2 and R_3 , which would be the optimal solution of the IS problem.

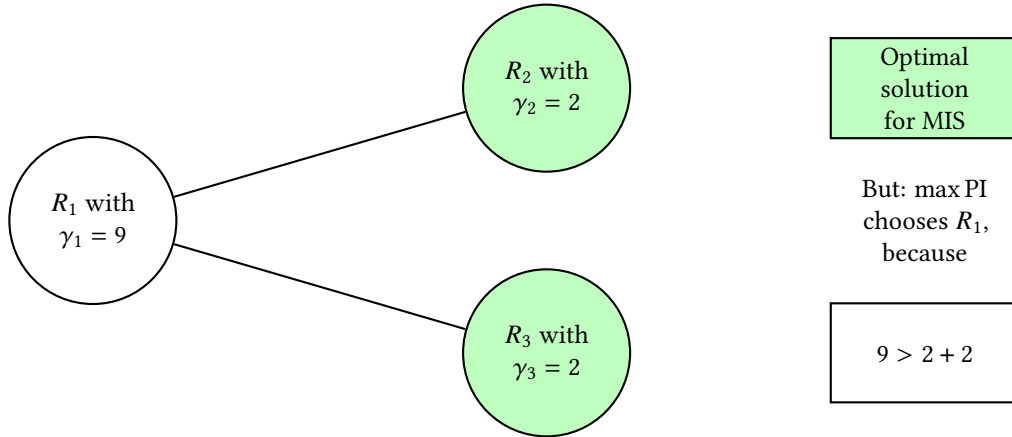


Figure 4.4: Example to illustrate the need of setting $\gamma_i = 1$ for all $i \in [n]$ in the \mathbb{NP} -completeness proof of the EBRs problem.

In the second figure 4.5 we illustrate the special case where no solution can be found due to an unfortunate constellation of dependencies. In the example, the maximum independent set is given by $I = \{R_1, R_2\}$. However, the EBRs algorithm cannot activate both R_1 and R_2 , since it has to activate R_3 when R_1 is activated. By assuming $D = \emptyset$ we exclude such cases and a maximal independent set can be found by the polynomial time subroutine of the EBRs problem.

Nonetheless, the \mathbb{NP} -completeness of the general EBRs problem—without the restriction of $\gamma_i, i \in [n]$ and $D = \emptyset$ —is still proven. To support this statement we refer to corollary 3.3.28.

4.3 Complexity-theoretical investigations of the EBRs problem

After defining the EBRs problem and proving that it is \mathbb{NP} -complete we now focus on the complexity-theoretical study of the problem. In theorem 4.2.13 we have shown that the EBRs problem is \mathbb{NP} -complete. To prove the theorem, we restricted the constraints and consequently there were no dependencies between the recommendations (i.e., $D = \emptyset$) and all weights were set to one (i.e., $\gamma_1 = \dots = \gamma_n = 1$).

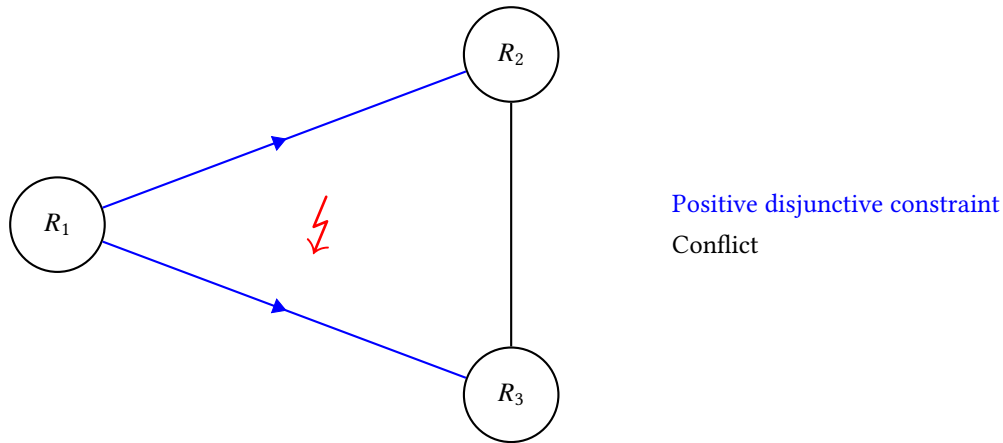


Figure 4.5: Example to illustrate the need of $D = \emptyset$ in the NP-completeness proof of the EBRs problem.

In addition, there is only one recommendation per model PI. The corresponding optimization problem has the following form.

Program 4.3.1.

$$\begin{aligned}
 & \max \sum_{i=1}^n x_i \\
 \text{s.t.} \quad & l \leq \sum_{i=1}^n x_i \leq u \\
 & x_i + x_{i'} \leq 1 \quad \forall \{R_i, R_{i'}\} \in C \\
 & x_i \in \{0, 1\} \quad \forall i \in [n]
 \end{aligned}$$

However, there exist other restrictions on the constraints of the EBRs problem. In the remainder of this thesis, we discuss different cases of constraint restrictions and evaluate whether the EBRs problem can be solved in polynomial time under these restricted constraints. For a practical application it is of course of great advantage if the correspondingly adapted EBRs problem could be solved in polynomial time. Therefore, in the further course of this section we ask the following question.

Are there cases of restricted constraints on the EBRs problem for which the problem is “easier” to solve?

Depending on the situation under consideration, there are a variety of restrictions on the constraints. Because of this variety, we do not claim comprehensiveness of all possible cases in this thesis.

For the proof of theorem [4.2.13](#), we have already restricted the constraints to consider only recommendations with conflicts. Therefore, as a first step, we observe the

cases with conflicting but independent recommendations, i.e., $C \neq \emptyset$ and $D = \emptyset$. We discuss these cases in subsection 4.3.1. In subsection 4.3.2, we study the cases of the EBRs problem with dependent but not conflicting recommendations. Finally, we examine the cases with independent recommendations without conflicts in subsection 4.3.3.

4.3.1 Existence of conflicts between recommendations but no dependencies

As already mentioned, we showed in the proof of theorem 4.2.13 that the EBRs problem without the constraint of dependent recommendations is NP-complete. In this chapter, we look for cases of the EBRs problem for which the problem is “easier” to solve, assuming that there are no dependent recommendations. i.e., it is $D = \emptyset$ and $C \neq \emptyset$. We refer to the EBRs problem under these constraints as EBRSC. We have identified three cases of EBRSC, which we present below.

4.3.1.1 Case: Pairwise conflicts

We start with a special case of the EBRSC problem, for which we consider the optimization version of the problem. More precisely, we consider the EBRSC problem as a 0-1-ILP $\max\{c^T : Ax \leq b\}$ for which a set of feasible solutions $P = \{x : x \in \{0, 1\}^n \wedge Ax \leq b\}$ is given, as described in subsection 4.2.1.2.

For this particular case, we can prove that the constraint matrix

$$A \in \{-1, 0, 1\}^{(2+2m+|C|+|D|) \times n}$$

is totally unimodular. We defined total unimodularity in section 3.6. There, we also showed the theorem of Hoffman and Kruskal 3.6.14, which states that for a totally unimodular constraint matrix A the corresponding polyhedron P is integer if b is integer. That is, the polyhedron P corresponds to its integer hull, i.e., $P = P_I$, and we can ignore the constraint $x \in \{0, 1\}^n$. Thus, the 0-1-ILP coincides with its LP relaxation and can therefore be solved in polynomial time.

This special case is defined by a set of conflicting recommendations C containing only pairwise conflicts, i.e., a recommendation conflicts with at most one other recommendation. Assuming that there are no conflicts between recommendations of the same model PI_j , no pair of conflicting recommendations does belong to the same model PI_j for all $j \in [m]$ (see remark 4.2.2). Moreover, according to $D = \emptyset$, there are no recommendations included in more than one model.

We illustrate the observed situation with the following example.

Example 4.3.2. In the example, we consider nine recommendations R_1, \dots, R_9 with activation variables $x_1, \dots, x_n \in \{0, 1\}$ distributed among three models PI_1, PI_2 and $PI_3, m = 3$, as shown in figure 4.6. Here, the blue lines indicate the conflicts between the recommendations. The red line is an example of a conflict that is not allowed in this special case. Each model consists of three recommendations and the set of conflicts is given by $C = \{\{R_2, R_4\}, \{R_3, R_7\}, \{R_6, R_9\}\}$.

The corresponding constraint matrix A is given as follows, with the zeros omitted.

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & & & & & & \\ & & & 1 & 1 & 1 & & & \\ -1 & -1 & -1 & & & & 1 & 1 & 1 \\ & & & -1 & -1 & -1 & & & \\ & & & & & & -1 & -1 & -1 \\ & 1 & & 1 & & & & & \\ & & 1 & & & & 1 & & \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{pmatrix} \in \{-1, 0, 1\}^{(2+2m+|C|) \times n}$$

Here, the first to rows of the matrix A correspond to the constraint that the total sum of activated recommendations is limited by the lower bound l and the upper bound u . The next $2m$ rows, i.e., a_3, \dots, a_8 , correspond to the constraints that the total sum per model PI_1, PI_2 , and PI_3 is limited by the lower bounds l_j and upper bounds $u_j, j \in [m]$, respectively. The last $|C| = 3$ rows refer to the conflicting recommendations and the corresponding constraint that only one conflicting recommendation can be activated.

Furthermore, it is

$$\begin{aligned} (a_{2i})_{i \in [n]} &= -(a_{1i})_{i \in [n]} \\ (a_{6i})_{i \in [n]} &= -(a_{3i})_{i \in [n]} \\ (a_{7i})_{i \in [n]} &= -(a_{4i})_{i \in [n]} \\ (a_{8i})_{i \in [n]} &= -(a_{5i})_{i \in [n]}. \end{aligned}$$

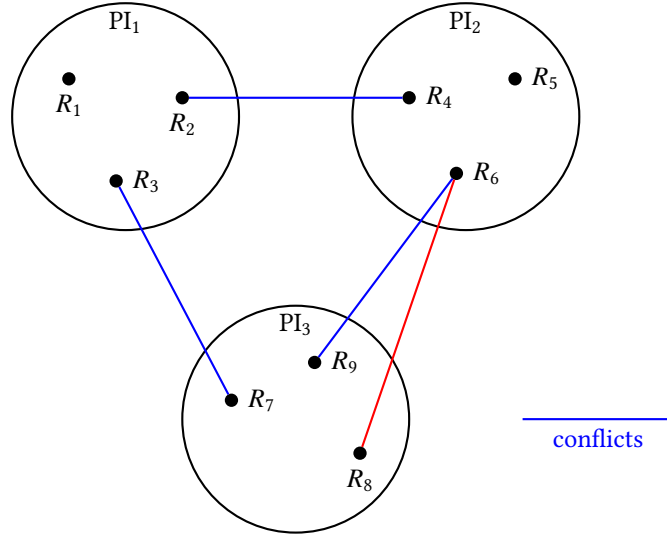


Figure 4.6: Example of the EBRS problem with a totally unimodular constraint matrix—with the red line an example of a conflict that is not allowed in this special case.

More generally, the constraint matrices for this special case are given as follows

$$A = \begin{pmatrix} 1 & \dots & 1 \\ -1 & \dots & -1 \\ a_1^u \\ \vdots \\ a_m^u \\ a_1^l \\ \vdots \\ a_m^l \\ a_1^C \\ \vdots \\ a_{|C|}^C \end{pmatrix} = \begin{pmatrix} 1^n \\ -1^n \\ A' \\ A^C \end{pmatrix} \in \{-1, 0, 1\}^{(2(m+1)+|C|) \times n}$$

where $a_1^u, \dots, a_m^u \in \{0, 1\}^n$ and $a_1^l, \dots, a_m^l \in \{-1, 0\}^n$ are the row vectors of the matrix. They assign the variables x_i to the corresponding model PI _{j} , $j \in [m]$. $a_1^C, \dots, a_{|C|}^C \in \{0, 1\}^n$ are the row vectors, each corresponding to a pair of conflicts according to the definition of C . For more details on the definition of these vectors we refer to section [3.6](#).

We combine the row vectors into matrices in the following way.

$$A' = \begin{pmatrix} a_1^u \\ \vdots \\ a_m^u \\ a_1^l \\ \vdots \\ a_m^l \end{pmatrix} \in \{-1, 0, 1\}^{2m \times n} \text{ and } A^C = \begin{pmatrix} a_1^C \\ \vdots \\ a_{|C|}^C \end{pmatrix} \in \{0, 1\}^{|C| \times n}$$

The corresponding vector b is given by

$$b = (u, l, u_1, \dots, u_m, l_1, \dots, l_m, 1, \dots, 1)^T \in \mathbb{Z}^{(2(m+1)+|C|) \times n}.$$

As we can see b is integer with $u, l, u_1, \dots, u_m, l_1, \dots, l_m \in \mathbb{N}_0$. Therefore, it only remains to show that A is TU.

Before proving that A is totally unimodular, we give a brief description of the matrix A . We denote with $S = \{R_1, \dots, R_n\}$ the set of all given recommendations and with S_1, \dots, S_m the set of recommendations per model PI_j , $j \in [m]$. With $D = \emptyset$ we know that S_1, \dots, S_m build a partition of S . Thus, the vectors a_1^u, \dots, a_m^u as well as a_1^l, \dots, a_m^l are linear independent and, moreover, by summing the vectors a_1^u, \dots, a_m^u as well as a_1^l, \dots, a_m^l we obtain a vector consisting only of 1's and -1's, respectively. However, a_1^u is linear dependent on a_1^l , a_2^u is linear dependent on a_2^l , and so on.

A^C is a submatrix, which represents the conflicts between the recommendations. Its rows have at most two non-zero entries. Furthermore, as a recommendation can at most be in conflict with one other recommendation, the rows of A^C , i.e., $a_1^C, \dots, a_{|C|}^C$, are linear independent.

Proposition 4.3.3. *A is totally unimodular.*

Proof. To show total unimodularity, we need to prove that $\det(U) \in \{-1, 0, 1\}$ for each quadratic submatrix $U \in \{-1, 0, 1\}^{k \times k}$ of A with $k = |I|$ and $I \subseteq \{1, \dots, 2 + 2m + |C|\}$. For $k = 1$ this is directly implied by

$$A = (a_{ji})_{\substack{j \in (2+2m+|C|) \\ i \in [n]}} \in \{-1, 0, 1\}^{(2+2m+|C|) \times n}.$$

Therefore, let $k > 1$. W.l.o.g., we assume that k rows are selected from the matrix A and build a new quadratic submatrix $U \in \{-1, 0, 1\}^{k \times k}$. We discuss the following four cases to prove that $\det(U) \in \{-1, 0, 1\}$, for all possible case constellations arising from drawing k rows from the constraint matrix A .

- a) All k rows are drawn from the submatrix A' , i.e., the sample consists of rows that assign the recommendations to their lower and upper bounds per model PI_j , $j \in [m]$.

Let $B \in \{-1, 0, 1\}^{k \times n}$ be a submatrix of A' . As S_1, \dots, S_m is a partition of S we know, that each recommendation is assigned to exactly one model PI_j . Thus, in each column of A' there exist at most two non-zero entries and all non-zero entries of A' are either 1 or -1 . With these conditions, we can find a partition (I_1, I_2) of I , $|I| = k$ as shown below.

In case a_j^u and a_j^l for $j \in [2m]$ are drawn, we know that the determinant is zero, as the vectors are linear dependent. By definition of A' , it is the only case where the submatrix B has two non-zero entries per column. Otherwise, there is only one non-zero entry, which is 1 or -1 . Moreover, we know that a_1^u, \dots, a_m^u as well as a_1^l, \dots, a_m^l are linear independent and therefore, there are no overlapping non-zero entries in the columns. Due to linear independence, it is possible to assign one of these rows to an index set I_1 or I_2 without affecting the assignment of the other rows. With this knowledge and due to the fact that at most one entry per column is non-zero, we can arbitrarily assign each row to I_1 or I_2 . Thus B is totally unimodular by lemma 3.6.9 and all quadratic submatrices U of B have determinants equal to -1 , 0 , and 1 .

If $k = 2m$ and thus all rows are drawn from the matrix A' , we have the special situation with linear dependent row vectors. The determinant of linear dependent vectors is zero.

- b) In this case, we assume that all k rows are drawn from the submatrix A^C , i.e., the submatrix containing all pairs of conflicting recommendations. And therefore we know that all columns are linear independent. The columns of matrix A^C contain at most one non-zero entry and all entries are either 0 or 1. Therefore, we can find a partition (I_1, I_2) of I by distributing the rows arbitrarily. Thus we can apply lemma 3.6.9 and have shown that A^C is totally unimodular. And therefore, the determinant of each quadratic submatrix U of A^C equals -1 , 0 , or 1 .
- c) None of the k rows drawn contains the rows $(a_{1i})_{i \in [n]} = 1^n$ or $(a_{2i})_{i \in [n]} = -1^n$, i.e., neither row is selected, which refers to the constraint that the total sum of all recommendations is bounded by a lower or upper bound, respectively.

Similar to the first case, if a_j^u and a_j^l with $j \in [2m]$ are drawn from A for any square submatrix $U \in \{-1, 0, 1\}^{k \times k}$, then the determinant of U is zero.

We consider in the following the submatrix $B \in \{-1, 0, 1\}^{k \times n}$ drawn from A as described in this case. For the remaining cases it holds, that the columns of B have at most two non-zero entries. To apply lemma 3.6.9 we need to show that there is a partition (I_1, I_2) of I .

W.l.o.g., we assign all rows of A^C to the index set I_1 , since all rows of A^C are linear independent. We now only need to assign the rows of the matrix A' to an index set. Since in this case the rows $(a_{1i})_{i \in [n]} = 1^n$ and $(a_{2i})_{i \in [n]} = -1^n$ are excluded, the rows of the remaining matrix A' are either linear independent or the columns with two non-zero entries have the values 1 or -1 , but in such a way that one row consists only of 1 and the other only of -1 . For better understanding, we give the following example for the case of columns with two non-zero entries.

$$\begin{pmatrix} 1 & 1 & & & \\ -1 & -1 & & & \\ & & 1 & 1 & 1 \\ & & -1 & -1 & -1 \end{pmatrix}$$

And therefore it is possible, that we assign all rows of A' with 1's to I_2 and with -1 's to I_1 . In summary, the assumptions of lemma 3.6.9 are fulfilled and thus, B is total unimodular, i.e., the determinant of each square submatrices of B is -1 , 0 , or 1 .

- d)** In the last case, it is also possible to draw from the first two rows $(a_{1i})_{i \in [n]} = 1^n$ and $(a_{2i})_{i \in [n]} = -1^n$ of the matrix A .

If both rows are drawn from the matrix A , we know that the determinant is zero because they are linear dependent. Therefore, we exclude all cases where both rows are drawn. In addition, we also exclude all cases discussed above

Let $B \in \{-1, 0, 1\}^{k \times n}$ be the drawn submatrix of A . After excluding the cases just mentioned, all columns of B have at most three non-zero entries. We can distinguish between two cases—in the first one $(a_{1i})_{i \in [n]} = 1^n$ is drawn, and in the second one $(a_{2i})_{i \in [n]} = -1^n$ is drawn. For both cases we show that we can find a partition (I_1, I_2) of I such that equation 3.7 holds.

- da)** Let $(a_{1i})_{i \in [n]} = 1^n$ be drawn first. W.l.o.g. let $b_{1i} = a_{1i}$ for all $i \in [n]$. Furthermore, let the index of $(b_{1i})_{i \in [n]}$ be assigned to I_1 , i.e., $1 \in I_1$. Then, all rows drawn from A' can be assigned to the partition by the following rules.

- daa)** All rows of B drawn from rows a_1^u, \dots, a_m^u of A are assigned to partition I_2 .
- dab)** All rows of B drawn from rows a_1^l, \dots, a_m^l of A are assigned to partition I_1 .

Finally, we have to assign the rows drawn from A^C to the partition. For this purpose, we consider the following figure 4.7. First, however, it should be noted that because of the remark 3.6.7 we can permute the columns and rows of the matrix B arbitrarily, so that we have the structure chosen in the figure. Again, we omit the zeros in the figure.

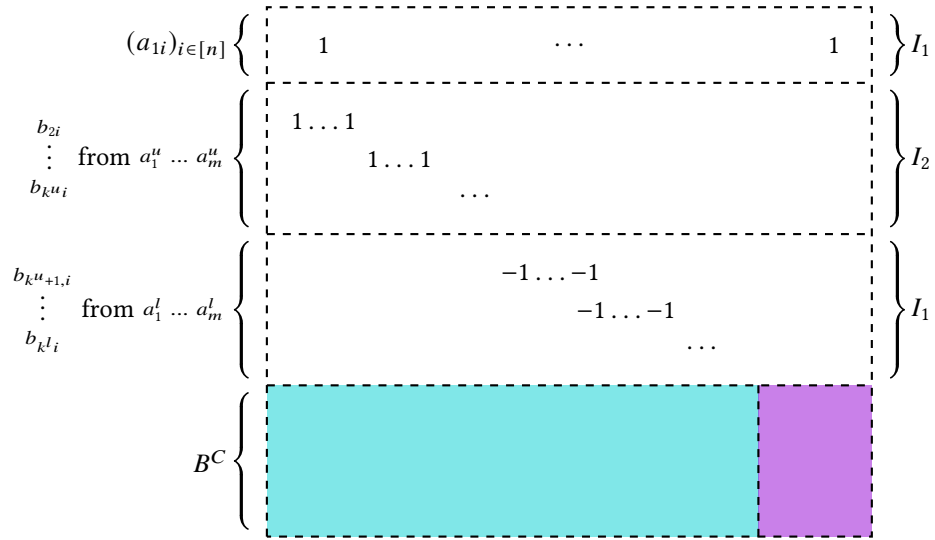


Figure 4.7: Assignment of the rows of the constraint matrix A^C to I_1 or I_2 of the partition $I = (I_1, I_2)$. For the matrix B^C , the non-zero turquoise part can be assigned to either I_1 or I_2 , since the row sum is already zero due to the assignment of a_1^u, \dots, a_m^u to I_2 and the assignment of a_1^l, \dots, a_m^l to I_1 . The rows of the pink part of B^C that are non-zero must be assigned to I_2 since $(a_{1i})_{i \in [n]}$ has already been assigned to I_1 .

By $B^C \in \{0, 1\}^{k^C \times n}$ we denote the submatrix of B generated by drawing any number k^C of rows of A^C .

Any non-zero entry of B^C is 1. Furthermore, the first row of the matrix $(a_{1i})_{i \in [n]}$ —which contains only 1’s—and all drawn rows from a_1^u, \dots, a_m^u —which also contain 1’s and 0’s—are assigned to I_1 and I_2 , respectively. So the turquoise part of the matrix B^C can be assigned to either I_1 or I_2 and the equation (3.7) is still satisfied. All rows of B^C with non-zero entries in the pink part must be assigned to I_2 . It remains to prove that the assignment of all rows with non-zero entries in the pink part to I_2 is possible.

Suppose that we cannot assign one of these rows to I_2 . Let w.l.o.g. $(b_{k^C i})_{i \in [n]}$ be this row, which is the last row of B^C . Each row of B^C consists of two non-zero entries, both equal to 1. It would not be possible to assign this row to I_2 , if an entry—the one not included in the pink area—must be assigned to I_1 . This is the case if there exists a second row vector of B^C —suppose it is w.l.o.g. $(b_{(k^C-1)i})_{i \in [n]}$ —for which

$$b_{k^C i} = b_{(k^C-1)i} = 1, \quad i \in [n].$$

This can only be the case if there exists a second conflict for recommendation R_i . Yet, such a case is excluded for the special case considered in this subsection.

db) Finally, let $(a_{2i})_{i \in [n]} = -1^n$ be drawn. The proof is similar to the case just discussed. However, the assignment differs in the following way.

W.l.o.g we assign $(a_{2i})_{i \in [n]}$ to the index set I_1 . The rows drawn from A' are assigned to the partition by the following rules.

dba) All rows of B drawn from rows a_1^u, \dots, a_m^u of A are assigned to partition I_1 .

dbb) All rows of B drawn from rows a_1^l, \dots, a_m^l of A are assigned to partition I_2 .

To satisfy equation (3.7) of theorem 3.6.11, the rows of B^C having non-zero entries in the columns of the pink area must be assigned to I_1 . The proof that the assignment of all rows with non-zero entries in the pink part is possible follows the proof of part da).

By proving both cases, we showed that theorem 3.6.11 is fulfilled for all submatrices $B \in \{-1, 0, 1\}^{k \times n}$ with $k = |I|$ and $I \subset [m]$.

Hence, we have proved the total unimodularity of A . □

With the total unimodularity of A , we know that the EBRSC problem is equivalent to its LP relaxation for the case where only pairwise conflicts exist between the recommendations, and therefore can be solved in polynomial time.

4.3.1.2 Case: Stellar conflicts

We continue to consider the EBRSC problem in this subsection. That is, dependencies between recommendations do not exist, but conflicts do. By further restricting the set of conflicts C to allow only pairwise conflicts between the recommendations, we showed in the last subsection that the associated constraint matrix A of the associated 0-1-ILP's $\max\{c^T : Ax \leq b\}$ of the EBRSC problem is totally unimodular. With A totally unimodular and since b is integer, the EBRSC problem is equivalent to its LP relaxation. To obtain the LP relaxation, we omit the $x \in \{0, 1\}^n$ constraint of the EBRSC problem and can treat the EBRSC problem as an LP. Thus, we find a solution in polynomial time (if one exists) and this solution is also a solution of the 0-1-ILP of the EBRSC problem. For more information about the ILP we refer to section 3.4 and subsection 4.3.1.1

In this case, we observe another special set of conflicts that is totally unimodular. Again, we observe pairwise conflicts, but in this case one of this recommendations can be in conflict with more than one recommendation. However, the reverse is not true, i.e., the other recommendations conflict only with the one recommendation. The one recommendation which is in conflict with others is called *center of star*,

A^C , i.e., for the example we get

$$(A^C)^T = \begin{pmatrix} 1 & 1 & 1 & 1 & & & \\ 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & 1 & 1 \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{pmatrix}. \quad (4.5)$$

This is admissible since we know by lemma 3.6.8 that A TU if and only if A^T TU. Here, the centers of the stars are given by the first and sixth row in the example matrix (4.5). Furthermore, each of the two lines contains the same number of 1's as there are stellar conflicts or corresponding number of spikes of the star. We will now prove that A^C is totally unimodular.

Proposition 4.3.4. A^C is totally unimodular.

Proof. To prove that A^C is TU we use lemma 3.6.9 and lemma 3.6.8.

We consider in the following $(A^C)^T \in \{0, 1\}^{n \times |C|}$. By definition of the set of conflicts $C = \{\{R_i, R_{i'} \in \binom{S}{2} \mid R_i \text{ and } R_{i'} \text{ in conflict, } i, i' \in [n]\}\}$ —see equation (4.1)—there are exactly two non-zero entries in each row of A^C . And therefore, the columns of $(A^C)^T$ has two non-zero entries.

We apply lemma 3.6.9 to show that $(A^C)^T$ is TU. Therefore, we need to find a partition (I_1, I_2) of the index set I , such that for all columns of $(A^C)^T$ with two non-zero entries with the same sign, we can assign one row to the index set I_1 and the other to index set I_2 (see third condition of lemma 3.6.9).

W.l.o.g. we assign each row of the matrix that belongs to a center of a star to I_1 , and each row that belongs to a spike of a star to I_2 .

The so defined partition (I_1, I_2) satisfies the third condition of lemma 3.6.9, which can be justified as follows.

According to the definition of the stellar conflicts, there exist centers, i.e., recommendations $R_i, i \in [n]$, which are in conflict with some other recommendations, and there exists spikes, which are only in conflict with the corresponding center. We denote by R_i^* the centers of the stellar conflicts.

Let $c_i, i \in [n]$, be the number of recommendations, with which center R_i^* is in conflict with. According to the definition of the set of conflicts, we further know that the i 'th row of the matrix $(A^C)^T$ that belongs to the recommendation R_i^* contains c_i 1's. The remaining rows belong to spikes.

Furthermore, we know that each row of a spike contains exactly one non-zero entry. With the definition of stellar conflicts and the set of conflicts C , there is no row of the matrix $(A^C)^T$ that contains no non-zero or more than two non-zero entries. Moreover, all rows belonging to a spike of a star are linear independent, i.e., the rows have no overlapping 1's in the columns.

W.l.o.g., we assigned all rows of the constraint matrix belonging to the spikes of the stars to I_2 . So, if we sum up all rows assigned to I_2 per stellar conflict, the resulting vector contains c_i non-zero entries. Because of the pairwise conflicts, these 1's are exactly in the same place as the non-zero entries of the corresponding row of the constraint matrix with the assignment of the recommendations to the center of the star.

This becomes more apparent when we look at matrix (4.5) again. The first row is the center of a star and the rows two through five are the corresponding spikes of the star. We can see that each column of these rows exists of either two ones or just zeros. In addition, we can see that we can split the rows so that the row corresponding to the center can be assigned to I_1 and the rows corresponding to the spikes can be assigned to I_2 .

In a similar way we can assign other centers of stars and corresponding spikes. With that we showed that a partition (I_1, I_2) exists, which satisfies the third condition of lemma 3.6.9.

Therefore, $(A^C)^T$ is TU. By remark 3.6.8 we also proved that A^C is TU. \square

Thus, we have proven that in the case of star-shaped conflicts, the matrix of constraints is totally unimodular and therefore, the EBRSC problem with stellar conflicts is equivalent to its LP relaxation. However, if we add the constraints of an upper and/or lower bound, the total unimodularity of the constraint matrix A does not hold anymore. We illustrate this with the following example.

Example 4.3.5. Let eight models PI_j , $j \in [m]$, with $m := 8$, each containing exactly one recommendation R_i , $i \in [n]$, and $n := m = 8$ be given. Furthermore, let the first four recommendations be in conflict to each other—with conflicts taking stellar shape. Here, let recommendation R_1 be the center of the star and R_2 , R_3 , and R_4 its spikes. Let the recommendations R_5 and R_6 be in conflict with each other, and let recommendations R_7 and R_8 not be in conflict with other recommendations. Each recommendation R_i can be activated via an activation variable x_i , $i \in [n]$, with $x_i \in \{0, 1\}$ for all $i \in [n]$. We illustrate the above described models in figure 4.9.

Beside the conflicts, there exists an upper bound $u := 4$ and a lower bound $l := 2$. Then, the set of feasible solutions is given by all $x \in \{0, 1\}^8$ that satisfy $Ax \leq b$,

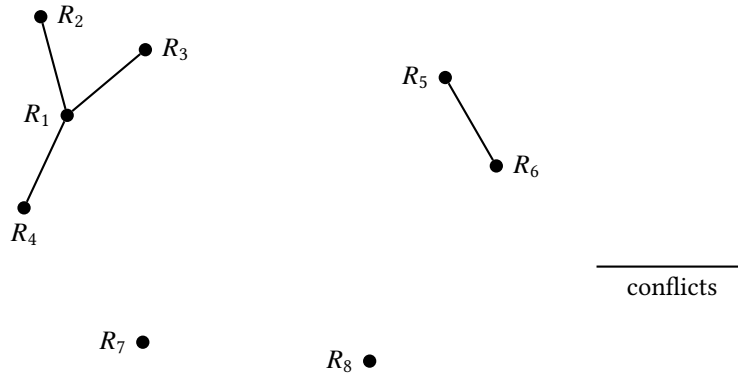


Figure 4.9: Example for stellar conflicts.

more specific

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & & & & & & \\ 1 & & 1 & & & & & \\ 1 & & & 1 & & & & \\ & & & & 1 & 1 & & \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_8 \end{pmatrix} \leq \begin{pmatrix} 4 \\ 2 \\ 1 \\ \vdots \\ 1 \end{pmatrix},$$

with $A \in \{-1, 0, 1\}^{6 \times 8}$ and $b \in \{1, 2, 4\}^6$.

To show that A is not TU, we use the statement of theorem [3.6.11](#). For this purpose, we assume that a nonempty set $I = \{2, 3, 4\} \subset [m]$ is given. A would be TU, if we can find a partition (I_1, I_2) of I , such that equation [\(3.7\)](#) holds.

We can split I as follows.

$$\begin{aligned} I_1 &= \{2\} \text{ and } I_2 = \{3, 4\} \\ I_1 &= \{3\} \text{ and } I_2 = \{2, 4\} \\ I_1 &= \{4\} \text{ and } I_2 = \{2, 3\} \end{aligned}$$

We omit the cases where we just swap the assignment to the partitions by renaming I_1 to I_2 and vice versa. This would lead to the same result of the equation [\(3.7\)](#), but with a different sign.

None of the above defined partitions satisfy

$$\left(\sum_{i \in I_1} a_i \right)^T - \left(\sum_{i \in I_2} a_i \right)^T \in \{-1, 0, 1\}^8.$$

For example, taking $I_1 = \{2\}$ and $I_2 = \{3, 4\}$ we get

$$\begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix} - \begin{pmatrix} 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -3 \\ -2 \\ -2 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \notin \{-1, 0, 1\}^8.$$

Thus, we cannot find a partition (I_1, I_2) for the above defined non-empty index set $I = \{2, 3, 4\} \subset [m]$, so that equation (3.7) is satisfied. And thus, it is shown that the constraint matrix A in general is not TU if one defines an upper and/or lower bound in the constraints besides the stellar conflicts.

Without the total unimodularity, an integer solution is not guaranteed when applying a solving algorithm to the LP relaxation.

The question remains open whether the EBRSC problem with stellar conflicts and lower l and/or upper bound u is in the complexity class \mathbb{P} or \mathbb{NP} . We will not discuss this question in this thesis.

4.3.1.3 Case: No solution

The last case of the EBRSC problem considers scenarios, for which no solution can be found. In these scenarios a special combination of conflicts and lower bounds has the effect that two conflicting recommendations are pronounced, which violates the conflict constraint and, therefore, no feasible solution is given. This particular case occurs in the following scenario.

In general, it is not possible to choose arbitrary combinations of activated recommendations when two or more recommendations are in conflict. This is because as soon as there is a conflict between two recommendations, we have to decide which one to activate. This is expressed by the already known constraint

$$x_i + x_{i'} \leq 1 \quad \forall \{R_i, R_{i'}\} \in C.$$

Here, the recommendations R_i and $R_{i'}$ are in conflict with $i, i' \in [n]$, x_i , and $x_{i'}$ are the corresponding activation variables and C is the set of conflicts defined by equation (4.1). Let S_j and $S_{j'}$ further be the sets of recommendations of the models PI_j and $\text{PI}_{j'}$, respectively—i.e., S_j contains the recommendations of model PI_j and $S_{j'}$ contains the recommendations of model $\text{PI}_{j'}$.

In the defined scenario, we consider two or more models PI_{j^*} , $j^* \in [m^*] \subset [m]$ and $|m^*| > 2$, for which the lower bound is given by $l_{j^*} := |S_{j^*}|$. Thus, it is required that all recommendations of the models PI_{j^*} should be activated.

Furthermore, for at least two models PI_{j^*} and $PI_{j'^*}$, $j^*, j'^* \in [m^*]$, for which all recommendations are to be activated, there exists at least one pair of recommendations $\{R_i, R_{i'}\}$ with $R_i \in S_{j^*}$ and $R_{i'} \in S_{j'^*}$ that is in conflict, i.e., $\{R_i, R_{i'}\} \in C$. We should therefore activate either R_i or $R_{i'}$.

However, according to the definition of the corresponding lower bounds, both R_i as well as $R_{i'}$ must be activated to meet the requirement of the lower bounds l_{j^*} and $l_{j'^*}$, respectively. Thus, we cannot satisfy the requirements of both constraints. And with that, we cannot find a feasible solution of the corresponding EBRSC problem.

For a better understanding we consider the following example.

Example 4.3.6. Suppose that three experts from an insurance company were interviewed—an accounting clerk $j = 1$, a manager $j = 2$, and a customer $j = 3$. Each of these three experts is assigned a model— PI_1 , PI_2 , and PI_3 . And therefore, $m := 3$.

Based on the interviews, we found ten recommendations $S = \{R_1, \dots, R_{10}\}$ and so $n := 10$. The recommendations are distributed among the three models as follows

$$S_1 = \{R_1, R_2, R_3, R_4\}, \quad S_2 = \{R_5, R_6, R_7\}, \quad S_3 = \{R_8, R_9, R_{10}\}.$$

Furthermore, the set of conflicts is given by

$$C = \{\{R_1, R_5\}, \{R_6, R_8\}, \{R_4, R_9\}\}.$$

They are no upper limits defined in the example.

In order to increase customer satisfaction while not losing sight of the company's goals, the lower limits per model are set to

$$l_1 := 1, l_2 := |S_2| = 3, \text{ and } l_3 := |S_3| = 3.$$

The scenario outlined above is illustrated in the following figure [4.10](#).

To satisfy the lower bounds, we need to activate all recommendations of the models PI_2 and PI_3 , which also implies the activation of recommendations R_6 and R_8 . But these are in conflict and therefore, according to the conflict constraint, either R_6 or R_8 can be activated. Thus, an algorithm cannot find a feasible solution for this scenario.

The following lemma shows that there exists no solution, if the scenario of this use case is given.

Lemma 4.3.7. *Let two or more models PI_{j^*} , $j^* \in [m^*] \subset [m]$ and $|m^*| > 2$ be given, where the lower bounds are defined by $l_{j^*} = |S_{j^*}|$ and $l_{j'^*} = |S_{j'^*}|$ for $j^*, j'^* \in [m^*] \subset [m]$ and $m^* > 2$. If there exists a conflict $\{R_i, R_{i'}\} \in C$ with $R_i \in PI_{j^*}$ and $R_{i'} \in PI_{j'^*}$, then, no solution to the problem exists.*

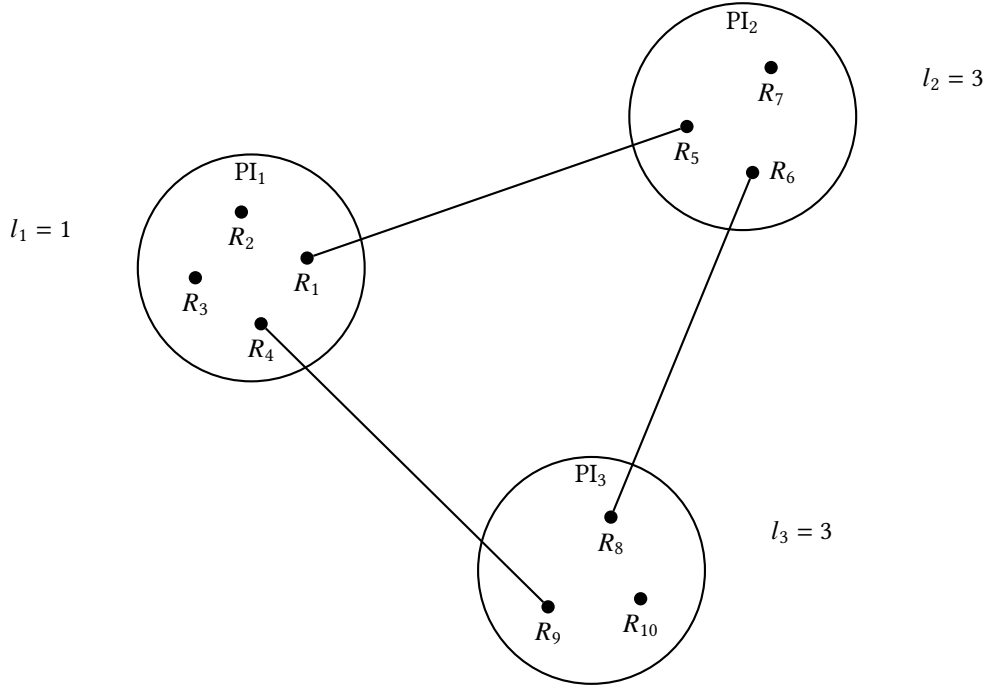


Figure 4.10: Example for the case “No solution” for conflicting recommendations.

Proof. Let w.l.o.g. PI_{j^*} and $PI_{j'^*}$ be two models, for which $l_{*j} := |S_{j^*}|$ and $l_{j'^*} := |S_{j'^*}|$ hold. Furthermore, there exists a conflict $\{R_i, R_{i'}\} \in C$ with $R_i \in PI_{j^*}$ and $R_{i'} \in PI_{j'^*}$.

To satisfy the lower bounds $l_{j^*} = |S_{j^*}|$ and $l_{j'^*} = |S_{j'^*}|$ all recommendations of PI_{j^*} and $PI_{j'^*}$ are activated, i.e., $x_i = 1$ for all $R_i \in S_{j^*}$ and $x_{i'} = 1$ for all $R_{i'} \in S_{j'^*}$. Thus, the inequality of the conflict constraint is no longer satisfied, because it is

$$x_i + x_{i'} = 2 > 1 \quad \forall R_i \in S_{j^*}, R_{i'} \in S_{j'^*}.$$

However, if we meet the inequality $x_i + x_{i'} \leq 1$ for all conflicting pairs of recommendations $\{R_i, R_{i'}\} \in C$, we cannot activate all $R_i \in S_{j^*}$ or $R_{i'} \in S_{j'^*}$. So the constraint belonging to the lower bounds cannot be satisfied.

Thus, no solution to the problem exists. \square

For the complexity theoretical investigation this case is noteworthy as we can prove in finite time that no solution exists. We show that by the following corollary.

Corollary 4.3.8. *The assumptions of lemma [4.3.7](#) can be checked in polynomial time.*

Proof. To prove the corollary, we describe a pseudo-algorithm that can be used to check whether the assumptions for the scenario are met. Further, we show that the running time of this pseudo-algorithm is polynomial.

The pseudo algorithm consists of the following two steps.

- a) In a first step the algorithm has to check if there exist at least two models PI_j and $PI_{j'}$, for which $l_j = |S_j|$ and $l_{j'} = |S_{j'}|$. Therefore, the algorithm has to compare for all m models, if $l_j = |S_j|$ for all $j \in [m]$.

To do so, the algorithm must first count all recommendations contained in each set S_j , $j \in [m]$. Since each set S_j contains at most n recommendations—in a worst case this can be done in $O(n)$.

For each model, secondly, the algorithm must check whether $l_j = |S_j|$. l_j as well as $|S_j|$ are natural numbers, and by \hat{n} we denote the longest number of digits of l_j and $|S_j|$, respectively, for all $j \in [m]$. In a worst case it takes $O(\hat{n})$ to check, if both natural numbers are equal.

In case it is $l_j = |S_j|$ the algorithm must remember it and continue counting in the background. If the count is greater than or equal to two, the checking algorithm proceeds, otherwise it stops.

The algorithm for this step has, in the worst case, a running time of $O(n + \hat{n} + 1)$.

- b) Let w.l.o.g. PI_j and $PI_{j'}$ be two models for which $l_j = |S_j|$ and $l_{j'} = |S_{j'}|$ holds. For all sets of conflicting pairs of recommendations, the algorithm checks if there exists a pair $\{R_i, R_{i'}\} \in C$ for which $R_i \in S_j$ and $R_{i'} \in S_{j'}$ or vice versa. Both sets of recommendations S_j and $S_{j'}$ contain at most n elements.

Let $c := |C| \in \mathbb{N}$. Then the algorithm must check for each recommendation per conflicting pair $\{R_i, R_{i'}\} \in C$, if it is contained in S_j or $S_{j'}$. Thus, the algorithm of step [b](#)) runs in $O(4 \cdot c \cdot n) = O(c \cdot n)$.

We have m models and for each model we have to run the above steps. Therefore, the total running time of the algorithm is $O(m \cdot ((n + \hat{n} + 1) + c \cdot n)) = O(m \cdot ((c + 1) \cdot n + \hat{n} + 1))$. So the running time of the pseudo algorithm is polynomial. \square

Of course, this scenario described above is one of many where the combination of lower bounds and conflicting recommendations leads to an intractable problem, all of which we will not list in this thesis. However, they are of great importance, especially in the practical context. When choosing lower bounds, we need to make sure that a feasible solution is still possible. In this case, we can also prove this in polynomial time

4.3.2 Existence of dependent but no conflicting recommendations

In the proof of theorem [4.2.13](#) we showed that the EBRS problem is NP -complete. To prove the NP -completeness we reduced the EBRS problem and excluded the consideration of dependent recommendations, i.e., the set of dependent recommendations was empty— $D = \emptyset$. In the latter subsection, we considered the EBRS

4.3 Complexity-theoretical investigations of the EBRs problem

problem taking into account only conflicting recommendations and looked for cases of the so-called EBRSC problem that can be solved in polynomial time.

In this subsection, we again look for cases of the EBRs problem that can be solved in polynomial time, considering dependencies between the recommendations but not conflicts. The goal of this subsection can be formulated as the following question.

Is the EBRs problem still NP -complete when there are dependent recommendations but no conflicts between them?

Dependent recommendations occur when experts recommend the same action or when recommendations from different models influence each other. The following example is intended to illustrate the concept of dependent recommendations in the context of insurance theory.

Example 4.3.9. We consider the processing of a motor vehicle claim where in special situations one expert—a clerk—recommends an estimate for repairing the car and another expert—a business manager—proposes to review all bills, estimates, and the charge of an expert in general. The recommendation of the business manager can, of course, only be activated, if there exists a bill, an estimate, or a charge of an expert.

Given that such a special situation occurs, the recommendation of the clerk is activated, and by activating the recommendation also the recommendation of the business manager has to be activated, i.e., the estimate should be reviewed. The manager's recommendation is thus dependent on the recommendation of the clerk.

In the insurance context, cases where several experts recommend the same action seem very likely—making the complexity-theoretical consideration of a recommendation system with dependent recommendations relevant. In particular, it would be beneficial to show that the EBRs problem without conflicts but with dependencies can be solved in polynomial time. However, we will prove that $\text{EBRS} \in \text{NP}$. This is even true in the case when experts recommend the same actions, i.e., when a recommendation is included in two or more models.

In order to better distinguish different cases of the EBRs problem, we refer to the EBRs problem with dependent but not conflicting recommendations as the EBRSD problem. By proving the following theorem the NP -completeness is shown.

Theorem 4.3.10. *The decision problem EBRSD is NP -complete for $D \neq \emptyset$ and $C = \emptyset$.*

Proof. To prove NP -completeness, we have to show that **a)** $\text{EBRSD} \in \text{NP}$, and **b)** EBRSD is NP -hard.

a) “EBRSD $\in \mathbb{NP}$ ”:

We already showed that EBRSD $\in \mathbb{NP}$ in theorem 4.2.13. So we found a polynomial time verifier for the problem, i.e., we can verify in polynomial time if a given certificate C is a solution to EBRSD. Without the constraint of conflicting recommendations, EBRSD is a restriction to EBRSD and, therefore, we can also find a polynomial time verifier for EBRSD. Thus, EBRSD $\in \mathbb{NP}$.

b) “EBRSD is \mathbb{NP} -hard”:

Similar to theorem 4.2.13 we use Karp reduction to show \mathbb{NP} -hardness. For more details on Karp reduction, we refer to definition 3.3.13. We therefore will polynomially reduce the perfect matching problem PM to EBRSD, i.e., $\text{PM} \leq_p \text{EBRSD}$. We have discussed the perfect matching problem in subsection 3.3.4.3. To prove that such a polynomial reduction exists, we will find a function that transforms a given instance of PM to a single instance of EBRSD in polynomial time and prove that both instances result in the same answer.

As an input of the PM problem an undirected graph $G = (V, E)$ with $m := |V|$ and $n := |E|$ and an integer $k \in \mathbb{N}_0$ is given (see problem 3.3.33). The transformation interprets every vertex $v_j \in V$ as a model PI_j for all $j \in [m]$. Each edge $e_i \in E$ with $v(e_i) = \{v_j, v_{j'}\} \in E$ is transformed to a recommendation $R_i \in S$ for all $i \in [n]$. Therefore, the set of recommendations is $S = \{R_1, \dots, R_n\}$ and $|S| = |E|$. There exist subsets $S_1, \dots, S_m \subseteq S$ and each set $S_j, j \in [m]$, consists of the edges that are linked to the corresponding vertices v_1, \dots, v_m . We interpret the connection of the vertices v_j and $v_{j'}$ by the edge $e_i = v(e_i) = \{v_j, v_{j'}\}$ as a recommendation R_i which is contained in PI_j as well as in $\text{PI}_{j'}$. Thus, we have $R_i \in S_j \cap S_{j'}$ and S_j as well as $S_{j'}$ are the sets of recommendations of PI_j and $\text{PI}_{j'}$, respectively.

Let M be a matching of size $|M| = k$. Then, $e_i \in M$ is interpreted as an activation of the recommendation R_i by setting the activation variable $x_i = 1$. If $e_i \notin M$ then the corresponding recommendation R_i is not activated and $x_i = 0$.

To guarantee that a perfect match is found the upper bounds u_j and the lower bounds l_j of every model $\text{PI}_j, j \in [m]$, are defined to be equal to one. Furthermore, the weight of every recommendation is equal to one, which is why $\gamma_i = 1$ for all $i \in [n]$.

The dependencies are given by activating the same recommendation of different models. That means, if we activate recommendation R_i we have to activate R_i in PI_j as well as in $\text{PI}_{j'}$ if the corresponding edge e_i connects the vertices v_j and $v_{j'}$, and the dependency set is given by

$$D = \{R_i \in S | R_i \in S_j \cap S_{j'} \neq \emptyset \text{ for } i \in [n] \text{ and } j, j' \in [m]\}.$$

Finally, no upper u and no lower bound l is defined.

Similar to the proof of theorem [4.2.13](#), the transformation is computable in polynomial time, since it consists of one-to-one translations. Because each vertex v_j of the undirected graph is transformed to a model PI_j for all $j \in [m]$ and each edge e_i is interpreted as a recommendation R_i for all $i \in [n]$ (and $n := |E| = |S|$).

We show the correctness of this transformation by proving the following claim. The claim also shows that an instance of PM and an instance of EBRSD have the same answer by applying the assumed subroutine for EBRSD.

Claim: Let $k \in \mathbb{N}_0$ (with $k \leq n$). An undirected graph $G = (V, E)$ has a perfect matching M of size at least k if and only if the sum over all activation variables of the corresponding recommendations is at least k under consideration of the given dependencies and the lower and upper bounds.

“ \Rightarrow ” Let M be a perfect matching of graph G with size at least k , i.e., $|M| \geq k$. Using this assumption, we show that the sum over all activation variables is also at least k under consideration of the constraints.

Every edge e_i is transformed to a recommendation R_i for all $i \in [n]$ and we know if $e_i \in M$ then the corresponding recommendation R_i is activated, and $x_i = 1$ according to the transformation. No two edges in M share a common vertex, which means that at most one recommendation per model PI_j is activated and, therefore, the constraints $u_j = 1$ for all $j \in [m]$ are satisfied. As M is a perfect match all vertices in the graph are linked to an edge of the matching and, thus, at least one recommendation is activated in every model PI_j and the constraints $l_j = 1$ are met for all $j \in [m]$. With $e_i \in M$ the recommendation $R_i \in S_j \cap S_{j'} \neq \emptyset$ is activated and the dependency constraint is satisfied, too. Finally, with $\gamma_i = 1$ for all $i \in [n]$ the sum over all activation variables is truly k or higher.

“ \Leftarrow ”: We assume that the sum over all activation variables is at least k and the constraints are met. Under this assumption we prove that there exists a perfect matching M of at least k edges for an undirected graph $G = (V, E)$ with $n = |E|$ edges $e_i, i \in [n]$, and $m = |V|$ vertices $v_j, j \in [m]$.

Let M consist of all activated recommendations R_i . With the activation of R_i the activation variable x_i is set to one in the models PI_j and $PI_{j'}$ due to the dependencies. Because of $u_j = 1$ for all $j \in [m]$, at most one variable can be set to one per model $PI_j, j \in [m]$. So no other recommendation can be activated in PI_j or $PI_{j'}$, respectively. Therefore, as every model corresponds to a vertex of the undirected graph no other edge is contained in M , which is linked to the vertex v_j or $v_{j'}$. Since the lower bound $l_j = 1$ for all $j \in [m]$, each vertex v_j is guaranteed to have an adjacent edge. Thus, the set M is a perfect match.

As the sum over all activation variables is at least k and $\gamma_i = 1$ for all $i \in [n]$, M has cardinality $|M| \geq k$. Therefore, M is a perfect match of size at least k .

In summary, it is shown that the EBRSD problem is NP-complete. □

We now know that the EBRSD problem is NP-complete in general. Since we consider this case of the EBRSD problem to be very likely, the question now is whether we can find cases where the EBRSD problem is solvable in polynomial time. As already mentioned, we could not find such cases in the context of this work. However, we think that further analysis could be quite interesting.

In the following subsection, we show that there are cases of the EBRSD problem for which no feasible solution exists—similar to the EBRSC problem.

4.3.2.1 Case: No solution

Similar to subsection [4.3.1.3](#), we also can find scenarios with dependent recommendations, for which in combination with lower and upper bounds no feasible solution can be found. Above all, these scenarios show that care must be taken when setting upper and lower bounds to ensure that they fit the given interdependencies between recommendations. By pronouncing a dependent recommendation, one or more other recommendations are activated, so an upper or lower bound may need to be adjusted accordingly. Otherwise, a constraint may not be met.

We give an example for such a scenario, but do not intend to list all possible scenarios for which no feasible solution exists. Similar to the case of conflicting recommendations, also in the case of dependent recommendations we can check in polynomial time if the lower and upper bounds are not set such that the algorithm cannot find a feasible solution—and refer to the proof of corollary [4.3.8](#) in subsection [4.3.1.3](#). Similar to that corollary, we can show that there is no feasible solution in the EBRSC problem.

One possible scenario, in which no feasible solution can be found, is given by two models, for which dependent recommendations exist. Let PI_1 and PI_2 be such two models and let

$$D' = \{(R_i, R_{i'}) \in S_1 \times S_2 \mid R_{i'} \text{ requires } R_i \text{ with } i, i' \in [n]\} \quad (4.6)$$

be the set of dependent recommendations between PI_1 and PI_2 . More precisely, D' is a set of directed recommendations, in which some recommendations of PI_2 require recommendations of PI_1 . Furthermore, let $d' := |D'| \in \mathbb{N}_0$ be the number of dependent recommendations between PI_1 and PI_2 . D' is defined by pairwise dependencies between recommendations. And therefore, if we activate one recommendation of a pair, we also activate the other one. So in case of activation of d' recommendations

(one of each pair), all corresponding recommendations are activated, too. And with that there are $2 \cdot d'$ recommendations activated.

For the scenario we further define that w.l.o.g. all recommendations of PI_1 should be activated and therefore, $l_1 = |S_1|$.

We cannot find a feasible solution, if the overall upper bound is defined by

$$u < 2 \cdot d' + (l_1 - d') = l_1 + d'.$$

Of course, a similar scenario can be defined with interacting recommendations. The following example illustrates a scenario where no feasible solution exists.

Example 4.3.11. In this example our goal is to increase the customer satisfaction and, therefore, all recommendations of the customers should be pronounced. Let PI_1 be the model of the customers with four recommendations $S_1 = \{R_1, R_2, R_3, R_4\}$. To increase the customer satisfaction we want to force that all recommendations of PI_1 are activated and, therefore, $l_1 = 4$. Let there be a second model PI_2 with six recommendations $S_2 = \{R_5, \dots, R_{10}\}$. Furthermore, the set of dependent recommendations is given by

$$D' = \{(R_2, R_5), (R_3, R_6), (R_4, R_7)\}.$$

To keep the claims processing manageable for the clerk, we define an overall upper bound with $u = 6 < 4 + 3 = l_1 + d'$. The defined scenario can be illustrated as shown in figure 4.11.

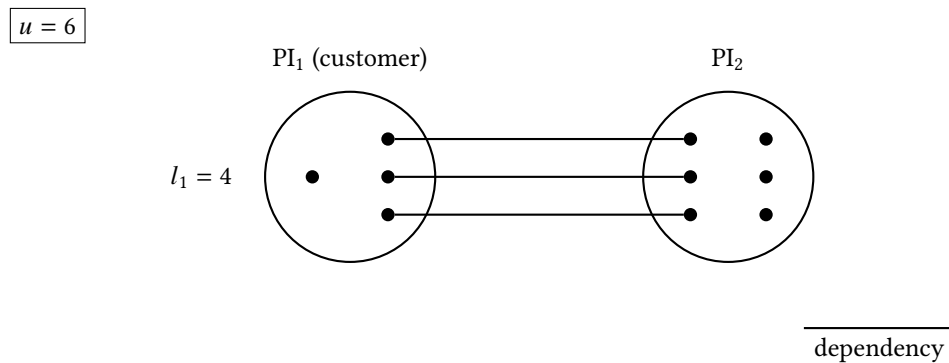


Figure 4.11: Example for the case “No solution” for dependent recommendations.

By activating all recommendations of PI_1 as it is demanded by $l_1 = 4$, we activate the recommendations R_1, \dots, R_7 , i.e., seven recommendations. We can activate no less than these seven recommendations, because in addition to $l_j = 4$, we must also consider the dependencies to the recommendations of the model PI_2 , and, therefore, R_5, R_6 , and R_7 have to be activated. Yet, this is in contradiction to $u = 6$. Thus, no feasible solution can be found.

Similar to this example, we can show that for the more general scenario no feasible solution exists.

Lemma 4.3.12. *Let PI_1 and PI_2 be two models, for which a set of dependent recommendations exists. The set of dependent recommendations is defined by equation (4.6) with $d' := |D'|$. Furthermore, w.l.o.g. the lower bound for model PI_1 is defined by $l_1 := |S_1|$, where S_1 is the set of recommendations of model PI_1 . For the total upper bound $u \in \mathbb{N}_0$ it holds that $u < 2 \cdot l_1 + d'$. Then, there exists no feasible solution.*

Proof. With $l_1 = |S_1|$ we have to activate all recommendations of model PI_1 . With the definition of D' , there are d' recommendations in PI_2 that depend on d' recommendations in PI_1 . With that, we have to activate $l_1 + d'$ recommendations, i.e., $\sum_{i=1}^n x_i = l_1 + d'$. With $u \in \mathbb{N}_0$ and $u < l_1 + d'$, we know that $u \leq l_1 + d' - 1$. But

$$\sum_{i=1}^n x_i = l_1 + d' > l_1 + d' - 1 \geq u$$

which is a contradiction to the constraint $\sum_{i=1}^n x_i \leq u$. And therefore, we cannot find a feasible solution to the defined scenario. \square

4.3.3 No conflicts and no dependencies

In this last subsection of the complexity-theoretical investigation, we turn to the case in which there are neither conflicts nor dependencies between the recommendations. Since no conflicts or dependencies have to be taken care of when activating the recommendations, but only the compliance with lower and upper bounds has to be considered, the conjecture is that such cases of the EBRS problem can be solved in polynomial time.

In fact, we can show that for the EBRS problem with constraints consisting only of lower and upper bounds, there exists a polynomial time algorithm to solve the problem. We discuss it in subsection (4.3.3.1).

However, we also want to discuss another case of the EBRS problem, where a weighted sum of all activation variables per model cannot exceed a certain upper bound. Such a constraint is useful and necessary if, for example, the influence of the recommendations on the quality of the claims processing should be evenly distributed among all models. In which we add a weighting of the activation variables into our model, the considered problem is no longer solvable in polynomial time, but NP-complete.

In subsection (4.3.3.1) we start with the discussion about the polynomial-time solution algorithm for the case where—without weighting—the upper and lower bounds must be satisfied. Following that, we describe the special case of a limited weighted sum per model PI_j , $j \in [m]$ in subsection (4.3.3.2).

4.3.3.1 Case: Only bounds

In the special case of the EBRs problem discussed here, there are no dependencies or conflicts between recommendations that constrain the maximization of quality, only lower and upper bounds. Therefore, we call it *Only Bounds* or OB problem.

Beside there are no dependencies between different recommendations R_i and $R_{i'}$, we also assume that no recommendation is included in more than one model, i.e., if $R_i \in S_j$ than $R_i \notin \{S_1, \dots, S_{j-1}, S_{j+1}, \dots, S_m\}$.

We prove, that there exists a polynomial time algorithm to solve the OB problem.

In a first step, we set up the program of the OB problem. We just mentioned that the OB problem is a restricted version of the EBRs problem without consideration of dependent or conflicting recommendations. Therefore, the program is given by

Program 4.3.13 (OB).

$$\begin{aligned}
 & \max \sum_{i=1}^n \gamma_i x_i \\
 \text{s.t.} \quad & l \leq \sum_{i=1}^n x_i \leq u \\
 & l_j \leq \sum_{R_i \in S_j} x_i \leq u_j \quad \forall j \in [m] \\
 & x_i \in \{0, 1\} \quad \forall i \in [n]
 \end{aligned}$$

with $\gamma_i \in \mathbb{R}$ for $i \in [n]$ and $l, u, l_j, u_j \in \mathbb{N}$ for $j \in [m]$.

In a second step, we describe the algorithm for solving the program. As already mentioned, there are no dependent recommendations and also no recommendations that exist in more than one model. Therefore, the sets of recommendations per model S_1, \dots, S_m build a partition of S , i.e., $S = S_1 \cup \dots \cup S_m$ and $S_1 \cap \dots \cap S_m = \emptyset$.

The algorithm which solves the OB program is a so-called *Greedy algorithm*. We will briefly explain what is generally meant by Greedy algorithms in the following.

Greedy algorithm is a general term for algorithms described by Dasgupta and colleagues [DPV06, p. 139] as follows.

“Greedy algorithms build up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit.”

In other words, the Greedy algorithm chooses in each step the best choice without considering any step before or after this step. This of course does not guarantee a global optimal solution in general. But there are problems for which such a global optimal solution exists. For example, a famous class of problems, for which the

Greedy algorithm finds a global optimal solution are problems with the property of a so-called *matroid*. This was proven by Rado [Rad57] and Edmonds [Edm71]. We will not discuss this in this thesis, but refer the interested reader for more details on Greedy algorithms and matroids to the works of Korte and Vygen [KV06], Dasgupta and colleagues [DPV06], Kleinberg and Tardos [KT13], and Cormen and colleagues [CLRS09].

The Greedy algorithm to solve the OB problem is given as follows.

Greedy Algorithm.

Algorithm 4.3.14. **a)** Sort the weights $\gamma_1, \dots, \gamma_n$ of the recommendations in descending order (independent of the model they are contained in) and number the recommendations, the activation variables, as well as the weights according to the new sorting by creating a second index $k \in \mathbb{N}$. With $k = 1$ denotes the recommendation with the highest weight and $k = n$ denotes the recommendation with the lowest weight. If two or more recommendations have the same weight, choose the order arbitrarily.

- b)** Meet the demands of the lower bounds l_1, \dots, l_m of the models by activating the recommendations in descending order of the corresponding weights accordingly.
- c)** For $k \in [n]$ iterate the following steps, starting by $k = 1$.
 - ca)** If $x_{i,k} = 1$, go a step **cf)**. Otherwise:
 - cb)** Activate recommendation $R_{i,k}$, i.e., $x_{i,k} = 1$.
 - cc)** Find model PI_j , $j \in [m]$, to which $R_{i,k}$ belongs.
 - cd)** Check, if the sum over the activated recommendations is greater or equal to the total lower bound l . If not, go to step **cf)**.
 - ce)** Check, if $\gamma_{i,k} \geq 0$. Otherwise, stop the process.
 - cf)** Check, if the sum over all activated recommendations of model PI_j (see step **cc)**) is lower or equal to the upper bound u_j of the corresponding model PI_j , $j \in [m]$. If the upper bound is not satisfied, deactivate recommendation $R_{i,k}$, i.e., $x_{i,k} = 0$ and go to step **ch)**. Otherwise:
 - cg)** Check, if the the sum over all activated recommendations is lower or equal to the overall upper bound u . If u is exceeded, deactivate $R_{i,k}$ and stop the process. Otherwise:
 - ch)** Activate $x_{i,k+1}$ and go to step **ca)**. If $k = n$ stop the process.

We know, that there exist no dependencies and no conflicts between the recom-

recommendations and, furthermore, S_1, \dots, S_m is a partition of S . So we do not have to consider any lower and upper bounds from other models when activating a recommendation. Moreover, as OB is a maximization problem, an algorithm would activate as much positive weighted recommendations until an upper bound is reached. Of course, if the algorithm has to choose between a higher weighted recommendation and a lower weighted recommendation, it would choose the higher weighted recommendation. So for example, in case no upper bounds exist and $\gamma_i \geq 0$ for all $i \in [n]$, an algorithm would activate all recommendations.

If we consider all this aspects together, the application of a Greedy algorithm to solve the OB problem seems appropriate. In the following, we will justify the appropriateness and describe in more details each step of the algorithm defined above.

It is plausible to sort the weights of the recommendations in descending order for solving the maximization problem. This allows us to generate an algorithm that works through the maximization step by step until an upper bound is reached.

Since S_1, \dots, S_m is a partition of S we can sort the weights without considering the assignment to the models. We apply the sorting algorithm and label the now re-sorted weights, associated recommendations, and activation variables with a new index $k \in \mathbb{N}$ in order to still know the original index $i \in [n]$, and, thus, the assignment to the respective model.

In case there exists only upper bounds in the OB problem, an algorithm would activate as much positive weighted recommendations as possible until the upper bounds of the models are exceeded or until the weighted sum over all recommendations does not increase anymore, as $\gamma_{i,k} < 0$. So the difficulty of solving this problem is given by the compliance with the lower bounds, especially when we have to consider them at every step where a recommendation is activated. Therefore, we satisfy the lower bounds of the models l_1, \dots, l_m in the second step of the algorithm. This means that we no longer have to take them into account during further processing.

Since we already have activated recommendations, we need to iterate the activation of additional recommendations step by step considering the overall lower bound l , the upper bounds of the models u_1, \dots, u_m , as well as the overall upper bound u . Therefore, we create a loop that proceeds according to the index $k \in [n]$ with the following steps.

In a first step, we need to check if the recommendation is already activated, i.e., we check whether $x_{i,k} = 1$. If this is true, the activation was necessary to satisfy a lower bound l_j and we can proceed immediately to the next step $k + 1$. Otherwise, we activate recommendation $R_{i,k}$ by setting $x_{i,k} = 1$ and check in the further steps whether we violate any of the other bounds by activating it. Therefore, we first look for the corresponding model PI_j by searching for the model set S'_j to which $R_{i,k}$ belongs (see step **cc**) of the algorithm).

The next steps **cd)** and **ce)** ensure that recommendations with negative weight are only activated if this is necessary due to the total lower bound. Otherwise, the algorithm excludes this by checking whether the activation of this recommendation actually increases the total weighted sum.

We check in step **cf)** of the algorithm whether the sum over all activated recommendation of the model PI_j is lower or equal to the upper bound u_j of the model PI_j to which $R_{i,k}$ belongs. If the upper bound is exceeded, we need to deactivate the recommendation, i.e., set $x_{i,k} = 0$, otherwise, the solution is not feasible.

In step **cg)** of the algorithm, we check whether the total upper bounds are satisfied. If u is exceeded, we deactivate the recommendation and the process stops.

In lemma [4.3.16](#) we show that this indeed leads to a global optimal solution. To better understand the operation and procedure of the algorithm, we first give the following example.

Example 4.3.15. Let the set of recommendations $S = \{R_1, \dots, R_{12}\}$ and the models PI_1, \dots, PI_4 be given. Let the sets of recommendations per models be given by

$$\begin{aligned} S_1 &= \{R_1, R_2\}, \\ S_2 &= \{R_3, R_4, R_5\}, \\ S_3 &= \{R_6, R_7, R_8, R_9\}, \\ S_4 &= \{R_{10}, R_{11}, R_{12}\}. \end{aligned}$$

The corresponding weights of the recommendations are defined as follows.

$$\begin{array}{lll} \gamma_1 = -5, & \gamma_5 = 4, & \gamma_9 = 1, \\ \gamma_2 = 10, & \gamma_6 = 10, & \gamma_{10} = 7, \\ \gamma_3 = -10, & \gamma_7 = 11, & \gamma_{11} = 5, \\ \gamma_4 = 5, & \gamma_8 = -5, & \gamma_{12} = -1. \end{array}$$

For the models the following lower and upper bounds are given.

$$\begin{array}{ll} l_1 = 2, & u_1 = 2, \\ l_2 = 0, & u_2 = 1, \\ l_3 = 1, & u_3 = 3, \\ l_4 = 0, & u_4 = 2. \end{array}$$

Finally, the overall lower bound is given by $l = 3$ and the upper bound is defined by $u = 7$.

We follow the instructions of algorithm [4.3.14](#).

- a) We obtain the following descending list of recommendations (including the running index k). In parentheses we additionally give the associated weight.

$$\begin{array}{lll} R_7 = R_{7,1}(11), & R_{11} = R_{11,5}(5), & R_{12} = R_{12,9}(-1), \\ R_6 = R_{6,2}(10), & R_4 = R_{4,6}(5), & R_1 = R_{1,10}(-5), \\ R_2 = R_{2,3}(10), & R_5 = R_{5,7}(4), & R_8 = R_{8,11}(-5), \\ R_{10} = R_{10,4}(7), & R_9 = R_{9,8}(1), & R_3 = R_{3,12}(-10). \end{array}$$

- b) We satisfy the demands of the lower bounds l_1, \dots, l_4 and activate the recommendations $R_{7,1}$, $R_{2,3}$ and $R_{1,10}$, i.e., $x_{7,1} = x_{2,3} = x_{1,10} = 1$.
- c) We repeat the following steps **ca)** to **cf)** of algorithm 4.3.14 until the process stops. We do not describe each step, but give an example of the first two iteration steps of the loop.

Iteration step 1:

- ca)** As $x_{7,1} = 1$, proceed with $k = 2$ beginning again with step **ca)**.

Iteration step 2:

- ca)** $x_{6,2} = 0$.
- cb)** Activate $R_{6,2}$, i.e., $x_{6,2} = 1$.
- cc)** Search the corresponding model. With $R_{6,2} \in S_3$ the model is PI_3 .
- cd)** With $l = 3$ and $x_{7,1} = x_{2,3} = x_{1,10} = 1$ the total lower bound is satisfied. Therefore, we go to step **ce)** of the algorithm.
- ce)** It is $\gamma_{6,2} = 10 > 0$ and, thus, the activation of recommendation $R_{6,2}$ increases the influence on the quality.
- cf)** Check the upper bound of model PI_3 , i.e., $\sum_{R_{i',k'} \in S_3} x_{i',k'} = 2 < 3 = u_3$.
- cg)** Check the overall upper bound, i.e., $\sum_{i=1}^n x_{i',k'} = 4 < 7 = u$.
- ch)** The process does not stop and recommendation $R_{2,3}$ is selected.
- d)** The process stops after six steps and activates the recommendations $R_{7,1}$, $R_{2,3}$, $R_{1,10}$, $R_{6,2}$, $R_{10,4}$, $R_{11,5}$, and $R_{4,6}$. Finally, the quality is

$$\sum_{i=1}^{12} \gamma_i x_{i,k} = 42.$$

With the following lemma we show that the algorithm 4.3.14 actually maximizes the program 4.3.13.

Lemma 4.3.16. *The algorithm 4.3.14 runs in polynomial time and provides an optimal solution to program 4.3.13*

Before we prove lemma 4.3.16, we give a short introduction to the so-called *Mergesort* algorithm, which is used in the proof of the lemma. We do not describe the algorithm in detail and refer the interested reader to the work of Dasgupta and colleagues [DPV06] for more details on Mergesort.

Mergesort is one of the *Divide and Conquer* algorithms. In short, the algorithm divides the sorting problem into subproblems (in case of Mergesort into subproblems containing just one element) and solve the subproblems. Subsequently, two subproblems are merged together, so that the resulting array (data elements of the same data type—here a list of weights) is still sorted. This procedure is repeated until the input array is finally sorted.

We now prove lemma 4.3.16.

Proof. In a first step, we prove that the algorithm runs in polynomial time.

In the context of this thesis, we consider it sufficient to illustrate the necessary steps of the prove. Further, it is not the goal of this chapter to find an efficient algorithm, but to show that a polynomial algorithm exists.

The steps of the algorithm 4.3.14 are as follows.

- a) *Sorting the weights of the recommendations in descending order and number the sorted recommendations, the activation variables, as well as the weights by a new index $k \in \mathbb{N}$:*

When applying, e.g., the *Mergesort* algorithm to sort the weights, this can be done in time complexity $O(n \log n)$. Adding a new (second) index to each element of the list of recommendations, is done in time complexity $O(n)$ with $n = |S|$. In total, this step runs in $O(n + n \log n)$.

- b) *Meet the demands of the lower bounds of the models l_1, \dots, l_m :*

We denote by S' the set of sorted recommendations and by

$$S'_j := \{R_{i'_1, k'_1}, \dots, R_{i'_{n_j}, k'_{n_j}}\}$$

the set of sorted recommendations of model PI_j . The number of recommendations in the sorted model set S'_j is given by $n_j := |S'_j|$.

For each $j \in [m]$ the algorithm activates the first $l_j \in \mathbb{N}_0$ recommendations of the sorted list S'_j . Therefore, it needs to look up and activate the first l_j elements of the list. With $l_j \in \mathbb{N}_0$ a constant, this takes $O(l_j) = O(1)$ time.

Overall, the time complexity of this step is $O(m)$, since the algorithm must perform this step m times.

- c) Iterate over all recommendations $R_{i,k}$ for $i, k \in [n]$ to find those recommendations that need to be activated such that the overall lower bound as well as all upper bounds are met:

The third step of the algorithm describes a loop. For each $k \in [n]$, the algorithm goes through the following steps.

- ca) Check, if $x_{i,k} = 1$:

Checking $x_{i,k} = 0$ or $x_{i,k} = 1$ can be done in $O(1)$.

- cb) Activate $x_{i,k}$:

To activate $x_{i,k}$ the algorithm assigns 1 to it—which can be done in $O(1)$.

- cc) Find the model to which $R_{i,k}$ belongs to:

The algorithm has to check for each set of recommendations of the corresponding model S_j , $j \in [m]$, if $R_{i,k}$ belongs to it. In a worst case, all m model sets have to be checked. In addition, each entry of the model set S'_j has to be checked. But as S_1, \dots, S_m and also S'_1, \dots, S'_m is a partition of S (the assignment to the models was not changed), the algorithm has to check at most n entries. Therefore, the algorithm is $O(m \cdot n)$.

- cd) Check, if the total lower bound l is already satisfied:

For that, the algorithm has to calculate $\sum_{i,k}^n x_{i,k}$ and compare the result to the total lower bound l . The calculation of the sum and the comparison with l takes $O(n + 1)$ time.

- ce) Check, if the corresponding weight is negative:

It can be checked in $O(1)$, if $\gamma_{i,k} \leq 0$.

- cf) Check, if the upper bound u_j , $j \in [m]$, is met:

With step cc) we know the index of the model $j \in [m]$ to which the activated recommendation $R_{i,k}$ belongs. We sum the activation variables of S'_j , and S_j respectively, i.e., we calculate

$$\sum_{R_{i',k'} \in S'_j} x_{i',k'},$$

and compare the result to u_j . The calculation of the sum takes $O(n)$ in a worst case, since $x_{i,k} \in \{0, 1\}$. The comparison takes $O(1)$. In total, this step is done in $O(n + 1)$.

cg) Check, if u is exceeded:

In a first step, calculate the sum

$$\sum_{i'=1}^n x_{i',k'}.$$

Similar to step **cd)** the calculation of the sum and the comparison takes $O(n + 1)$.

ch) Select the activation variable of recommendation $R_{i,k+1}$:

As we have a sorted list, we select the next element of the list with index $k + 1$. This can be done in $O(1)$ time.

In total, the running time of the loop is asymptotically

$$\begin{aligned} O(n \cdot (1 + 1 + m \cdot n + (n + 1) + 1 + (n + 1) + (n + 1) + 1)) &= O((m + 3)n^2 + 9n) \\ &= O((m + 1)n^2 + n). \end{aligned}$$

In total, the algorithm consist of steps, which run in polynomial time and therefore, the whole algorithm runs in polynomial time. In a worst case, the running time of the algorithm is $O((m + 1)n^2 + 2n + n \log n + m)$.

In a second step, we prove that the algorithm finds an optimal solution.

Given the recommendations are already sorted in descending order, we exclude w.l.o.g. the case where the recommendations $R_{i_1,k}, \dots, R_{i_p,k+p}$ with $p \subset n$ have equal weights, i.e., $\gamma_{i_1,k} = \dots = \gamma_{i_p,k+p}$, and the algorithm stops within steps $k, \dots, k + p$. In this case there would be no unique solution, because if the weights are equal, the algorithm will sort the corresponding recommendations arbitrarily (see step **a)** of the algorithm [4.3.14](#)). We can exclude such cases in our consideration, since they do not affect the quality $\sum_{i=1}^n \gamma_i x_i$ of the procedure.

The OB program is a (linear) maximization problem and therefore, the goal of the algorithm is to activate as much recommendations with positive weight $\gamma_i > 0$, $i \in [n]$, as possible. Without restrictions by the lower and upper bounds, it would activate all recommendations with positive weight—note that there exists no conflict or dependencies. However, due to the lower bounds per model l_1, \dots, l_m and the total lower bound l , the algorithm may be forced to activate recommendations with negative weights $\gamma_i \leq 0$, $i \in [n]$ as well (see also example [4.3.15](#)), and the upper bounds limit the activation of all positively weighted recommendations.

A maximal solution of the algorithm is found if the recommendations with the highest (positive) weighting are activated—taking into account all upper and lower bounds.

Therefore, the Greedy algorithm considered checks at each step whether it can activate a recommendation with positive weight, i.e., $\gamma_{i,k} > 0$, or whether it has to activate a recommendation with negative weight to satisfy a lower bound. Since the Greedy algorithm proceeds stepwise, by sorting the recommendations in descending order of their weights, we guarantee that the recommendations with the highest positive weights are activated first.

In the second step of algorithm [4.3.14](#) we activate all recommendations per model in descending order to satisfy the requirement of lower bounds l_1, \dots, l_m . By sorting, we ensure that the recommendations with the highest positive weights per model PI_j are activated. Of course, it is possible that in order to satisfy the lower bounds, the algorithm must activate a recommendation with negative weighting.

In the next steps of the loop, the algorithm checks in each step whether the total lower bound l is already satisfied (i.e., l recommendations are activated). Only if l is not satisfied, the algorithm does not check whether the currently considered recommendation $R_{i,k}$ has a weight greater than zero, i.e., $\gamma_{i,k} > 0$. That is, only to satisfy the lower bounds, it is allowed to activate recommendations with negative weights.

In case l is fulfilled, the algorithm activates recommendations until

- a) u or $u_j, j \in [m]$, is reached, or
- b) $\gamma_{i,k} \leq 0$ for $i, k \in [n]$.

If u is reached or the considered recommendation has weighting $\gamma_{i,k} \leq 0$, the algorithm stops. Otherwise, it continues.

Thus, we have shown that activating the recommendation with the highest weight leads to an optimal solution unless a constraint is violated. \square

4.3.3.2 Case: Limited weighted sum per model $PI_j, j \in [m]$

In a real-world application, it may be reasonable to limit the weighted sum of activated recommendations per model rather than just limiting the number of recommendations per model—if the goal is to improve, e.g., the quality of the claims processing across all models equally. In that case, even if the number of recommendations per model are quite similar, there can be still a great difference in the impact on the quality. Another example of the need to consider a weighted sum is to increase the satisfaction of one group of experts—e.g., customers—more than other expert groups. Since the impact on quality improvement depends on the weight γ_i of a recommendation $R_i, i \in [n]$, it may be necessary to consider the weighing and not only the number of activated recommendations. In the last case of our complexity-theoretical study, we only consider the constraint of an upper bound, but we can convert the constraint into a lower bound by multiplying it

by -1 . Given that, the recommendation system must then fulfill a certain level of satisfaction if

$$\sum_{R_i \in S_j} \gamma_i x_i \geq l_j^\gamma$$

holds for $i \in [n]$ and $j \in [m]$. By $l_j^\gamma \in \mathbb{R}_0^+$ we denote correspondingly the lower bounds adjusted to the weighted sum.

We define the limited weighted sum per model PI_j by

$$\sum_{R_i \in S_j} \gamma_i x_i \leq u_j^\gamma \quad \forall j \in [m].$$

Thereby, γ_i are the weights of recommendation R_i , $i \in [n]$, and $u_j^\gamma \in \mathbb{R}_0^+$ is the upper bound of each weighted sum $\sum_{R_i \in S_j} \gamma_i x_i$.

To show that considering a constrained weighted sum of recommendations per model rather than a constraint on the number of activations can be useful, we give the following example.

Example 4.3.17. Let two models PI_1 and PI_2 be given with $S_1 = \{R_1, R_2, R_3, R_4\}$ and $S_2 = \{R_5, R_6, R_7\}$, and the corresponding weights $\gamma_1 = 10$, $\gamma_2 = 9$, $\gamma_3 = 15$, $\gamma_4 = 5$, $\gamma_5 = 1$, $\gamma_6 = 4$, and $\gamma_7 = 5$. With that, it is $m := 2$ and $n := 7$.

First, we consider the case in which the upper bounds per model are equivalent, i.e., $u_1 = 1$ and $u_2 = 1$. The corresponding program is given by

$$\begin{array}{ll} \max & \sum_{i=1}^7 \gamma_i x_i \\ \text{s.t.} & \sum_{R_i \in S_j} x_i \leq 1 \quad \forall j \in [m] \\ & x_i \in \{0, 1\} \quad \forall i \in [n]. \end{array}$$

In order for $\gamma^T x$ to be maximal under the given constraints, the recommendations R_3 and R_7 are activated. With that we have $\sum_{i=1}^7 \gamma_i x_i = 15 + 5 = 20$.

To maximize the weighted sum under consideration of a limit for that weighted sum $\sum_{R_i \in S_j} \gamma_i x_i$ we observe the following program:

$$\begin{array}{ll} \max & \sum_{i=1}^7 \gamma_i x_i \\ \text{s.t.} & \sum_{R_i \in S_j} \gamma_i x_i \leq u_j^\gamma \quad \forall j \in [m] \\ & x_i \in \{0, 1\} \quad \forall i \in [n] \end{array}$$

with $u_1^\gamma = u_2^\gamma = 10$ the weighted upper bounds. Then, by activating R_1, R_5, R_6 and R_7 the program under consideration of the constraints is maximum. We get $\sum_{i=1}^7 \gamma_i x_i = 10 + 10 = 20$.

In both cases the quality $\gamma^T x$ is the same after maximizing it, but the impact of each model on the quality is different. In the first case, the most impact—measured by γ_i —is given by model PI_1 . However, the impact on the quality in the second case is equal. Since the quality, i.e., $\sum_{R_i \in S_j} \gamma_i x_i$, measures the satisfaction of the corresponding expert group, this makes a significant difference in the output.

The EBRS problem with limited weighted sum of activated recommendations per model $PI_j, j \in [m]$ is given by the following linear integer programming formulation. We call it *Limited weighted sum problem* or LWS problem.

Program 4.3.18 (LWS).

$$\begin{array}{ll} \max & \sum_{i=1}^n \gamma_i x_i \\ \text{s.t.} & \sum_{R_i \in S_j} \gamma_i x_i \leq u_j^\gamma \quad \forall j \in [m] \\ & x_i \in \{0, 1\} \quad \forall i \in [n] \end{array}$$

Before we prove that the LWS problem is \mathbb{NP} -complete, we formulate the decision version of it as follows.

Problem 4.3.19. Given an integer $k \in \mathbb{N}_0$. Can the weighted sum over all activation variables x_1, \dots, x_n sum up to a value of at least k by activating a set of appropriate recommendations, so that the weighted sum of activated recommendations per model $PI_j, j \in [m]$ does not exceed a predefined value u_j^γ ?

The LWS problem is quite similar to the famous Knapsack problem described in subsection [3.3.4.4](#). Even more, with $w_i = p_i$ for all $i \in [n]$, it is similar to the Subset Sum problem (SUBSET-SUM) problem. Note that the SUBSET-SUM problem is described in subsection [3.3.4.5](#).

Therefore, we prove in the following the \mathbb{NP} -completeness of the LWS problem by showing that the LWS problem is a generalization of the SUBSET-SUM problem. We define the SUBSET-SUM problem similar to Martello and Toth [\[MT07\]](#) and thus similar to the Knapsack problem with $p_i = w_i$ for all $i \in [n]$.

Proposition 4.3.20. *The LWS problem is \mathbb{NP} -complete.*

Proof. To show that the LWS problem is NP-complete, we apply a method other than Karp reduction. This method is described in subsection 3.3.3.2. To show NP-completeness, we need to prove the two statements **a)** $LWS \in NP$, and **b)** LWS is a generalization of SUBSET-SUM.

a) “ $LWS \in NP$ ”:

Similar to the proof of theorem 4.2.13 we can show that $LWS \in NP$.

b) “LWS is a generalization of SUBSET-SUM”:

Let us assume that there exists a polynomial time subroutine for the LWS problem.

Given an instance of SUBSET-SUM with a set of positive integers $S = \{s_1, \dots, s_n\}$ and a target value or knapsack capacity $c \in \mathbb{Z}^+$, we find a transformation with which the given instance of SUBSET-SUM is transformed to the LWS problem in the following way.

The set of positive integers $S = \{s_1, \dots, s_n\}$ is interpreted as a set of recommendations $S = \{R_1, \dots, R_n\}$. By program 3.3.38 we know that $s_i = w_i \cdot x_i$ for all $i \in [n]$, where $w_i \geq 0$ is the weight of item i for all $i \in [n]$. x_i is defined by

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is selected or “packed”,} \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, the weight $w_i \geq 0$ is interpreted as the weight $\gamma_i \geq 0$ for all $i \in [n]$ and x_i is interpreted as

$$x_i = \begin{cases} 1 & \text{if recommendation } R_i \text{ is activated or selected,} \\ 0 & \text{otherwise.} \end{cases}$$

As SUBSET-SUM does not differentiate between different models, let $m := 1$. With that, there exists only one upper bound $u_1^\gamma = u^\gamma := c \geq 0$, where c is the capacity of the knapsack.

We define a set of activated or selected recommendations $S' \subseteq S$, i.e., a recommendation $R_i \in S'$, if $x_i = 1$ for $i \in [n]$, and with that $S_{j=1} := S'$. That is, instead of looking at the assignment to a model $PI_{j=1}$ —we consider only one model—we interpret the activation as a selection of packed items or in the sense of LWS problem as a selection of activated recommendations.

This is a polynomial time transformation as in each step, we reinterpret the instance of SUBSET-SUM to an instance of LWS.

We now prove that a solution of the LWS problem for an instance of SUBSET-SUM is truly a solution of the SUBSET-SUM problem.

Let $m = 1$ and $S' \subseteq S$ be a set of activated recommendations, for which

$$\sum_{R_i \in S'} \gamma_i x_i \leq u^\gamma$$

holds. Since $\sum_{i=1}^n \gamma_i x_i$ should be maximized, we activate as many recommendations so that the total weight is closest to u^γ without exceeding. That is, the set of activated recommendations S' contains those recommendations, for which the sum $\sum_{R_i \in S'} \gamma_i x_i$ is nearest to u^γ , but does not exceed the upper bound. With the above defined transformation this is a solution to the SUBSET-SUM problem and we have therefore found a subset $S' \subseteq S$, such that the sum of packed items of S is closest to its knapsack capacity c , without exceeding it.

□

We know from the literature and as already mentioned above that the SUBSET-SUM problem can be seen as a special case of the Knapsack problem [MT07, CLRS09]. Also in case of the LWS problem we find a generalization, for which we can show that the Knapsack problem can be reduced to it. With that it is still NP-hard. We can motivate this generalization of the LWS problem by the following consideration.

We already discussed in subsection 4.2.1.1 that an upper bound for activated recommendations is reasonable, as a clerk can only process a certain number of recommendations. Yet, not only the number of activated recommendations can be a limiting factor. Among other things, also costs, time and the request of support of other divisions or external companies—e. g. in case of fraud detection—play an important role in the processing of a claim. Of course these “resources” are not unlimited and it can therefore be reasonable to include this limiting aspect in the optimization of the LWS problem. We denote all possible limiting resources as *effort* $e_i \geq 0$ for all $i \in [n]$. We will use the term effort to refer to all resources that can be expended during a claims processing.

When a recommendation $R_i \in S$, $i \in [n]$, is activated, this is directly associated to an effort e_i . When activating a recommendation we have to make sure—regardless of how we define “effort”—that a previously defined overall upper bound u^e is not exceeded. Of course, it is $e_i \geq 0$ for all $i \in [n]$. Furthermore, we assume that the upper bound is a natural number, i.e., $u^e \in \mathbb{N}_0$. We discuss the effort and especially the definition of the upper bound in more detail in section 5.1.

According to these considerations we can now set up the following program.

Program 4.3.21 (LWSE).

$$\begin{aligned} & \max \sum_{i=1}^n \gamma_i x_i \\ \text{s.t.} & \sum_{i=1}^n e_i x_i \leq u^e \\ & x_i \in \{0, 1\} \quad \forall i \in [n] \end{aligned}$$

So that the problem cannot be solved trivially, we assume that

$$\begin{aligned} & e_i \leq u^e \quad \forall i \in [n], \text{ and} \\ & \sum_{i=1}^n e_i > u^e. \end{aligned}$$

As in the EBR program (see program 4.2.8) in general as well as in the cases discussed above, we want to maximize the quality of the claims processing $\sum_{i=1}^n \gamma_i x_i$ by activating appropriate recommendations. Since the resources we can use during a claims processing are limited, it may be reasonable to take into account the effort involved in making a recommendation. To express this limitation, we define an overall upper bound $u^e \in \mathbb{N}_0$. The constraint is thus defined by

$$\sum_{i=1}^n e_i x_i \leq u^e.$$

This is somehow similar to the Knapsack problem (see program 3.3.35). We can interpret the overall upper bound u^e as the knapsack capacity c . The weight limit for the knapsack in our case is the limit of the resources we can use. The maximization of the value of the packed items in the knapsack can be interpreted as the maximization of the quality of the claims processing by activating appropriate recommendations. In both cases we activate a recommendation—or pack an item, respectively—by setting $x_i = 1$, and $x_i = 0$ otherwise for $i \in [n]$.

Or, to put it more figuratively, we put our recommendation into the knapsack of activated recommendations and want to fill this knapsack as valuable as possible without exceeding the knapsack capacity or knapsack resources, respectively.

The big difference between both programs is, that the weights $\gamma_i \in \mathbb{R}$ are not positive integers like the values p_i of the KP program. Nevertheless, we can show that the LWSE problem is NP-hard by reducing the KP problem to the LWSE problem. We will not prove the NP-hardness of the LWSE problem as the proof is similar to the proof of proposition 4.3.20. Instead, we draft the proof.

If we assume, that there exists a polynomial time algorithm to solve the LWSE problem than we would also have a polynomial time solver for the KP program

by simply restrict γ_i to $\gamma_i \geq 0$ for all $i \in [n]$. Beside the transformed weights γ_i , $i \in [n]$ the polynomial time solver accepts the rest of the instance of the KP problem without transformation. Of course, this can be done in polynomial time, which shows the NP-hardness.

To show $\text{LWSE} \in \text{NP}$ we can proceed similarly as in the proof of the theorem [4.2.13](#). Finally, we state the following proposition without proof.

Proposition 4.3.22. *The LWSE problem is NP-complete.*

In practice, there are cases where dependent and/or conflicting recommendations exist in addition to the constraint of limited resources—as discussed in the LWSE problem. Furthermore, it also might be reasonable to limit the resources per model PI_j by a lower bound l_j^e , $j \in [m]$. Consider, for example, an insurance company that is more willing to invest greater resources in customer satisfaction than in the satisfaction of other experts' needs.

We will not discuss this generalization of the LWSE problem in detail in this thesis, but will describe its assumptions in an outlook in section [5.1](#).

5 Outlook and Conclusions

We will end this thesis with an outlook on the utility and necessity of considering “effort” in maximizing the EBRS problem. We have already introduced the basic idea in subsection [4.3.3.2](#). In the LWSE problem, we considered only lower and upper bounds in the constraints of the problem. We now deepen and extend this idea to a generalized form that also considers dependencies and conflicts between recommendations. Thus, we extend the EBRS problem to include the idea of an effort in section [5.1](#). In the final section [5.2](#), we summarize the results of this work and provide suggestions for further analysis on this topic.

5.1 The effort—a generalization of the EBRS problem

Making a recommendation involves a certain amount of *effort*, which can vary depending on the recommendation. In the context of claims processing in the insurance industry, the term effort can be understood primarily in terms of cost and time, whereby other aspects of effort are also possible and reasonable.

In case of costs, this primarily refers to claims adjustment expenses, which are defined by Wagner [\[Wag17\]](#) as costs incurred by an insurance company in connection with the claims processing. Among them are internal costs (e.g., staff and material costs) as well as external costs, arising when engaging third parties such as external consultants or lawyers. Depending on the recommendation, such costs arise when the recommendation is activated. For example, by activating the recommendation “Engage an external consultant”, the insurance company has to pay the consultant. Of course, there are staff costs with any claims processing. The more time a claim handler needs to process the recommendation, the higher the personnel costs.

As further discussed in subsection [2.1.1](#), customer satisfaction plays a central role in the digitalization of the insurance industry. According to Köneke and colleagues [\[KMPF15\]](#), customers rate a claims processing as more satisfactory if the payout amount is appropriate, and if the time to payout is as short as possible. Of course, this depends on the severity of the accident. In case of a tragic accident, a customer will most likely want personal and human attention, whereas in the case of glass damage, they will most likely want a fast payout that can be handled without

personal contact. We have the ability to map this through the recommendations of the customer model. However, for some insurance companies, it may as well be important to consider the time taken to process a recommendation as (part of) the effort in the optimizing of problem EBRs.

The general rule here is that the more recommendations—and the more time-consuming each recommendation—the longer the processing time.

In terms of efficiency, there is an interaction between time and cost. As discussed in subsection [2.1.1](#), the longer the claims processing takes, the higher the costs. That is why including time—beside costs—as a limiting factor in the optimization within the term effort (instead of considering it via a recommendation) seems plausible, and may ensure efficient claims processing.

Yet, the term *effort* is not limited to time and cost. Depending on the objective and the situation of the insurance company under consideration, there may be other factors that need to be taken into account as limiting factors.

We define the effort of each recommendation of the EBRs problem as

$$e_i \geq 0 \text{ for all } i \in [n].$$

As we mentioned earlier, there is an upper limit to the amount of effort an insurance company can expend. To reflect this amount of effort in the program, the upper limit must be adjusted. Instead of placing an upper limit on the number of recommendations $u \in \mathbb{N}_0$, we now define the upper limit as a maximum amount of money that a company is willing to spend or a maximum duration for the processing of a claim, for example. Therefore, the upper limit is defined by

$$u^e \geq 0.$$

We assume, w.l.o.g., that

$$\begin{aligned} e_i &\leq u^e && \text{for all } i \in [n], \text{ and} \\ \sum_{i_1}^n e_i &> u^e. \end{aligned}$$

Otherwise, the problem could be solved trivially.

We call the so-defined problem *Limited weighted sum problem under consideration of effort* (LWSE) as already mentioned in subsection [4.3.3.2](#), where we have already shown that the LWSE problem is NP-complete by reducing the Knapsack problem to it.

In this section, we show that there exist further generalizations of the LWSE problem that may be important to represent a real claims processing of an insurance company. We will also briefly show that the generalizations of the Knapsack problem can also be reduced to these further generalizations of the LWSE problem.

Figuratively speaking, this can be justified in the way that instead of packing a knapsack with items, we now pack a recommendation set by activating the recommendations accordingly. The items—now recommendations—have both a value and a weight, the latter being the effort in the LWSE problem.

For the LWSE problem in subsection 4.3.3.2, we excluded dependencies and conflicts between recommendations. However, the case that neither conflicts nor dependencies exist seems unlikely. If we take into account the opinions of customers, managers and clerks in the respective models, for example, it seems very likely that there exist pairs of recommendations that are in conflict with each other—thinking about the amount of payments, the speed of the process, or of the repair of a car in the authorized workshop, etc. In addition, there also may exist dependencies between the recommendations.

In the remainder of this section, we consider the LWSE program extended by conflicts and dependencies. We call it *generalized* LWSE problem, or shortly g-LWSE. It is defined as follows.

Program 5.1.1 (g-LWSE).

$$\begin{array}{ll}
 \max & \sum_{i=1}^n \gamma_i x_i \\
 \text{s.t.} & \sum_{i=1}^n e_i x_i \leq u^e \\
 & x_i + x_{i'} \leq 1 \quad \forall \{R_i, R_{i'}\} \in C \\
 & x_i \leq x_{i'} \quad \forall (R_i, R_{i'}) \in D \\
 & x_i \in \{0, 1\} \quad \forall i \in [n]
 \end{array}$$

with $\gamma_i \in \mathbb{R}$ for all $i \in [n]$, $u^e \geq 0$ and $e_i \geq 0$ for all $i \in [n]$. The sets of conflicts C and dependencies D are defined as in equations (4.1) and (4.2), respectively, which can be found in subsection 4.2.1.1.

We assume (without further proof) that the g-LWSE problem is also NP-complete. On the one hand, our assumption is confirmed by the fact that the Knapsack problem can be reduced to the LWSE problem, which is the g-LWSE problem without the constraints of dependent and conflicting recommendations (see subsection 4.3.3.2). On the other hand, it is strengthened by the fact that the g-LWSE problem is similar to the following generalizations of the Knapsack problem—the *Precedence-constrained Knapsack Problem* (PCKP) and the *Knapsack problem with conflict graph* (KCG). Here, the PCKP problem is an extension of the KP problem to include dependencies, and the KCG problem is an extension of the KP problem to include conflicts. For more details on both models, we refer to subsection 3.3.4.6. The PCKP as well as the KCG problem are both NP-complete. Assuming that the parameters γ , e , and u^e of the g-LWSE problem are positive integers, the transformation between

the g-LWSE problem and the PCKP problem, or the KCG problem is more or less a one-to-one transformation.

Beside the fact that we can prove the NP -completeness of the g-LWSE problem with the PCKP and the KCG problem, it is also noteworthy that approximation algorithms are known for both models. Pferschy and Schauer discussed a *polynomial-time approximation scheme* (PTAS) and a *fully polynomial-time approximation scheme* (FPTAS) in this context [PS16].

Since the g-LWSE problem is very similar to the PCKP and the KCG problem, we conjecture that we can transfer the approximation algorithm of the PCKP or the KCG problem to our g-LWSE problem to some extent. It would therefore be reasonable to check whether a so-called *approximation-preserving reduction* (or APX reduction) to the g-LWSE problem exists. An APX reduction can be described in simplified terms as follows. Suppose that a solution algorithm exists (at least approximately) for an optimization problem Π' . Now, if an instance of an optimization problem Π' can be reduced to an instance of an optimization problem Π , and the solution of problem Π' can be recovered to problem Π to some extent, then we say that an APX reduction exists between the two problems. APX reductions are not part of this thesis, but bear great potential for (approximately) solving NP -hard problems. We refer the interested reader for more informations on approximation-preserving reductions to the works of Crescenzi [Cre97], Ausiello and colleagues [AMSC⁺99], and Woeginger [Woe05].

It could therefore be of further interest to investigate whether and to what extent APX reductions exist for the g-LWSE problem.

In summary, the g-LWSE problem may be of interest if the effort of a claims processing should be taken into account. This can also impact the efficiency as well as the effectiveness of the processing of claims when considered in the optimization.

5.2 Conclusion

Digitalization offers new opportunities in a wide range of industries, including the insurance industry. Digitalization can also be understood as helping to improve processes with the help of appropriate recommendations. More and more data becomes available through various digital tools. Analyzing these data in a meaningful way may help to better understand processes and to uncover opportunities to optimize these processes. This can be done using appropriate statistical methods of data analysis. Above that, we wanted to add an additional source of knowledge when analyzing processes in this thesis—experts. The experience and knowledge of experts are indispensable and extend the knowledge that can be obtained only with the help of data. If one includes the expert knowledge in the optimization

of a process, one even has the possibility to make individual implicit and isolated knowledge of experts explicit and generally available.

Here, we considered the claims processing in the insurance industry. However, the method developed throughout this thesis can of course also be applied to other areas of an insurance company—as well as to other industries, or in general to other fields of research where decisions have to be done incorporating data and expert knowledge.

A first goal of this thesis was to set up an optimization problem to improve the *quality of a claims processing*. A novel approach we took in this thesis was to define (and estimate) the not-yet-known *quality* of a claims processing. As a first step, we needed to include the target variable of our optimization problem in the dataset. We achieved this by surveying experts using appropriate statistical methods—in our case we applied the *conjoint analysis*. Based on this expert survey, we could set up an optimization problem, which is why we called the optimization problem *expert-based recommendation system* or EBRS, for short.

To determine the quality of a claims process, we focused at completed claims. This means that we looked at the processes *ex post*, firstly. In this way, we obtained an assessment not only of the quality of the claims processing, but also of various perspectives—from the customer (or claimant) to the claims processor, and the manager. At the same time, finding an appropriate selection of claims processes is somewhat challenging. On the one hand, we needed to make sure that the full spectrum of different scenarios in the claims processing is covered. On the other hand, we had to be careful not to overwhelm the experts with too many cases. To cover both aspects, we developed the so-called *conditional sampling from k clusters*, which is based on the k -medoids method.

Using this method, we found a suitable selection of so-called *stimuli* and surveyed the experts with them. The conjoint analysis now provided us with suitable statistical methods to estimate the quality of the claims processing based on the survey results. In addition, we were able to identify the estimation parameters as recommendations. We described the methods used to determine the quality of the claims processing and identify appropriate recommendations in section [4.1](#).

With *quality* implemented in the dataset, and the recognition of appropriate recommendations, it was possible to set up the EBRS problem. In section [4.2](#), we formulated the problem and defined the constraints.

The second goal of this thesis was to study the optimization problem in terms of complexity theory. We first showed that the EBRS problem is in general NP-complete. Thus, it is not solvable in polynomial time in general. We proved this in subsection [4.2.3](#). However, we have worked out cases for which a polynomial solution algorithm can be found throughout this thesis.

We divided our complexity-theoretic investigation into the following parts. In subsection [4.3.1](#), firstly—and similar to the NP -completeness proof—we focused on cases where there are no dependencies between recommendations. That is, we have excluded the possibility that the activation of one recommendation requires the activation of another or several other recommendations. Instead, there are only conflicts between recommendations. That is, when one recommendation is activated, another recommendation (or several) can no longer be activated. Here, we could show for two special cases that a polynomial solution algorithm can be found. In the first special case, there are only pairwise conflicts, and in the second special case, one recommendation can conflict with several other recommendations, but the other recommendations in each case conflict only with that one recommendation. If we draw this structure by connecting the conflicting recommendations with a stroke (or edge in graph-theoretic terms), these conflicting recommendations appear like a star. Hence the name *stellar conflicts*.

The situation is different when we consider cases where there are dependencies but no conflicts. We proved in subsection [4.3.2](#) that the EBRs-problem is NP -complete even in this case. However, we have not yet been able to find a special case in which the EBRs-problem restricted to dependent but no conflictual recommendations can be solved polynomially. Further research is needed in this regard.

For both the EBRs-problem with conflicting recommendations and the EBRs-problem with dependent recommendations, we found cases for which no solutions can be found. We also showed that we can identify these special cases in polynomial time.

Finally, in subsection [4.3.3](#) we discussed the case where neither conflicts nor dependencies need to be considered in solving the EBRs problem. For this special case, we found a polynomial solution algorithm for the EBRs-problem.

In both, subsection [4.3.3.2](#) and section [5.1](#), we also considered cases where we extended the EBRs-problem by a so-called *effort*. Considering effort seems reasonable to us because, among other things, making a recommendation involves cost and time to process that recommendation. With the activation, resources—in our case, those of the insurance company, which are generally not available in unlimited quantities—are consumed.

However, even without considering conflicts and dependencies between the recommendations in the constraints, the considered problem is NP -complete if the EBRs-problem is extended by the effort $e_i \geq 0, i \in [n]$. We proved the NP -completeness of the so-called *Limited weighted sum problems* or LWS problems in subsection [4.3.3.2](#). In doing so, we reduced it to the well-known *Knapsack problem* or KP-problem. In section [5.1](#), we recognized again (now including dependencies and conflicts between the recommendations) the similarity between the EBRs-problem extended by effort—called now g-LWSE-problem—and the generalizations of the KP-problem.

Focusing on this similarity, we formulated the question whether we can find an APX reduction for the g-LWSE-problem—especially since there are such investigations also for the generalizations of the KP-problem [PS16]. In the spirit of coping with NP-completeness, it is worthwhile to find out whether we can still solve the problem approximately well. Moreover, APX reduction may be of interest not only for the g-LWSE problem, but also for the EBRS problem.

We conclude this thesis with one last relevant question, which we recommend to be investigated in further work. In our complexity-theoretic investigations, we found that there is no polynomial solution algorithm for the EBRS problem once dependent recommendations must be considered in the solution. Here, it would be worth investigating if our conjecture is confirmed—or if there is a special case with dependent recommendations that can be solved in polynomial time.

If the conjecture that the EBRS problem with dependent recommendations is not polynomially solvable—even with further restrictions—is confirmed, further efforts to find approximate solutions would be worth pursuing.

5 Outlook and Conclusions

Bibliography

- [Add62] Sidney Addelman, *Orthogonal main-effect plans for asymmetrical factorial experiments*, *Technometrics* **4** (1962), no. 1, 21–46.
- [AMSC⁺99] Giorgio Ausiello, Alberto Marchetti-Spaccamela, Pierluigi Crescenzi, Giorgio Gambosi, Marco Protasi, and Viggo Kann, *Complexity and approximation*, Springer Berlin Heidelberg, 1999.
- [BCC⁺18] Pia Brüggemann, Tanguy Catlin, Jonas Chinczewski, Johannes-Tobias Lorenz, and Samantha Prymaka, *Claims in the digital age: How insurers can get started*, <https://www.mckinsey.com/industries/financial-services/our-insights/claims-in-the-digital-age>, 2018.
- [BEPW18] Klaus Backhaus, Bernd Erichson, Wulff Plinke, and Rolf Weiber, *Multivariate Analysemethoden*, Springer Berlin Heidelberg, 2018.
- [BG20] Andreas Brieden and Peter Gritzmann, *Predicting show rates in air cargo transport*, 2020 International Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT), IEEE, 2020.
- [BKM21] Andreas Brieden, Christian Krams, and Vanessa Mindl, *Innovatives Schadenmanagement für das digitale Zeitalter*, *Zeitschrift für Versicherungswesen* **72** (2021), no. 4, 102–104.
- [BP04] Johannes Berger and Bernd Postai, *Ertragssteigerung durch besseres Schadenmanagement*, *Versicherungswirtschaft* **10** (2004), 759–762.
- [Cag21] Derin Cag, *Insurtech firms: Are they a threat to insurance companies?*, <https://insurtechdigital.com/insurtech/insurtech-firms-are-they-threat-insurance-companies>, 12 2021.
- [CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to algorithms*, 3 ed., MIT Press, 2009.
- [Coo71] Stephen A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the third annual ACM symposium on Theory of computing - STOC '71, ACM Press, 1971.
- [CRB09] Eugene M. Caruso, Dobromir A. Rahnev, and Mahzarin R. Banaji, *Using conjoint analysis to detect discrimination: Revealing covert preferences from overt choices*, *Social Cognition* **27** (2009), no. 1, 128–137.

Bibliography

- [Cre97] P. Crescenzi, *A short guide to approximation preserving reductions*, Proceedings of Computational Complexity. Twelfth Annual IEEE Conference, IEEE Comput. Soc, 1997.
- [DAV19] DAV Arbeitsgruppe Tarifierungsmethodik, *Aktuarieller Umgang mit Big Data in der Schadenversicherung*, Deutsche Aktuarvereinigung e.V., 5 2019.
- [Die17] Reinhard Diestel, *Graph theory*, Springer Berlin Heidelberg, 2017.
- [DPV06] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*, McGraw-Hill Education, 2006.
- [DT97] George B. Dantzig and Mukund N. Thapa, *Linear programming*, Springer, 1997.
- [Edm71] Jack Edmonds, *Matroids and the greedy algorithm*, Mathematical Programming **1** (1971), no. 1, 127–136.
- [EG77] Jack Edmonds and Rick Giles, *A min-max relation for submodular functions on graphs*, Studies in Integer Programming (P. L. Hammer, E. L. Johnson, B. H. Korte, and G. L. Nemhauser, eds.), North-Holland, 1977, pp. 185–204.
- [FKLM13] Ludwig Fahrmeir, Thomas Kneib, Stefan Lang, and Brian Marx, *Regression*, Springer Berlin Heidelberg, 2013.
- [FMF12] A. Field, J. Miles, and Z. Field, *Discovering statistics using r*, Sage, 2012.
- [Gen21] Görkem Gençer, *3 ways ai enables efficient claims processing in insurance*, <https://research.aimultiple.com/insurance-claims-ai/>, 12 2021.
- [Gen22] ———, *Top 7 technologies that improve claims processing in 2022*, <https://research.aimultiple.com/claims-processing>, 7 2022.
- [GH62] A. Ghouila-Houri, *Caract ´erisation des matrices totalement unimodulaires*, Comptes Rendus Hebdomadaires des S ´eances de l’Acad ´emie des Sciences **154** (1962), 1192–1194.
- [GHH07] Anders Gustafsson, Andreas Herrmann, and Frank Huber (eds.), *Conjoint measurement*, Springer Berlin Heidelberg, 2007.
- [GJ79] Michael R. Garey and David S. Johnson, *Computers & intractability: A guide to the theory of np-completeness*, W. H. Freeman and Company, 1979.
- [GLK10] Keith Goffin, Fred Lemke, and Ursula Koners, *Identifying hidden needs*, Palgrave Macmillan UK, 2010.
- [GLS93] Martin Grötschel, László Lovász, and Alexander Schrijver, *Geometric algorithms and combinatorial optimization*, Springer, 1993.

- [Gow71] J. C. Gower, *A general coefficient of similarity and some of its properties*, *Biometrics* **27** (1971), no. 4, 857.
- [Gri13] Peter Gritzmann, *Grundlagen der mathematischen Optimierung*, Springer, 2013.
- [Gru18] Volker Gruhn, *Versicherungen: Von Natur aus für Künstliche Intelligenz geeignet*, *Wirtschaftsinformatik & Management* **4** (2018), 104–110.
- [GS78] Paul E. Green and V. Srinivasan, *Conjoint analysis in consumer research: Issues and outlook*, *Journal of Consumer Research* **5** (1978), no. 2, 103.
- [GS90] ———, *Conjoint analysis in marketing: New developments with implications for research and practice*, *Journal of Marketing* **54** (1990), no. 4, 3–19.
- [HMMR15] Christian Hennig, Marina Meila, Fionn Murtagh, and Roberto Rocci (eds.), *Handbook of cluster analysis*, Chapman and Hall, dec 2015.
- [Hof74] A. J. Hoffman, *A generalization of max flow—min cut*, *Mathematical Programming* **6** (1974), no. 1, 352–359.
- [HT57] I. Heller and C. B. Tompkins, *An extension of a theorem of dantzig’s, Linear Inequalities and Related Systems. (AM-38)* (Harold William Kuhn and Albert William Tucker, eds.), Princeton University Press, dec 1957, pp. 247–254.
- [HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The elements of statistical learning*, Springer New York, 2009.
- [JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, *An introduction to statistical learning*, Springer, 2013.
- [Kar72] Richard M. Karp, *Reducibility among combinatorial problems*, *Complexity of Computer Computations*, Springer, 1972, pp. 85–103.
- [Kha80] L.G. Khachiyan, *Polynomial algorithms in linear programming*, *USSR Computational Mathematics and Mathematical Physics* **20** (1980), no. 1, 53–72.
- [KMPF15] Vanessa Köneke, Horst Müller-Peters, and Detlef Fetchenhauer, *Betreuung im Schadensfall – Negativen Erfahrungen und Racheakten vorbeugen*, *Versicherungsbetrug verstehen und verhindern*, Springer, 2015, pp. 377–397.
- [KPP04] Hans Kellerer, Ulrich Pferschy, and David Pisinger, *Knapsack problems*, Springer, 2004.
- [KR05] Leonard Kaufman and Peter J. Rousseeuw, *Finding groups in data: An introduction to cluster analysis*, Wiley, 2005.

Bibliography

- [KT13] J. Kleinberg and E. Tardos, *Algorithm design*, Pearson Education Limited, 2013.
- [KV06] Bernhard Korte and Jens Vygen, *Combinatorial optimization*, Springer, 2006.
- [Mal82] Naresh K. Malhotra, *Structural reliability and stability of nonmetric conjoint analysis*, *Journal of Marketing Research* **19** (1982), no. 2, 199–207.
- [MF20] Sushma MacGeoch and Deirdre Fenton, *Professional indemnity claims management: conflicts of interest*, Online Articles and Briefings by Womble Bond Dickinson, <https://www.womblebonddickinson.com/uk/insights/articles-and-briefings/professional-indemnity-claims-management-conflicts-interest>, 05 2020.
- [MG07] J. Matousek and B. Gärtner, *Understanding and using linear programming*, Springer, 2007.
- [MK08] Jerko Markovina and Damir Kovacic, *The importance of apple attributes: A comparison of self-explicated and conjoint analysis results*, 12th EAAE Congress "People, Food and Environments: Global Trends and European Strategies", Gent (Belgium), 26–29 August 2008, European Association of Agricultural Economists, 2008.
- [Mor16] Marco Morawetz, *Der telematische Irrweg der Kfz-Versicherung*, *Versicherungswirtschaft* (2016), 1–8.
- [MPS14] Anna Makarewicz, Piotr Pikuta, and Dominik Szałkowski, *Properties of the determinant of a rectangular matrix*, *Annales UMCS, Mathematica* **68** (2014), no. 1, 31–41.
- [MRS⁺22] Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert, and Kurt Hornik, *cluster: Cluster analysis basics and extensions*, 2022, R package version 2.1.4.
- [MT07] S. Martello and P. Toth, *Knapsack problems: Algorithms and computer implementations*, Wiley, 2007.
- [PS16] Ulrich Pferschy and Joachim Schauer, *Approximation of knapsack problems with conflict and forcing graphs*, *Journal of Combinatorial Optimization* **33** (2016), no. 4, 1300–1323.
- [PZ80] David A. Plaisted and Samuel Zaks, *An NP-complete matching problem*, *Discrete Applied Mathematics* **2** (1980), no. 1, 65–72.
- [Rad57] R. Rado, *Note on independence functions*, *Proceedings of the London Mathematical Society* **s3-7** (1957), no. 1, 300–320.

- [Rad66] M. Radi, *A definition of determinant of rectangular matrix*, Glasnik Mate-maticki **1** (1966), no. 21, 17–22.
- [Rou87] Peter J. Rousseeuw, *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis*, Journal of Computational and Applied Mathematics **20** (1987), 53–65.
- [RVPOR⁺11] Jorge A. Ruiz-Vanoye, Joaquín Pérez-Ortega, Rodolfo A. Pazos R., Ocotlán Díaz-Parra, Juan Frausto-Solís, Hector J. Fraire Huacuja, Laura Cruz-Reyes, and José A. Martínez F., *Survey of polynomial transformations between NP-complete problems*, Journal of Computational and Applied Mathematics **235** (2011), no. 16, 4851–4865.
- [Sch98] Alexander Schrijver, *Theory of linear and integer programming*, Wiley, 1998.
- [Sch03] ———, *Combinatorial optimization. polyhedra and efficiency*, Springer, 2003.
- [SM18] M. Steiner and M. Meißner, *A user’s guide to the galaxy of conjoint analysis and compositional preference measurement*, Journal of Research and Management **40** (2018), no. 2, 3–25.
- [SS73] V. Srinivasan and Allan D. Shocker, *Linear programming techniques for multidimensional analysis of preferences*, Psychometrika **38** (1973), no. 3, 337–369.
- [Swe10] John Sweller, *Element interactivity and intrinsic, extraneous, and germane cognitive load*, Educational Psychology Review **22** (2010), no. 2, 123–138.
- [SY00] N. Samphaiboon and Y. Yamada, *Heuristic and exact algorithms for the precedence-constrained knapsack problem*, Journal of Optimization Theory and Applications **105** (2000), no. 3, 659–676.
- [Tha21] Jitesh J. Thakkar, *Multi-criteria decision making*, Springer Singapore, 2021.
- [Tho19] Timothy Thornton, Jr., *Claims handling conflicts*, Online Article by the American Bar Association TIPS Insurance Institute, https://grayduffylaw.com/wp-content/uploads/2019/05/TIPS_Insurance_Institute_Conflicts_Panel_Paper.pdf, 4 2019.
- [Ver20] Versicherungsbote, *Schadenbearbeitung der Versicherer: Fast keiner kommt ohne Papier aus*, <https://www.versicherungsbote.de/id/4892661/Schadenbearbeitung-Versicherer-Digitalisierung>, 2020.
- [Vri95] M. Vriens, *Conjoint analysis in marketing: developments in stimulus representation and segmentation methods*, Ph.D. thesis, Capelle, 1995.

Bibliography

- [Wag17] Fred Wagner (ed.), *Gabler Versicherungslexikon*, 2 ed., Springer, 2017.
- [Wal17a] Frank Walthes, *Der digitale Aufbruch steckt voller Chancen*, 2017.
- [Wal17b] ———, *Kundenorientierung: Die Reise lohnt sich*, 2017.
- [Wes00] D. West, *Introduction to graph theory*, 2 ed., Prentice-Hall, 2000.
- [Woe05] Gerhard J. Woeginger, *Combinatorial approximation algorithms: a comparative review*, *Operations Research Letters* **33** (2005), no. 2, 210–215.
- [YKW02] T. Yamada, S. Kataoka, and K. Watanabe, *Heuristic and exact algorithms for the disjunctively constrained knapsack problem*, *IPSJ Journal* **43** (2002), no. 9, 2864–2870.
- [YY07] Byungjun You and Takeo Yamada, *A pegging approach to the precedence-constrained knapsack problem*, *European Journal of Operational Research* **183** (2007), no. 2, 618–632.

