The integrated treatment of planning problems which are usually considered separately and sequentially has been studied for a long time, due to the better solutions one may find in the extended decision space of an integrated problem. This thesis considers the integrated locomotive scheduling and driver assignment problem in rail freight transport. We also consider the generalization of this problem, which we call graph 2-list-colouring with compatibility constraints. The motivation to study this problem originates from our collaboration with DB Cargo Polska within the ROMSOC Programme. The thesis consists of two parts. Part I focuses on modelling and solving the integrated locomotive scheduling and driver assignment problem in rail freight transport. After a literature review, we present a novel optimization model for the problem studied and a way to improve its formulation. Next, we introduce the decomposition-based solution approach we derive for the problem. To ensure the global feasibility of the solutions to the decomposed subproblems, we devise four classes of valid inequalities. We also develop a presolve heuristic. We then test our algorithm against two sets of instances. In general, the methods presented enabled the creation of locomotive timetables and driver assignments in less than two hours. In Part II, we study a generalization of the integrated locomotive scheduling and driver assignment problem, which we call graph 2-list-colouring with compatibility constraints. We begin by defining the problem studied and putting it in the context of other, more famous combinatorial problems. Then we present two formulations for the problem and discuss how they may be tightened. We also study a case for which we may find a complete polyhedral description. Next, we present a decomposition-based solution approach which adapts the algorithm introduced in Part I. We then test the performance of our method against a set of standard instances drawn from the literature, which were appropriately modified. Altogether, our work is a practical contribution to the solvability of the integrated locomotive scheduling and driver assignment problem. We also show how the developed method may be extended to successful use in more general graph-theoretic contexts.

The integrated treatment of planning problems which are usually considered separately and sequentially has been studied for a long time, due to the better solutions one may find in the extended decision space of an integrated problem. This thesis considers the integrated locomotive scheduling and driver assignment problem in rail freight transport. We also consider the generalization of this problem, which we call graph 2-list-colouring with compatibility constraints. The motivation to study this problem originates from our collaboration with DB Cargo Polska within the ROMSOC Programme. The thesis consists of two parts. Part I focuses on modelling and solving the integrated locomotive scheduling and driver assignment problem in rail freight transport. After a literature review, we present a novel optimization model for the problem studied and a way to improve its formulation. Next, we introduce the decomposition-based solution approach we derive for the problem. To ensure the global feasibility of the solutions to the decomposed subproblems, we devise four classes of valid inequalities. We also develop a presolve heuristic. We then test our algorithm against two sets of instances. In general, the methods presented enabled the creation of locomotive timetables and driver assignments in less than two hours. In Part II, we study a generalization of the integrated locomotive scheduling and driver assignment problem, which we call graph 2-list-colouring with compatibility constraints. We begin by defining the problem studied and putting it in the context of other, more famous combinatorial problems. Then we present two formulations for the problem and discuss how they may be tightened. We also study a case for which we may find a complete polyhedral description. Next, we present a decomposition-based solution approach which adapts the algorithm introduced in Part I. We then test the performance of our method against a set of standard instances drawn from the literature, which were appropriately modified. Altogether, our work is a practical contribution to the solvability of the integrated locomotive scheduling and driver assignment problem. We also show how the developed method may be extended to successful use in more general graph-theoretic contexts.
Jonasz Staszek

Modelling and solving the integrated locomotive scheduling and driver assignment problem
Modelling and solving the integrated locomotive scheduling and driver assignment problem

with an extension to graph 2-list-colouring problem with compatibility constraints

Erlangen
FAU University Press
2023
Modelling and solving the integrated locomotive scheduling and driver assignment problem with an extension to graph 2-list-colouring problem with compatibility constraints

Über das Modellieren und Lösen des integrierten Lokomotivplanungs- und Lokführerzuweisungsproblems mit einer Erweiterung auf das Problem der 2-Listen-Färbung von Graphen mit Kompatibilitätsnebenbedingungen

Der Naturwissenschaftlichen Fakultät der Friedrich-Alexander-Universität Erlangen-Nürnberg zur Erlangung des Doktorgrades Dr. rer. nat.

vorgelegt von

Magister Jonasz Staszek aus Tichau (Tychy, Polen)
Als Dissertation genehmigt
von der Naturwissenschaftlichen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen
Prüfung: 25.01.2023

Vorsitzender des
Promotionsorgans: Prof. Dr. Wolfgang Achtziger

Gutachter: Prof. Dr. Alexander Martin
Prof. Dr. Ralf Borndörfer
Tym którym zawdzięczam życie:

Dziadkowi Józefowi – dziękując za ciche, skuteczne towarzyszenie
Babci Monice – dziękując za inspirację do ciężkiej, niestrudzonej pracy
Dziadkowi Zygmuntowi – dziękując za przykład wielkiego serca i szerokich horyzontów
Babci Helenie – dziękując za miłość która więcej robi niż mówi

Tacie Markowi – dziękując za każde drzwi, które mi otwarłeś
Mamie Elżbiecie – dziękując za każde słowo, każdy kilometr i każdą łzę

z wdzięcznością dedykuję tę pracę.
Acknowledgements

This thesis could not have been written without the tremendous support of Prof. Dr. Alexander Martin. He guided me along the way, taking the time to patiently explain both the trivial and the less trivial concepts. He gave me a lot of freedom. On a side note – he took quite a risk in hiring a person without a degree in mathematics to pursue a doctoral degree in this field. Hopefully this dissertation will show that the gamble paid off.

I would also like to thank Prof. Dr. Ralf Borndörfer for reviewing this thesis. I am very grateful to Prof. Dr. Michael Stingl and Prof. Dr. Frauke Liers for their willingness to serve as examiners on my committee. I would also like to thank Prof. Dr. Francisco Javier Zaragoza Martinez for our many fruitful discussions and Prof. Dr. Timm Oertel for our joint didactic efforts. My sincere thanks also go to Dr. Dieter Weninger for his patient introduction to the world of mathematical optimization.

I gratefully acknowledge the funding I received from European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement 765374 (ROMSOC), led by Prof. Dr. Volker Mehrmann and Dr. Lena Scholz from TU Berlin. My work has also received funding from the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics – Data – Applications (ADA-Center) within the framework of “BAYERN DIGITAL II” (20-3410-2-9-8).

Many thanks are due to my colleagues, especially Dr. Oskar Schneider, Dr. Patrick Gemander¹, Lukas Hager, Katrin Halbig¹, Dr. Martina Kuhlbauer, Alexander Müller¹, Kristin Braun¹, Tobias Kuen¹, Jan Krause¹, Dr. Lukas Glomb, Lukas Hümbs¹ and Florian Rösel. Thank you very much for making this experience a pleasant one, even though – due to COVID pandemic – we did not see each other very often! Special thanks are also due to Beate Kirchner, Regine Stirnweis, Edeltraud Balser, Sonja Biswenger, Christina Weber and Susanne Hiltl for their support in administrative matters.

The experience of my doctorate would not be the same without my ROMSOC cohort: Marcus Bannenberg, Marco Martinolli, Umbero Morelli, Giorgi Rukhaia, Bernadett Stadler, Ashwin Nayak, Onkar Jadhav, Nirav Shah. I will long remember the time we spent traveling around Europe, enjoying mathematics and good food.

An important part of my doctoral experience was the secondment to DB Cargo Polska. I would like to thank Michal Batko, Pawel Pucek, Robert Nowakowski,

¹ Thank you for proof-reading parts of this dissertation.
Ziemowit Zawarczyński, Paweł Wiechoczek, Jacek Babiuch, Piotr Cieśliński, Miłosz Warda, Joanna Dubiel, Sławomir Cichoński, Tomasz Kleszcz, Tomasz Nawrat, Krzysztof Czaja, Aleksandra Zdziebłowska, Przemysław Kilianński and many others, for their warm welcome and for introducing me to the fascinating world of railways. Thanks are due to Sławomir Kucharczyk, Jerzy Skotnicki and Anna Sobczuk from Nodus for their readiness to share the experience of digitization of the railway industry.

I would also like to thank my master’s student Linda Schneider for her contribution to our research, as well as the helpful crew of the Erlangen HPC Center for access to their computing resources and kind support.

Last but not least, I would like to thank my family and friends. In particular, I would like to mention my grandparents and parents, my siblings Agata and Nikodem and their spouses Bartek and Beata (thanks for your patience with your older brother(-in-law)), Zbigniew Borowski (thanks for our friendship despite the growing number of kilometers between us) and Daniel Vjačka (thanks for our numerous car trips and all the discussions we had).

Finally, my wife Monika and my son Józef deserve my deepest gratitude. They believed in me and in the eventual success of this dissertation from the very beginning, even in moments when I myself doubted it. Especially during the time devoted to the finalization of this dissertation, you tolerated my physical and mental absence from the normal life of a young family. For that I am beyond grateful. Thank you for your unconditional love and support in all my endeavors.

Erlangen, 18.04.2023

Jonasz M. Staszek
Abstract

The integrated treatment of planning problems which are usually considered separately and sequentially has been studied for a long time. Researchers’ interest in this field is sparked by the better solutions that may be found in the extended decision space of an integrated problem. This thesis considers the integrated locomotive scheduling and driver assignment problem in rail freight transport. The motivation to study this problem originates from our collaboration with DB Cargo Polska – the Polish subsidiary of Deutsche Bahn, within the ROMSOC Programme. We also consider the generalization of this problem, which we call graph 2-list-colouring with compatibility constraints. The thesis consists of two closely related parts. Their contents are presented in the following.

Part I focuses on modelling and solving the integrated locomotive scheduling and driver assignment problem in rail freight transport. We begin by discussing the relevance of the problem in practice. We also present the rich body of literature discussing applications of Mathematical Optimization in the railway industry. Then we present a novel optimization model for the problem at hand and a way to improve its formulation. Next, we discuss the decomposition-based solution approach we derive for the problem. To ensure the global feasibility of the solutions to the decomposed subproblems, we devise four classes of valid inequalities. We also develop a presolve heuristic, allowing for further reductions in solution times. We then test our algorithm against two sets of instances. One of them was provided to us by our industry partner, the other simulates cases with less favourable resource availability. In general, the methods presented enabled the creation of locomotive timetables and driver assignments in less than two hours. These short solution times exceeded the expectations of our industry partner and enable iterative what-if analyses.

In Part II, we study a generalization of the integrated locomotive scheduling and driver assignment problem, which we call graph 2-list-colouring with compatibility constraints (G2LC-CC). We begin our considerations by defining the problem studied and putting it in the context of other, more famous combinatorial problems. Then we present two formulations for the problem and discuss how they may be tightened. We also study a case for which we may find a complete polyhedral description. Next, we present a decomposition-based solution approach which adapts the algorithm introduced in Part I. We then test the performance of our method against a set of standard instances drawn from the literature, which were appropriately modified.
Altogether, our work is a practical contribution to the solvability of the integrated locomotive scheduling and driver assignment problem. We also show how the developed method may be extended to successful use in more general graph-theoretic contexts.
Zusammenfassung


In Teil II untersuchen wir eine Verallgemeinerung des integrierten Lokomotivplanungs- und Lokführerzuweisungsproblems, das wir als Problem der 2-Listen-Färbung von Graphen mit Kompatibilitätsnebenbedingungen (G2LC-CC) bezeichnen. Wir beginnen unsere Überlegungen mit der Definition des untersuchten Problems und stellen es in den Kontext

Insgesamt ist unsere Arbeit ein praktischer Beitrag zur Lösbarkeit des integrierten Lokomotivfahrplanungs- und Lokführerzuordnungsproblems. Wir zeigen auch, wie die von uns entwickelte Methode für einen erfolgreichen Einsatz in allgemeineren graphentheoretischen Kontexten erweitert werden kann.
# Contents

Introduction .......................................................... 1

I The integrated locomotive scheduling and driver assignment problem .......................................................... 7

1 Motivation of the problem ........................................... 11

1.1 “Demand factors” – a need for an efficient use of resources . 11

1.1.1 Increasing the share of rail in land freight transport 11

1.1.2 Combating climate change ................................. 13

1.2 “Supply factors” – use cases of MO in rail planning problems 15

1.2.1 Strategic level decisions ................................. 15

1.2.2 Tactical level decisions ................................. 16

1.2.3 Operational level decisions ............................... 20

2 Modelling the integrated locomotive scheduling and crew assignment problem ............................................ 23

2.1 State of the art in the integrated resource planning: a literature overview .................................................... 23

2.2 Overview of modelling requirements and assumptions . 27

2.2.1 Driver restrictions and requirements .................. 28

2.2.2 Locomotive restrictions and requirements ............ 30

2.2.3 Compatibilities of locomotives, drivers and trains . 32

2.3 The optimization model .................................................. 32

2.3.1 Set definitions ............................................... 33

2.3.2 Multi-commodity flow part of the model – the locomotive assignment .................................................. 40

2.3.3 Definitions of variables ...................................... 41

2.3.4 Model formulation ........................................... 42

2.4 Model preprocessing – clique tightening .................... 46

3 A solution algorithm for the integrated vehicle scheduling and crew assignment ........................................... 51

3.1 Decomposition into locomotive master problem and driver subproblem .................................................. 52

3.2 Classes of valid cutting planes ...................................... 53

3.2.1 Valid cutting planes derived from the time conflict constraints .................................................. 55
### Contents

3.2.2 Valid cutting planes derived from the break conflict constraints ................................................. 57  
3.2.3 Valid cutting planes derived from the compatibility constraints ......................................................... 59  
3.2.4 Valid cutting planes derived from the constraints related to Sunday breaks ........................................... 61  
3.3 Preprocessing the driver subproblem ......................................................... 62  
3.4 A heuristic to presolve the driver subproblem ......................................................... 63  
4 Computational results .......................................................................................... 65  
4.1 Implementation details ..................................................................................... 65  
4.2 Performance of the method against instances supplied by DB Cargo Polska .......................................... 68  
4.2.1 Analysis of solution times ................................................................................ 70  
4.2.2 Assessment of solution quality ........................................................................ 71  
4.3 Performance of the method under less favourable conditions ......................................................... 72  
4.3.1 Exploring the optimal values with a decreasing number of locomotives and drivers .................................. 72  
4.3.2 Limiting the availability of drivers and locomotives upfront ......................................................... 74  

II The graph 2-list-colouring problem with colour compatibilities ................................................. 79  
5 Introduction, problem definition and literature survey ..................................................................... 83  
5.1 Basic definitions and problem statement ......................................................................................... 83  
5.2 Literature overview ......................................................................................................................... 84  
5.2.1 Graph colouring ....................................................................................................................... 84  
5.2.2 Stable Set, Maximum Clique and Set Packing Problems .......................................................... 86  
6 Formulations of the problem and some polyhedral results ............................................................ 93  
6.1 Formulations of the problem ........................................................................................................... 93  
6.1.1 One-variable formulation ........................................................................................................... 93  
6.1.2 Two-variable formulation .......................................................................................................... 94  
6.2 Improvements to the formulations .................................................................................................. 94  
6.2.1 Clique tightening of one-variable formulation ........................................................................... 95  
6.2.2 Clique tightening of two-variable formulation and a valid equation class ..................................... 96
6.3 A special case of \((G2LC-CC)\) on a chordal graph with a special property .................................................. 98

7 A decomposition-based solution algorithm .................................. 105
  7.1 Decomposition scheme ................................................. 106
  7.2 Preprocessing of \(c\)-subproblem and \(d\)-subproblem ........... 107
  7.3 Valid inequality for the \(c\)-subproblem ............................... 107
  7.4 Preprocessing of the \(d\)-subproblem ................................. 108

8 Computational results ......................................................... 111
  8.1 Implementation details .................................................. 111
  8.2 Instances introduction .................................................. 112
  8.3 Results and their interpretation ...................................... 115

Summary and outlook ............................................................ 121
Introduction

Throughout its history, the railway sector has always posed interesting research questions. Scientists from nearly all areas of knowledge have explored the challenges connected to that industry. Today, 193 years after the introduction of the Rocket steam locomotive by George Stephenson, we still find many exciting avenues which call for a scientific exploration. One of the most active and prominent areas in railways-related research is the use of Mathematical optimization (MO) techniques to improve the operations of railway companies. Although the first works in this field date back to 1930s, it gained traction only in 1950s, and the first MO-based algorithms were introduced for practical use in 1990s.

This growth in popularity has been due to the tremendous increase in the available computing power over the last few decades as well as the efficiency of solution algorithms. In fact, both have grown so much that planning problems which were previously considered to be computationally intractable can now be solved. One of such challenges was posed by the resource planning problems, which needed to be solved sequentially, i.e. as separate optimization problems, with the solution of the first one being “affixed” to the second. Such approaches – criticized already in the 1980s (cf. [Raff, 1983]) – now cease to be the only viable option.

This thesis is devoted to studying two interrelated problems: the integrated locomotive scheduling and driver assignment problem, and its generalization which we call graph 2-list-colouring with compatibility constraints (G2LC-CC). These problems are related in several aspects, in particular by similar formulations and analogous solution approaches. We devote a separate part of the dissertation to each of these two topics. Let us now discuss each of these problems in more detail.

As the name suggests, the integrated locomotive scheduling and driver assignment problem consists in the joint treatment of two planning problems: locomotive scheduling and driver assignment. The locomotive scheduling problem focuses on the creation of schedules for locomotives for a given period of time, taking into account the requirements of trains such as wagons’ weight or the availability of catenaries. The driver assignment problem (characteristic for rail freight) focuses on allocating individual trains to drivers in such a way, that working time constraints or route licensing requirements are met.

We believe that the integrated locomotive scheduling and driver assignment problem poses an exciting research challenge. There are two reasons for this.
Introduction

First, the exceptional complexity of the integrated locomotive scheduling and driver assignment problem results in numerous modelling and algorithmic challenges which need to be resolved to arrive at a functional solution approach. Second, the problem is drawn from the planning practice of our industrial partner, DB Cargo Polska, which makes the deliberations presented in this dissertation relevant to the planning practice of actual rail carriers.

To accurately reflect the numerous planning constraints at hand, we have tested a number of model formulations. In this thesis, we present a custom modelling for the integrated locomotive scheduling and driver assignment problem. It is a combination of a multicommodity flow problem (to account for the locomotive scheduling part) and a set packing problem with multiple choice and compatibility constraints (for the driver assignment part). We also present a formulation strengthening technique – the so-called clique tightening.

The solution of the integrated locomotive scheduling and driver rostering problem is then made possible by the tailor-made, decomposition-based algorithm we devise. It consists in relaxing the locomotive-driver compatibility constraints, and then solving the two resultant subproblems sequentially. To ensure the global feasibility of the solutions to both subproblems, we introduce a number of valid inequalities, which are then added to one of the subproblems. To further reduce the computation times, we also devise a presolve heuristic for one of the subproblems.

To check the practicality of our algorithm, we then test it on real-world data supplied by our industry partner, DB Cargo Polska. These correspond to an actual planning challenge, which the planners had to deal with in February 2020. Based on that instance, we generate a number of other instances, which can be characterized by smaller numbers of drivers and locomotives available. These instances are then used to explore to what extent our algorithm can be generalized.

In the second part of this dissertation, we present a generalization of the integrated locomotive scheduling and driver assignment problem. We call it graph 2-list-colouring with colour compatibilities (G2LC-CC). To the best of our knowledge, this is the first time such a problem is introduced and studied. After comparing it to the other popular combinatorial problems, we present two formulations for the problem. We also discuss the ways to tighten both of them. For a special case, we show that the tightening actually allows for a complete description of the underlying polyhedron.

To solve the (G2LC-CC), we devise a decomposition-based algorithm which is an adaptation of the procedure used to solve the integrated locomotive
scheduling and driver assignment problem. It comprises one class of valid inequalities thanks to which the solutions to the decomposed subproblems are globally feasible. We also test the performance of the two formulations we introduce as well as the algorithm we devise. We measure it against a number of instances which have been adapted from a popular graph colouring library.

Structure of this thesis

This thesis is composed of two parts. Their composition is “symmetrical” – both of them contain four chapters. The first chapter of each part focuses on introducing the problem, its motivation and the existing literature. In the second one, formulation(s) are presented and discussed. Third chapters are devoted to the introduction of the solution methods, while the fourth ones present computational results. The links between Chapters 2 and 6 as well as Chapters 3 and 7 are particularly strong: Chapter 6 presents a relaxation of the formulation studied in Chapter 2 and adds some polyhedral results, while the solution approach introduced in Chapter 7 is actually a generalization of the algorithm discussed in Chapter 3. Let us now briefly describe the contribution of each part and chapter.

Part I focuses on the problem of integrated locomotive scheduling and driver assignment in rail freight. As discussed, it consists of four chapters, namely Chapters 1-4.

In Chapter 1 we discuss the motivation of the problem and its significance to modern rail freight companies. In Section 1.1 we discuss the global phenomena which underline the need for the algorithm developed in this thesis. Section 1.2 is devoted to a broad discussion about the large potential of MO applications in the railway industry.

Chapter 2 is devoted primarily to the introduction of the optimization model for the integrated locomotive scheduling and driver assignment problem. In particular, Section 2.1 presents the state of the art in the integrated resource planning for rail freight companies. Later, in Section 2.2, the requirements and restrictions considered in the modelling are discussed. Next, in Section 2.3, we introduce an Integer Programming (IP) model which considers all the requirements collected. Finally, in Section 2.4 we introduce clique tightening as a formulation improvement technique.

In Chapter 3, a solution algorithm to the integrated locomotive scheduling and driver assignment is presented. We begin with an introduction of the decomposition scheme in Section 3.1. Then, in Section 3.2, four classes of valid inequalities are introduced. Their purpose is to ensure the global feasibility of the solutions to the decomposed subproblems. Later, in Section 3.3, we
discuss a preprocessing technique for one of the subproblems, whose aim is to preclude some infeasible assignments upfront. Finally, in Section 3.4 a preprocessing heuristic is presented. Its purpose is to shorten the solution times for one of the subproblems by deciding on values of some variables upfront.

Chapter 4 focuses on the presentation of two case studies for the algorithm introduced in Chapter 3. In particular, Section 4.1 presents the details of the implementation of our method. Then, Section 4.2 discusses the performance of the method against the data set supplied by our industrial partner. In Section 4.3, in order to see the extent to which our method can be generalized, two other realistic scenarios are introduced and the method is tested against them.

Part II focuses on the graph 2-list-colouring problem with colour compatibilities \((G_{2LC-CC})\). We devote Chapters 5-8 to study that problem.

In particular, Chapter 5 is devoted to introducing and defining the problem and placing it in the context of other combinatorial problems. In Section 5.1, after introducing the necessary definitions, the graph 2-list-colouring problem with colour compatibilities is stated. Later, in Section 5.2 the problem is put in the context of other, popular combinatorial problems.

Chapter 6 focuses on the IP formulations to \((G_{2LC-CC})\), as well as their improvements. In Section 6.1, we introduce two formulations to the problem. Then, in Section 6.2 we also show a formulation strengthening technique for both of them. Additionally, in Section 6.3 we present a special case for which a complete polyhedral description of the problem may be derived.

In Chapter 7 we present a decomposition-based solution algorithm for \((G_{2LC-CC})\), which is an adaptation of the algorithm presented in Chapter 3. In particular, Section 7.1 is devoted to introducing the decomposition scheme, which lies at the core of the method. Later, in Section 7.2, a preprocessing scheme is introduced for each of the decomposed subproblems. It is performed before the optimization starts. Then, in Section 7.3 a valid equation class is introduced for one of the subproblems. Finally, Section 7.4 presents one more preprocessing scheme, suitable for one of the subproblems. It is performed whenever the solution to the other subproblem is available.

Chapter 8 will be devoted to the presentation of a computational study. In Section 8.1, the details about the implementation of solution methods will be given. Then, in Section 8.2, the instances used in the computational study will be presented. The results – together with a brief discussion – follow in Section 8.3.
Introduction

At the end of the thesis, we will present the conclusions of its findings. For that end, we will briefly summarize the findings of each chapter. We will also discuss the limitations of our work as well as give indications regarding future research.

Incorporation of Joint Work with other Authors

This thesis incorporates collaborative work with other authors that has been submitted for publication in several joint works. Chapters 3, 4, 5 and 6 are based on the article:


which is referenced here as [Bärmann et al., 2021b]. It is joint work with Andreas Bärmann and Alexander Martin and was submitted for publication in EURO Journal on Transportation and Logistics. The conceptualization of this publication as well as the presented mathematical approaches are the result of joint contributions of all its authors. The proofs of the theorems are significantly based on ideas of the author of this dissertation, complemented by weekly joint discussions with Andreas Bärmann. The author of this thesis also did the implementations for the computational experiments.

Additionally, an early version of the integrated optimization model introduced in Chapter 2 as well as its performance has been published as a chapter in the report:


which is cited here as [Bannenberg et al., 2020]. This chapter is a joint work with Andreas Bärmann and Alexander Martin. The modelling presented there is a result of joint contribution of all authors. The collection of the requirements for modelling, as well as the computational experiments are due to the author of this dissertation.
Further, an early version of the locomotive master problem introduced in Chapter 2 has been discussed in a chapter of the report:


referred to as [Bärmann et al., 2021c]. This chapter is also a joint work with Andreas Bärmann and Alexander Martin. The decomposition approach which allowed for the “extraction” of the locomotive master problem has been developed by all the authors. The author of this dissertation contributed the computational experiments.

Additionally, the anonymized data used in the case study in Chapter 4 was made available in the report:


which we cite as [Bannenberg et al., 2021]. This chapter is also a joint work with Andreas Bärmann and Alexander Martin. All authors contributed to the ideas implemented in the benchmark codes. The author of this dissertation derived the instances from the data made available by the industrial partner. He also prepared the implementations of the solution algorithms tested.

This work has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 765374 (ROMSOC). This research was further supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics – Data – Applications (ADA-Center) within the framework of “BAYERN DIGITAL II” (20-3410-2-9-8).

The author gratefully acknowledges the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) – 440719683.
Part I

The integrated locomotive scheduling and driver assignment problem
In this part of the thesis, we will consider the integrated locomotive scheduling and driver assignment problem. It consists in simultaneously solving two important planning challenges: locomotive scheduling and driver assignment. On top of the constraints relevant to each of the problems itself, the integrated locomotive scheduling and driver assignment needs to comply with additional compatibility requirements. This ensure that both a suitable locomotive and an appropriate driver are selected for a train. They also ask that the locomotive and the driver are mutually compatible.

This part is composed of four chapters: Chapters 1 to 4.

Chapter 1 will be devoted to the presentation of the problem and its place it in the context of the existing literature, as well as to the discussion of its relevance for the global economy and modern rail freight companies. In Section 1.1 we will show that although rail freight transportation is the most CO2-efficient mode of transportation, its share in land transport in the EU is very small. Section 1.2 is devoted to a broad discussion about the large potential of MO applications in the railway industry. We show how the MO methods may be applied across the planning perspectives in that sector.

Then, in Chapter 2, we will present the optimization model for the integrated locomotive scheduling and driver assignment problem. We will also discuss the state of the art in the integrated resource planning problems in rail freight (cf. Section 2.1). Section 2.2 will be devoted to a thorough discussion of the requirements and restrictions considered in the modelling of the problem studied. Next, in Section 2.3, an Integer Programming (IP) model which considers all the requirements collected will be presented. The chapter will conclude with a discussion of clique tightening – a formulation improvement technique (cf. Section 2.4).

Later, in Chapter 3, we will focus on the solution algorithm to the integrated locomotive scheduling and driver assignment problem. In Section 3.1, we introduce the decomposition scheme which underlies the algorithm we develop. It consists in relaxing some of the constraints and solving the two resultant subproblems one after another. Section 3.2 is devoted to the study of four classes of valid inequalities, whose aim it to ensure the global feasibility of the solutions to the decomposed subproblems. Next, in Section 3.3, a preprocessing technique for one of the subproblems is introduced. It allows for the removal of some infeasible assignments upfront. Finally, in Section 3.4 a preprocessing heuristic is presented. Its aim is to decide about the values of some of the variables before solving the associated subproblem, and this way shorten the solution time.
Next, in Chapter 4, we present the computational results for the algorithm introduced in this part. The experiments are carried out for two sets of instances – real, supplied by DB Cargo Polska, and realistic, derived from the real ones. In Section 4.1, we discuss the details of the implementation of our method, as well as the software and hardware characteristic of our computations. Later, in Section 4.2, we turn our attention to the study of the performance of method against the data set supplied by our industrial partner, DB Cargo Polska. We find out that our method allows for a satisfactory solution to the planning challenge at hand. Finally, Section 4.3 is devoted to checking to what extent the method introduced in this part can be generalized. For that end, two other realistic scenarios are introduced. Both of them assume that a smaller number of drivers and locomotives (compared to the instances studied in Section 4.2). We then test our approach against these instances and see the limits to which our method can be extended.
1 Motivation of the problem

In this chapter, we will state the motivation of studying the problem of integrated locomotive scheduling and driver assignment. We will do it in two perspectives. Firstly, we will show the global reasons for the necessity of operational improvements among rail freight carriers. In particular, we will point to the low modal share of rail freight in land transportation, and to its CO2-efficiency. Secondly, we will present the rich body of literature concerning the applications of Mathematical Optimization (MO) in the railway industry. In particular, we will discuss some of the most important works which apply to the planning problems in the railway industry. The papers will be divided by three planning perspectives – strategic, tactical and operational.

1.1 “Demand factors” – a need for an efficient use of resources among rail freight carriers

In this section, we will focus on two phenomena which amplify the need to come up with an algorithm for the integrated locomotive scheduling and driver assignment problem. We will show that rail freight transportation has a relatively small market share. We will also discuss some of the reasons for this state of matter. Then, we will show how rail is a much more CO2-efficient mode of transportation than road transport, and how its broad adaptation may contribute to combating climate change. We will conclude by showing that these considerations warrant a search for the most efficient use of the production resources available.

1.1.1 Increasing the share of rail in land freight transport

Based on the data of [European Court of Auditors, 2016], Figure 1 presents the share of individual modes of transportation in the twenty-seven European Union countries (EU27) between 2011 and 2020. It is visible that road transport accounts for more than three quarters of the transport performance in the EU27 countries. It further shows that the rail freight transport had lost 2.4 percentage points of the market share to the benefit of road transport in the discussed period. Moreover, the decreasing trend appears to be stable throughout the years. We can conclude that rail is not a dominating transportation mode for freight in EU.

[European Court of Auditors, 2016] mentions several reasons for the low modal share of rail in freight transport in the EU. A common denomina-
1 Motivation of the problem

Figure 1: Modal split of freight transport in EU27 countries (as % in total freight transport performance measured in tonne-kilometres). Own elaboration based on [Eurostat, 2020].

The main reason for all of them is the fact that the European rail market is fragmented into several national segments. As a consequence, rail freight suffers a lack of competition in the market in some countries. Further, rail traffic management procedures are frequently not adapted to the needs of rail freight transport. The state of matter is further worsened by problems with infrastructure, which has been neglected for years in favour of road transport. Finally, there are certain technical and administrative constraints which do not help improve the competitive stance of rail freight carriers.

To remediate the situation, [European Court of Auditors, 2016] suggests to continue the works on market liberalization. It also recommends further works to simplify the traffic management procedures, lifting some of the “administrative and technical constraints”, monitoring and ensuring the transparency of the performance of the sector and ensuring a fair competition between different transportation modes.

In our work, we aim at addressing some of the technical constraints mentioned above. These directly influence the competitive position of rail freight transport. An example of such a technical constraint is given by the fact that – unlike truck drivers – locomotive drivers require a license to both the locomotive type and the route they drive, and hence are much less flexible in terms of jobs, to which they may be assigned (see Chapter 2 for more details). Our work aims at improving the way locomotive drivers are assigned to trains, in order to maximize the number of performed trains.
1.1.2 Combating climate change

Combating the climate change has been on the agenda of numerous institutions for many years now. This is due to the grave threats to many people, including the most vulnerable, which it poses (cf. [Caney, 2010]). We will now briefly discuss the way in which rail freight transportation may contribute to combating climate change. We will also mention how solving the integrated locomotive scheduling and driver assignment problem can contribute to this aim.

In their report, [Dantec et al., 2018] mention four benefits of rail transport to combating climate change. First of all, railway transport enables significant economies of scale which reduce the energy consumption and greenhouse gas emission per unit transported. Secondly, the possibility of using electric energy “gives carbon-free and renewable energies an important place for the future of the sector”. Thirdly, railway transportation may, on one hand, contribute additional electricity by recovering the braking energy, and on the other hand, optimize the use of the electrical energy drawn from the network, thereby contributing to its resilience. Finally, high-speed trains (esp. passenger trains) may successfully compete with less energy-efficient and more polluting transportation modes, such as air transport. Let us now consider the four aspects separately.

**Economies of scale**  With regard to the economies of scale allowing for per-unit reduction of CO2 emissions, [IEA, 2017] shows that transport is responsible for 19.1% of the total emissions from fuel combustion in EU27 countries in 2017. Moreover, road transport is responsible for 85.7% of the total transport emissions, while rail transport only accounts for 5.0% of them. With regard to CO2 efficiency of individual transportation modes, Figure 2 presents the CO2 emissions per tonne-km transported by the most popular transportation modes in Europe in 2013. It clearly shows that – in terms of CO2 emissions – rail transport, regardless of the fuel of the locomotive, is two to four times as efficient as the road transport.

**Possibility to use renewable energy sources**  Rail carriers indeed pay a lot of attention to ensuring that the electric energy they use does originate from renewable sources. For instance, DB Cargo Polska has decided to source 50% of the tractive electric energy from renewable sources in 2022 and 100% in 2023 (cf. [Kuzior and Staszek, 2021]). In a similar fashion, the German entities of the Deutsche Bahn group strive to switch to green energy sources. Since 2018, all the express passenger trains (categories ICE, IC and EC) use green electricity (i.e. originating from renewable sources). Moreover, in 2019 approx. 62% of the electric energy used to power trains in Germany originates


1 Motivation of the problem

![Figure 2: CO2 emissions (in grammes) per tonne-km and per mode of transport in the European Economic Areas countries. Own elaboration based on [EEA, 2013].](image)

from renewable sources. DB aims at increasing this percentage to 80% in 2030, and to 100% in 2038 (cf. [Deutsche Bahn, 2019]).

**Contributing energy and stabilizing the network** During its operations (esp. braking), railways may return electric energy to the network. This effect has been applied in the underground railway system in Nürnberg, allowing for savings amounting to hundreds of thousands of euro annually (cf. [Bärmann et al., 2021a]). For other approaches in this field, we refer to [Scheepmaker et al., 2017].

**Viable alternative to less CO2-efficient transportation modes** As shown by [Miller, 2021], considerable CO2 savings are possible thanks to using existing electrified rail as an alternative to air travel in the northeastern US. Similarly, by [Prussi and Lonza, 2018], significant decreases in CO2 emissions may be achieved on many short-haul European flight routes if the passengers travelled by rail instead of flying. They show an example of a connection between Frankfurt and Paris CDG, which – despite a promising greenhouse gases emission reduction potential – are now too long in duration to allow for a real substitution.

To summarize, both the report of [European Court of Auditors, 2016], and the findings of [Caney, 2010] underline the need to increase the modal share of rail in the freight transport in Europe. While these sources suggest primarily regulatory or economic measures, railway freight carriers also need to contribute on their part to be able to accommodate the increased number of orders. In this thesis, we develop an algorithm which – by integrating some of the planning steps – may unlock additional transport capacities without adding new resources i.e. locomotives and drivers.
1.2 “Supply factors” – use cases of MO in rail planning problems

In this section, we will review the works which use MO to optimize problems occurring in the railway industry. We will discuss some of the most important works in the field, dividing them by the planning perspective they relate to – strategic, tactical or operational.

1.2.1 Strategic level decisions

MO can be applied to numerous problems on the strategic planning level in the railway industry. In the following, we discuss two types of such problems – namely capacity assessment and railway network design. They have received considerable attention in the past and are still of interest to both scientists and practitioners. Let us now deal with each of them separately.

Network capacity assessment

Network capacity assessment deals with evaluating the occupation of the existing railway infrastructure and determining possible bottlenecks. This knowledge can in turn be used to better understand the existing timetable constraints and to guide possible investments in infrastructure. [Bešinović and Goverde, 2018] offer a great overview of the topic.

There are two MO-based approaches used for capacity assessment in Europe. Both of them utilize the so-called timetable compression method discussed thoroughly in [Delorme et al., 2009]. The first approach – proposed by the International Railway Association (UIC) – is known as the UIC 406 capacity method (cf. [UIC, 2004]). The second one is the British Capacity Utilization Index (CUI) method as suggested by [Gibson et al., 2002]. In the US, such approaches have not been applied yet – see [Pouryousef et al., 2015].

While these methods can easily be applied to the capacity assessment of corridors (cf. [Landex, 2009, Abril et al., 2005, Abramović et al., 2004, Abril et al., 2008]), their extension to nodes (i.e. stations) is not as straightforward, and requires the use of different mathematical structures, such as the so-called max-plus automata (cf. [Gaubert and Mairesse, 1999, Goverde, 2007, Heidergott et al., 2006]). Finally, the capacity assessment of entire railway networks is not commonly seen; although some approaches have been suggested, see for example [Goverde, 2010] or [Ekman, 2011].

For completeness, we also mention that some approaches to capacity assessment utilize simulation (cf. [Jensen et al., 2017]), queuing theory (cf. [Schwanhäußer, 1994, Büker and Seybold, 2012, Huisman et al., 2002, Weik et al., 2016, Wendler, 2007, Yuan and Hansen, 2007]), parametric mod-
Motivation of the problem

Railway network design The problem of railway network design consists in finding the optimal composition of a railway network. A common objective when solving railway network design problems is to determine the least-cost extension of the existing network, while accounting for constraints such as upgrade cost, transportation cost and rejected orders (cf. [Bärmann et al., 2017]). Some works which use such an objective are [Lai and Shih, 2013] for the case of North America and [Kuby et al., 2001] for the case of China. [Petersen and Taylor, 2001] present a different approach where they focus on expanding the network in such a way that the profit of the railway operator is maximized. They study a case of a Brazilian railway. A case of the extension of the Spanish high-speed railway network is studied by [Blanco et al., 2011]. Some works aim at maximizing the users’ access to the transportation network (see for example [Marín and Jaramillo, 2008]). There are also some works focusing on locating new stations within an existing network. We refer the reader to [Repolho et al., 2013], who deal with such a problem for a Portuguese case, as well as its references.

Railway network design belongs to the family of capacitated network design problems, which are known to be NP-Hard (see [Trukhanov et al., 2010]). There are several solution approaches to the railway network design problem based on Lagrangian relaxation (cf. [Crainic et al., 2001, Gendron et al., 1999]), Benders decomposition (cf. [Costa, 2005]) as well as aggregation (cf. [Bärmann et al., 2015]).

1.2.2 Tactical level decisions

In the following, we will focus on the research works which apply MO methods to tactical level decisions in the railway industry. In particular, we will consider the problems of timetabling, rolling stock scheduling. We will also discuss the crew scheduling, duty generation, crew rostering and crew assignment problems. All of them have received considerable attention in the past and are still of interest to both scientists and practitioners. Let us now deal with each of them separately.

Timetabling Timetabling problems deals with determining an appropriate timetable for the trains envisioned to run in a given period of time, while taking into account numerous factors such as minimal separation between trains or the desired frequency of connections between stations. Following [Cacchiani and Toth, 2012], we will
now discuss several approaches to this problem present in the literature. First of all, some authors seek to construct cyclic timetables – see for example [Serafini and Ukovich, 1989, Nachtigall, 1994, Odijk, 1996, Peeters, 2003, Zimmermann and Lindner, 2003, Liebchen et al., 2008, Liebchen and Möhring, 2008, Nachtigall and Voget, 1996]. There are also works where cyclicity of the timetable is not required – for example [Borndörfer et al., 2006, Brännlund et al., 1998, Cacchiani et al., 2010] as well as [Cai and Goh, 1994, Caprara et al., 2007, Caprara et al., 2002] or [Carey and Lockwood, 1995, Higgins et al., 1997, Jovanović and Harker, 1991]. Secondly, while some works focus on constructing schedules for the entire railway network (for example [Schrijver and Steenbeek, 1994, Nachtigall and Voget, 1996]), other focus on a single line only (also known as corridor) – for example [Cacchiani et al., 2008, Oliveira and Smith, 2000]. Thirdly, some works concentrate on freight traffic (e.g. [Cacchiani et al., 2010]). Others – in turn – consider the passenger traffic primarily (for example [Peeters, 2003, Nachtigall and Voget, 1996]). Finally, the approaches present in the literature may be differentiated by the objective functions, for example, cost of infrastructure use ([Serafini and Ukovich, 1989]), sum of profits of trains in the schedule ([Brännlund et al., 1998]), or minimizing the total delay ([Oliveira and Smith, 2000]).

For completeness, we also mention the numerous approaches which take into account the robustness of a given timetable. It is an important consideration to ensure that the timetables generated are resistant to adverse external factors – delays, rolling stock failures, network maintenance etc. (cf. [Heidt et al., 2016]). The following techniques are used to attain robustness in the timetables: stochastic optimization (cf. [Kroon et al., 2008]), light robustness (cf. [Fischetti and Monaci, 2009]), recoverable robustness (cf. [Liebchen et al., 2009]), delay management (cf. [Cicerone et al., 2009]), bi-criteria and Lagrangian-based approaches (cf. [Schöbel and Kratz, 2009]) as well as metaheuristics (cf. [Tormos et al., 2008]).

**Rolling stock scheduling**  
Rolling stock scheduling is an umbrella term which is used to denote the optimization of the schedules of rolling stock of all types. In the following, we will consider the problem from the perspective of locomotives (also known as engines) and wagon sets. For completeness, we also mention the case of optimizing the rotations of electric multiple units, which are – by their nature – similar to locomotive rotations (cf. [Cacchiani et al., 2012]).

The problem of locomotive scheduling (cf. [Vaidyanathan and Ahuja, 2015]), also known as engine scheduling (cf. [Florian et al., 1976]), has received much attention from researchers throughout the world. An excel-
lent survey of the topic was given by [Piu and Speranza, 2014]. Following them, we can divide the existing literature by a number of criteria. First of all, some works deal with assigning single locomotives to trains (cf. [Lübbecke and Zimmermann, 2003, Illès et al., 2005, Illès et al., 2006, Fügenschuh et al., 2006]). There is also a large number of articles considering the case of multiple locomotives, or locomotive consists (cf. [Florian et al., 1976, Ziarati et al., 1997, Ziarati et al., 1999, Noble et al., 2001, Vaidyanathan et al., 2007]). Secondly, while some works deal with rail freight traffic (cf. [Florian et al., 1976, Ziarati et al., 1999, Ahuja et al., 2006, Vaidyanathan et al., 2007, Fügenschuh et al., 2006, Fügenschuh et al., 2008, Frisch et al., 2021]), others focus on the passenger traffic (cf. [Cordeau et al., 2000, Illès et al., 2005, Illès et al., 2006, Paoletti and Cappelletti, 2006]). Thirdly, we notice some authors concentrate their efforts on railways in North America (see [Charnes and Miller, 1956, Florian et al., 1976, Vaidyanathan et al., 2007]), while others deal with cases of European railways (for example [Booler, 1980, Illès et al., 2005, Paoletti and Cappelletti, 2006, Fügenschuh et al., 2008]).


Crew scheduling, duty generation, crew rostering and crew assignment

The four problems – crew scheduling, duty generation, crew rostering and crew assignment – deal with finding an appropriate assignment of crews to trains. Crew scheduling is defined as the creation of anonymous duties, which cover all the trains a driver has to serve in a given period of time (e.g. one work day). As the number of feasible duties may get extremely large, some authors use the additional step of duty generation to generate high-quality duties beforehand. Then, in the crew rostering step, the duties are combined to weekly or monthly sequences which are later assigned to individual crew members. In the context of freight transportation, the highly unstable nature of order books (cf. [Jütte and Thonemann, 2015]) warrants the need to assign crew members to trains directly in a planning step called crew assignment (performed instead of the three steps mentioned before). Although these
four problems are quite diverse, they all aim at the generation of an optimal attribution of drivers to jobs. Hence we consider them jointly in this section.

An excellent literature survey in this field is due to [Heil et al., 2020]. Following them, we will now briefly describe the body of the literature in this field.

The works in this field may be classified by the planning problem they deal with: some works deal solely with crew scheduling (cf. [Abbink et al., 2011, Chen and Shen, 2013, Desaulniers et al., 2002, Fuentes et al., 2019, Grötschel et al., 2003]), while others combine it with crew rostering (cf. [Abbink, 2014, Abbink et al., 2018, Dalal and Jensen, 2001, Ernst et al., 2001, Khmeleva et al., 2018]). Some works focus only on crew assignment (cf. [Jütte and Thonemann, 2015]). Further division may be drawn by differentiating between studies dealing with passenger transport (cf. [Ernst et al., 2001, Hoffmann et al., 2017, Koniorczyk et al., 2015, Kroon et al., 2009]) and freight transport (cf. [Bach et al., 2016, Dalal and Jensen, 2001, Vaidyanathan et al., 2007]). We can also differentiate between works focusing only on drivers (cf. [Albers, 2009, Banihashemi and Haghani, 2001, Ernst et al., 2001, Laplagne et al., 2005, Yaghini et al., 2016]), conductors (or other non-driver crew) ([Derigs et al., 2010]) or both (cf. [Abbink et al., 2011, Balakrishnan et al., 2016, Bengtsson et al., 2007]).

From the modelling perspective, we notice that some works use Set Covering formulation (cf. [Hoffmann et al., 2017, Jütte and Thonemann, 2015, Khmeleva et al., 2018, Kroon et al., 2009]), while others utilize Set Partitioning (cf. [Desaulniers, 2007, Grötschel et al., 2003]). Authors also use various objectives to evaluate their schedules. The most popular ones are total cost (for example [Abbink, 2014, Chen and Shen, 2013, Derigs et al., 2010, Grötschel et al., 2003, Jütte and Thonemann, 2015]), total number of duties (see for example [Alfieri et al., 2007]) or total idle time (for instance [Shi, 2015]). Last but not least, the approaches in the literature vary by the solution method they employ. The most frequent ones include column generation (for instance [Abbink et al., 2011, Caprara, 2015, Grötschel et al., 2003, Hoffmann et al., 2017]), heuristics (cf. [Alefragis et al., 2000, Balakrishnan et al., 2016]), integer programming methods (cf. [Boschetti et al., 2004, Chew et al., 2001]) and meta-heuristics (cf. [Cabrera and Rubio, 2009, Elizondo et al., 2010]).
1 Motivation of the problem

1.2.3 Operational level decisions

There is also a number of problems on the operational planning level which can be successfully modelled and solved with the use of MO techniques. In the following, we will consider two such problems – real-time train rescheduling and delay management (also known as dispatching). Each of them will be treated in a separate paragraph.

Real-time train rescheduling Every now and then, situations happen which prevent the train timetable from being realized as planned. This may have a serious impact on the entire traffic – both on the affected railway corridor and in the entire network. Real-time train rescheduling deals with minimizing the impact of such adverse events. [Cacchiani et al., 2014] defines two types of disruptions: disturbances, which are relatively small perturbations of the railway system that can be handled by modifying the timetable, but without modifying the assignments for rolling stock and crew, and disruptions, which are relatively large incidents, requiring both the timetable and the assignments for rolling stock and crew to be modified. [Cacchiani et al., 2014] offers a great survey of the works dealing with real-time train rescheduling. Following them, we will now present the most important contributions to the field. The existing works focus on rescheduling the timetables, the rolling stock rotations or the crew rosters. There are also some integrated approaches. In the following, we will deal with each of them separately.

With regard to timetable rescheduling, [Cacchiani et al., 2014] suggest a division into macroscopic perspective (i.e. considering the railway network at a higher level, in which stations can be represented by nodes of a graph and tracks by arcs) and a microscopic perspective (i.e. accounting also for block sections and signals i.e. components of a railway line). Some works which deal with disturbances on microscopic level are [Bocci et al., 2013, Corman et al., 2009, Corman et al., 2010a, Corman et al., 2010b, Lusby et al., 2013, Rodriguez, 2007]. There are also papers dealing with disturbances on macroscopic level (see for example [Acuna-Agost et al., 2011, Dollevoet et al., 2012] as well as [Schachtebeck and Schöbel, 2010, Schöbel, 2009]). Works considering disruptions are less numerous, both on the microscopic (see for example [Corman et al., 2011, Hirai et al., 2009]) and macroscopic level (for instance [Albrecht et al., 2013, Louwerse and Huisman, 2014, Narayanaswami and Rangaraj, 2013]).

The literature pertaining to the rescheduling of rolling stock or crew is not as rich. Nonetheless there are some works which should be mentioned in our context. Some of the most important approaches to rolling stock reschedul-
1.2 “Supply factors” – use cases of MO in rail planning problems

ing are given by [Budai et al., 2010, Fioole et al., 2006, Nielsen et al., 2012, Kroon et al., 2015]. With regard to crew rescheduling, there is a group of authors who use Set Covering in their modelling [Huisman, 2007, Potthoff et al., 2010, Rezanova and Ryan, 2010, Sato and Fukumura, 2011]. Some of them – namely [Rezanova and Ryan, 2010] and [Potthoff et al., 2010] – allow for task cancellation at the expense of incurring a penalty. We should also mention the agent-based system approach of [Abbink et al., 2009] here. Finally, with regard to integrated approaches, there are works which tackle timetable and crew rescheduling jointly (cf. [Walker et al., 2005, Veelenturf et al., 2012]). There are also works dealing with timetable and rolling stock rescheduling simultaneously (cf. [Cadarso et al., 2013, Fekete et al., 2011, Veelenturf et al., 2017]).

**Delay management** Once a delay has occurred, it is likely to spread across the entire network. Delay management is the task of controlling this phenomenon. It was introduced by [Schöbel, 2001] and [Suhl et al., 2001], who deal with the so-called wait-depart decision: should a connecting train wait for a delayed feeder and propagate the delay in the network or depart on time so that transferring passengers will miss their connection? This problem has received a lot of attention from researchers in the last two decades. [König, 2020] has offered a great survey of this topic. Following them, we will now mention several works concerning delay management, divided by a number of criteria.

First of all, the objectives of the models introduced may focus on the passenger (cf. [Corman et al., 2017, Dollevoet and Huisman, 2014, Dollevoet et al., 2015, Heilporn et al., 2008, Schöbel, 2007]), on the train (for example [Cicerone et al., 2012, Goerigk et al., 2014]) or both (cf. [Corman et al., 2012, Dollevoet et al., 2014]). Recall that in the paragraph devoted to real-time train rescheduling we defined the terms disturbances (minor) and disruptions (major). The existing body of the literature about delay management can be divided similarly. Some works focus on the delays resulting from disturbances (for example [Berger et al., 2011, Schmidt, 2013, Xu et al., 2018]), while others consider the large disruptions (cf. [Malucelli and Tresoldi, 2019]). Additionally, as suggested by [Cacchiani et al., 2014] and discussed in the paragraph considering real-time train rescheduling, the perspective taken by the approaches may be macroscopic (cf. [Schöbel, 2001, Schöbel, 2007, Heilporn et al., 2008, Kliwer and Suhl, 2011]) or microscopic (cf. [Corman et al., 2012, Corman et al., 2017, Xu et al., 2018]). Next, we can divide the works by the type of input: deterministic (cf. [Schöbel, 2007, Heilporn et al., 2008]), stochastic (cf. [Berger et al., 2011,
Motivation of the problem

Schön and König, 2018]) or incomplete (for example [Bauer and Schöbel, 2014, Kliewer and Suhl, 2011, Schachtebeck and Schöbel, 2010]). Finally, different works introduce various solution methods: either exact (for instance [Schöbel, 2001, Dollevoet et al., 2014]) or heuristic (cf. [Kliwer and Suhl, 2011] or [Bauer and Schöbel, 2014]).

In this chapter, we described the demand and supply factors. It is clear that a need exists for our solution, and that the existing body of literature warrants the use of MO in search of the solution to the problem considered.
2 Modelling the integrated locomotive scheduling and crew assignment problem in rail freight

In this chapter, we will focus on modelling the integrated locomotive scheduling and driver assignment problem. In the beginning, we will put the problem in the context of other, similar integrated approaches present in the literature. We will also discuss how our approach contributes to the state of art in the field. Later, we will present the list of requirements and restrictions to include in our modelling. Some of the planning aspects included originate from our industrial partner, DB Cargo Polska. They will be treated by the resource group they pertain to – locomotives, drivers and their compatibilities. Equipped with this knowledge, we will proceed to defining sets, variables and finally presenting the optimization model. This chapter will conclude with a discussion on a model preprocessing technique – clique tightening.

2.1 State of the art in the integrated resource planning: a literature overview

In this section, we will discuss some of the works considering integrated resource planning. As mentioned in the introduction to this thesis, the topic of joint vehicle and crew scheduling in general has been treated quite extensively in the literature. However, the primary focus of the research works in this field lies on urban transportation and airlines. Only a few works considering joint vehicle and crew scheduling in the railway context can be found, although there is a long history of vehicle scheduling and crew scheduling approaches studied individually here. These were discussed extensively in the previous chapter.

In the following, we present the state of the art in the integrated resource planning. We begin though with a brief rationale for the reasons which allow us to skip the duty scheduling step. Then, we group the integrated approaches by three dimensions: exact, heuristic and robust. Later, we briefly mention the approaches in which vehicle scheduling, crew scheduling and crew rostering are considered. Finally, we consider two papers in which locomotive scheduling was integrated with crew scheduling in the railway context. We also relate them to the contributions of this thesis.

Reasons for skipping the duty scheduling The duty scheduling, discussed in more detail in Section 1.2.2, is a complex step in the planning
process in the railway industry, especially for rail passenger traffic. Unlike passenger trains, rail freight transportation is – to a significant extent – a last-minute business (see [Jütte and Thonemann, 2012]). The existing literature suggests that constantly changing order books are normal to all rail freight carriers. For example, [Heil et al., 2020] mentions a case of a large European freight railway carrier, for whom at most 80% of all trains in the order book are regular trains, with the remaining 20% being added, cancelled or changed at short notice. Similar proportions are reported by our industrial partner. In some cases, the order books are completely irregular – see [Kumar et al., 2009]. As a result, the standard operational planning horizon for rail freight carriers ranges between a day and a month. Overall, the discussed phenomenon of frequent changes in the order book limits the usability of pre-planned rosters in the planning practice of rail freight carriers.

Taking into account the highly unstable nature of order books, the approach we introduce skips the step of duty generation. To incorporate the working time constraints, which are usually dealt with in the stage of duty generation, we consider the most popular types of such constraints (i.e. “short” breaks after each shift, “long” breaks once a week, non-working days on weekends etc.) in the step of driver assignment (cf. Section 2.2) Hence, the omission of duty generation causes no harm to the quality of driver rosters – they are compliant with working time regulations.

**Integrated vehicle and crew scheduling** First, it can be noted that a vast majority of the works in this field studies bus or urban transportation systems. An excellent overview in this context was published by [Ibarra-Rojas et al., 2015]. Exemplarily, we mention the works of [Haase et al., 2001, Freling et al., 2003, Huisman et al., 2005, Mesquita and Paías, 2008, Steinzen et al., 2007, Borndörfer et al., 2008, Laurent and Hao, 2008, Amberg et al., 2019, Perumal et al., 2021]). Some works consider planning in airlines, see for example [Cordeau et al., 2001b, Mercier et al., 2005, Mercier and Soumis, 2007, Dunbar et al., 2014]. Other areas for which integrated vehicle and crew scheduling models were developed include postal delivery ([Hollis et al., 2006]) and road transportation ([Drexl et al., 2013]).

Following [Ibarra-Rojas et al., 2015], we can divide the existing approaches to integrated vehicle and crew scheduling problem into exact and heuristic ones.

**Exact approaches** Most of the exact solution methods are based on column generation. It is frequently used together with Lagrangian heuris-
tics to solve set partitioning/covering formulations of the problem, see e.g. [Haase et al., 2001],[Freling et al., 2003] or [Huisman et al., 2005]). These approaches are able to supply good-quality solutions to relatively small instances within a three-hour period on a personal computer. Column generation is also used with a different, quasi-assignment model by [Gaffi and Nonato, 1999]. A different approach was suggested by [Borndörfer et al., 2008], who developed a Lagrangian relaxation method to solve the problem. Based on these works, [Mesquita and Paias, 2008] developed a solution incorporating column generation for a partitioning/covering model and dedicated branching rules. This allows them to solve larger instances and to reduce computation time compared to e.g. [Huisman et al., 2005] or [Borndörfer et al., 2008].

**Heuristic approaches** An important contribution to heuristic approaches to the integrated vehicle and driver scheduling problem was made by [Huisman, 2004]. He proposes a kind of moving-horizon modelling which is solved via Lagrangian-relaxation-based heuristics, column generation as well branch and bound algorithms. Other heuristic approaches in the literature include Greedy Randomized Adaptive Search Procedures (GRASP) ([Laurent and Hao, 2008, De Leone et al., 2011]), evolutionary algorithms ([Steinzen et al., 2007]) as well as local search algorithms ([Valouxis and Housos, 2002]).

We note that a considerable number of studies considers crews to be uniform or capable to work with any vehicle (e.g. [Gaffi and Nonato, 1999], [Freling et al., 2003], [Laurent and Hao, 2008] or [Perumal et al., 2021]), whereas the studies considering a limited crew-vehicle comparability are not as numerous (see e.g. [Huisman, 2004], [Hollis et al., 2006] or [Boyer et al., 2018]). Similarly, while some works focus on only one type of vehicles (such as [Haase et al., 2001] or [Mesquita and Paias, 2008]), others include multiple types (see [Gaffi and Nonato, 1999], [Cordeau et al., 2001a], [Borndörfer et al., 2008] or [Amberg et al., 2019]).

**Robust approaches** In the context of integrated vehicle and driver scheduling for airlines, robust optimization approaches play a significant role. For example, the works of [Weide et al., 2010], [Dück et al., 2012], [Petersen et al., 2012] and [Dunbar et al., 2014] present models which attain robustness against delay propagation in the schedule. While the methods developed in [Dück et al., 2012], [Petersen et al., 2012] and [Dunbar et al., 2014] minimize the sum of propagated delays, [Weide et al., 2010] focuses on cost minimization.
Approaches to integrated vehicle scheduling, crew scheduling and rostering There are also some studies which combine vehicle scheduling, driver scheduling and driver rostering. In [Mesquita et al., 2013], the authors propose a formulation combining set covering, multi-commodity flow and covering-assignment, which is then solved by Benders decomposition. There are also some heuristic approaches to the combined problem, such as [Shen and Xia, 2009] based on local search or [Mesquita et al., 2011] based on an iterative MIP (mixed-integer programming) heuristic. For the sake of completeness, we also mention that some works consider vehicle and crew routing as well as scheduling jointly – on top of assigning crews and vehicles, they also decide on the departure and arrival times of individual connections (see e.g. [Lam et al., 2020]). We also mention the work of [Borndörfer et al., 2017], who show an example of planning mobile tours of toll inspectors on German motorways, for which they present an integrated vehicle routing and crew rostering model as well as a real-world case study.

Integrated vehicle and crew scheduling in railway planning and our contribution In a thorough study of the available literature, we only found two works which attempt to solve the integrated vehicle and crew scheduling problem in the context of railway transport. To the best of our knowledge, the first work in this field is due to [Aksoy and Altan, 2013]. Using a multi-commodity flow formulation with node demands, they optimize the flows of locomotives and crews between yards (to which trains are assigned). Their objective is then to ensure the required number of locomotives and crews are available at a yard at the beginning of a given day of a week while minimizing the costs of moving each without a train. However, their model is a very coarse representation of the necessities of the real-world planning process. It mainly captures the flow balance for drivers and locomotives between yards in order to perform a capacity planning matching the number of required drivers and locomotives on each day of the week. Their model does not include an assignment of concrete drivers and locomotives to the trains to be staffed. Further, it does not include many of the critical constraints for ensuring the feasibility of such an assignment, like the working time constraints of the drivers. The computational results they show only feature a trivial instance with 3 yards, with no mentioning where the data came from.

Compared to [Aksoy and Altan, 2013], the model we present includes a detailed representation of the standard operational requirements, including various locomotive types and non-uniform skills of the drivers. Moreover, we derive a novel, efficient algorithmic solution approach based on problem decomposition, cutting planes and a dedicated preprocessing. The quality of the resulting model and the performance of the dedicated solution algorithm
2.2 Overview of modelling requirements and assumptions

are demonstrated at the hand of real-world, country-scale problem instances provided by DB Cargo Polska.

Our work is further related to that of [Dauzère-Pérès et al., 2015], who focus on vehicle scheduling and crew scheduling in railway passenger traffic. Similar to us, they start from the relaxation of the coupling constraints between driver and locomotive assignment. Then they use a Lagrangian relaxation heuristic to obtain high-quality solutions. They also describe their computational and implementation experience gathered in collaboration with their industrial partner, the French national railway carrier SNCF. In comparison to [Dauzère-Pérès et al., 2015], our case study is significantly more comprehensive in that it is based on country-wide data for a whole month, whereas they focus on a single region and a time horizon of one week.

Most notably, we consider the integrated vehicle scheduling and driver assignment problem in the railway industry. This means, unlike the other two works discussed above, our model includes assigning concrete drivers to concrete trains to be staffed, including detailed working time requirements – which are not explicitly stated in [Dauzère-Pérès et al., 2015]. Further, our approach, enhanced by cutting planes, can solve instances which are about five times larger than those presented in [Dauzère-Pérès et al., 2015].

2.2 Overview of modelling requirements and assumptions

In the following, we will introduce the modelling requirements and restrictions we took into account to construct a first, basic version of the model. We will consider them for locomotives and their drivers separately. Additionally, we will consider the requirements and restriction pertaining to the mutual compatibility of locomotives, drivers and trains.

A key question for optimizing the use of locomotives and drivers is: how many of the planned trains can be carried by compatible locomotives and served by drivers with appropriate licenses given the scarce resources at hand at a railway company? The model developed here is thus intended to find such an assignment of drivers and locomotives to the pre-scheduled trains, with the aim that as many of them as possible can run.

As mentioned above, our model considers three distinct blocks of requirements and restrictions; they are related to (i) drivers, (ii) locomotives and (iii) mutual compatibilities between locomotives, drivers and the trains. All necessary information about the trains to be performed is given in the so-called order-book – a data set containing details about the origin and destination
station of each train, its planned departure and arrival times as well as locomotive and driver requirements. In our modelling, we include numerous classical working time constraints, encountered across numerous geographies. This includes maximum shift length, minimum break between shifts, a longer break once a week and a specified day off every few weeks. We also account for the fact that the drivers are usually assigned to a planned region and that some shifts need to start or end in that region. Additionally, to distinguish between individual weeks in longer time horizons, we will use the notion of calculation weeks – which are subsequent, non-overlapping planning periods, spanning seven days each.

They are needed to ensure that at least one long break, explained in more detail later, is fully included in each of the calculation weeks. In this work, we will consider time frames spanning from one week to one month. More details about the individual instances considered are included in Chapter 8. Given the above information, we can use an indirect modelling of time and space to keep the size of the resulting model small. In particular, we will model time and space via mutual compatibilities and conflicts of individual trains.

In the next subsections, we will introduce the modelling requirements and restrictions we took into account when deriving the model presented later. They pertain to all the three constraint groups mentioned above. The requirements stem from physical and legal conditions and hence must not be violated. The restrictions reflect the planning practice of our industrial partner; therefore, we provide a short rationale for each of them and also mention in which way they could be extended to consider other planning practices. Thus, although our model strongly reflects the tasks that are faced by planners in Poland, it is still a general approach which can be adapted to other regions via modifications of the assumptions stated below.

2.2.1 Driver restrictions and requirements

This subsection presents restrictions and requirements related to the planning of drivers. They comprise the standard types of driver-related limitations (e.g. working time, assignment to regions etc.) as well as the planning practice of the industrial partner.

Requirements  One of the commonly encountered requirements which we include in our work prescribes that drivers work for some maximal number of hours. We model this maximal length of a shift with a parameter $c_{\text{shift}}$. It is also a standard requirement that the driver has to rest for at least a stipulated minimal time after each shift. Such a minimal rest time will be defined by
2.2 Overview of modelling requirements and assumptions

a parameter $c_{\text{short}}$. We will call a break following each shift a short break. Yet another standard operational limitation consists in ensuring that each driver has at least one long break, defined as a longer period of uninterrupted rest, per calculation week. The duration of the long break is modelled by the parameter $c_{\text{long}}$. We also assume that the entire duration of the long break must be incorporated in full within one week. Another constraint which is frequently seen in the literature requires the driver to have a specific day of week off once every few weeks. In our approach, we assume that the day off is Sunday, although it could well be any other day of week. Moreover, the frequency in which the day off shall be granted will be modelled by the parameter $c_{\text{sunday}}$, which denotes the maximal number of working Sundays per month.

Another common planning practise is to assume that every driver has a set of “licensed” train routes (e.g. origin-destination pairs). We may assign the driver only to trains traversing these routes. We also need to consider the fact that each driver is licensed to a limited number of locomotive classes, and only these locomotive classes can be assigned to the considered driver. We require that this be unconditionally respected. Table 1 presents a summary of the parameters used for modelling the working time constraints.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{\text{shift}}$</td>
<td>Maximal duration of shift in hours</td>
</tr>
<tr>
<td>$c_{\text{short}}$</td>
<td>Minimal duration of short break in hours</td>
</tr>
<tr>
<td>$c_{\text{long}}$</td>
<td>Minimal duration of long break in hours</td>
</tr>
<tr>
<td>$c_{\text{sunday}}$</td>
<td>Maximal count of working Sundays in a month</td>
</tr>
</tbody>
</table>

Restrictions Our industrial partner assigns drivers to one of three planning regions, corresponding to a geographic subdivision of Poland into three regions – roughly south, west and east. We assume that we are free to schedule drivers to any train they are licensed to drive. However, per company directive a driver’s first job in the planning period must start in the home region of the driver. Similarly, the drivers’ last job in the planning period has to end in the respective home region to which they are assigned. As a consequence, we have to let drivers rest in hotels if they end their work away from their home region. Based on real-world practice, we take it that sometimes drivers are brought to the first train in the shift by car. We also assume that drivers rest in the location of the destination station of the last train of their shift. In our
modelling, we grant the drivers a short break plus transportation time if a transport is required. We could easily extend the model to accommodate to other rules regarding work location by adapting the availability of drivers to drive trains far away from their planning region.

As per the planning practice of our industrial partner, we do not consider intra-shift breaks in the following. We would like to point out though that our approach can easily be generalized to account for at least some of such rules – for example by enforcing a break of fixed length after each served train. This could be achieved by a slight redefinition of one of the sets required for constraint construction (discussed later in this section). For a discussion of cases where intra-shift breaks are actually required, we refer to [Bach et al., 2016].

We further do not consider the possibility to perform training rides (aiming at maintaining and extending the set of routes to which a driver is licensed) as this would result in an overmodelling – since the validity of a route in the set of “licensed” routes in Poland is 180 days, such decisions need to be made on a tactical or even strategic level, depending on the planned order portfolio. They could easily be integrated, however, by excluding the drivers from serving any trains during their training periods, and by directing appropriate locomotives to the corresponding stations (by fixing the relevant variables). A similar argument holds for extending the drivers’ certification to a new locomotive type – given the price of new locomotives, such decisions are usually made on a strategic level, whereas we focus on the operational activities.

We assume that at most one driver can be staffed to a train. Unlike public transport and airline industry, this is standard operational practice at our industrial partner. Nevertheless, our model could be extended to allow for multiple drivers in one train by changing one constraint group from set packing to set cover. While we would gain another degree of freedom to transport crew via deadheading, this would result in a different problem structure and would require a separate study of polyhedral properties.

2.2.2 Locomotive restrictions and requirements

Next, we will discuss the restrictions and requirements related to the planning of locomotives. We include many of the constraints frequently encountered in this field as well as the particular planning practice of our industrial partner.

Requirements Concerning the locomotives, the basic requirements we consider comprise their tractive power and their source of energy (electricity or diesel). A locomotive needs to have sufficient horsepower to carry a given
2.2 Overview of modelling requirements and assumptions

train, and the availability of catenaries needs to be taken into account. Further, unlike drivers, locomotives may only pick up trains in the location where their previous train has ended.

**Restrictions** For the sake of simplicity, we assume that we are free to select the starting location of each locomotive. It is, however, straightforward to consider the actual starting locations of the locomotives by restricting the respective set of initial trains they can cover. Additionally, we assume that locomotives only move when carrying a train. Note here that we do not plan any so-called *empty runs* of locomotives in our model, i.e. trips of locomotives without a train in order to place them where they are needed next. We instead assume that empty runs are pre-planned and thus already part of the order book. The reason for such an approach lies in the organization of the planning process at our industrial partner, who plans the empty runs on an aggregate level, before the individual client orders are confirmed or rejected. As a result, we sometimes see a slightly less-than-100% coverage of the trains scheduled therein. While differentiating between “normal” trains and empty runs would be interesting, for the ease of exposition, we chose not to incorporate this distinction, as this would have required additional notation. On the other hand, it is straightforward to include it in the model.

We further take it that only one locomotive is required to carry each train. This limitation is again drawn from the operational practice of our industrial partner and could be at least partially lifted by a slight change in the modelling approach. Further, since some trains in the instances supplied are carried by locomotives which are not the property of our industrial partner, we require that for each of these trains a “virtual” locomotive exists which is capable of carrying only that one train. For such trains, we only need to find a suitable driver.

Finally, it should be mentioned that the maintenance needs of locomotives are not included directly in our model, since planning the time and location of maintenance periods is largely a different planning problem. We should also note that maintenance constraints constitute a critical part of the planning chain, both for freight and passenger transportation, where the vehicles frequently do not return to their depots for several weeks. However, we could easily accommodate for at least the basic, daily maintenance schedules by making sure that the locomotive stays at the maintenance station for a sufficiently long time between carrying trains.
2 Modelling the integrated locomotive scheduling and crew assignment problem

2.2.3 Compatibilities of locomotives, drivers and trains

Last but not least, we summarize the requirements and the restrictions imposed by our industry partner which are related to the mutual compatibilities between locomotives, drivers and trains.

**Requirements** We require that each train be served by a driver who is licensed to the route of the train and to the employed locomotive type as described in Subsection 2.2.1. We also enforce that a locomotive selected for a given train has sufficient horsepower and that catenaries are available if an electric locomotive is to be used, see Subsection 2.2.2.

**Restrictions** Normally, the planners make an indication of the locomotive type which shall serve each train already at the stage of constructing the order-book. We relax this assumption and allow that a train is carried by any locomotive having sufficient power and an appropriate energy source. Therefore, in many cases we can benefit from the ability to choose between more locomotive types than just the stipulated one.

2.3 The optimization model

In this section, we will model the optimization problem described above as a combination of a set-packing problem with compatibility, conflict, and multiple-choice constraints and a multi-commodity-flow problem. In particular, we will briefly define the inputs to the model, as well as an initial preprocessing technique. We will also explain some of the modelling assumptions. Later, we will formally define sets, variables and parameters required for the model construction. Finally, the model will be presented and explained.

The inputs to the model are a set $T$ of trains to be performed, a set of locomotives $L$ and a set of drivers $D$. Moreover, let us define the set $S$ of stations which contains the origins and destinations of all trains. For each train $t \in T$, we consider its origin $o(t) \in S$, its arrival station $a(t) \in S$ as well as its departure time $s(t) \in \mathbb{R}$ and arrival time $e(t) \in \mathbb{R}$. Additionally, for each driver $d \in D$ let $H(d) \subset S$ denote the stations which belong to the home region of that driver. To denote the subsets of locomotives compatible with a driver $d \in D$ or a train $t \in T$, we use $L^d$ and $L^t$ respectively. Let $D^l$ and $D^t$ represent the sets of drivers compatible with a locomotive $l \in L$ or a train $t \in T$ respectively. We also define $T^l$ and $T^d$ as the subsets of trains compatible with a locomotive $l \in L$ or a driver $d \in D$ respectively.
2.3 The optimization model

We notice that some of the locomotives $l \in \mathcal{L}$ have exactly the same set of compatible trains $T^l$ as other locomotives in $\mathcal{L}$. In order to make the locomotive part more compact, we partition $\mathcal{L}$ into subsets $L$ in which all locomotives $l \in L$ have the same sets of compatible trains $T^l$ and compatible drivers $D^l$. We call this partition $\mathcal{L}$, and a subset $L \in \mathcal{L}$ is called a locomotive class. Further, we define $T^L \subseteq T$ as the set of trains compatible with a locomotive class $L \in \mathcal{L}$ and denote by $\mathcal{L}^t \subseteq \mathcal{L}$ the set of locomotive classes compatible with a given train $t \in T$. Additionally, $\mathcal{L}^d \subseteq \mathcal{L}$ shall denote the subset of locomotive classes compatible with a given driver $d \in D$, and $D^L$ shall denote the set of drivers compatible with a locomotive class $L$. Finally, we use $W \subseteq \mathbb{N}$ to denote the set of calculation weeks.

Before building the model, we also perform an initial preprocessing of the drivers, locomotives and trains sets. If, for a train $t \in T$ and for a locomotive class $L \in \mathcal{L}^t$ no suitable driver can be found (or $D^t \cap D^L = \emptyset$), we remove the locomotive class $L$ from $\mathcal{L}^t$. We also remove the train $t$ from the set $T^L$. Similarly, if for a train $t \in T$ and for a driver $d \in D^t$ no suitable locomotive can be found (or $\mathcal{L}^t \cap \mathcal{L}^d = \emptyset$), we remove the driver $d$ from $D^t$. We also remove the train $t$ from the set $T^d$.

As a consequence of the indirect modelling of time indicated in the previous subsection, we assume that each shift of a driver has a unique first and a unique last train (job). Moreover, as we assume that we may plan at most one driver to drive each train, any train $t \in T^d$ can be a first job or a last job in a shift for at most one driver $d \in D$. This assumption does not apply to the jobs which we later denote as the last ones before the long break. For these we allow that the arrival time of one train determines the beginning time of the long break for an arbitrary number of drivers. This is due to a different nature of the long break – rather than at the end of each shift, it needs to be fully included in the respective calculation week and not between the two of them. In other words, for each driver, at least one long break needs to be included fully in each calculation week. Hence, only a subset of trains within the calculation week may serve as an indicator of the beginning of the long break. We also allow the long break to actually be longer than $c_{\text{long}}$ hours to the benefit of the drivers. This matter will be discussed in more detail in Section 2.3.1.

2.3.1 Set definitions

We also introduce a number of sets required to build the constraints of the model. They represent the relationships between trains to incorporate the assumptions and requirements discussed in Subsection 2.2. For example, they list the trains which run simultaneously to another one, or trains which
could be served successively. In Table 2, we introduce parameters which allow for the modification of sets and constraints pertaining to the working time regulations. This way, our approach is general and hence it may easily be applied for various companies in different countries.

In the next paragraphs, exact definitions of the sets required for the constraints construction will be given. For ease of notation, we write \( t_1 \leq t_2 \) for two trains \( t_1, t_2 \in T \) if \( t_2 \) departs at the same point in time or later than \( t_1 \).

**Table 2: Summary of parameters required for set construction**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s(t) )</td>
<td>Starting time of a train ( t \in T )</td>
</tr>
<tr>
<td>( e(t) )</td>
<td>Ending time of a train ( t \in T )</td>
</tr>
<tr>
<td>( s^{\text{week}}(w) )</td>
<td>Starting time of a calculation week ( w \in W )</td>
</tr>
<tr>
<td>( e^{\text{week}}(w) )</td>
<td>Ending time of a calculation week ( w \in W )</td>
</tr>
<tr>
<td>( s^{\text{sunday}}(w) )</td>
<td>Starting time of a Sunday falling in week ( w \in W )</td>
</tr>
<tr>
<td>( e^{\text{sunday}}(w) )</td>
<td>Ending time of a Sunday falling in week ( w \in W )</td>
</tr>
<tr>
<td>( o(t) )</td>
<td>Origin station of a train ( t \in T )</td>
</tr>
<tr>
<td>( a(t) )</td>
<td>Destination station of a train ( t \in T )</td>
</tr>
<tr>
<td>( \tau^{s_1,s_2} )</td>
<td>Transit time between stations ( s_1, s_2 \in S )</td>
</tr>
<tr>
<td>( c^{\text{shift}} )</td>
<td>Maximal duration of shift in hours</td>
</tr>
<tr>
<td>( c^{\text{short}} )</td>
<td>Minimal duration of short break in hours</td>
</tr>
<tr>
<td>( c^{\text{long}} )</td>
<td>Minimal duration of long break in hours</td>
</tr>
<tr>
<td>( c^{\text{sunday}} )</td>
<td>Maximal count of working Sundays in a month</td>
</tr>
</tbody>
</table>

**Forward-looking daily break set** \( T_{t,d}^{B^+} \) The set \( T_{t,d}^{B^+} \) is used to represent, for each driver \( d \in D \) and for all \( t \in T^d \), the trains \( t_1 \in T^d \) which cannot be assigned to driver \( d \) if \( t \) is the last train in the shift before a short break. We can formally state the set \( d \in D \) and \( t \in T^d \) as

\[
T_{t,d}^{B^+} := \{ t_1 \in T^d : s(t_1) \geq e(t) \land e(t_1) \leq e(t) + c^{\text{short}} \} \\
\cup \{ t_1 \in T^d : s(t_1) > s(t) \land s(t_1) < e(t) + c^{\text{short}} \land e(t_1) \geq e(t) + c^{\text{short}} \} \\
\cup \{ t_1 \in T^d : s(t_1) > s(t) \land e(t) + c^{\text{short}} + \tau^{a(t), o(t_1)} > s(t_1) \}.
\]

An illustration of this set is presented in Figure 3. The capital letter B used in the name of this set (as well as in the name of the set \( T_{t,d}^{B^-} \) introduced below)
point towards the fact that its contains the trains which would violate a break constraint if they were assigned to driver \(d\) along with train \(t\). This entails both trains departing before or after train \(t\).

\[ \text{Weekly break set } T_{t,d}^{\text{LB}+} \]
For each driver \(d \in D\) and train \(t \in T^d\) we require the set \(T_{t,d}^{\text{LB}+}\) to denote those trains \(t_1 \in T^d\) which cannot be assigned to driver \(d\) if train \(t\) is the last job before a long break. Formally, we define this set as:

\[
T_{t,d}^{\text{LB}+} := \{ t_1 \in T^d : t_1 > t \land s(t_1) \geq e(t) \land e(t_1) \leq e(t) + c^{\text{long}} \}
\]
\[
\cup \{ t_1 \in T^d : t_1 > t \land s(t_1) < e(t) + c^{\text{long}} \land e(t_1) \geq e(t) + c^{\text{long}} \}
\]
\[
\cup \{ t_1 \in T^d : t_1 > t \land e(t) + c^{\text{long}} + \tau a(t) o(t_1) > s(t_1) \}
\]

for all \(d \in D\) and \(t \in T^d\).

An illustrative example of such a set is presented in Figure 4.

\[ \text{Backward-looking trains blocked set } T_{t,d}^{\text{B}+} \]
In the set \(T_{t,d}^{\text{B}+}\), defined for all \(d \in D\) and \(t \in T^d\), we group together all trains \(t_1 \in T^d\) which cannot be served by the driver \(d\) if train \(t\) is selected to be the first in one of the shifts of driver \(d\). More formally, we have

\[
T_{t,d}^{\text{B}+} := \{ t_1 \in T^d : s(t_1) \leq s(t) - c^{\text{short}} \land e(t_1) \geq s(t) - c^{\text{short}} \land e(t_1) \leq s(t) \}
\]
\[
\cup \{ t_1 \in T^d : s(t_1) \geq s(t) - c^{\text{short}} \land e(t_1) \leq s(t) \}
\]
\[
\cup \{ t_1 \in T^d : t_1 < t \land s(t) - c^{\text{short}} - \tau a(t) o(t) < e(t_1) \}
\]
2 Modelling the integrated locomotive scheduling and crew assignment problem

Figure 4: Illustration of a $T_{t,d}^{L}$ set for some $d \in D$ and $t \in T^d$

for all $d \in D$ and $t \in T^d$. An illustrative example of such a set is presented in Figure 5.

Figure 5: Illustration of a $T_{t,d}^{B-}$ set for some $d \in D$ and $t \in T^d$

Long break beginning set $T_{w,d}^{week}$ For every driver $d \in D$ and each week $w \in W$, we construct a set $T_{w,d}^{week}$ to collect those trains $t \in T^d$ which belong to calculation week $w$ and which could serve as the last one before a long break of driver $d$. We need to make sure that at least one long break is scheduled in each calculation week. Hence, $T_{w,d}^{week}$ shall contain all the jobs $t \in T^d$ which end more than $c_{long}$ hours before the end of the calculation week.

This set can be formally defined as

$$T_{w,d}^{week} := \{t \in T^d : e(t) \geq s_{week}(w) \land e(t) + c_{long} \leq e_{week}(w)\}$$
2.3 The optimization model

for all \( w \in W \) and \( d \in D \).

An illustrative example of such a set is presented in Figure 6.

![Figure 6: Illustration of a \( T_{w,d}^{\text{week}} \) set for some \( d \in D \) and \( w \in W \)](image)

**Calculation week set** \( T_{w,d}^{\text{week,asgt}} \) For each calculation week \( w \in W \) and for each driver \( d \in D \), we need the set \( T_{w,d}^{\text{week,asgt}} \) to indicate the trains \( t \in T^d \) which belong to calculation week \( w \). It can be defined as follows:

\[
T_{w,d}^{\text{week,asgt}} := \{ t \in T^d : s(t) \geq s^{\text{week}}(w) \land e(t) \leq e^{\text{week}}(w) \}
\]

for all \( w \in W \) and \( d \in D \).

The letters “asgt” in the set’s name are an abbreviation of the word “assignment”. An illustrative example of such a set is presented in Figure 7.

![Figure 7: Illustration of a \( T_{w,d}^{\text{week,asgt}} \) set for some \( d \in D \) and \( w \in W \)](image)
Feasible shift beginnings and ends $T_{t,d}^{\text{shift\_beginning}}$ and $T_{t,d}^{\text{shift\_end}}$ The two sets $T_{t,d}^{\text{shift\_beginning}}$ and $T_{t,d}^{\text{shift\_end}}$ are required for all drivers $d \in D$ and all trains $t \in T^d$ in order to accumulate those trains $t_1 \in T^d$ which could have been assigned to the driver $d$ along with train $t$ as the first and last train in a shift respectively. For a formal definition, for each $d \in D$ and each train $t \in T^d$ let us first introduce the following set of potential next jobs:

$$T_{t,d}^{\text{next\_driver}} := \{ t_1 \in T^d : (s(t_1) > e(t) \land e(t_1) < s(t) + c_{\text{shift}} \land o(t_1) = a(t)) \} \cup \{ t_1 \in T^d : s(t_1) > e(t) + \tau^{o_{t_1},a(t)} \land e(t_1) < s(t) + c_{\text{shift}} \}.$$ 

Based on this set, we define the directed graph $G_d^{\text{shift}} = (T^d, A_d^{\text{shift}})$ with

$$A_d^{\text{shift}} := \{ (t_1, t_2) : t_1, t_2 \in T^d \land t_2 \in T_{t,d}^{\text{next\_driver}} \}.$$ 

Based on this graph, we formally define the two required index sets:

$$T_{t,d}^{\text{shift\_beginning}} := \{ t_1 : t_1 \in \delta^{-}(t) \land s(t_1) \geq e(t) - c_{\text{shift}} \} \cup \{ t \}$$

and

$$T_{t,d}^{\text{shift\_end}} := \{ t_1 : t_1 \in \delta^{+}(t) \land e(t_1) \leq s(t) + c_{\text{shift}} \} \cup \{ t \}$$

for all $d \in D$ and $t \in T^d$. Here, $\delta^{+}(t)$ and $\delta^{-}(t)$ denote the set of outgoing and incoming arcs of a node $t \in T^d$ respectively.

Trains in time conflict set $T_{t,d}^{\text{time}}$ Using the set $T_{t,d}^{\text{time}}$, we gather, for all drivers $d \in D$ and for each train $t \in T^d$, the trains $t_1 \in T^d$ which are in time conflict with the train $t \in T^d$. This will allow us to prohibit situations which would require a driver to serve two trains at the same time. Formally, we introduce this set as

$$T_{t,d}^{\text{time}} := \{ t_1 : t_1 \in T^d : s(t_1) \leq s(t) \land e(t_1) \geq e(t) \} \cup \{ t_1 : t_1 \in T^d : s(t_1) \geq s(t) \land s(t_1) \leq e(t) \land e(t_1) \geq e(t) \} \cup \{ t_1 : t_1 \in T^d : s(t_1) \geq s(t) \land e(t_1) \leq e(t) \} \cup \{ t_1 : t_1 \in T^d : s(t_1) \leq s(t) \land e(t_1) \leq e(t) \land e(t_1) \geq s(t) \} \cup \{ t_1 : t_1 \in T^d : e(t) + \tau^{o(t_1),a(t)} \leq s(t_1) \land e(t_1) \leq s(t) + c_{\text{shift}} \}$$

for all $d \in D$ and $t \in T^d$.

An illustrative example of such a set is presented in Figure 8.
2.3 The optimization model

Feasible next shift beginnings and ends \( T_{t,d}^\text{after break} \) and \( T_{t,d}^\text{before break} \) For all drivers \( d \in D \) and trains \( t \in T^d \), these two sets \( T_{t,d}^\text{after break} \) and \( T_{t,d}^\text{before break} \) are used to group together the trains \( t_1 \in T^d \) which can be the first jobs of the next shift after the short break following the train \( t \) and, respectively, the trains \( t_1 \in T^d \) which can be the last job of the previous shift before the short break preceding train \( t \). For a formal definition, for each \( d \in D \) let us first introduce a graph \( G^b_d = (T^d, A^b_d) \) with

\[
A^b_d := \{(t_1, t_2) : t_1, t_2 \in T^d \land o(t_1) = a(t) \land s(t_1) > e(t) + c_{\text{short}}\} \\
\cup \{(t_1, t_2) : t_1, t_2 \in T^d \land s(t_1) > e(t) + \tau^a(t_1,a(t)) + c_{\text{short}}\}.
\]

Based on this graph, we formally define

\[
T_{t,d}^\text{after break} := \delta^+(t)
\]

and

\[
T_{t,d}^\text{before break} := \delta^-(t).
\]

for all \( d \in D \) and \( t \in T^d \). The two sets \( \delta^+(t) \) and \( \delta^-(t) \) denote the outgoing and incoming arcs of a node \( t \in T^d \) respectively.

Locomotive potential next trains \( T_{t,L}^\text{next} \) For all locomotive classes \( L \in \mathcal{L} \) and for all trains \( t \in T^L \), the set \( T_{t,L}^\text{next} \) is used to gather all trains \( t_1 \in T^L \) which can be selected as the successors to locomotive of locomotive class \( L \) if it serves train \( t \). It can be formally stated as

\[
T_{t,L}^\text{next} := \{t_1 \in T^L : s(t_1) > e(t) \land o(t_1) = a(t)\}.
\]
2 Modelling the integrated locomotive scheduling and crew assignment problem

for all \( t \in T \) and \( L \in L^t \).

An illustrative example of such a set is presented in Figure 9.

![Illustration of a set for some \( L \in L \) and \( t \in T^L \)](image)

**Sunday set** \( t_{w,d}^{\text{Sunday}} \) The set \( t_{w,d}^{\text{Sunday}} \) is used to determine which trains \( t_1 \in T \) are scheduled for the period falling between Sunday, 6:00 a.m., and Monday, 6:00 a.m. (referred to as ‘Sunday’) in a calculation week \( w \) for a given driver \( d \in D \) in order to make sure that at least every fourth Sunday is off. We define this set as

\[
\begin{align*}
T_{w,d}^{\text{Sunday}} & := \{ t_1 \in T^d : s(t_1) \geq s^{\text{Sunday}}(w) \land e(t_1) \leq e^{\text{Sunday}}(w) \} \\
\cup \{ t_1 \in T^d : s(t_1) \geq s^{\text{Sunday}}(w) \land s(t_1) \leq e^{\text{Sunday}}(w) \land e(t_1) \geq e^{\text{Sunday}}(w) \} \\
\cup \{ t_1 \in T^d : s(t_1) \leq s^{\text{Sunday}}(w) \land e(t_1) \geq s^{\text{Sunday}}(w) \land e(t_1) \leq e^{\text{Sunday}}(w) \}
\end{align*}
\]

for all \( d \in D \) and \( w \in W \).

2.3.2 Multi-commodity flow part of the model – the locomotive assignment

We consider the set \( L \) of locomotive classes as commodities which need to be “routed” through a directed graph \( G = (\mathcal{V}, \mathcal{A}) \), defined via

\[
\mathcal{V} := T \cup \{\Sigma, \Theta\}
\]

and

\[
\mathcal{A} := \bigcup_{L \in L} \mathcal{A}^L,
\]
2.3 The optimization model

with

\[ \mathcal{A}^L := \{(t_1, t_2) : t_1 \in T^L \land t_2 \in T_{t_1}^{\text{next}} \}, \]

for all \( L \in \mathcal{L} \), with \( \Sigma, \Theta \in V \) being the source and sink nodes of \( \mathcal{G} \) respectively. They are artificial nodes and do not represent any actual trains. In our model, we will represent each locomotive class as a separate commodity. The choice of an arc \( a = (t_1, t_2) \in \mathcal{A} \) for a locomotive class \( L \) means that a locomotive of locomotive class \( L \) first serves train \( t_1 \) and directly afterwards train \( t_2 \). As an abbreviation for the outgoing and the incoming arcs of a node \( t \in \mathcal{V} \) in the graph \( \mathcal{G} \), we use \( \delta^+(t) \) and \( \delta^-(t) \) respectively.

Per definition of the set \( \mathcal{A} \), a locomotive class may be chosen for a given arc if it is compatible with both the first and the second corresponding train. Further, each arc shall have unit capacity, i.e. it can be chosen by at most one locomotive class.

2.3.3 Definitions of variables

In our model, we need to make sure that each train \( t \in T \) is staffed with exactly one suitable driver \( d \in D^t \) and one locomotive of suitable class \( L \in \mathcal{L}^t \) if it shall run. The decision to assign driver \( d \in D^t \) to a train \( t \in T \) is modelled by the binary variables \( x^t_d \in \{0, 1\} \). The binary variable \( f^t_{L_1, L_2} \in \{0, 1\} \) models the decision to have locomotive class \( L \in \mathcal{L}^t \cap \mathcal{L}^t \) carry the two trains \( t_1, t_2 \in T \) in direct succession.

To comply with the working time requirements, we need to consider the first and the last job in the shift of a driver explicitly. The binary variable \( y^t_d \) asks if train \( t \in T^d \) is the first job of a driver \( d \in D \) on the respective shift. Similarly, the binary variable \( v^t_d \) models the choice of the last job in a shift before a short break. The binary variable \( z^t_d \) asks if the arrival time of train \( t \) indicates the beginning of the long break for driver \( d \). We also need to know whether a driver \( d \in D \) works on the Sunday of week \( w \). This is determined by the binary variable \( h^w_d \).

Finally, for modelling purposes we also need to know which trains \( t \in T^d \) are the first and the last job for driver \( d \in D \) in the planning period. We do that with the help of the binary variables \( \alpha^t_d \) and \( \omega^t_d \), respectively. While several of the choices are similar in structure, we believe that this level of detail in the presentation is important to precisely convey the information about the individual working time constraints. All in all, the variables are summarized in Table 3.
### 2.3.4 Model formulation

We can now state a full formulation of the integrated locomotive scheduling and driver assignment problem we consider in this work as a binary optimization problem:

\[
\begin{align*}
\text{max} & \quad \sum_{t \in T} \sum_{d \in D^t} x_d^t \\
\text{s.t.} & \quad x_{d_{t_1}} \leq \sum_{L \in L^t \cap L^d} f_{L_{t_1}} \quad \forall t_1 \in T, \quad \forall d \in D_{t_1} \\
& \quad \sum_{d \in D^t} f_{L_{t_1}} \leq \sum_{d \in D^t \cap D_{t_1}} x_d^t \quad \forall t_1 \in T, \quad \forall L \in L^t_{t_1} \\
& \quad \sum_{d \in D^t} x_d^t \leq 1 \quad \forall t \in T \\
& \quad \sum_{t \in T} \alpha_{d} \leq 1 \quad \forall d \in D \\
& \quad \sum_{t \in T^d} \omega_{d} \leq 1 \quad \forall d \in D \\
& \quad x_{d_{t_1}} + x_{d_{t_2}} \leq 1 \quad \forall t \in D^d, \quad \forall t_1 \in T_{t,d}^\text{time}
\end{align*}
\]
2.3 The optimization model

\[ y_d^t + x_d^t \leq 1 \quad \forall d \in D \quad (I8) \]
\[ v_d^t + x_d^t \leq 1 \quad \forall t \in T_d^d \quad (I9) \]
\[ z_d^t + x_d^t \leq 1 \quad \forall t \in T_d^d \quad (I10) \]
\[ v_d^t \leq \omega_d^t \quad \forall d \in D \quad (I11) \]
\[ y_d^t \leq \alpha_d^t + \sum_{t_1 \in T_{t,d}^{\text{after\_break}}} v_d^{t_1} \quad \forall d \in D \quad (I12) \]
\[ x_d^t \leq \sum_{t_1 \in T_{t,d}^{\text{shift\_beginning}}} y_d^{t_1} \quad \forall d \in D \quad (I13) \]
\[ x_d^t \leq \sum_{t_1 \in T_{t,d}^{\text{shift\_end}}} v_d^{t_1} \quad \forall d \in D \quad (I14) \]
\[ x_d^t \leq \sum_{t_1 \in T_{w,d}^{\text{week\_asgt}}} z_d^{t_1} \quad \forall w \in W \quad (I15) \]
\[ \alpha_d^t \leq \sum_{t_2 \in T_{d,t_1}^{d\_t_2 \geq t}} \omega_d^{t_2} \quad \forall d \in D \quad (I16) \]
\[ x_d^t \leq \bar{h}_d \quad \forall w \in W \quad (I17) \]
\[ \sum_{w \in W} h_d^w \leq c_{\text{sunday}} \quad \forall d \in D \quad (I18) \]
\[ y_d^t \leq x_d^t \quad \forall d \in D \quad (I19) \]
Modelling the integrated locomotive scheduling and crew assignment problem

\[ v_d^t \leq x_d^t \]
\[ \forall d \in D \]
\[ \forall t \in T_d \ (l_{20}) \]

\[ \alpha_d^t \leq x_d^t \]
\[ \forall d \in D \]
\[ \forall t \in T_d \ (l_{21}) \]

\[ \omega_d^t \leq x_d^t \]
\[ \forall d \in D \]
\[ \forall t \in T_d \ (l_{22}) \]

\[ \sum_{t_0 \in T_0} f_{L}^{t_0,t_1} - \sum_{t_1 \in T_1} f_{L}^{t_1,t_2} = 0 \]
\[ \forall t_1 \in T \]
\[ \forall L \in L_1 \ (l_{23}) \]

\[ \sum_{t_1 \in T_1} f_{L}^{t_1,t_2} \leq 1 \]
\[ \forall t_2 \in T \ (l_{24}) \]

\[ \sum_{t \in T} f_{L}^{\Sigma_t} \leq |L| \]
\[ \forall L \in L \ (l_{25}) \]

\[ \sum_{t \in T^L} f_{L}^{\Sigma_t} - \sum_{t \in T^L} f_{L}^{t,\theta} = 0 \]
\[ \forall L \in L \ (l_{26}) \]

\[ x_d^t \in \{0, 1\} \]
\[ \forall t \in T \]
\[ \forall d \in D_t \ (l_{27}) \]

\[ y_d^t \in \{0, 1\} \]
\[ \forall t \in T \]
\[ \forall d \in D_t \ (l_{28}) \]

\[ v_d^t \in \{0, 1\} \]
\[ \forall t \in T \]
\[ \forall d \in D_t \ (l_{29}) \]

\[ z_d^t \in \{0, 1\} \]
\[ \forall t \in T \]
\[ \forall d \in D_t \ (l_{30}) \]

\[ \alpha_d^t \in \{0, 1\} \]
\[ \forall t \in T \]
\[ \forall d \in D_t \ (l_{31}) \]

\[ \omega_d^t \in \{0, 1\} \]
\[ \forall t \in T \]
\[ \forall d \in D_t \ (l_{32}) \]

\[ h_d^w \in \{0, 1\} \]
\[ \forall d \in D \]
\[ \forall w \in W \ (l_{33}) \]

\[ f_{L}^{t_1,t_2} \in \{0, 1\} \]
\[ \forall (t_1, t_2) \in S^L \]
2.3 The optimization model

With objective function (I1), we maximize the number of trains running. Constraints (I2) and (I3) make sure that either both a locomotive and a driver are assigned to a train or none of them; they also take care that driver and locomotive are mutually compatible. With (I4), we ensure that at most one driver is assigned to a train. Constraints (I5) and (I6) enforce that each driver has a unique first and last job respectively. Using constraint (I7), we ensure that no two trains which run simultaneously are assigned to the same driver. With (I8) and (I9), we model that the minimal length of a short break shall not be violated. Additionally, with (I10) we require the integrity of the long break. Using (I11), we make sure that each last job \( t \) in a shift is succeeded by a first job of the next shift, or that the job \( t \) is the last one assigned to driver \( d \) in the plan. Similarly, with (I12) we ensure that each first job \( t \) in a shift is preceded by a last job of the previous shift, or that the job \( t \) is the first one assigned to driver \( d \) in the plan. Constraints (I13), (I14) incorporate the integrity of drivers’ shifts and model the maximal length of a shift. Via (I15), we make sure that at least one long break per week is assigned to each driver in each week in the planning period. With the help of (I16), we achieve that the last job of a driver in the plan is either the same or a later one as the first job in the plan. Using constraints (I17) and (I18), we guarantee that a driver works on at most \( e^{\text{sunday}} \)-many Sundays in a given planning period. Constraints (I19), (I20), (I21) and (I22) tie each shift variable to the corresponding staffing variable.

For the locomotive part of the model, (I23) ensures that a locomotive that serves a train \( t \) arrives at its origin station and in the due time, and similarly it later departs from the arrival station of train \( t \). With constraint (I24), we make sure that at most one locomotive class \( L \in \mathcal{L}^2 \) is selected to serve a train \( t_2 \in T \). Constraint (I25) ensures that the number of locomotives of a given class used in the plan does not exceed the cardinality of the locomotive class. Via (I26), we assure the integrity of the locomotive schedule. Finally, constraints (I27) – (I34) require all decision variables to be binary.

After solving model (I) its trivial to concretize the assignment of a locomotive class to arcs and nodes in the flow network \( \mathcal{G} \) to actual, individual locomotives. We obtain a solution in the form of directed paths through the network, connecting the source node \( \Sigma \) to the sink node \( \Theta \) (as defined in Subsection 2.3.4) for all the locomotive classes \( L \in \mathcal{L} \). To make the locomotive assignment concrete, we only need to assign each of the paths to an actual locomotive in the class \( L \in \mathcal{L} \). More details about an algorithm which assigns individual paths to “actual” locomotives is introduced in Appendix A.
2 Modelling the integrated locomotive scheduling and crew assignment problem

2.4 Model preprocessing – clique tightening

Model (I) is very complex, mostly due to its size. We will now consider a way to reduce the number of constraints needed, which usually greatly benefits the computations times. Namely, we will use clique tightening to improve the formulation of the conflict constraints.

Consider the conflict constraints (I7), (I8), (I9) and (I10). They all refer to a situation in which at most one of the considered trains can be assigned to the same driver. With a growing number of trains, the number of such constrains will grow cubically. For all drivers \( d \in D \), we construct a graph \( G_{\text{time}}^d = (V_{\text{time}}^d, E_{\text{time}}^d) \), where \( V_{\text{time}}^d := T^d \) and \( E_{\text{time}}^d := \{(t_1, t_2) : t_1 \in T^d \land t_2 \in T_{\text{time}}^d\} \). For the exemplary case depicted in Figure 10a, the resulting conflict graph \( G_{\text{time}}^d \) is presented in Figure 10b.

![Train schedule and conflict graph](image)

Figure 10: Example of a time conflict graph \( G_{\text{time}}^d \) for some driver \( d \in D \)

We notice that a grouping of the nodes in this graph to cliques (defined as complete subgraphs of a given graph) can be employed to come up with stronger constraints. Obviously, each edge of the graph itself induces a 2-clique in the graph. We are now interested in finding fewer, but larger cliques in order to express an equivalent, smaller set of constraints. In order to do so, we need to make sure that each edge is covered by at least one of the cliques. This can be achieved by searching for a (minimal) clique edge cover of the graph \( G_{\text{time}}^d \).

**Proposition 2.4.1.** Let \( C_{\text{time}}^d \) be a clique edge cover of the graph \( G_{\text{time}}^d \). Then constraint (I7) is equivalent to the following constraint:

\[
\sum_{\ell \in C} x_{\ell}^d \leq 1 \quad \forall d \in D, \forall C \in C_{\text{time}}^d. \tag{I7a}
\]
2.4 Model preprocessing – clique tightening

In the example from Figure 10b, we can use the two cliques \( C_1 := \{t_1, t_3, t_4, t_5\} \) and \( C_2 := \{t_1, t_2, t_3\} \) to construct two conflict constraints which are equivalent to the eight constraints induced by the individual edges. This reformulation also results in a tighter description of the underlying convex hull of feasible solutions (see e.g. [Brito and Santos, 2021]).

For the remaining constraints (I8), (I9) and (I10), we can further extend this concept by a slight redefinition of the conflict graph. We outline the derivation of the backward-break conflict constraints, represented by constraint (I8).

For all drivers \( d \in D \), we construct a graph \( G^d_{\text{back\_break}} = (V^d_{\text{back\_break}}, E^d_{\text{back\_break}}) \).

The vertex set is defined as

\[ V^d_{\text{back\_break}} := V^d_X \cup V^d_Y, \quad \text{with} \quad V^d_X := T^d \times \{1\}, \quad V^d_Y := T^d \times \{2\}, \]

i.e. \( V^d_{\text{back\_break}} \) contains two copies of each node in \( T^d \). For ease of notation, we will write \( t_X \in V^d_X \) and \( t_Y \in V^d_Y \) instead of \( (t, 1) \in V^d_X \) and \( (t, 2) \in V^d_Y \) respectively, with the understanding that \( t \in T^d \) holds in each case. The vertices in \( V^d_X \) correspond to the respective \( x \)-variables, while those in \( V^d_Y \) correspond to the respective \( y \)-variables. The edge set is then defined as

\[
E^d_{\text{back\_break}} := \{ \{t_X, t_Y\} : t_X \in V^d_X \land t_Y \in V^d_Y \land t_X \in T^d_{t_Y} \} \\
\cup \{ \{t^1_Y, t^2_Y\} : t^1_Y, t^2_Y \in V^d_Y \land t^1_Y \in T^d_{t^2_Y} \}
\]

There are two kinds of edges in \( E^d_{\text{back\_break}} \). The first of them corresponds to conflicts between \( x \)-variables and \( y \)-variables, and the second represents conflicts only between \( y \)-variables.

For the illustrative case of a driver \( d \in D \) with \( T^d := \{t_1, t_2, t_3, t_4\} \), \( T^d_{t_3} := \{t_2, t_3\} \), \( T^d_{t_5} := \{t_1, t_2\} \), an exemplary graph \( G^d_{\text{back\_break}} \) is presented in Figure 11. In particular, if train \( t_4 \) is selected as the first one in a shift of driver \( d \), this driver may not serve trains \( t_2 \) and \( t_3 \) as well. Similarly, trains \( t_1 \) and \( t_2 \) may not be assigned to driver \( d \) if train \( t_3 \) was chosen as the first one in the shift.

Since for a given driver \( d \in D \) no nodes in \( V^d_X \) are connected to each other, each clique in \( G^d_{\text{back\_break}} \) contains at most one node from the vertex set \( V^d_X \). Similarly, each node in \( V^d_X \) will appear in at most one maximal clique. For the construction of constraints, we are interested in the cliques which contain at least one node from \( V^d_Y \), i.e. one \( x \)-variable. Such cliques will be referred to as constraint-generating cliques. Note that we can safely omit cliques containing only nodes from \( V^d_Y \), since – thanks to constraint (I9) – the resulting conflict constraints will be dominated by the constraints generated from cliques.
including nodes from both \( \mathcal{V}_X^d \) and \( \mathcal{V}_Y^d \). One obvious extension of the approach would be to include all the edges induced by time conflicts (from \( G_{\text{time}}^d \)) also between the nodes in \( \mathcal{V}_X^d \). To save computation time, we decided to include only those conflicts which are not already represented in \( G_{\text{time}}^d \) – this reduces the number of edges, which led to a smaller number of cliques to be considered in our experiments in Section 4.

We now define \( G_{\text{back\_break}}^d \) to be the set of all maximal cliques in \( G_{\text{back\_break}}^d \) for all drivers \( d \in D \). Based on this set, we define a set of constraint-generating cliques for each train \( t \in T^d \) as the subset of maximal cliques containing the node \( t_X \in \mathcal{V}_X^d \):

\[
\mathcal{C}_{\text{back\_break}}^{t,d} := \{ C \in \mathcal{G}_{\text{back\_break}}^d : t \in C \land t_X \in \mathcal{V}_X^d \}.
\]

A very similar derivation for constraints (I9) and (I10) leads to corresponding graphs \( G_{\text{forward\_break}}^d \) and \( G_{\text{long\_break}}^d \) for all drivers \( d \in D \) and respective sets of constraint-generating cliques \( \mathcal{C}_{\text{forward\_break}}^{t,d} \) and \( \mathcal{C}_{\text{long\_break}}^{t,d} \). With a slight abuse of notation, we also introduce the respective node sets \( \mathcal{V}_Y \) and \( \mathcal{V}_Z \) as well as the respective node types \( t_Y \) and \( t_Z \) for the trains \( t \in T^d \). They allow us to simplify model (I) further.

**Proposition 2.4.2.** The constraints (I8)–(I10) are equivalent to the following set of constraints:

\[
\sum_{t_Y \in C} y_d^{t_Y} + \sum_{t_X \in C} x_d^{t_X} \leq 1 \quad \forall d \in D, \forall t \in T^d, \forall C \in \mathcal{C}_{\text{back\_break}}^{t,d} \quad (I8a)
\]

\[
\sum_{t_Y \in C} v_d^{t_Y} + \sum_{t_X \in C} x_d^{t_X} \leq 1 \quad \forall d \in D, \forall t \in T^d, \forall C \in \mathcal{C}_{\text{forward\_break}}^{t,d} \quad (I9a)
\]
2.4 Model preprocessing – clique tightening

\[
\sum_{t_x \in C} z_{d}^{t_x} + \sum_{t_x \in C} x_{d}^{t_x} \leq 1 \quad \forall d \in D, \quad \forall t \in T^d, \quad \forall C \in C_{long\_break}^{t,d}, \quad (110a)
\]

To save computation time, in our Case Study we will determine the required clique edge covers in a heuristic fashion. More details will be given in Chapter 4.
3 A solution algorithm for the integrated vehicle scheduling and crew assignment problem

This chapter is devoted to developing a solution algorithm based on decomposition and cutting planes for the integrated locomotive scheduling and driver assignment problem introduced in the previous chapter. Figure 12 gives an outline of the approach as a flowchart; its ingredients are detailed in the following.

First, in Section 3.1 the general decomposition scheme will be presented. A number of preliminary computational experiments had shown that the locomotive assignment part of the problem is far easier to solve than the driver part. This led to the idea to design the solution method in such a way that it first computes a best possible feasible locomotive assignment, relaxing all driver-related constraints.

Afterwards, in Section 3.2, we will derive cutting planes in the locomotive-flow variables which are valid for the integrated problem. These cutting planes encode common reasons for the infeasibility of the driver part (as encountered for the real-world instances presented in Chapter 4), expressed in terms of the locomotive assignment variables. In our solution algorithm, we first solve the locomotive part as a master problem to obtain a candidate locomotive assignment. Then we search for violated cutting planes to cut the current locomotive assignment off from the locomotive master problem and solve it again. This procedure is iterated until no further cutting planes are found.

The next step is to fix the resulting locomotive assignment in the integrated problem. This gives rise to the driver subproblem, whose task it is to find a feasible driver assignment that is compatible to the fixed locomotive assignment. If this is possible, we have solved the problem to global optimality. Sometimes, the driver part is infeasible but we cannot generate one of our problem-specific cutting planes to ensure feasibility of the driver subproblem. We then generate a combinatorial Benders cut instead that precisely cuts off the current locomotive assignment from the locomotive master problem and reiterate. This way, the algorithm will eventually converge to a global optimal solution to the joint locomotive and driver problem.

Finally, we will present some algorithmic enhancements to reduce computation times. These are a preprocessing scheme (Section 3.3) as well as a
heuristic (Section 3.4) whose aim it is to facilitate the solution of the driver subproblem, which is still hard to solve, even though the locomotive assignment is already fixed there.

---

**Figure 12:** Flowchart of the exact version of the solution algorithm to the integrated locomotive scheduling and driver assignment problem introduced in this work

---

### 3.1 Decomposition into locomotive master problem and driver subproblem

In this section, we will present the decomposition approach which underlines the solution algorithm we develop in this chapter. In particular, we establish the locomotive master problem and the driver subproblem for our decomposition approach.

As already discussed in Subsection 2.3.4, we can subdivide the constraints of the joint model (I) into three groups: (i) compatibility constraints between drivers and locomotives (I2) and (I3), (ii) the driver-related constraints
3.2 Classes of valid cutting planes

(I4)–(I6), (I7a)–(I10a), (I11)–(I22) and (I27)–(I33), and (iii) the locomotive-related constraints (I23)–(I26) and (I34). When relaxing the compatibility constraints, model (I) decomposes into two independent subproblems. We call the model given by the original objective function (I1) and constraints of group (ii) discussed above the driver subproblem.

In order to define the corresponding locomotive master problem, we first note that although the locomotive-related constraints are exactly those of a pure binary multi-commodity-flow problem, we need a slightly different kind of objective function. Rather than maximizing the flow through the network (which would amount to maximizing the number of used locomotives), we have to make sure that as many nodes/trains in the network are “visited” by some commodity/locomotive of some locomotive class. To this end, we introduce a new binary decision variable which checks whether a given train was served by a compatible locomotive or not:

\[ \lambda_t = \begin{cases} 1, & \text{if train } t \in T \text{ is served by a compatible locomotive } L \in \mathcal{L}^t \\ 0, & \text{otherwise.} \end{cases} \]

Using the \( \lambda \)-variables, we can state the objective function of the locomotive master problem maximizing the number of served trains:

\[ \max \sum_{t \in T} \lambda_t. \]  

(L1)

We also need an additional constraint that couples the newly introduced \( \lambda \)-variables with the existing \( f \)-variables:

\[ \lambda_{t_1} \leq \sum_{L \in \mathcal{L}} \sum_{(t_1, t_2) \in \mathcal{L}^L} f_{t_1, t_2}^{t_1, t_2} \quad (\forall t_1 \in T). \] 

(L2)

Finally, we need to make sure that the \( \lambda \)-variables are binary:

\[ \lambda_t \in \{0, 1\} \quad (\forall t \in T). \] 

(L3)

Altogether, the locomotive master problem has the objective function (L1) as well as the constraints (I23) – (I26), (I34), (L2) and (L3).

3.2 Classes of valid cutting planes

In this section, we will present four classes of valid inequalities, which are then implemented as cutting planes. They encode common reasons for the infeasibility of the driver part (as encountered for the real-world instances
presented in Chapter 4), expressed in terms of the locomotive assignment variables.

The driver subproblem contains ten constraint types which are potential sources of infeasibility in model (I) when fixing the \( f \)-variables corresponding to a given solution to the locomotive master problem. These are all the conflict constraints (I5), (I6), (I7a), (I8a), (I9a) and (Ioa), the compatibility constraints (I11), (I12), long-break enforcement constraint (I15) as well as constraint (I18), which pertains to ensuring the Sunday breaks. Based on our computational experience, we determined that only a subset of these are violated for the real-world instances supplied by our industrial partner. For these, we explicitly show how an infeasibility can arise. Further, we present corresponding valid inequality classes for the locomotive master problem to ensure that its solution will not result in an infeasible driver subproblem. For the remaining, potentially violated driver constraints, the valid locomotive inequalities can be derived in a similar fashion.

We start by noticing that all constraints of the driver subproblem not mentioned above cannot be a cause of infeasibility in model (I) when fixing a solution to the locomotive part of the problem.

Lemma 3.2.1. Let \((\bar{f}, \bar{\lambda})\) be a feasible solution to the locomotive master problem given by constraints (L1), (I23)–(I26), (L2) and (L3). Then we can find values \(\bar{x}, \bar{y}, \bar{v}, \bar{\alpha}, \bar{\omega}, \bar{h}\) such that \((\bar{f}, \bar{x}, \bar{y}, \bar{v}, \bar{\alpha}, \bar{\omega}, \bar{h})\) is a solution to the constraint system given by the driver-related constraints (I2), (I3), (I13), (I14), (I16), (I17), (I19), (I20), (I21) and (I22).

Proof: Feasibility of the second driver-locomotive compatibility constraint (I3) can be ensured for any feasible locomotive assignment, because it is sufficient to set at most one \(x\)-variable on its right-hand side to one in order to fulfil it. This means we choose any compatible driver for the locomotive-train assignment. For constraints (I13), (I14), (I16), (I17) we can choose the \(y\)-, \(v\)-, \(\alpha\)-, \(\omega\)- and \(h\)-variable corresponding to the chosen \(x\)-variables and set them to one. By doing so, we do not have to set any other \(x\)-variables occuring on the right-hand sides of constraints (I19), (I20), (I21) and (I22) to one. Therefore, the first driver-locomotive compatibility constraint (I2) cannot be violated either.

As shown in Observation 3.2.1, no solution \((\bar{f}, \bar{\lambda})\) to the locomotive master problem may cause an infeasibility in the driver subproblem with regard to constraints (I2), (I3), (I13), (I14), (I16), (I17), (I19), (I20), (I21) and (I22).
Hence, we do not need to focus on these constraints as we look for infeasibilities in the driver subproblem which could result from a fixed solution to the locomotive subproblem \((\bar{f}, \bar{\lambda})\).

Let us now focus on the ten potentially problematic constraints mentioned in the beginning of this subsection. As they might be violated by a given locomotive assignment, we need to consider them already at the stage of solving the locomotive master problem. We will now explain how such an infeasibility can arise and introduce additional, valid constraints for the locomotive master problem to ensure the feasibility of the locomotive assignment when computing a compatible driver assignment. As we will see, many of the infeasibilities targeted by these cuts are caused by an insufficient availability of drivers to certain train-locomotive combinations. Such a situation may occur if the total number of drivers is very small, or if some train-locomotive combinations can only be served by a small number of drivers. This could be the case when, for example, the railway carrier has only begun to serve a new client, whose location is by far different to the locations of the existing customers. This requires the locomotive drivers to obtain the appropriate route knowledges and – at least in the initial phase – only a small number of drivers may be available for assignment to trains for that client.

In the case study in Chapter 4, we will see that these cutting planes are sufficient to ensure the feasibility of the driver assignment for the real-world instances provided by our industry partner, which means that we obtain globally optimal solutions via our method. As mentioned above, it might occur for general instances that a given locomotive solution cannot be cut off by the cutting planes we derive here. In this case, we can ensure the feasibility and global optimality of the decomposition approach by cutting off the current locomotive assignment from the locomotive master problem via a combinatorial Benders cut, as described in [Codato and Fischetti, 2006]. Altogether, this allows us to cut off any locomotive assignment which cannot be completed to a full, feasible solution to model (I). Therefore, our algorithm will in the end converge to a feasible, globally optimal solution to the integrated locomotive scheduling and driver assignment problem. In the following, we will introduce four classes of valid inequalities for the model (I).

3.2.1 Valid cutting planes derived from the time conflict constraints

Constraint (I7a) ensures that no driver is staffed to drive two trains which run simultaneously. We need to make sure that the computed optimum of the locomotive subproblem also respects these constraints. Recall that in Subsection 2.4 we introduced the graph \(G_{\text{time}}^d\) to represent the time conflicts.
between trains for a given driver \( d \in D \). Let us now generalize this graph to cover time conflicts between all of the trains. For this purpose, we introduce the graph \( G_{\text{time}} = (T, E_{\text{time}}) \), where

\[
E_{\text{time}} := \{(t_1, t_2) : t_1 \in T \land \exists d \in D : t_2 \in T_{t_1, d}^{\text{time}}\}.
\]

Now define \( \mathcal{C}_{\text{time}} \) as the set of all maximal cliques in \( G_{\text{time}} \). As the set of drivers available for a train depends both on the train itself and on the locomotive assigned to carry it, we need to consider all possible assignments of locomotives for all trains present in the cliques in \( \mathcal{C}_{\text{time}} \). These assignments are represented by the set

\[
\mathcal{C}_{\text{time}} \text{ assignments} := \{(t, L) : t \in C \land L \in L^t \} : C \in \mathcal{C}_{\text{time}},
\]

which allows us to derive valid cutting planes.

**Theorem 3.2.2.** The following inequalities are valid for the model (I):

\[
\sum_{(t_1, L) \in C} \sum_{t_2 : (t_1, t_2) \in \mathcal{A}^L} f_{L, t_1, t_2} \leq \left| \bigcup_{(t_1, L) \in C} D_{t_1} \cap D^L \right| \forall C \in \mathcal{C}_{\text{time}} \text{ assignments}. \tag{V1}
\]

**Proof.** Let \((\tilde{f}, \tilde{\lambda}, \tilde{x}, \tilde{y}, \tilde{v}, \tilde{z}, \tilde{a}, \tilde{\omega})\) be a feasible solution to model (I). Suppose this solution violates constraint (V1). Then there exists a clique \( C \in \mathcal{C}_{\text{time}} \text{ assignments} \) such that

\[
\sum_{(t_1, L) \in C} \sum_{t_2 : (t_1, t_2) \in \mathcal{A}^L} f_{L, t_1, t_2} > \left| \bigcup_{(t_1, L) \in C} D_{t_1} \cap D^L \right| \tag{1}
\]

holds. Define \( C_{\text{trains}} := \{t_1 : (t_1, L) \in C\} \). By definition of \( \mathcal{C}_{\text{time}} \text{ assignments} \), all of the trains in \( C_{\text{trains}} \) are in time conflict. Hence, to ensure that all these trains can be served, we need at least \(|C_{\text{trains}}|\)-many drivers compatible with the trains in \( C_{\text{trains}} \) and the respective locomotive classes. However, inequality (1) implies that fewer compatible drivers are available. Therefore, the considered solution is infeasible, which is a contradiction. \( \Box \)

To better illustrate the context in which the cut is being used, we introduce the following example.

**Example 3.2.3.** Consider a subset \( S := \{t_1, t_2, t_3\} \) of a larger set of trains. Assume that the trains in \( S \) are mutually in time conflict. Then the set \( S \) constitutes a clique in \( \mathcal{C}_{\text{time}} \text{ assignments} \). Let us assume that this clique is maximal.
3.2 Classes of valid cutting planes

An example of such a case is depicted in the Further, we assume the following locomotive compatibilities: \( \mathcal{L}^{t_1} = \{L_1, L_2\} \), \( \mathcal{L}^{t_2} := \{L_1, L_2, L_3\} \), \( \mathcal{L}^{t_3} := \{L_2, L_3\} \). With regard to drivers, let \( D^{t_1} = D^{t_2} = D^{t_3} := \{d_1, d_2, d_3, d_4\} \), and \( D^{L_1} := \{d_1, d_2, d_3\} \), \( D^{L_2} := \{d_1, d_2\} \), \( D^{L_3} := \{d_2, d_3, d_4\} \).

Now, consider the following assignment of locomotives to trains: \( \{(t_1, L_2), (t_2, L_2), (t_3, L_2)\} \). Such a locomotive assignment will cause the need to assign two drivers \( d_1 \) and \( d_2 \) to three trains in time conflict, which is obviously infeasible (and violates constraint (I7a)). Such an infeasibility may be cut off by a cut introduced in this section. For the example considered, we first notice that:

\[
|(D^{t_1} \cap D^{L_2}) \cup (D^{t_2} \cap D^{L_2}) \cup (D^{t_3} \cap D^{L_2})| = 2,
\]

and hence the inequality would look as follows:

\[
\sum_{t_a : (t_1, t_a) \in S} f_{L_1}^{t_1, t_a} + \sum_{t_b : (t_2, t_b) \in S} f_{L_2}^{t_2, t_b} + \sum_{t_c : (t_3, t_c) \in S} f_{L_2}^{t_3, t_c} \leq 2.
\]

This way, we ensure that at most two of the trains in \( S \) are assigned locomotives of class \( L_2 \), thereby cutting off the infeasible solution.

3.2.2 Valid cutting planes derived from the break conflict constraints

Consider constraints (I8a), (I9a) and (I10a). As they are relatively similar in structure, we will only treat constraint (I9a) exemplarily. Here we deal with situations in which only one driver \( d \in D \) is available to serve a certain set of train-locomotive combinations, whereas – due to working time regulations – the driver may be assigned to at most one of them. If the solution to the locomotive subproblem enforced an assignment of driver \( d \) to more than one of these trains, some of the working time constraints would be violated, and thus the locomotive assignment would be infeasible. Thanks to the cutting planes developed in this section, such locomotive assignments can be cut off.

Recall that for each driver \( d \in D \) and for each train \( t \in T^d \), all the constraint-defining cliques \( C^{t,d}_{\text{forward}_\text{break}} \in \mathcal{C}^{d}_{\text{forward}_\text{break}} \) contain exactly one node from the node set \( \mathcal{V}^d_X \) and at least one node from node set \( \mathcal{V}^d_Y \). For each driver \( d \in D \), for each train \( t \in T^d \) and for each conflict set \( C^{t,d}_{\text{forward}_\text{break}} \in \mathcal{C}^{d}_{\text{forward}_\text{break}} \), we now define a set \( S^{t,d}_X \) of train-locomotive assignments with \( S^{t,d}_X := S^{t,d}_X \cup S^{t,d}_Y \), where

\[
S^{t,d}_X := \{(t, L) : L \in \mathcal{L}^t \land D^t \cap D^L = \{d\}\}
\]
A solution algorithm for the integrated vehicle scheduling and crew assignment

and

\[
S_{t,d}^v := \{(t_v, L) : t_v \in c_{\text{forward\_break}}^t \land t_v \in \mathcal{V}_v^d \land L \in \mathcal{L}_v^t \land D_v^t \cap D_L^t = \{d\} \land T_{t_v,d}^{\text{shift\_end}} = \{t_v\}\}.
\]

Again, we derive a set of valid cutting planes from this construction.

**Theorem 3.2.4.** The following inequalities are valid for the model (I):

\[
\sum_{(t,L) \in S_{t,d}^v} \sum_{t_1, t_1} f_{L,t_1}^{t,t_1} \leq 1 \quad \forall d \in D, \forall t \in T_d^d. \quad (V2)
\]

**Proof.** Let \((\tilde{t}, \tilde{L}, \tilde{x}, \tilde{y}, \tilde{z}, \tilde{\alpha}, \tilde{\omega})\) be a feasible solution to model (I) and suppose this solution violates constraint (V2). Then there exists a set \(S_{t,d}^v\) for some \(d \in D\) and \(t \in T_d^d\) for which we have:

\[
\sum_{(t,L) \in S_{t,d}^v} \sum_{t_1, t_1} f_{L,t_1}^{t,t_1} > 1.
\]

By definition of \(S_{t,d}^v\), at most one of the \(f\)-variables pertaining to the pairs \((t, L) \in S_{t,d}^v\) may be selected, as otherwise constraint (I9a) would be violated. However, it follows from inequality (2) that more than one \(f\)-variable is set to one, which is a contradiction. Therefore, the solution is infeasible and we have a contradiction. \(\square\)

The next example illustrates the cut.

**Example 3.2.5.** Consider a subset \(\{t_1, t_2\}\) of a larger set of trains \(T\). Moreover, let \(t_2 \in c_{\text{forward\_break}}^t\) for all \(d \in D\). Additionally, assume \(T_{t_2,d}^{\text{shift\_end}} = \{t_2\}\) for all \(d \in D\). Further, we assume the following locomotive compatibilities: \(\mathcal{L}_{t_1} = \{L_1, L_2\}, \mathcal{L}_{t_2} = \{L_1, L_2, L_3\}\). With regard to drivers, let \(D_{t_1} = D_{t_2} = \{d_1, d_2, d_3, d_4\}\), and \(D_{t_1} = \{d_1\}, D_{t_2} = \{d_2, d_3, d_4\}\).

Now, consider the following assignment of locomotives to trains: \(\{(t_1, L_2), (t_2, L_2)\}\). Such a locomotive assignment will cause the need to assign the same driver \(d_1\) to the two trains \(t_1\) and \(t_2\). Since \(t_2 \in c_{\text{forward\_break}}^t\) for all \(d \in D\) and \(T_{t_2,d}^{\text{shift\_end}} = \{t_2\}\) for all \(d \in D\), we will see an infeasibility with regard to constraint (I9a) – we may not assign the same driver to \(t_1\) and \(t_2\).
3.2 Classes of valid cutting planes

To get rid of such an infeasibility, we use the cut introduced in this section. For the example considered, it would look as follows:

\[
\sum_{t_a : (t_1, t_a) \in \mathcal{O}^2} f_{L_2}^{t_1, t_a} + \sum_{t_b : (t_2, t_b) \in \mathcal{O}^2} f_{L_2}^{t_2, t_b} \leq 1.
\]

For constraints (I8a) and (I10a), the respective sets of valid constraints each can be constructed analogously.

3.2.3 Valid cutting planes derived from the compatibility constraints

Let us now turn to constraints (I11) and (I12). These two are again similar in structure, so that we only treat constraint (I11) explicitly here. As discussed in Section 2.2, the \(\omega\)-variables only exist for trains which end in the home region of the driver. This means that for a driver \(d \in D\) and for a train \(t \in T_d\) which does not end in the driver’s home region, we need to ensure that at least one train \(t_1 \in T_{t,d}^{\text{after break}}\) may be performed by driver \(d\). This requires us to assign locomotives to trains \(t_1 \in T_{t,d}^{\text{after break}}\) in such a way that the driver is compatible with at least one train-locomotive assignment \((t_1, L)\) with \(t_1 \in T_{t,d}^{\text{after break}}\) and \(L \in L^{t_1}\). More formally, for each driver \(d \in D\), we define a subset \(\mathcal{O}^d \subseteq T^d\) of trains which do not end in the driver’s home region (and hence no corresponding \(\omega\)-variable exists):

\[
\mathcal{O}^d := \{t \in T^d : a(t) \in S \setminus H(d)\}.
\]

For the trains \(t \in \mathcal{O}^d\), we consider all train-locomotive assignments which may be served only by driver \(d\). These will be stored in a set \(S^d\), defined as

\[
S^d := \{(t, L) : t \in \mathcal{O}^d \land L \in L^t \cap L^d \land D^t \cap D^L = \{d\}\}.
\]

Now, for each \((t, L) \in S^d\) we introduce the set \(\mathcal{T}_{t,L}^d\), which contains all train-locomotive assignments for which driver \(d \in D\) is able to drive a train after a break:

\[
\mathcal{T}_{t,L}^d := \{(t_1, L_1) : t_1 \in T_{t,d}^{\text{after break}} \land L_1 \in L^{t_1} \cap L^d\}.
\]

We use it to construct valid inequalities cutting off solutions which meet two conditions jointly for a driver \(d \in D\) and a train \(t \in \mathcal{O}^d\):

1. For the train \(t\) they assign such a locomotive class \(L \in L^t\) that driver \(d\) is the only available one, i.e. \(D^L \cap D^t = \{d\}\).
3 A solution algorithm for the integrated vehicle scheduling and crew assignment

2. For each of the trains $t_1 \in T_{t,d}^{\text{after break}}$, they assign such a locomotive class $L_1 \in \mathcal{L}^{t_1}$ that driver $d$ may not be assigned to them (i.e. $d \notin D^{t_1} \cap D^{L_1}$ for all $t_1 \in T_{t,d}^{\text{after break}}$ and $L_1 \in \mathcal{L}^{t_1}$).

Such solutions result in the violation of constraint (I11) and hence are infeasible.

**Theorem 3.2.6.** The following inequalities are valid for the model (I):

$$
\sum_{t_1: (t, t_1) \in \mathcal{F}^{L}} f^{t_1, t} \leq \sum_{(t, L_2) \in \mathcal{F}^{d}_{t, L_2}} \sum_{t_3: (t_2, t_3) \in \mathcal{F}^{L}_2} f^{t_2, t_3} \quad \forall d \in D, \forall (t, L) \in S^d.
$$

(V3)

**Proof.** Let $(\tilde{f}, \tilde{\lambda}, \tilde{x}, \tilde{y}, \tilde{v}, \tilde{z}, \tilde{\alpha}, \tilde{\omega})$ be a feasible solution to model (I). Suppose this solution violates constraint (V3). Then there exists a set $S^d$ for some $d \in D$ and a train-locomotive assignment $(t, L) \in S^d$ such that

$$
\sum_{t_1: (t_1, t_3) \in \mathcal{F}^{L}} f^{t_1, t} > \sum_{(t_2, L_2) \in \mathcal{F}^{d}_{t_2, L_2}} \sum_{t_3: (t_2, t_3) \in \mathcal{F}^{L}_2} f^{t_2, t_3}
$$

holds. By definition, for all drivers $d \in D$ and for any solution $(t, L) \in S^d$ we need to select at least one of the assignments $(t_1, L_1) \in \mathcal{F}^{d}_{t_1, L_1}$, as otherwise constraint (I11) would be violated. Since in inequality (3) we saw that the opposite follows from our assumption, the solution is infeasible. Hence, we have a contradiction. \(\square\)

The next example illustrates the cut.

**Example 3.2.7.** Let us consider a set of three trains $\{t_1, t_2, t_3\} \subset T$, as well as a set of three drivers $\{d_1, d_2, d_3\} \subset D$. Moreover, assume that $t_1 \in \mathcal{O}^d$ for all $d \in \{d_1, d_2, d_3\}$. Additionally, let $\{t_2, t_3\} \subset T_{t,d}^{\text{after break}}$ for all $d \in \{d_1, d_2, d_3\}$. Further, we assume the following locomotive compatibilities: $\mathcal{L}^{t_1} := \{L_1, L_2\}$, $\mathcal{L}^{t_2} := \{L_1, L_2, L_3\}$, $\mathcal{L}^{t_3} := \{L_2, L_3\}$. With regard to drivers, let $D^{t_1} = D^{t_2} = D^{t_3} := \{d_1, d_2, d_3, d_4\}$, and $D^{L_1} := \{d_2, d_3\}$, $D^{L_2} := \{d_1\}$, $D^{L_3} := \{d_2, d_3, d_4\}$.

For an assignment $\{(t_1, L_1), (t_2, L_1), (t_3, L_3)\}$, we notice that the only driver who is able to drive $t_1$ is $d_1$. Since $t_1 \in \mathcal{O}^{d_1}$, we need to assign locomotives to trains $t \in T_{t,d}^{\text{after break}}$ in such a way that the driver $d_1$ is compatible with at least one such assignment.

In the presented context, this is not the case, as for train $t_2$ the set of feasible drivers is $D^{t_2} \cap D^{L_1} = \{d_2, d_3\}$, and for train $t_3$ it is $D^{t_3} \cap D^{L_3} = \{d_2, d_3, d_4\}$. Hence, we get a solution which is infeasible with regard to constraint (I11).
3.2 Classes of valid cutting planes

To prevent this, we can use the cut introduced in this section. It will then make sure that the locomotives are assigned in such a way that for at least one of the trains \( t \in T_{t_1,d_1}^{\text{after break}} \) driver \( d_1 \) may be chosen as well.

In the context of our example, the cut would look the following way:

\[
\sum_{t_d^i \in \mathcal{L}} f_{L_2}^{t_1,t_d^i} \leq \sum_{t_b^i \in \mathcal{L}} f_{L_2}^{t_2,t_b^i} + \sum_{t_c^i \in \mathcal{L}} f_{L_2}^{t_3,t_c^i}.
\]

3.2.4 Valid cutting planes derived from the constraints related to Sunday breaks

The purpose of constraint (I18) is to enforce that no driver works on more than \( c^{\text{sunday}} \) many consecutive Sundays. We consider the set \( T_{w,d}^{\text{sunday}} \) for each driver \( d \in D \) and for each week \( w \in W \). Now let us define the graphs \( G_d^{\text{sunday}} = (V_d^{\text{sunday}}, E_d^{\text{sunday}}) \) for all \( d \in D \), by

\[
\begin{align*}
V_d^{\text{sunday}} &:= \{(t,L) : D^t \cap D^L = \{d\} \land \exists w \in W : t \in T_{w,d}^{\text{sunday}}\}, \\
E_d^{\text{sunday}} &:= \{(t_1,L_1), (t_2,L_2) : t_1, t_2 \in V_d^{\text{sunday}} \land w_1, w_2 \in W \land t_1 \in T_{w_1,d}^{\text{sunday}} \land t_2 \in T_{w_2,d}^{\text{sunday}} \land 0 < |w_2 - w_1| \leq c^{\text{sunday}}\}.
\end{align*}
\]

Further, for each \( d \in D \) let \( C_d^{\text{sunday}} \) be the set of all \((c^{\text{sunday}} + 1)\)–cliques in \( G_d^{\text{sunday}} \). Each such clique corresponds to a combination of train-locomotive assignments that would lead to a driver \( d \) working on \((c^{\text{sunday}} + 1)\)-many consecutive Sundays. This idea leads to a set of inequalities whose validity is easy to see.

**Theorem 3.2.8.** The following inequalities are valid for the model (I):

\[
\sum_{(t,L) \in C} \sum_{t_i \in \mathcal{L}} f_{L}^{t,t_i} \leq c^{\text{sunday}} \quad \forall d \in D \forall C \in C_d^{\text{sunday}}. \tag{V4}
\]

In the following we present an example for the cut.

**Example 3.2.9.** Consider a subset \( \{t_1, t_2, t_3, t_4\} \) of a larger set of trains \( T \). We assume that the set of trains \( T \) spans over one month with four Sundays and that \( t_1 \) is scheduled to run on the first Sunday of the month, \( t_2 \) is planned to run on the second Sunday, \( t_3 \) will run on the third Sunday, and finally \( t_4 \) is expected on the fourth Sunday.
Moreover, assume that the parameter \( c_{\text{Sunday}} \) takes the value of 3. Now, assume that locomotives of four types \( L_1, L_2, L_3, L_4 \) are assigned to the trains \( t_1, \ldots, t_4 \), respectively, and that – as a result – there is only one driver \( d \) capable of serving the chosen assignments, i.e. \( D^{t_1} \cap D^{t_1} = \{d\}, D^{t_2} \cap D^{t_2} = \{d\}, D^{t_3} \cap D^{t_3} = \{d\}, D^{t_4} \cap D^{t_4} = \{d\} \).

Such a locomotive assignment will result in an infeasibility in constraint (I18) – we set the maximal number of working Sundays (represented by parameter \( c_{\text{Sunday}} \)) to three. To prevent such infeasibilities from occurring, we add the cut of the form (V4). In the presented context, the inequality would look as follows:

\[
\sum_{t_a : (t_b, t_a) \in \mathcal{A}_{t_1}} f_{L_1}^{t_a, t_a} + \sum_{t_b : (t_2, t_b) \in \mathcal{A}_{t_2}} f_{L_2}^{t_2, t_b} + \sum_{t_c : (t_3, t_c) \in \mathcal{A}_{t_3}} f_{L_3}^{t_3, t_c} + \sum_{t_d : (t_4, t_d) \in \mathcal{A}_{t_4}} f_{L_4}^{t_4, t_d} \leq 3.
\]

### 3.3 Preprocessing the driver subproblem

In this section, we will describe two preprocessing mechanisms we use to reduce the size of the driver subproblem after the locomotive master problem has been solved, aiming at simplifying the solution process of the driver subproblem. The first one is the way to remove some unnecessary driver-related variables. The second consists in changing the sense of one of the constraints.

**Removing unnecessary driver-related variables** As the assignment of locomotive-classes to the trains has already been performed by the locomotive master problem, part of the variables relevant to the train-driver assignment can be eliminated. In particular the variables which pertain to drivers who are unable to drive both a given train and its assigned locomotive.

Let \((\tilde{f}, \tilde{\lambda})\) be a solution to the locomotive master problem. Using that solution, for all trains \( t \in T \) we can enumerate the subset of drivers who are compatible with both the train \( t \) and the locomotive class \( L \in \mathcal{L}^t \) which was assigned to it. More formally, for all \( t \in T \) let us denote the locomotive class selected to perform the train \( t \) by \( \tilde{L}^t := L \), where \( L \) is the unique \( L \in \mathcal{L}^t \) with \( \sum_{t_1 : (t, t_1) \in \mathcal{A}} f_{L_1}^{t_1, t_1} = 1 \).

Now as we know the selected locomotive class for each train \( t \in T \), we can introduce the corresponding set of feasible drivers as

\[
\tilde{D}^t := D^t \cap D_{\tilde{L}^t}.
\]
With these sets, we can now precisely generate those variables $x, y, z, v, \alpha$ and $\omega$ for all trains $t \in T$ and for all drivers $d \in \tilde{D}^t$ compatible with both the train and the selected locomotive. This means we can use the solution of the locomotive master problem to reduce the number of variables generated. Furthermore, those constraints of the driver subproblem which only include variables for drivers $D^t \setminus \tilde{D}^t$ on the left-hand side become trivial and can be eliminated as well.

**Changing the sense of one of the constraints** Recall the multiple-choice constraints (I4) for the driver assignment, which are part of the driver subproblem. As the locomotive master problem already determines which trains shall be performed, we can change the optimization sense of the driver subproblem from maximization (of objective function (I1)) into a mere feasibility problem if we change constraint (I4) into an equation as follows:

$$\sum_{d \in \tilde{D}^t} x_d^t = 1 \quad \forall t \in T.$$

(I4b)

In our early experiments, this reformulation has proved to be computationally more efficient than the maximization version of the driver subproblem.

### 3.4 A heuristic to presolve the driver subproblem

Despite of the preprocessing, the driver subproblem can be still be difficult to solve. Therefore, we now describe a heuristic which is able to find feasible solutions more quickly in many cases. It is based on solving an auxiliary MIP including constraints (I10a), (I15), (I17) and (I18), which make sure that the drivers’ requirements with regard to longer breaks (i.e. Sundays off and the long breaks) are respected. In our computational experiments, we saw that the driver subproblem was significantly easier to solve when these constraints had been relaxed. Accordingly, if we decide upfront which of the Sundays shall be a free day for each driver and after which train a long break shall begin, the solution to the remainder of the driver subproblem usually becomes much easier. To be precise, the mentioned auxiliary IP maximises objective function (I1) over the above-mentioned constraints (I10a), (I15), (I17) and (I18), as well as the following two constraints:

$$\sum_{d \in \tilde{D}^t} x_d^t \geq \left\lfloor \frac{|\tilde{D}^t|}{2} \right\rfloor + 1 \quad \forall t \in T \quad \text{ (H1)}$$

$$\sum_{t \in T_{\text{week}}} z_d^t \geq 1 \quad \forall w \in W, \forall d \in D. \quad \text{ (H2)}$$
The purpose of constraint (H1) is to ensure that there is sufficient choice of drivers for each train after fixing the breaks – we want to make sure that slightly more than half the drivers are available to the restricted driver subproblem solved afterwards, which empirically proved to be an adequate value. Constraint (H2) is required to make sure that at least one long break for each week \( w \in W \) is selected for each driver \( d \in D \) (even if the driver was not pre-assigned to any train via constraint (H1)). An optimal solution \((\bar{x}, \bar{z}, \bar{h})\) to this auxiliary model contains three important pieces of information for the driver subproblem. Firstly, it determines which Sunday shall be off for each driver in the given month. Secondly, it fixes the points in time when the weekly 35-hour breaks of the drivers start. But most importantly, it indicates for all drivers \( d \in D \) which trains \( t \in T^d \) cannot be served when respecting the selected Sunday or 35-hour breaks. Based on the latter, we restrict the set of available drivers for a given train \( t \in T \) to

\[
D^t_{\text{available}} := \{ d \in \tilde{D}^t : \tilde{x}^t_d = 1 \}.
\]

If the above auxiliary MIP is feasible, we heuristically preprocess out further \( x-, y-, v-, \alpha- \) and \( \omega\)-variables accordingly and fix the decisions concerning long breaks to obtain a restricted driver subproblem. Should the heuristic preprocessing fail (i.e. either auxiliary MIP or restricted driver subproblem are infeasible), which it seldom did in our experiments, we instead solve the full, unrestricted driver subproblem directly.

We would like to point out that in more general settings, the efficiency of the heuristic introduced in this section – in its current shape – will probably decrease with more working time constraints. However, it is straightforward to extend it to accommodate other constraint types and this way to adjust its performance.
4 Computational results

In this chapter, we demonstrate the efficiency of our methods for solving the integrated locomotive scheduling and driver assignment problem in a real-world case study. We begin by describing how we will assess the enhancements to the solution algorithm presented in Chapter 3 in terms of reducing the computation times. We also mention some of the details which lead to a well-performing implementation. Then we describe the case study itself, performed at the hand of a country-wide problem instance stemming from Polish rail freight traffic. We also mention the values assigned to parameters critical to the generation of the working time constraints. After a description of the input data provided by our industry partner DB Cargo Polska, we analyse both the solution times of our algorithmic approaches and the quality of the computed solutions in terms of covering the order book as far as possible. To further show the efficiency of our approach, we also present how the number of trains running in the optimal solution changes under less favourable resource availability conditions. We present two scenarios, in which the number of locomotives and drivers are reduced. In the first one, we iteratively remove the least-used resources and reoptimize. In the second, we randomly remove some of the resources upfront.

4.1 Implementation details

In this section, we present the details of the implementation of our method, as well as the hardware and software used in our computational study.

We have run all the computations presented in Sections 4.2 and 4.3 on a compute server with two Intel Xeon E5-2643 v4 processors using all 12 cores and 256 GB of memory. Further, we have used Gurobi 9.5 ([Gurobi, 2021]) to solve the arising binary optimization problems. The models are built and solved via its Python interface. Finally, we have used NetworkX ([Hagberg et al., 2008]) to represent the underlying graph structures.

Recall that for generating constraints (I7a)–(I10a) in the improved version of model (I), we need to find minimal clique edge covers in certain graphs. As this problem is NP-hard in general (see [Kou et al., 1978]), we use the maximal-clique enumeration algorithm developed by [Bron and Kerbosch, 1973] as adapted by [Tomita et al., 2006] and discussed in [Cazals and Karande, 2008] to solve it heuristically. This algorithm is implemented in the Python NetworkX package (see [Hagberg et al., 2008]). The choice of this method is justified by its good running time behaviour ([Tomita et al., 2006] showed that its worst-case time complexity amounts to $O(3^{n/3})$ for a graph with
n nodes), its ease of use as part of our implementation as well as the good results we obtained with it.

To further decrease model generation times, we also use a slightly simplified approach to the clique tightening (as described in Section 2.4). Namely, each of the graphs $G^d_{\text{time}}$, $G^d_{\text{back\_break}}$, $G^d_{\text{forward\_break}}$ and $G^d_{\text{long\_break}}$ generated for each driver $d \in D$, contains only trains relevant to the driver $d$ as its nodes. In our implementation, we generate the graphs $G^d_{\text{time}}$, $G^d_{\text{back\_break}}$, $G^d_{\text{forward\_break}}$ and $G^d_{\text{long\_break}}$ introduced in Section 3.2 instead, which contain all trains $t \in T$. Then we use the maximum-clique algorithm mentioned in the previous paragraph to enumerate all the maximal cliques of these larger graphs. For each of the four sets of maximal cliques, we generate an adjusted version of the cliques for each driver $d \in D$ by removing the trains which are not compatible with the driver $d$. This simplification allows us to only perform the clique enumeration four times instead of $4 \cdot |D|$ times. As a result, we obtain a significant decrease in model generation time, at the expense of a slightly higher number of constraints generated.

For our computational experiments, we used a slightly redefined version of the set $T_{t,d}^{\text{l,B+}}$ as it allowed for a simpler implementation. It is different from the above definition by the fact that instead of using exact transportation times between stations, we use maximal transportation time between stations which are the origin and destination stations of trains $t \in T$. Since the heuristic version of the set includes more trains than necessary, it is actually to the benefit of the drivers – their breaks could potentially be longer than the assumed $c_{\text{long}}$ hours plus the transportation time. This restriction still allowed us to obtain optimal solutions for all the instances in our real-world case study. We could deduce that from a comparison of the number of trains covered by the locomotive master solution and the driver subproblem solution respectively.

In order to assess the impact of each of the improvements we developed on top of our basic solution approach, we will perform numerical experiments subdivided into three different groups. In the following, we describe each of these computational scenarios in detail.

**Clique tightening**  As a first step, we will attempt to solve the initial formulation of model (I) extended by clique tightening. This means we replace constraints (I7), (I8), (I9) and (I10) with (I7a), (I8a), (I9a) and (I10a) respectively (see Section 2.4). From now on, we will refer to this scenario as CLQ.
4.1 Implementation details

**Decomposition**  Here we consider the decomposition approach as described in Section 3.1. To recall, it comprises the following three steps between which the algorithm iterates until convergence:

1. Solve the locomotive master problem given by objective function (L1) and constraints (I23)–(I26), (I34), (L2) and (L3) extended by the additional valid inequalities developed in Section 3.2. To be precise, inequalities (V3) are all added from the beginning, while inequalities (V1), (V2) and (V4) are added as lazy constraints. In other words, whenever a feasible solution to locomotive master problem is found, all the inequalities (V1), (V2) and (V4) are tested for correctness. If any of them is violated, it is added to the locomotive master problem and the optimization restarts. This procedure is iterated until no violated inequalities are found.

2. Preprocess the driver subproblem, as discussed in Section 3.3.

3. Solve the driver subproblem, i.e. the feasibility problem given by constraints (I2), (I3), (I4b), (I5)–(I6), (I7a)–(I10a), (11)–(122) and (127)–(132), with the f-variables fixed according to the solution of the locomotive master problem. If the problem is infeasible, we generate a combinatorial Benders cut for the locomotive master problem and iterate. Otherwise, the algorithm terminates with a globally optimal solution.

We refer to this implementation as **DECOMP**.

**Decomposition + Heuristic**  In the final, most advanced implementation, we use the complete algorithmic scheme shown in Figure 12 on page 52. Mainly, this is implementation **DECOMP** enhanced by the long-break heuristic described in Section 3.4. In particular, the procedure loops over the following four steps until convergence:

1. Solve the locomotive master problem as in implementation **DECOMP**.

2. Preprocess the driver subproblem, as discussed in Section 3.3.

3. Solve the long-break presolve heuristic for the driver subproblem described in Section 3.4.

4. Solve the driver subproblem as in implementation **DECOMP** with the following modification: we do not only fix the f-variables according to the solution of the locomotive master problem, but also the (x, y, v, a, ω, h)-variables as per the solution of the long-break presolve heuristic (if it is successful, otherwise we solve the full driver subproblem with only the f-variables fixed).
This implementation will be called DECOMP+HEUR in the following.

### 4.2 Performance of the method against instances supplied by DB Cargo Polska

In this section, we will present the first case study for the method developed in Chapter 3. It will be based on the data set supplied by our industrial partner, DB Cargo Polska. We will also discuss the results, both from the perspective of the solution times and quality.

Our industry partner provided us with a high-quality real-world data set for the problem, covering a full month of planning (for February 2020). Firstly, it comprises the complete order book for this month, i.e. a list of all the trains that need to be run, including their origin and destination stations, departure and arrival times and the respective calculation weeks they are counted to. The data set covers four calculation weeks, which always start on a Saturday and last until the next Friday. Further, we have obtained the full list of drivers, together with their respective licenses to locomotives, knowledge of routes and home regions. We were also provided with the list of available locomotives, stating their respective tractive power and energy source, as well as a mapping of stations to regions in Poland. To estimate the travel times for the assumed “car rides” of drivers between stations (as discussed in Subsection 2.2.1), we used the data available on 14 February 2020 from the Google Maps API.

To reflect the requirements of the Polish Labour Code and the Polish Railway Transport Act, our industrial partner gave us the following values for parameters used in the generation of sets required for constraints construction. They are presented in Table 4.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{\text{shift}}$</td>
<td>12</td>
</tr>
<tr>
<td>$c_{\text{short}}$</td>
<td>11</td>
</tr>
<tr>
<td>$c_{\text{long}}$</td>
<td>35</td>
</tr>
<tr>
<td>$c_{\text{Sunday}}$</td>
<td>3</td>
</tr>
</tbody>
</table>

Since the data about the route knowledge of the drivers was limited to only inner-Polish routes, we restrict ourselves to trains departing and arriving in the territory of Poland. In cases where a given train was to terminate at the
first station past the Polish border, we artificially shortened the route to the passed border station on the Polish side. The trains which only “commute” between two neighbouring border terminals of Poland and a neighbouring country were not taken into account, since they are rather to be considered as shunting connections, which are not in the focus of this work. In total, we needed to sort out 390 out of 2941 trains, which leaves us with a total of 2551 trains to be served.

Based on the data received from the industrial partner, we derived ten problem instances corresponding to different planning horizons, ranging in length from one week to a full month. In each instance, we have assumed all 217 drivers and 112 locomotives of DB Cargo Polska to be available. Table 5 below presents a summary of the instances. Their names correspond to the time period they entail (e.g. 1M – the whole month, 1W.1 – the first week of the month, 2W.3 – the third two-week period of the month, i.e. weeks 3 and 4, and so on). Note that for the instances spanning less than one month, we excluded the $h$-variables and the constraints pertaining to Sunday breaks. With a growing instance size, the computational complexity of model generation increases exponentially due to the effort required to perform the clique tightening. Especially for the larger instances, the model generation is a challenging task on its own. Here we report arithmetic averages of the model generation times over three runs, since they proved to be similar regardless of the chosen version of the algorithm.

<table>
<thead>
<tr>
<th>Instance</th>
<th>#Days</th>
<th>#Trains</th>
<th>Avg. model generation time (in s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1W.1</td>
<td>7</td>
<td>629</td>
<td>320</td>
</tr>
<tr>
<td>1W.2</td>
<td>7</td>
<td>610</td>
<td>293</td>
</tr>
<tr>
<td>1W.3</td>
<td>7</td>
<td>615</td>
<td>314</td>
</tr>
<tr>
<td>1W.4</td>
<td>7</td>
<td>613</td>
<td>306</td>
</tr>
<tr>
<td>2W.1</td>
<td>14</td>
<td>1239</td>
<td>798</td>
</tr>
<tr>
<td>2W.2</td>
<td>14</td>
<td>1242</td>
<td>782</td>
</tr>
<tr>
<td>2W.3</td>
<td>14</td>
<td>1228</td>
<td>800</td>
</tr>
<tr>
<td>3W.1</td>
<td>21</td>
<td>1854</td>
<td>1427</td>
</tr>
<tr>
<td>3W.2</td>
<td>21</td>
<td>1838</td>
<td>1401</td>
</tr>
<tr>
<td>1M</td>
<td>29</td>
<td>2551</td>
<td>2358</td>
</tr>
</tbody>
</table>
4 Computational results

4.2.1 Analysis of solution times

For the computational experiments, we have used the instances from Table 5 and compare their solutions times for the three different implementation scenarios described in Section 4.1. We have performed three runs of each implementation on each instance and report the average solution times in Table 6. The first column presents the name of the instance. Columns 2 to 4 display the performance of the respective implementation against the listed instances. A dash denotes that no solution has been found within the time limit of 7200 seconds for all three runs of each scenario-instance combination. A dagger (†) means that the presented result pertains to only two of the computations and that the remaining one timed out. Similarly, a double dagger (‡) indicates that the presented result pertains to only one of the computations and that the remaining two timed out. An integer number denotes the average time in seconds required to find an optimal solution.

Overall, it is obvious from Table 6 that the use of the decomposition approach is necessary to obtain a solution for the largest instances. The integrated model is only able to solve the instances spanning one week to optimality, but even these are never solved in all three runs. For the longer planning horizons, it returned no integer-feasible solution within the allotted time. We can also observe that employing the long-break heuristic has produced significant speed-ups in the solution times. These solution times even exceeded the expectations of our industrial partner, for whom it would already have been
sufficient to obtain a monthly schedule in less than 12 hours. For shorter planning horizons, they are even short enough to allow for an interactive use, e.g. for performing what-if analyses.

Finally, it should be mentioned that for all the experiments presented in this section, already one iteration of the algorithm was sufficient. Moreover, since the instances span over realistic resource availabilities, there was no need to generate the valid inequalities discussed in Section 3.2, except for those added upfront.

### 4.2.2 Assessment of solution quality

In Table 7, we compare the number of trains in each instance to the number of trains served in the optimal solution. The first column states the name of the instance, with the second column repeating the corresponding number of trains in that instance. The third column presents the globally optimal solution obtained via the decomposition approach developed in this work. Finally, the fourth column gives the coverage obtained with this solution, computed as the number of the trains performed in the optimal solution, divided by the total number of trains in the instance. Overall, we can conclude that the solutions our approach produces enable our industrial partner to perform all or nearly all trains in the order book, regardless of the time horizon of the instance. These results were satisfying to the industrial partner, because

<table>
<thead>
<tr>
<th>Instance</th>
<th>#Trains</th>
<th>Optimum</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1W_1</td>
<td>629</td>
<td>629</td>
<td>100.0%</td>
</tr>
<tr>
<td>1W_2</td>
<td>610</td>
<td>610</td>
<td>100.0%</td>
</tr>
<tr>
<td>1W_3</td>
<td>615</td>
<td>615</td>
<td>100.0%</td>
</tr>
<tr>
<td>1W_4</td>
<td>613</td>
<td>613</td>
<td>100.0%</td>
</tr>
<tr>
<td>2W_1</td>
<td>1239</td>
<td>1238</td>
<td>99.9%</td>
</tr>
<tr>
<td>2W_2</td>
<td>1242</td>
<td>1224</td>
<td>98.6%</td>
</tr>
<tr>
<td>2W_3</td>
<td>1228</td>
<td>1228</td>
<td>100.0%</td>
</tr>
<tr>
<td>3W_1</td>
<td>1854</td>
<td>1845</td>
<td>99.5%</td>
</tr>
<tr>
<td>3W_2</td>
<td>1838</td>
<td>1829</td>
<td>99.5%</td>
</tr>
<tr>
<td>1M</td>
<td>2551</td>
<td>2528</td>
<td>99.1%</td>
</tr>
</tbody>
</table>
the few remaining trains can likely be served by repositioning locomotives in a similar fashion as we did with the drivers. Together with the relatively short solution times we achieve, our approach can prove very favourable if integrated in a decision support tool, where planners can manually add such repositionings (“empty runs”) to make sure that all the trains can be run. An example illustration of a schedule assigned to a locomotive and a shift assigned to a driver can be found in Appendix A.

4.3 Performance of the method under less favourable conditions

In this section, we will test the performance of the method introduced in this thesis against a set of instances derived from the 1M instance with limited availability of drivers and locomotives for the trains. We will consider two scenarios. In the first one, we will explore how the optima returned by our algorithm change when the least used resources are being removed. The second scenario will consider an upfront removal of random locomotives and drivers.

4.3.1 Exploring the optimal values with a decreasing number of locomotives and drivers

In the first scenario, we will explore how the optima change when only the assets which have performed the most work are kept in the asset base.

Description of the scenario  In this scenario, we run our algorithm against the 1M instance in an iterative fashion. We begin with all the resources available. For each consecutive run, we remove approx. 10% of the least used resources (rounded to the nearest integer), resolving draws randomly. For locomotives, we ensure that at least one locomotive of each type remains present throughout the experiment. Since the setting involved an iterative approach, we let the evaluation process run for 24 hours, a maximum allowed by the compute cluster we used for our experiments.

Discussion of the results  Table 8 presents the results of running the iterative scenario described above for 24 hours. The first column denotes the number of iterations of the algorithm. Iteration 0 means that the algorithm runs with all the resources available. The second and third columns represent the number of locomotives resp. drivers in consecutive iterations. Similarly, the fourth and fifth column represent the share of locomotives and drivers available in each instance to their respective total numbers. The sixth
4.3 Performance of the method under less favourable conditions

column represents the value of the objective function after each iteration, while the seventh presents it as a percentage of all trains in the order book. In the seventh column, the solution times for each iteration are shown. Since the model generation times are similar to the ones reported in Section 4.2, we do not display them here.

Table 8: Changes in optimal value with decreasing availability of drivers and locomotives

| Iter. | $|\mathcal{L}|$ | $|D|$ | %$|\mathcal{L}|$ | %$|D|$ | Value | %$|T|$ | Time (s) |
|-------|-------------|-----|---------|--------|-------|-------|--------|
| 0     | 112         | 215 | 100     | 100    | 2528  | 99.1  | 1811   |
| 1     | 101         | 193 | 90      | 90     | 2512  | 98.5  | 1726   |
| 2     | 90          | 172 | 80      | 80     | 2509  | 98.4  | 1319   |
| 3     | 79          | 150 | 71      | 70     | 2498  | 97.9  | 1505   |
| 4     | 68          | 128 | 61      | 60     | 2484  | 97.4  | 977    |
| 5     | 58          | 106 | 52      | 49     | 2467  | 96.7  | 3654   |
| 6     | 48          | 85  | 43      | 40     | 2434  | 95.4  | 2154   |

Table 8 clearly shows that, despite decreasing resource availability by up to 60%, solutions which cover some 95% of the order book may still be achieved. Such an assignment is much more efficient due to the much higher utilization of the employed locomotives and drivers as well as the more economic choice of trains to serve. Therefore, the presented procedure may well serve as a core plan which may be used as a starting point for the creation of a full plan in the company. The planner could use the solution from Iteration 6 (as presented in Table 8) and use the remaining assets to power as many of the unserved trains as desirable. Further, the freed-up assets could be assigned to tasks which are not included in the instance. This may help determine the maintenance plan – unused locomotives may now be scheduled for (long-term) maintenance. With regard to the algorithmic performance, we also remark that for the presented results only one iteration of the solution algorithm was sufficient in each reduction iteration. Moreover, since the considered resource availabilities are not extremely tight, there was no need to generate the valid inequalities discussed in Section 3.2. When reducing the availability of locomotives and drivers below 40%, the instances could not be solved within the limit.
4 Computational results

4.3.2 Limiting the availability of drivers and locomotives upfront

In the second scenario, we will consider the impact of decreasing the asset base by removing a certain percentage of randomly selected assets before proceeding with the optimization.

Description of the scenario In this scenario, we again run our algorithm against the order book of the 1M instance. We test the performance of the method when the availability of the resources is limited. To this end, we limit the availability of assets in steps of 10%. The reduction of the locomotive availability is performed by randomly removing locomotives from the original data. With regard to drivers, we simulate holiday patterns which could be requested by the drivers affected. These include requesting one or two periods off in a month. If one period is selected, it may last 1, 2, 3, 7, 14, 21 or 28 days. If two periods off are requested, each one may last 1, 2, 3, 7 or 14 days. We randomly select the subset of drivers whose availability becomes limited. For these, we randomly select the number of periods of time off and then their respective duration and starting date(s).

Discussion of the solution Table 9 presents the optimal solutions to the 1M order book with limited resource availability. Rows correspond to different availabilities of locomotives, while columns represent various degrees to which drivers are available. The numbers reported correspond to an average of the discussed result from three runs. If a dagger (†) is shown next to a number, it means that the number is an average of two results, since one experiment failed. If there is no number in a cell, then no experiment for such a configuration was conducted. Such a table structure follows throughout this paragraph.

Table 9 itself shows that decreasing only the number of locomotives did not make the problem much more difficult to solve – the algorithm always found an optimal solution within the allotted time. As the number of locomotives available decreases, so does the optimum. Even with only 20% of the locomotives and all drivers, the algorithm can still cover the order book to a level of 77% (1961 out of 2551 trains). In the opposite case, with all the locomotives present and a decreasing availability of the drivers, the optimal solution does not change down to a little as 50% of the original count of drivers. Instances with smaller percentages of drivers could not be solved within the allotted time of three hours. For the much more realistic case of proportional decreases (equal shares of the original locomotive and driver sets), the optima
4.3 Performance of the method under less favourable conditions

Table 9: Irregular reduction – optimal values

<table>
<thead>
<tr>
<th>% of locomotives remaining</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1357</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1961</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2271</td>
</tr>
<tr>
<td>40</td>
<td>2416</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2437</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>2458</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2458</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td>2483</td>
<td></td>
<td></td>
<td></td>
<td>2483</td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td>2504</td>
<td></td>
<td></td>
<td>2504</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2518</td>
<td></td>
<td>2518†</td>
</tr>
<tr>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2521</td>
<td>2521</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2528</td>
</tr>
</tbody>
</table>

returned for successful experiments do not differ for all but one case from the scenario in which only the locomotive set was reduced.

Table 10 presents the average solution times across the instances. We may conclude that removing drivers complicates the model much more than removing locomotives and generally contributes to longer solution times. We may also conclude that solution times vary greatly for similar settings – for example, while the instance comprising 60% of drivers and 60% of locomotives solves to optimality in 3159 seconds on average, a more “relaxed” instance consisting of 60% of drivers and 100% of locomotives requires 4073 seconds on average. This is partially explained by Table 12, which presents the different patterns of the solution algorithm behaviour. It shows that for a majority of the successful experiments, the driver problem heuristic turned out to be successful. For the ones where the heuristic failed, the full driver subproblem was solved within the allotted time. For the mentioned case, the two runs of the “tighter” instance comprising 60% of drivers and 60% of locomotives solved to optimality after a successful driver heuristic run, while for the “more relaxed” instance consisting of 60% of drivers and 100% of locomotives the driver heuristic failed in all three runs and hence the more complex full driver subproblem needed to be solved, and this resulted in longer solution times.
Finally, Table 11 presents the average number of combinatorial Benders cuts added in the solution process of each individual instance (rounded to the nearest integer). We see that the combinatorial Benders cuts and the resulting need to run the algorithm loop from the beginning occurred primarily for instances which ultimately did not solve to optimality within the allotted time (recognizable from the fact that they are not depicted in Tables 9 – 12). We saw in our experiments that this also applies to the problem-specific cutting planes from Section 3.2. Overall, our computational results clearly show that our method is capable of solving the integrated locomotive scheduling and driver assignment problem on a real-world scale.

To conclude, we can say that the solutions our approach produces enable the railway company to perform all or nearly all trains in the order book, regardless of the time horizon of the instance. These results were satisfying to the industrial partner, because the few remaining trains can likely be served by repositioning locomotives in a similar fashion as we did with the drivers. Together with the relatively short solution times we achieve, our approach can prove very favourable if integrated in a decision support tool, where planners can manually add such repositionings (“empty runs”) to make sure that all the trains can be run.
### 4.3 Performance of the method under less favourable conditions

Table 11: Average number of combinatorial Benders cuts added in the solution process

<table>
<thead>
<tr>
<th>% of drivers remaining</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>% of locomotives remaining</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>70</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 12: Algorithmic behaviour of solution process (driver heuristic successful / driver heuristic failed)

<table>
<thead>
<tr>
<th>% of drivers remaining</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>% of locomotives remaining</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>0/3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>0/3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>2/1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>70</td>
<td>70</td>
<td>0/3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>3/0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>90</td>
<td>90</td>
<td>3/0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>0/3</td>
<td>0/3</td>
<td>3/0</td>
<td>3/0</td>
<td>3/0</td>
<td>3/0</td>
</tr>
</tbody>
</table>
Part II

The graph 2-list-colouring problem with colour compatibilities
In this part of the thesis, we will consider a problem which is a generalization of the integrated locomotive scheduling and driver assignment problem, studied in Part I. We call this problem *graph 2-list-colouring with colour compatibilities (G2LC-CC)*. While graph colouring lends an illustrative metaphor which facilitates the understanding of the decision challenge at hand, the problem we study is different to classical graph colouring problem in several dimensions. First of all, given a simple undirected graph $G = (V, E)$, the problem studied here requires that two colours (labels) are assigned to each vertex $v \in V$. Further, each of the two colours assigned must originate from a different colour set (which we call *colour family*). Additionally, the two colours selected for each vertex must be mutually compatible. They also need to be compatible with the node they are selected for. On top of this, we require that no two nodes connected by an edge may share any of the two colours assigned. Unlike the classical graph colouring problem, we allow a situation in which some nodes remain uncoloured.

The composition of this part corresponds to the way in which Part I is structured: it contains four chapter, namely Chapters 5–8.

Chapter 5 will be devoted to the presentation of the problem and its place in the context of the existing literature. It corresponds to Chapter 1 in the Part I of this thesis. We will start with introducing the relevant definitions and formally stating the problem. This will be done in Section 5.1. Later, in Section 5.2, (G2LC-CC) will be put into the context of other popular combinatorial problems, such as “classical” graph colouring (with its generalizations), stable set, maximum clique and set packing problems. Basing on [Padberg, 1973] and [Balas and Padberg, 1976], we will also discuss some of the polyhedral properties of the set packing problem, which will be of use in the later chapters.

Then, in Chapter 6, we will present two formulations for the problem studied. They will be introduced in Section 6.1. While one of them uses only one “type” of binary variables, the other one requires two, as well as an additional type of constraints. In Section 6.2, we will also discuss how these formulations may be improved using clique tightening. Finally, we will devote Section 6.3 to present a case for which a complete description of a relaxed version of (G2LC-CC) can be given. This chapter corresponds roughly to the Chapter 2 in the Part I of this thesis.

Later, in Chapter 7, we will introduce a decomposition-based solution approach for one of the formulations to (G2LC-CC). It is a generalization of the solution approach presented in Chapter 3 in the Part I of this thesis. In the very beginning of that chapter, we will present a schematic overview of the solution algorithm introduced therein, as well as its description. In
Section 7.1, we will present the decomposition scheme which underlies the solution method we develop. Then, in Section 7.2, the first preprocessing scheme will be discussed. It allows to remove some of the infeasible assignments upfront. Section 7.3 will be devoted to the presentation of a valid inequality class, which contributes to ensuring that the solutions to the two decomposed subproblems are globally feasible. Finally, in Section 7.4, we will introduce one more preprocessing technique, which allows to use the solution to one of the subproblems to remove the infeasible assignment in the other one.

Finally, in Chapter 8, we will test the solution algorithm against a set of instances drawn from [Gualandi and Chiarandini, 2013], which are then appropriately adjusted to match the context of \((G2LC-CC)\). In particular, we will begin with discussing the details about our implementation in Section 8.1. Then, we devote Section 8.2 to discuss the instances we use for our computational experiments. As mentioned above, they are standard, graph-colouring instances commonly seen in the literature, which are appropriately adjusted to the requirement of \((G2LC-CC)\). Next, in Section 8.3 we discuss the performance of the solution method introduced in Chapter 7 against the two formulations presented in Chapter 6. Structurally this chapter corresponds to the Chapter 4 in Part I of this thesis.
5 Introduction, problem definition and literature survey

This chapter is devoted to the proper definition of the problem we study throughout Part II of this thesis, as well to setting it in the context of other combinatorial problems. In particular, we will begin with introducing the relevant definitions and formally stating the problem \((G_2 LC-CC)\) in Section 5.1. Then, we will devote the Section 5.2 to put \((G_2 LC-CC)\) into the context of other popular combinatorial problems, such as “classical” graph colouring (with its generalizations), stable set, maximum clique and set packing problems. Basing on [Padberg, 1973] and [Balas and Padberg, 1976], we will also discuss some of the polyhedral properties of the set packing problem, which we will then exploit in Chapter 6.

5.1 Basic definitions and problem statement

This brief section is devoted to introducing the appropriate definitions, which will be used throughout this part of this thesis. We will then be able to properly state the problem, which we will study.

In both this chapter and the subsequent ones, the following (standard) notation and terminology will be used. We deal with a simple (undirected, loopless) graph \(G = (V, E)\), with \(V\) denoting the vertex set and \(E\) denoting the set of edges. A clique is a set of pairwise adjacent nodes. A clique is maximal if it may not be extended by further nodes. A stable set is a set of pairwise disjoint nodes. Further, the complement of a graph \(G = (V, E)\) is the graph \(G' = (V, E')\), where \(E' = \{\{i, j\} \in V : \{i, j\} \notin E\}\). A chordal graph \(G = (V, E)\) is a simple undirected graph in which every graph cycle of length four and greater has a chord.

We also have two sets of colours \(C\) and \(D\) (we also refer to them as colour families). For each \(v \in V\), we define two sets of node-compatible colours \(C^v \subseteq C\) and \(D^v \subseteq D\), i.e. only the colours in \(C^v\) and \(D^v\) may be selected for a vertex \(v \in V\). Additionally, for each \(c \in C\) we also have sets \(D_c \subseteq D\) of compatible colours \(d \in D\); we similarly define \(C_d \subseteq C\) as a set of compatible colours \(c \in C\) for all \(d \in D\). To simplify the notation, we will refer to colours \(c \in C\) as c-colours; and colours \(d \in D\) will also be called d-colours. Let us also use \(\delta(v)\) for all \(v \in V\) to denote the set of nodes neighbouring a node \(v \in V\). Additionally, for every set \(S\) we define its power set \(\mathcal{P}(S)\) as the set that contains all subsets of a given set, including the empty set and the set \(S\) itself as well.
Problem statement  Based on the definitions above, we are now ready to state the problem we will be studying in the next chapters. We are given a graph \( G = (V, E) \) and two colour families \( C \) and \( D \). We also know the colour compatibilities \( C_d \) for all \( d \in D \) and \( D_c \) for all \( c \in C \). Moreover, we also know the sets of \( c \)-colours \( C^v \) and \( d \)-colours \( D^v \) compatible with each vertex \( v \in V \). We are looking for such a selection of both \( c \)-colours and \( d \)-colours for nodes \( v \in V \) that as many of them are coloured as possible. Moreover, we demand that the nodes are either coloured with both a \( c \)-colour and a \( d \)-colour or by no colours at all. We also require that the chosen colours are compatible with the node \( v \in V \) they are selected for. Moreover, for each node \( v \in V \), we ask that the selected colours are compatible with each other. Finally, we require that for any two nodes \( (i, j) \in E \), neither the \( c \)-colour nor the \( d \)-colour may be the same. We will call the problem a graph 2-list-colouring with colour compatibilities (G2LC-CC).

5.2 Literature overview

In the following, we will discuss the most important aspects of some combinatorial problems which are related to (G2LC-CC). On top of the “classical” Graph Colouring Problem (GCP), we will consider Stable Set, Maximum Clique and Set Packing Problems. As shown by [Cornaz and Jost, 2008] and [Delle Donne and Marenco, 2016], these problems are equivalent to the graph colouring problem. Moreover, as we will see in the next chapter, the problem we consider in this part of the thesis – (G2LC-CC) – will actually assume the form of a stable set problem. This justifies the need to study its polyhedral properties in more detail.

5.2.1 Graph colouring

Graph Colouring Problem (GCP) is one of the longest studied problems in graph theory. While it is known to be NP-hard for arbitrary graphs, see [Karp, 1972], it is polynomially solvable for some graph classes, such as perfect graphs – see [Grötschel et al., 1993, Johnson, 1985]. In the following, we will discuss both the “classical” (GCP) and its generalizations.

“Classical” graph colouring problem  The “classical” graph colouring problem (GCP) consists in labeling each of the vertices \( v \in V \) with colours \( c \in C \) in such a way that no edge connects two vertices with the same label (colour) and the total number of different labels used is minimized.

Following [Coll et al., 2002], we now give the standard formulation of the problem. For each node \( v \in V \) and for each colour \( c \in C \) we define a binary
variable $x^y_v$. It assumes a value 1 if a colour $c$ is selected for node $v$ and 0 otherwise. Further we define a binary variable $w_c$, which takes a value 1 if colour $c$ is used in the solution and 0 otherwise. The formulation is given by model (5):

\[
\min \sum_{c \in C} w_c \quad \text{(5a)}
\]

\[
\text{s.t.} \quad \sum_{c \in C} x^y_v = 1 \quad \forall v \in V \quad \text{(5b)}
\]

\[
x^u_v + x^w_v \leq w_c \quad \forall \{v, w\} \in E \quad \forall c \in C \quad \text{(5c)}
\]

\[
x_v \in \{0, 1\} \quad \forall v \in V \quad \text{(5d)}
\]

\[
w_c \in \{0, 1\} \quad \forall c \in C. \quad \text{(5e)}
\]

The objective function (5a) minimizes the number of colours used in the solution to the problem. Constraint (5b) ensures that exactly one colour is selected for each node $v \in V$. With the help of constraint (5c), we make sure that no pair of adjacent vertices share the same colour, and that $w_c = 1$ if some vertex is assigned colour $c \in C$. It also implies that none of the vertices $v \in V$ is assigned the colour $c \in C$ if the corresponding variable $w_c$ is equal to 0. Constraints (5d) and (5e) ensure that the variables are binary.

For other formulations to (GCP), we refer to [Borndörfer et al., 1998], as well as [Delle Donne, 2009, Campêlo et al., 2004, Burke et al., 2010]. Many classes of valid inequalities, as well as facet-defining inequalities for (GCP) are introduced by [Coll et al., 2002]. The applications of (GCP) are very numerous. For example, it is used in scheduling, timetabling, bandwidth allocation and sequencing. A comprehensive overview is given by [Lewis, 2016].

**Generalizations of (GCP)** Over time, more and more variants of (GCP) were introduced in the literature. The most important of them include Precolouring Extension ([Biró et al., 1992]), $\mu$-Colouring ([Bonomo and Cecowski, 2005]), $(\gamma, \mu)$-Colouring ([Bonomo et al., 2009]) and List Colouring ([Tuza, 1997]). A detailed discussion is given by [Delle Donne and Marenco, 2016].

A proper generalization of (GCP), the Graph Multicolouring Problem (GMP), has also received quite a lot of attention. Given a graph $G = (V, E)$, (GMP)
5 Introduction, problem definition and literature survey

seeks to assign a number of labels (colours) to each vertex. The number of colours required by each of the vertices is specified by the length (or colour requirement) parameter of that vertex in the input. At this place, we should mention also the Non-Preemptive Multicolouring, which requires that colours assigned to a vertex are consecutive, as well as Sum-Multicolouring, which seeks to minimize the sum of the multicolourings, or the sum of the largest colour assigned to each vertex (assuming colours are natural numbers). We refer to [Halldórsson and Kortsarz, 2004] for more details.

5.2.2 Stable Set, Maximum Clique and Set Packing Problems

We will now briefly introduce the Stable Set, Maximum Clique and Set Packing Problems and discuss the relationships between them. As we will see, they are closely related to each other. For the Set Packing Problem, we will present the polyhedral properties relevant to our considerations in the next chapter. Some excellent surveys in this topic were given by [Bomze et al., 1999, Grötschel et al., 1993, Pardalos and Xue, 1994, Prosser, 2012, Wu and Hao, 2015].

Stable Set Problem The Stable Set Problem (SSP) is one of the oldest and most-studied problems in combinatorial optimization ([Nemhauser and Wolsey, 1999]). It consists in finding the stable set of maximal cardinality in a graph \( G = (V, E) \).

Let us begin with introducing the “classical” formulation of (SSP), as introduced by [Padberg, 1973]. For each node \( v \in V \) we define a binary variable \( x_v \). It assumes a value 1 if node \( v \) is selected to be included in the stable set and 0 otherwise. The formulation is given by model (6):

\[
\begin{align*}
\text{max } & \sum_{v \in V} x_v \\
\text{s.t. } & x_i + x_j \leq 1 \quad \forall \{i, j\} \in E \\
& x_i \in \{0, 1\} \quad \forall i \in V.
\end{align*}
\]

The objective function (6a) maximizes the number of vertices entering the stable set. Constraint (6b) ensures that no two adjacent vertices are selected for the stable set. With the help of (6c) we ensure that the variables are binary.

Many families of valid inequalities are known for (SSP). More details may be found in [Borndörfer, 1998, Canovas et al., 2000,

Later on, we will pay special attention to one of the families of valid inequalities, *clique inequalities*. Their significance stems from the fact that – in some cases – they are sufficient for a complete description of STAB($G$) (cf. [Padberg, 1973]). These valid inequalities will be discussed in more detail in the paragraph pertaining to Set Packing Problem.

We will also now refer to a few different formulations of (SSP). Some of the alternative formulations replace constraint (6b) with different constraints (cf. [Della Croce and Tadei, 1994, Murray and Church, 1997]). Other formulations use additional variables (e.g. [Balas et al., 1996, Burer and Vandenbussche, 2006, Lovász and Schrijver, 1991]) There are also some formulations which may be improved using semi-definite programming ([Dukanovic and Rendl, 2007, Giandomenico and Letchford, 2006, Grötschel et al., 1993]. Finally, some pose the Stable Set Problem as a non-linear optimization problem with continuous variables ([Gruber and Rendl, 2003, Busygin, 2006, Motzkin and Straus, 1965]).

**Maximum Clique Problem** The maximum clique problem (MCP) consists in finding a clique of maximal cardinality in a graph. [Karp, 1972] lists its decision version as one of the first 21 NP-complete problems. It has attracted a lot of attention ever since.

The “classical” formulation of (MCP), as discussed by [Wu and Hao, 2015], is given by model (7). For each node $v \in V$ a binary variable $x_v$ is defined, which assumes a value 1 if node $v$ is selected to be included in the clique and 0 otherwise.

\[
\begin{align*}
\text{max} & \quad \sum_{v \in V} x_v & (7a) \\
\text{s.t.} & \quad x_i + x_j \leq 1 & \forall \{i,j\} \in E' & (7b) \\
& \quad x_i \in \{0, 1\} & \forall i \in V. & (7c)
\end{align*}
\]

The objective function (7) maximizes the number of vertices entering the clique. Constraint (7b) ensures that no pairwise non-adjacent vertices are selected for the clique. With the help of (7c) we ensure that the variables are binary.
There are also other formulations of \((MCP)\). [Bomze et al., 1999], [Butenko, 2003] and [Pardalos and Xue, 1994] provide a good overview in this matter. Some of the works which discuss the polyhedral aspects of \((MCP)\) are [Balas et al., 1996, Pardalos and Xue, 1994, Sørensen, 2004].

The relationship between \((SSP)\) and \((MCP)\) is based on the fact that a set \(K \subseteq V\) is a maximum clique of a graph \(G\) if and only if \(K\) is also a stable set of maximal cardinality in \(G'\). This equivalence is illustrated in Figure 13. Such a close relationship means that all the characteristics of \((SSP)\) are also valid for \((MCP)\) and vice-versa. Moreover, a result that is true for \((SSP)\) in \(G\) will also hold for \((MCP)\) in \(G'\) and vice versa.

\[\begin{align*}
\text{Set Packing Problem} & \quad \text{Suppose one has a finite set } S \text{ and a family of its subsets } \mathcal{P}(S). \text{ The set packing problem consists in finding } k \text{ subsets in } \mathcal{P}(S) \text{ which are pairwise disjoint. The optimization version of this problem seeks the maximum number of pairwise disjoint subsets in } \mathcal{P}(S). \text{ This problem has been studied for many decades (see for example [Paschos, 1997]). A simple formulation of the problem is given by model } (8). \text{ For each subset } S \in \mathcal{P}(S) \text{ a binary variable } x_S \text{ is defined, which assumes a value 1 if a subset } S \text{ is selected to be included in the set packing and 0 otherwise.}
\end{align*}\]

\[\begin{align*}
\max \; & \sum_{S \in \mathcal{P}(S)} x_S \\
\text{s.t.} \; & \sum_{S \in \mathcal{P}(S) : i \in S} x_S \leq 1 \quad \forall i \in S \\
& x_S \in \{0, 1\} \quad \forall S \in \mathcal{P}(S).
\end{align*}\]
The objective function (8a) seeks to maximize the number of sets entering the set packing. Constraint (8b) ensures that no two sets which share at least one element are selected for the set packing. With the help of (8c) we ensure that the variables are binary.

(SPP) is known to be strongly NP-Hard (cf. [Garey and Johnson, 1979]). The works of [Padberg, 1973] and [Balas and Padberg, 1976] contributed greatly to the understanding of polyhedral properties of (SPP). These results are frequently used in branch-and-cut schemes to solve (SPP) to optimality (cf. [Nemhauser and Wolsey, 1999]). For an extensive treatment of the problem, we refer the reader to [Grötschel et al., 1993] and [Borndörfer, 1998].

(SSP) can be reduced to (SPP) and the other way round one-to-one and in polynomial time:

• (SPP) → (SSP): Given a (SPP) on a collection of subsets \( \mathcal{P}(S) \) of a set \( S \). We build a graph \( G = (V, E) \) such that for each \( S \in \mathcal{P}(S) \), we define a vertex \( v_S \in V \) and \( V := \bigcup_{S \in \mathcal{P}(S)} \{v_S\} \). Additionally, an edge \( \{v_S, v_T\} \in E \) exists if and only if \( S, T \in \mathcal{P}(S) \) and \( S \cap T \neq \emptyset \). Then a stable set in \( G \) corresponds to a set packing on \( \mathcal{P}(S) \).

• (SSP) → (SPP): Given a (SSP) on a graph \( G = (V, E) \). We now build a set \( E := G \) and a collection of its subsets \( \mathcal{P}(S) := \bigcup_{v \in V} \{S_v\} \) with \( S_v \) containing all edges adjacent to \( v \) (defined for all \( v \in V \)). Then, a set packing on \( \mathcal{P}(S) \) corresponds to a stable set in \( G \).

**Polyhedral aspects of the Set Packing Problem**  Basing on [Padberg, 1973] and [Balas and Padberg, 1976], we will now discuss some of the polyhedral characteristics of (SPP), which will be of use later in the next chapters.

We will consider (SPP) in its standard form. For that end, let us define \( A \) as an \( m \times n \) matrix of zeros and ones (originating from model (8)), \( c \) as an arbitrary \( n \)-vector and \( 1 \) as an \( m \)-vector of ones. Then, the Set Packing Problem can be defined as:

\[
\begin{align*}
\text{max} \quad & c^T x \\
\text{s.t.} \quad & Ax \leq 1 \quad (9a) \\
& x_j \in \{0, 1\} \quad \text{for all } j \in \{1, \ldots, n\}. \quad (9c)
\end{align*}
\]

Let us also define the set of fractional solutions to (9) as:

\[
P = \{x \in \mathbb{R}^n : Ax \leq 1, x \geq 0\} \quad (10)
\]
Let $P_i$ be the convex hull of integral points of $P$, i.e.
\[ P_i = \text{conv}(\{x \in P : x \in \mathbb{Z}^n\}). \] (11)

With each problem formulated as (9) we can associate the intersection graph $\mathcal{G} = (N, E)$, with a node $j \in N$ for every column of $A$ and $(i, j) \in E$ if and only if $a^i a^j \geq 1$ i.e. columns $i$ and $j$ have at least one $+1$ entry in common.

The convention of naming $\mathcal{G}$ an intersection graph can also be found in [Edmonds, 1962] and [Harary, 2001].

Based on this graph, we can now represent problem (9) as:

\[
\begin{align*}
\text{max} & \quad c x \\
\text{s.t.} & \quad A_{\mathcal{G}} x \leq 1_{\mathcal{G}} \\
& \quad x_j \in \{0, 1\} \quad \text{for all } j \in \{1, \ldots, n\},
\end{align*}
\] (12a)

with $A_{\mathcal{G}}$ being the incidence matrix of $\mathcal{G}$, and $1_{\mathcal{G}}$ being a vector of ones dimensioned compatibly with $A_{\mathcal{G}}$.

Note that every feasible solution to (12) is a feasible solution to (9) and vice-versa, and in particular, every optimal solution to (12) is an optimal solution to (9) and vice-versa.

Let us now define $\mathcal{K}(\mathcal{G})$ as the set of all maximal cliques in the graph $\mathcal{G}$.

We can introduce a class of facet-defining inequalities, derived by [Padberg, 1973]:

**Theorem 5.2.1.** ([Padberg, 1973]) An inequality
\[ \sum_{j \in K} x_j \leq 1 \] (13)

is a facet of $P_i$ if and only if $K$ is the node set of a maximal clique in $\mathcal{G}$, i.e. $K \in \mathcal{K}(\mathcal{G})$.

The following proof is also due to [Padberg, 1973].
Proof. ([Padberg, 1973]) As $K$ is the node set of a clique, we see that $(i, j) \in E$ for all $i, j \in K$. Hence, the inequality (13) holds for all $x \in P_t$.

We also notice that at least $n$ linearly independent zero-one solutions to $P_t$ satisfying (13) with equality exist. This can be seen from constructing $|K|$ solutions which set $x_j = 1$ for exactly one $j \in K$ at a time and $x_j = 0$ otherwise. Now, since $K$ is the node set of a maximal clique, for every $k \in N \setminus K$, at least one node $j \in K$ such that $(k, j) \notin E_{\text{int}}$ must exist. Then, $x_k = 1, x_j = 1, x_i = 0$ otherwise yields a feasible solution to $P_G$ which satisfies (13) with equality. If we do so for every $k \in N \setminus K$, we obtain $|N \setminus K|$ feasible solutions to $P_t$ which satisfy (13) with equality. Now, the matrix made up of the corresponding $n$ solutions is triangular (modulo a permutation of its rows and columns) with determinant equal to $\pm 1$.

It remains to prove the “only if” part of the Theorem 5.2.1. Assume w.l.o.g. that $K$ could be extended by a single node $j \in N$ to a maximal clique. Then

$$\sum_{k \in K \cup \{j\}} x_k \leq 1 \quad (14)$$

is satisfied with equality by all feasible solutions to $P_G$ satisfying (13) with equality and at least one solution more, namely $x_j = 1, x_i = 0$ otherwise. Hence, it is affinely independent to the other feasible solutions to $P_G$ and in this way, (14) would be a proper face with dimension $n$, which is a contradiction.

Let us now define $A_C$ as the clique matrix of the intersection graph $\mathcal{G}$, with columns corresponding to nodes and rows to all maximal cliques of $\mathcal{G}$. Additionally let $1_C$ be a vector of ones dimensioned compatibly with $A_C$. We can now state the corresponding problem:

$$\begin{align*}
\max & \quad cx \\
\text{s.t.} & \quad A_C x \leq 1_C \\
& \quad x_j \in \{0, 1\} \quad \text{for all } j \in \{1, \ldots, n\},
\end{align*}$$

and the set of its fractional solutions:

$$P_C = \{x \in \mathbb{R}^n : A_C x \leq 1_C, x \geq 0\}. \quad (16)$$
As a consequence of Theorem 5.2.1, we can state that $P_l \subseteq P_c \subseteq P$. Although $P_c$ is usually different from both $P_l$ and $P$, a class of matrices $A$ exists for which the three matrices coincide i.e. $P = P_c = P_l$. Such matrices are called perfect. [Balas and Padberg, 1976] showed that clique-matrices of perfect intersection graphs are all perfect.

Even if the intersection graph $\mathcal{G}$ is not perfect, we can still exploit the fact that $P_c$ is at least as tight as $P$, and – in many cases – such a reformulation allows for a better description of the underlying polyhedron and hence a faster solution process.
6  Formulations of the problem and some polyhedral results

In this chapter, we will present two formulations for $(G_2LC-CC)$. Later we will show how they can be reformulated into Set Packing Problems. For both formulations, we will present a formulation strengthening technique – clique tightening. We will also consider a special case of $(G_2LC-CC)$ on chordal graphs with a special property, which we call one-colour-chord property. We will see that the resulting intersection graph is perfect and hence the polyhedron induced by the corresponding model has integer vertices. This chapter corresponds to Chapter 2 in the Part I.

6.1  Formulations of the problem

In this section, we will present two formulations for the problem we consider. Both of them will take the form of a Set Packing Problem, with one having additional compatibility constraints.

6.1.1  One-variable formulation

Let us now introduce the first formulation. It has one family of binary variables $a_{c,d}^v \in \{0,1\}$ which assume 1 if both $c \in C^v$ and $d \in D^v$ are assigned to node $v \in V$ and 0 otherwise. The formulation reads as follows:

$$\text{max} \quad \sum_{v \in V} \sum_{c \in C^v} \sum_{d \in D^v \cap D_c} a_{c,d}^v$$

$$\text{s.t.} \quad \sum_{d \in D^v \cap D_c} a_{c,d}^{v_1} + \sum_{d \in D^v \cap D_c} a_{c,d}^{v_2} \leq 1 \quad \forall \{v_1, v_2\} \in E, \forall c \in C^{v_1} \cap C^{v_2}$$

$$\sum_{c \in C^v \cap C_d} a_{c,d}^{v_1} + \sum_{c \in C^v \cap C_d} a_{c,d}^{v_2} \leq 1 \quad \forall \{v_1, v_2\} \in E, \forall d \in D^{v_1} \cap D^{v_2}$$

$$\sum_{c \in C^v} \sum_{d \in D^v \cap D_c} a_{c,d}^v \leq 1 \quad \forall v \in V$$

$$a_{c,d}^v \in \{0,1\} \quad \forall v \in V, \forall c \in C^v, \forall d \in D^v \cap D_c.$$
This formulation has $O(|V| \times (|C| + |D|))$ variables and $O(|V| + |E|)$ constraints.

### 6.1.2 Two-variable formulation

The second formulation of $(G2LC-CC)$ that we consider is much closer related to the integrated locomotive scheduling and driver assignment model (I), presented in Part I of this thesis. It similarly uses two families of binary variables. Firstly, we define a variable $x_c^v \in \{0, 1\}$ for all $v \in V$ and $c \in C^v$ to denote whether a $c$-colour $c$ is selected for a node $v$ or not. Secondly, with a variable $y_d^v \in \{0, 1\}$, introduced for all $v \in V$ and $d \in D^v$ we decide if a $d$-colour $d$ is selected for a node $v$ or not. The formulation reads as follows:

\[
\begin{align*}
\text{max} & \sum_{v \in V} \sum_{c \in C^v} x_c^v \\
\text{s.t.} & \quad x_c^v \leq \sum_{d \in D^v \cap C^v} y_d^v \quad \forall v \in V, \forall c \in C^v \\
& \quad y_d^v \leq \sum_{c \in C^v \cap C_d} x_c^v \quad \forall v \in V, \forall d \in D^v \\
& \quad x_c^{v_1} + x_c^{v_2} \leq 1 \quad \forall \{v_1, v_2\} \in E, \forall c \in C^{v_1} \cap C^{v_2} \\
& \quad y_d^{v_1} + y_d^{v_2} \leq 1 \quad \forall \{v_1, v_2\} \in E, \forall d \in D^{v_1} \cap D^{v_2} \\
& \quad \sum_{c \in C^v} x_c^v \leq 1 \quad \forall v \in V, \forall c \in C^v \\
& \quad \sum_{d \in D^v} y_d^v \leq 1 \quad \forall v \in V, \forall d \in D^v \\
& \quad x_c^v \in \{0, 1\} \quad \forall v \in V, \forall c \in C^v \\
& \quad y_d^v \in \{0, 1\} \quad \forall v \in V, \forall d \in D^v.
\end{align*}
\]

This formulation has $O(|V| \times (|C| + |D|))$ variables and $O(|V| + |E|)$ constraints.

### 6.2 Improvements to the formulations

In this section, we will apply the techniques described in Section 5.2.2 (i.e. clique tightening) to the two formulations introduced in the previous section. We will see that while for one-variable formulation such a clique tightening is straightforward, it does require a little more effort for the two-variable formulation. In particular, we will present a decomposition scheme which will
allow the clique tightening to be performed on the two-variable formulation as well. Additionally, we will present a valid equation class for that formulation.

### 6.2.1 Clique tightening of one-variable formulation

First of all, we consider the Model (17). Recall that in Section 5.2.2 we introduced the concept of an intersection graph. In case of the one-variable formulation, it takes the form of

\[ G = (\mathcal{U}_\text{int}, \mathcal{W}_\text{int}) \]

where

\[ \mathcal{U}_\text{int} := \{(u, v, d) : u \in U \land v \in V \land d \in D^v \cap D_c\} \]

and

\[ \mathcal{W}_\text{int} := \{(v_1, c, d_1), (v_2, c, d_2) : (v_1, v_2) \in E \land c \in C^{v_1} \cap C^{v_2} \land d_1 \in D^{v_1} \cap D_c \land d_2 \in D^{v_2} \cap D_c\} \]

Based on this intersection graph, we may present the model (17) as a classical Set Packing model:

\[
\begin{align*}
\text{max} & \quad \sum_{v \in V} \sum_{c \in C_v} \sum_{d \in D^v \cap D_c} a_{c,d}^v \quad \text{(19a)} \\
\text{s.t.} & \quad a_{c_1,d_1}^{v_1} + a_{c_2,d_2}^{v_2} \leq 1 \quad \forall\{(v_1, c_1, d_1), (v_2, c_2, d_2)\} \in \mathcal{W}_\text{int} \quad \text{(19b)} \\
& \qquad a_{c,d}^v \in \{0, 1\} \quad \forall v \in V, \forall c \in C_v, \forall d \in D^v \cap D_c. \quad \text{(19c)}
\end{align*}
\]

Further, as discussed in Section 5.2.2, to strengthen the model (19), we can generate the set of maximal cliques \( \mathcal{K}(G) \subset \mathcal{P}(\mathcal{U}_\text{int}) \) of the intersection set \( G \). Then, model (19) may be reformulated as follows:
6 Formulations of the problem and some polyhedral results

\[
\begin{align*}
\text{max} & \quad \sum_{v \in V} \sum_{c \in C^v} \sum_{d \in D^v \cap D_c} a^v_{c,d} \\
\text{s.t.} & \quad \sum_{(v,c,d) \in K} a^v_{c,d} \leq 1 \quad \forall K \in \mathcal{K}(G) \\
& \quad a^v_{c,d} \in \{0,1\} \quad \forall v \in V, \forall c \in C^v, \forall d \in D^v \cap D_c.
\end{align*}
\]

### 6.2.2 Clique tightening of two-variable formulation and a valid equation class

Let us now turn our attention to the two-variable formulation (18), introduced in Subsection 6.1.2. In this case, we are unable to apply the clique tightening described in Section 5.2.2 directly. This is due to the coupling constraints (18b) and (18c). But if these two constraint were relaxed, we would deal with two decoupled set packing problems, one given by constraints (18d), (18f) and (18h), and the other defined by constraints (18e), (18g) and (18i). This – in turn – allows us to seek ways to strengthen the formulation of the two problems.

In the following we will present a way to decompose the model (18). We will also present a valid equation class for model (18).

**Clique tightening** Let us now define two intersection graphs. First of all, we define \( G_C := (V_{int}^C, E_{int}^C) \) with

\[
V_{int}^C := \{(v, c) : v \in V \land c \in C^v\},
\]

\[
E_{int}^C := \{(v_1, c_1), (v_2, c_2) : (v_1, v_2) \in E \land c \in C^{v_1} \cap C^{v_2}\}
\]

\[
\cup \{(v, c_1), (v, c_2) : v \in V \land c_1, c_2 \in C^v\}.
\]

Further, we define \( G_D := (V_{int}^D, E_{int}^D) \) as:

\[
V_{int}^D := \{(v, d) : v \in V \land d \in D^v\},
\]

\[
E_{int}^D := \{(v_1, d_1), (v_2, d_2) : (v_1, v_2) \in E \land d \in D^{v_1} \cap D^{v_2}\}
\]

\[
\cup \{(v, d_1), (v, d_2) : v \in V \land d_1, d_2 \in D^v\}.
\]
6.2 Improvements to the formulations

Based on these graphs, we define the respective sets of maximal cliques \( \mathcal{K}(G_c) \subseteq \mathcal{P}(V_{\text{int}}^c) \) and \( \mathcal{K}(G_d) \subseteq \mathcal{P}(V_{\text{int}}^d) \).

As shown in Section 5.2.2, we may reformulate the constraints (18d) and (18f) as

\[
\sum_{(v,c) \in K} x_c^v \leq 1 \quad \forall K \in \mathcal{K}(G_c). \tag{21}
\]

The constraints (18e) and (18g) could be presented as

\[
\sum_{(v,d) \in K} y_d^v \leq 1 \quad \forall K \in \mathcal{K}(G_d). \tag{22}
\]

All in all, the polyhedron described by the clique inequalities is at least as tight as the one described by conflict inequalities resulting from individual edges of the respective intersection graphs. Hence, we will replace constraints (18d) and (18f) with constraint (21). We will also use constraint (22) instead of constraints (18e) and (18g). In the next chapter, we will also explore a decomposition scheme suggested here, consisting in relaxing the constraints (18b) and (18c) and solving the two resulting subproblems sequentially, while ensuring that the solutions are globally feasible.

Valid equation class  We define an equation family which is valid for (18):

\[
\sum_{c \in C^v} x_c^v - \sum_{d \in D^v} y_d^v = 0 \quad \forall v \in V. \tag{23}
\]

**Theorem 6.2.1.** Equation (23) is valid for model (18)

**Proof.** Assume that a feasible solution \((x, y)\) to model (18) which violates equation (23) exists.

Then, for some \(v \in V\), either of the following is true:

1. \(\sum_{c \in C^v} x_c^v > \sum_{d \in D^v} y_d^v\)
2. \(\sum_{c \in C^v} x_c^v < \sum_{d \in D^v} y_d^v\)

Consider case 1. By constraints (18f) and (18g) we know that the value of both \(\sum_{c \in C^v} x_c^v\) and \(\sum_{d \in D^v} y_d^v\) can be at most 1.
Moreover, since both \( x \) and \( y \) variables are binary, the smallest possible value for both \( \sum_{c \in E^v} x_c^v \) and \( \sum_{d \in D^v} y_d^v \) is 0.

Hence, since \( \sum_{c \in E^v} x_c^v > \sum_{d \in D^v} y_d^v \), it follows that \( \sum_{c \in E^v} x_c^v = 1 \) and \( \sum_{d \in D^v} y_d^v = 0 \).

But then, substituting to the constraint (18b), it follows that:

\[
1 = \sum_{c \in E^v} x_c^v \leq \sum_{c \in E^v} \sum_{d \in D^v \cap D_c} y_d^v \leq \sum_{d \in D^v} y_d^v = 0
\]

which is a contradiction. Hence, the solution \((x, y)\) is not a feasible solution. For point 2, an analogous proof can be carried out.

6.3 A special case of \((G2LC-CC)\) on a chordal graph with a special property

In this section we consider a special case of \((G2LC-CC)\), for which we will be able to come up with a complete description of the underlying polyhedron. We will study the case of \((G2LC-CC)\) on chordal graphs that possess an additional property which we call one-colour-chord property. We will show that – if it holds – the corresponding intersection graph is perfect, and we can fully describe the underlying polyhedron (cf. Section 5.2.2). The practical relevance of this property could be related to some of the graphs which only have a few colours available for many nodes.

Throughout this section, we assume that \( G = (V, E) \) is a chordal graph. The intersection graph of the \((G2LC-CC)\) takes the form of \( \mathcal{G}_{\text{int}}(G, C) = (V_{\text{int}}, E_{\text{int}}) \), with

\[
V_{\text{int}} := \{(v, c) : v \in V \land c \in C^v\}
\]

and

\[
E_{\text{int}} := \{(v_1, c), (v_2, c) : (v_1, v_2) \in E \land c \in C^{v_1} \cap C^{v_2}\}
\]

\[
\cup \{(v, c_1), (v, c_2) : v \in V \land c_1, c_2 \in C^v\}
\]

The perfection of \( \mathcal{G}_{\text{int}}(G, C) \) will be assured by the following property:

**Definition 6.3.1.** An intersection graph \( \mathcal{G}_{\text{int}}(G, C) \) underlying a graph-list colouring problem on a chordal graph \( G \) for a colour set \( C \) is said to possess one-colour-chord property, if, for each cycle of cardinality greater than or
6.3 A special case of (G2LC-CC) on a chordal graph with a special property

equal to 4 in G, the nodes connected by a chord \((v_1, v_2)\) have identical colour compatibility sets, i.e. \(C^{v_1} = C^{v_2}\). Moreover, \(|C^{v_1}| = |C^{v_2}| = 1\).

We will now show that the intersection graph \(\mathcal{G}_{\text{int}}(G, C)\) defined above is perfect.

For the purpose of the proof, let us re-define \(\mathcal{G}_{\text{int}}(G, C)' = (\mathcal{V}_{\text{int}}, E_{\text{int}}')\) as the complement of the graph \(\mathcal{G}_{\text{int}}(G, C)\). For completeness, we define:

\[
E_{\text{int}}' = \{((v_1, c), (v_2, c)) : (v_1, v_2) \notin E \land v_1 \neq v_2 \land c \in C^{v_1} \cap C^{v_2} \}
\]

\[
\bigcup \{((v_1, c_1), (v_2, c_2)) : v_1 \neq v_2 \land c_1 \neq c_2 \}
\]

For each subset \(H \subseteq \mathcal{V}_{\text{int}}\), we define the following sets:

\[
V(H) := \{v \in V : \exists (v, c) \in H \text{ for some } c \in C^v\},
\]

\[
E_{\text{int}}(H) := \{((v_1, c_1), (v_2, c_2)) \in E_{\text{int}} : (v_1, c_1), (v_2, c_2) \in H\},
\]

\[
E_{\text{int}}'(H) := \{((v_1, c_1), (v_2, c_2)) \in E_{\text{int}}' : (v_1, c_1), (v_2, c_2) \in H\}.
\]

Additionally, for each subset \(H \subseteq \mathcal{V}_{\text{int}}\) and for each colour \(c \in C\), we define

\[
V^c(H) := \{v \in V : (v, c) \in H\},
\]

and for all \(H \subseteq \mathcal{V}_{\text{int}}\) and for each node \(v \in V\), we define

\[
N_{\text{int}}^V(H) := \{(v_1, c_1) \in H : v = v_1\}.
\]

Finally, for each subset \(H \subseteq \mathcal{V}_{\text{int}}\) we let:

\[
C(H) := \{c \in C : \exists (v, c) \in H \text{ for some } v \in V^c\}.
\]

**Theorem 6.3.2.** Let \(G\) be a chordal graph, and \(C\) a set of feasible colours for each node of \(G\). Further, let \(\mathcal{G}_{\text{int}}(G, C)\) possess one-colour-chord property. Then, \(\mathcal{G}_{\text{int}}(G, C)\) is perfect.

**Proof.** Let us assume the opposite case that the graph \(\mathcal{G}_{\text{int}}(G, C)\) is not perfect. By the Strong Perfect Graph Theorem ([Chudnovsky et al., 2006]), a perfect graph is also a Berge graph. Hence it does not induce an odd hole or an odd antihole. As we assume that \(\mathcal{G}_{\text{int}}(G, C)\) is not perfect, we are able to find such
a subset of nodes $H \subseteq V_{\text{int}}$ which induces an odd hole, or a subset of nodes $H' \subseteq V_{\text{int}}$ which induces an odd anti-hole in $\mathcal{G}_{\text{int}}(G, C)$.

Let us deal with finding a hole $H$ in $\mathcal{G}_{\text{int}}(G, C)$.

First of all, we consider the case of $|V(H)| = |H|$. Then, by construction of $\mathcal{G}_{\text{int}}(G, C), C(H) = \{c\}$, or $|C(H)| = 1$. But then recall that – by construction of $\mathcal{G}_{\text{int}}(G, C) - ((v_1, c), (v_2, c)) \in E_{\text{int}}(H)$ if and only if $(v_1, v_2) \in E$. This means that $V(H)$ induces a hole in $G$. As $G$ is a chordal graph and hence perfect, it does not contain odd holes, and we have a contradiction. Such a case is illustrated in Figure 14.

Before moving ahead, let us now consider a case of an odd-hole inducing set $H$ such that $|N^v_{\text{int}}(H)| \geq 3$ holds for some one $v \in V(H)$. Assume w.l.o.g. that $n$ is as integer such that $n \geq 3$ and $\{(v, c_1), (v, c_2), \ldots, (v, c_n)\} \subseteq H$ for some $v \in V(H)$. By construction of $\mathcal{G}_{\text{int}}(G, C)$, the nodes $(v, c_1), (v, c_2), \ldots, (v, c_n)$ are pairwise connected by edges, one of which, in turn, induces a chord in the odd hole induced by $H$. Since holes contain no chords, we have a contradiction. For the upcoming cases, we can assume w.l.o.g. that $0 < |N^v_{\text{int}}(H)| \leq 2$ for each $v \in V(H)$. Such a situation is illustrated in Figure 15.

Now, consider the case of $|V(H)| \in (1, \lceil \frac{|H|}{2} \rceil)$, with $|V(H)| \in \mathbb{Z}$. But then for at least one vertex $v \in V(H), |N^v_{\text{int}}(H)| \geq 3$ would hold and a chord would be induced (cf. previous paragraph). Hence we can state that node sets $H$ such that $|V(H)| \in (1, \lceil \frac{|H|}{2} \rceil)$ holds may not induce an odd hole. Moreover, we can also assume w.l.o.g. that $|H| > |V(H)| \geq \lceil \frac{|H|}{2} \rceil + 1$.

In light of the remark above, the case we should now consider is $|V(H)| \geq \lceil \frac{|H|}{2} \rceil + 1$. By definition, the smallest possible value for $|H|$ is 5, and so we start with the case of $|V(H)| = 3$. Assume $V(H) = \{v_1, v_2, v_3\}$. Since we
6.3 A special case of (G2LC-CC) on a chordal graph with a special property

Figure 15: Example of a hole $H$ in $\mathcal{G}_{\text{int}}(G, C)$ with $|N^v_{\text{int}}(H)| \geq 3$ for one $v \in H$

showed that $0 < |N^v_{\text{int}}(H)| \leq 2$ for all $v \in V$, we can state w.l.o.g. that $|N^v_1(H)| = 2, |N^v_2(H)| = 2, |N^v_3(H)| = 1$. Now, let $N^v_1(H) = \{(v_3, c_1)\}$. Since $|N^v_3(H)| = 1$, $(v_3, c_1)$ must – by construction – be connected to some $(v_1, c_1) \in N^v_1(H)$ and $(v_2, c_1) \in N^v_2(H)$. It follows that $\{(v_1, v_3), (v_2, v_3)\} \subseteq E$. Moreover, by construction of $\mathcal{G}_{\text{int}}(G, C)$, $(v_1, c_1)$ must be connected to some $(v_1, c_2)$. Now, the only node which can connect both $(v_1, c_2)$ and $(v_2, c_1)$ to “close the cycle” is $(v_2, c_2)$. It follows that $(v_1, v_2) \in E$. But this also means that $((v_1, c_1), (v_2, c_1)) \in E_{\text{int}}$, and the cycle induced by $H$ has a chord, and hence is not a hole. This contradicts our assumption. For an example, cf. Figure 16.

Figure 16: Example of a hole $H$ in $\mathcal{G}_{\text{int}}(G, C)$ with $|V(H)| = 3$

Now, let us consider the case of $4 \leq |V(H)| < |H|$. For the ease of notation, let $|V(H)| = n$. We can also state that $V(H) = \{v_1, v_2, ..., v_n\}$. Similarly to previous case, w.l.o.g., let $0 < |N^v_1(H)| \leq 2, 0 < |N^v_2(H)| \leq 2, ..., |N^v_n(H)| = 1$, and $N^v_n(H) = \{(v_n, c_1)\}$. It is also easy to see that in the considered case, $|C(H)| \geq 2$. Now – by construction of $\mathcal{G}_{\text{int}}(G, C)$ – we can w.l.o.g. state that for
the node \((v_n, c_1)\), the neighbouring nodes are \((v_1, c_1)\) and \((v_{n-1}, c_1)\). Then the remaining nodes in \(H\) form a path graph \(P\) of length no smaller than 2. Moreover such that the two terminal vertices of \(P\), \((v_{t_1}, c_{t_1})\) and \((v_{t_2}, c_{t_2})\) are connected to \((v_1, c_1)\) and \((v_{n-1}, c_1)\) respectively. By definition, \(P\) contains a path between \((v_{t_1}, c_{t_1})\) and \((v_{t_2}, c_{t_2})\). But then it follows from the construction of \(\mathcal{R}_{\text{int}}(G, C)\) that the subgraph of \(G\) induced by \(V(P)\) also contains a path between \(v_{t_1}\) and \(v_{t_2}\). This also means that \(V(H) = V(P) \cup \{v_1, v_{n-1}, v_n\}\) induces a cycle of length greater than or equal to 4 in \(G\). By definition of \(G\), all such cycles induce a chord. Hence, we can w.l.o.g. assume that for some two nonconsecutive nodes \(v_{t_1}, v_{t_2} \in V(H)\), \((v_{t_1}, v_{t_2}) \in E\). By one-colour-chord property we know that the nodes connected by a chord have identical sets of compatible colours, whose cardinality is 1. Assume \(C^{v_k} = C^{v_n} = \{c\}\). We can then state that \(((v_k, c), (v_n, c)) \in E_{\text{int}}\) and the cycle induced by \(H\) in \(\mathcal{R}_{\text{int}}(G, C)\) is not a hole. This contradicts our assumption. An example for such a case is presented in Figure 17.

![Figure 17: Example of a hole \(H\) in \(\mathcal{R}_{\text{int}}(G, C)\) with \(4 \leq |V(H)| < |H|\)](image)

Finally, recall that the case of \(|V(H)| = |H|\) has been dealt with in the beginning of the proof.

Now, consider a case of \(|V(H')| = 2\). Since the length of the hole induced by \(H'\) in \(\mathcal{R}_{\text{int}}(G, C)'\) is odd, at least one edge in \(E_{\text{int}}'(H')\) will assume the form \(((v, c_1), (v, c_2))\) for some \(v \in H'\) and \(c_1, c_2 \in C^v\). But by construction of \(\mathcal{R}_{\text{int}}(G, C)'\), such edges don't exist and we have a contradiction. An example of such a case may be found in Figure 18.
6.3 A special case of $(G_{2LC-CC})$ on a chordal graph with a special property

From this, we derive that for the odd holes $H'$ in $\mathcal{G}_{\text{int}}(G, C)'$ we consider, $|V(H')| \geq 3$. Let us now consider such odd holes, looking at them from the angle of various cardinalities of the associated sets $C(H')$.

In case $|C(H')| = 1$, each of the nodes in $H'$ points to the same colour in $C$. We can also infer that each node in $H'$ refers to a different node in $V$. Suppose now that $|H'| = n$, where $n$ is an odd positive integer greater than or equal to five. We can then assume w.l.o.g. that $V(H') = \{v_1, v_2, ..., v_{n-1}, v_n\}$, and that $E'_{\text{int}}(H') = \{(v_1, v_2), ..., (v_{n-1}, v_n), (v_n, v_1)\}$. In terms of the original graph $G$, it follows that each node in $V(H')$ is connected to all the other nodes but its neighbours in the cycle. But this means that $H'$ induces an odd anti-hole in the original graph $G$, and we have a contradiction.

If $|H'| > |C(H')| \geq 2$, we can trivially find such two nodes $(v_1, c_1), (v_2, c_2) \in H'$, such that $v_1 \neq v_2$ and $c_1 \neq c_2$ and $((v_1, c_1), (v_2, c_2)) \notin E_{\text{int}}(H)$. In other words, $(v_1, c_1), (v_2, c_2)$ are non-consecutive. By construction of $\mathcal{G}_{\text{int}}(G, C)'$, such two nodes are connected by an edge. This means that they also induce a chord in the cycle $H'$, and we have a contradiction. An example of such a case may be found in Figure 19.

Finally, the case of $|C(H')| = |H'|$ has been dealt with in the paragraph devoted to the situation in which $|V(H')| = 1$ (as these cases are equivalent).
6 Formulations of the problem and some polyhedral results

Figure 19: Example of a hole $H'$ in $\mathcal{G}_{\text{int}}(G, C)'$ with $|H'| > |C(H')| \geq 2$
In this chapter, we will present a solution approach for (G2LC-CC) based on a decomposition scheme mentioned briefly in Subsection 6.2.2. We will also use other findings from the previous chapter. As mentioned earlier, this solution approach is an adjusted version of the algorithm introduced in Chapter 3. Figure 20 below presents a simplified flowchart of the algorithm we introduce in this chapter – the following presents the details of the ingredients of our solution approach.

First, in Section 7.1, the general decomposition scheme will be presented. Then, in Subsection 7.2, we will present a preprocessing scheme, allowing us to remove some of the infeasible solutions upfront. Later, in Subsection 7.3, we will derive a family of inequalities which are valid for two-variable formulation of (G2LC-CC) (i.e. model (18)). In the next step, as discussed in Section 7.4, we will fix the assignment of one variable family in the integrated model, and remove the infeasible assignments in the other variable family. Sometimes, the remainder of the problem is infeasible but we cannot generate one of our problem-specific cutting planes to ensure feasibility of the second subproblem. We then generate a combinatorial Benders cut instead which precisely cuts off the current locomotive assignment from the locomotive master problem and reiterate. This way, the algorithm will eventually converge to a global optimal solution to (G2LC-CC).

![Flowchart of the solution algorithm to (G2LC-CC) introduced in this work](image.png)

**Figure 20**: Flowchart of the solution algorithm to (G2LC-CC) introduced in this work
7 A decomposition-based solution algorithm

7.1 Decomposition scheme

In this section, we will present the decomposition scheme, which underlines the solution approach for \((G2LC-CC)\) which we develop in this part of the thesis.

Let us now revisit model (18), together with the tightened constraints (21) and (22). It comprises (i) constraints which refer to the colour family \(C\) (i.e. (21) and (18h)), (ii) constraints which refer to the colour family \(D\) (i.e. (22) and (18i)), and (iii) compatibility constraints ((18b) and (18c)). Recall also the definitions of the associated intersection graph \(\mathcal{G}_C\) and \(\mathcal{G}_D\) as well as the sets of their maximal cliques \(\mathcal{K}(\mathcal{G}_C)\) and \(\mathcal{K}(\mathcal{G}_D)\) respectively. If the compatibility constraints were relaxed, the model (18) decomposes into two independent subproblems – the \(c\)-subproblem and the \(d\)-subproblem.

The \(c\)-subproblem can be presented as follows:

\[
\begin{align*}
\text{max} & \quad \sum_{v \in V} \sum_{c \in C^v} x_c^v \\
\text{s.t.} & \quad \sum_{(v,c) \in K} x_c^v \leq 1 \quad \forall K \in \mathcal{K}(\mathcal{G}_C) \\
& \quad x_c^v \in \{0, 1\} \quad \forall v \in V, \ \forall c \in C^v.
\end{align*}
\]

To define the corresponding \(d\)-subproblem, we need an objective function, which we define below as

\[
\text{max} \sum_{v \in V} \sum_{d \in D^v} y_d^v.
\]

The \(d\)-subproblem can be presented as follows:

\[
\begin{align*}
\text{max} & \quad \sum_{v \in V} \sum_{d \in D^v} y_d^v \\
\text{s.t.} & \quad \sum_{(v,d) \in K} y_d^v \leq 1 \quad \forall K \in \mathcal{K}(\mathcal{G}_D) \\
& \quad y_d^v \in \{0, 1\} \quad \forall v \in V, \ \forall d \in D^v.
\end{align*}
\]

Now, as we have two separate problems, we may attempt at solving them independently.
7.2 Preprocessing of the c-subproblem and the d-subproblem – exclusion of infeasible assignments upfront

In this section, we will discuss a preprocessing method for both subproblems introduced so far.

Before solving any of the two problems (24) or (25), we may preclude some of their infeasible solutions upfront. This may be achieved by removing variables pertaining such assignments of \( c \)-colour to a node \( v \in V \), for which no suitable \( d \)-colour may be found and the other way round. More formally, for each node \( v \in V \) we define two sets of feasible colours:

\[
\tilde{C}^v := \{ c \in C^v : D^v \cap D_c \neq \emptyset \}, \\
\tilde{D}^v := \{ d \in D^v : C^v \cap C_d \neq \emptyset \}.
\]

The preprocessing procedure consists in removing all the variables which denote assigning a colour \( c \notin \tilde{C}^v \) (resp. \( d \notin \tilde{D}^v \)) to a node \( v \in V \).

**Changing the sense of a subset of the constraints**  Recall the multiple-choice constraints (18g) of the initial \( d \)-subproblem. The solution to the \( c \)-subproblem already determines which nodes shall be assigned a \( c \)-colour, we can change the optimization sense of the \( d \)-subproblem from maximization (of objective function (25a)) into a mere feasibility problem if we change some of the constraints (25b) into equations, in the following way:

\[
\sum_{d \in \tilde{D}^v} y_d^v = 1 \quad \forall v \in V.
\]  

(26)

It can easily be seen that – by construction – for each \( v \in V \) the set \( \tilde{D}^v \) corresponds to one of the sets \( K \in \mathcal{K}(\mathcal{D}) \). This reformulation has proved to be computationally more efficient than the original, maximization version of the \( d \)-subproblem.

7.3 Valid inequality for the c-subproblem

In this section, we will introduce a valid inequality class for \( c \)-subproblem. It will help us ensure that the solutions to the two subproblems are feasible with regard to the integrated model (18).

The \( d \)-subproblem contains one constraint which is a potential source of infeasibility in model (18) when fixing the \( x \)-variables corresponding to a
A decomposition-based solution algorithm

given solution of c-subproblem, namely the conflict constraint (25b). To ensure that the solution to c-subproblem also respects these constraints, we need to derive a special class of valid inequalities. Recall that in Section 5.2.2 and Subsection 6.2.2 we defined the intersection graphs \( \mathcal{G}_C \) and \( \mathcal{G}_D \) as well as the sets of maximal cliques in these graphs \( \mathcal{K}(\mathcal{G}_C) \) and \( \mathcal{K}(\mathcal{G}_D) \).

Based on these sets, we now define

\[
K_C^{\text{valid}} := \{ ((v, c) : v \in K \land c \in C^v) : K \in \mathcal{K}(\mathcal{G}_D) \}.
\]

**Theorem 7.3.1.** The following inequalities are valid for model (18):

\[
\sum_{(v, c) \in K} x^v_c \leq \left| \bigcup_{(v, c) \in K} D^v \cap D_c \right| \quad \forall K \in K_C^{\text{valid}}
\] (27)

**Proof.** Let \((\bar{x}, \bar{y})\) be a feasible solution to model (18). For the purpose of the proof, define a set \( V_K := \{ v : (v, c) \in K \} \) for all \( K \in K_C^{\text{valid}} \). Suppose this solution violates the constraint (27). Then there exists a set \( K \in K_C^{\text{valid}} \) such that

\[
\sum_{(v, c) \in K} x^v_c > \left| \bigcup_{(v, c) \in K} D^v \cap D_c \right| \quad (28)
\]

holds. By its definition, it includes only mutually exclusive vertices. Hence, we need at least \( |V_K| \)-many \( d \)-colours compatible with nodes in \( V_K \) and with the potentially selected \( c \)-colours to ensure that each of the nodes is assigned two colours. But in inequality (28) we supposed that there are fewer compatible \( d \)-colours. Therefore, the considered solution is infeasible, which is a contradiction. \( \square \)

### 7.4 Preprocessing the d-subproblem – solution-based exclusion of infeasible assignments

In this section, we introduce one more preprocessing scheme. It consists of removing some of the variables of the \( d \)-subproblem based on the solution to the \( c \)-subproblem.

As the assignment of \( c \)-colours to the nodes of the graph has already been performed by the \( c \)-subproblem, a part of variables pertaining to the assignment of \( d \)-colours may be eliminated – in particular the ones which pertain to the \( d \)-colours which are not compatible with both a node \( v \in V \) and the \( c \)-colour \( c \in C^v \) selected for the node \( v \).
7.4 Preprocessing of the $d$-subproblem

Let $\bar{x}$ be the solution to the $c$-subproblem. Using that solution, for each node $v \in V$ we may enumerate the subset of $d$-colours which are compatible with both the node $v$ and the colour $c \in C^v$ which was assigned to it. More formally, for all $v \in V$, let us denote the $c$-colour assigned to the node $v$ by:

$$\bar{c}^v := c,$$

where $c$ is the unique $c \in C^v$ with $\bar{x}_c^v = 1$.

Now as we know the selected $c$-colour for each node $v \in V$, we may introduce the corresponding set of feasible $d$-colours as

$$\bar{D}^v := D^v \cap D_{\bar{c}^v}.$$

Based on these sets, we may enumerate the $y$-variables which are compatible both with the node $v \in V$ and the selected $c$-colour. Hence, we may use the solution of the $c$-subproblem to reduce the number of $y$-variables generated. Moreover, since some $y$-variables may be omitted, some of the constraints (18c) will become trivial and hence may be eliminated as well.
8 Computational results

In this chapter, we will test the performance of the algorithm introduced in the previous chapter. For that end, we derive a number of instances for \((G2LC-CC)\). These are based on the popular graph colouring instances from the literature (cf. [Gualandi and Chiarandini, 2013]). In Section 8.1, we discuss the details of our implementation. Then, we devote Section 8.2 to introduce the instances we used in the computational experiments. Their results are presented and discussed in Section 8.3.

8.1 Implementation details

In this section, we will present the details about the implementations we used to conduct our experiments. We will also discuss the details about software and hardware used in our computations.

Similar to Chapter 4, we have run all the experiments presented in this chapter on a compute server with two Intel Xeon E5-2643 v4 processors using all 12 cores and 256 GB of memory. They were made available to us by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). Further, we have used Gurobi 9.5 ([Gurobi, 2021]) to solve the arising binary optimization problems. The models are built and solved via its Python interface. To represent the underlying graph structures, we have used NetworkX package for Python 3.7 ([Hagberg et al., 2008]). To generate maximal cliques, we will use the maximal-clique enumeration algorithm developed by [Bron and Kerbosch, 1973] as adapted by [Tomita et al., 2006] and discussed in [Cazals and Karande, 2008], implemented in NetworkX.

We will run our experiments in three scenarios. In the following, we describe each of them in greater detail.

One-variable formulation  As a first step, we will attempt at solving the one-variable formulation of the problem extended by clique tightening, i.e. problem (17).

We will refer to this scenario as ONE-VAR.

Two-variable formulation  We will also attempt at solving the two-variable formulation of the problem extended by clique tightening, i.e. problem (18), with constraint (21) used instead of constraints (18d) and (18f). Further, constraints (18e) and (18g) will be replaced by constraint (22).
We refer to this implementation as **TWO-VAR**.

**Decomposition**  In the final, most advanced implementation, we use the complete algorithmic scheme shown in Figure 20 on page 105.

This implementation will be called **DECOMP** in the following.

For our experiments, we have set a time limit of 10800 seconds (or three hours). We have applied each solution scenario to each instance three times, to account for the potential variability in the solution times.

### 8.2 Instances Introduction

In this section, we will present the instances used in our computational study.

The computational study was performed on a subset of graph colouring instances, which were appropriately modified to suit the context of our problem. We have gathered the instances from [Gualandi and Chiarandini, 2013]. Each of the instances used in the computational study contained information about the number of nodes, number of edges and the chromatic number (i.e. value of optimal solution to \((GCP)\) over a given graph).

For no instance considered in this computational study a polynomial time algorithm is known, hence the corresponding problems belong to \(NP\) class. Following [Gualandi and Chiarandini, 2013] (who in turn follow [Koch, 2015]), we divide the instances into two classes: \(NP-s\) (vertex colouring problems on such instances may be solved in seconds) and \(NP-m\) (vertex colouring problems on such instances may be solved in minutes).

Further, to adjust the instances to the context studied in this thesis, we generated two colour sets for each graph, both of cardinality \(\chi\). For each node of the graph considered, we also generated two sets of colours compatible with each node, each of cardinality \(70\% \cdot \chi\), choosing colours randomly.

Tables 13 and 14 below summarize the basic information about instances we used for our computational study. They present the name of instance, number of vertices (\(|V|\)) and edges (\(|E|\)) of the associated graphs, as well as their chromatic number (\(\chi\)).

Table 13: \(NP-s\) instances used in the computational study

| Instance | \(|V|\) | \(|E|\) | \(\chi\) |
|----------|--------|--------|--------|
| 1-FullIns_3 | 30     | 100    | 4      |

Continued on next page
### Table 13 – continued from previous page

| Instance          | $|V|$  | $|E|$  | $\chi$ |
|-------------------|------|-------|--------|
| 1-Insertions_4    | 67   | 232   | 5      |
| 2-FullIns_3       | 52   | 201   | 5      |
| 2-Insertions_3    | 37   | 72    | 4      |
| 3-Insertions_3    | 56   | 110   | 4      |
| anna              | 138  | 493   | 11     |
| ash331GPIA        | 662  | 4181  | 4      |
| david             | 87   | 406   | 11     |
| DSJC125.1         | 125  | 736   | 5      |
| DSJR500.1         | 500  | 3555  | 12     |
| fpsol2.i.1        | 496  | 11654 | 65     |
| fpsol2.i.2        | 451  | 8691  | 30     |
| fpsol2.i.3        | 425  | 8688  | 30     |
| games120          | 120  | 638   | 9      |
| homer             | 561  | 1628  | 13     |
| huck              | 74   | 301   | 11     |
| inithx.i.1        | 864  | 18707 | 54     |
| inithx.i.2        | 645  | 13979 | 31     |
| inithx.i.3        | 621  | 13969 | 31     |
| jean              | 80   | 254   | 10     |
| le450_25a         | 450  | 8260  | 25     |
| le450_25b         | 450  | 8263  | 25     |
| le450_5c          | 450  | 9803  | 5      |
| le450_5d          | 450  | 9757  | 5      |
| miles1000         | 128  | 3216  | 42     |
| miles1500         | 128  | 5198  | 73     |
| miles250          | 128  | 387   | 8      |
| miles500          | 128  | 1170  | 20     |
| miles750          | 128  | 2113  | 31     |
| mug88_1           | 88   | 146   | 4      |

Continued on next page
## Computational results

Table 13 – continued from previous page

| Instance     | |V| | |E| | \(\chi\) |
|--------------|---|---|---|---|
| mug88_25     | 88 | 146 | 4 |
| mulsol.i.1   | 197 | 3925 | 49 |
| mulsol.i.2   | 188 | 3885 | 31 |
| mulsol.i.3   | 184 | 3916 | 31 |
| mulsol.i.4   | 185 | 3946 | 31 |
| mulsol.i.5   | 186 | 3973 | 31 |
| myciel3      | 11  | 20  | 4 |
| myciel4      | 23  | 71  | 5 |
| myciel5      | 47  | 236 | 6 |
| qg.order30   | 900 | 26100 | 30 |
| queen5_5     | 25  | 160 | 5 |
| queen6_6     | 36  | 290 | 7 |
| queen7_7     | 49  | 476 | 7 |
| queen8_12    | 96  | 1368 | 12 |
| queen8_8     | 64  | 728 | 9 |
| queen9_9     | 81  | 1056 | 10 |
| r1000.1      | 1000 | 14378 | 20 |
| r125.1       | 125  | 209 | 5 |
| r125.1c      | 125  | 7501 | 46 |
| r125.5       | 125  | 3838 | 36 |
| r250.1       | 250  | 867 | 8 |
| r250.1c      | 250  | 30227 | 64 |
| school1      | 385  | 19095 | 14 |
| will199GPIA  | 701  | 6772 | 7 |
| zeroin.i.1   | 211  | 4100 | 49 |
| zeroin.i.2   | 211  | 3541 | 30 |
| zeroin.i.3   | 206  | 3540 | 30 |
8.3 Results and their interpretation

This section is devoted to the presentation and discussion of the results of the computational study we performed.

Tables 15 and 16 below present the performance of all three solution approaches against the selected instances. Each of them first introduces the name of the instance, then the number of nodes in the associated graph (|\(V|\)) Further, the value of optimal solution is reported (Opt.). Then, the average times in seconds taken by each of the three algorithms are presented (in columns 1-VAR [s], 2-VAR [s] and DECOMP [s] respectively). A hyphen (-) denotes that the optimal solution was not found in the allotted time. A dagger (†) means that one of the three computations did not find an optimal solution in the allotted time and a mean of two computations is presented.

Very importantly, none of the instances required the use of a combinatorial Benders cuts, hence these are not reported in the result tables below.

Table 14: NP-m instances used in the computational study

| Instance          | |\(V|\) | |\(E|\) | \(\chi\) |
|-------------------|------|------|------|------|
| 1-FullIns_4       | 93   | 593  | 5    |
| 2-FullIns_4       | 212  | 1621 | 6    |
| 3-FullIns_3       | 80   | 346  | 6    |
| 4-FullIns_3       | 114  | 541  | 7    |
| 4-Insertions_3    | 79   | 156  | 4    |
| 5-FullIns_3       | 154  | 792  | 8    |
| ash608GPIA        | 1216 | 7844 | 4    |
| ash958GPIA        | 1916 | 12506| 4    |
| le450_15a         | 450  | 8168 | 15   |
| mug100_1          | 100  | 166  | 4    |
| mug100_25         | 100  | 166  | 4    |
| myciel6           | 95   | 755  | 7    |
| qg.order40        | 1600 | 62400| 40   |
| school1_nsh       | 352  | 14612| 14   |
| wapo5a            | 905  | 43081| 50   |
### 8 Computational results

Table 15: Performance of the solution methods against NP-s instances

| Instance       | |V|  | Opt. | 1-VAR [s] | 2-VAR [s] | DECOMP [s] |
|----------------|-------------------|----------|-------------------|----------|----------|-----------|
| 1-FullIns_3    | 30                | 28       | 0.22              | 0.15     | 0.11     |
| 1-Insertions_4 | 67                | 67       | 1.16              | 0.19     | 0.14     |
| 2-FullIns_3    | 52                | 52       | 1.04              | 0.16     | 0.15     |
| 2-Insertions_3 | 37                | 35       | 0.18              | 0.17     | 0.10     |
| 3-Insertions_3 | 56                | 54       | 0.26              | 0.27     | 0.12     |
| anna           | 138               | 138      | 92.83             | 0.60     | 0.46     |
| ash331GPIA     | 662               | 594      | 19.80             | 70.02    | 3.34     |
| david          | 87                | 87       | 166.19            | 0.49     | 0.27     |
| DSJC125.1      | 125               | 125      | 413.77            | 80.97    | 9.70     |
| DSJR500.1      | 500               | 500      | -                 | 16.35    | 3.61     |
| fpsol2.i.1     | 496               | 269      | -                 | 209.23   | 96.54    |
| fpsol2.i.2     | 451               | 363      | -                 | 352.89   | 16.15    |
| fpsol2.i.3     | 425               | 363      | -                 | 368.28   | 16.18    |
| games120       | 120               | 120      | 110.35            | 0.53     | 0.40     |
| homer          | 561               | 556      | 2162.01           | 8.24     | 3.64     |
| huck           | 74                | 74       | 58.26             | 0.29     | 0.21     |
| inithx.i.1     | 864               | 519      | -                 | 886.63   | 105.01   |
| inithx.i.2     | 645               | 558      | -                 | 808.16   | 29.19    |
| inithx.i.3     | 621               | 559      | -                 | 943.06   | 29.20    |
| jean           | 80                | 77       | 46.32             | 0.28     | 0.24     |
| le450_25a      | 450               | 450      | -                 | 1660.77  | 25.35    |
| le450_25b      | 450               | 450      | -                 | 1808.88  | 18.16    |
| le450_5c       | 450               | 450      | -                 | 962.28   | 180.41   |
| le450_5d       | 450               | 227      | -                 | 1587.81  | 78.35    |
| miles1000      | 128               | 128      | -                 | 102.24   | 12.29    |
| miles1500      | 128               | 128      | -                 | 179.85   | 58.10    |
| miles250       | 128               | 125      | 22.87             | 0.39     | 0.28     |
| miles500       | 128               | 128      | -                 | 2.41     | 1.32     |

Continued on next page
### 8.3 Results and their interpretation

#### Table 15 – continued from previous page

| Instance   | $|V|$ | Opt. | 1-VAR [s] | 2-VAR [s] | DECOMP [s] |
|------------|-----|------|-----------|-----------|------------|
| miles750   | 128 | 128  | -         | 11.07     | 4.19       |
| mug88_1    | 88  | 78   | 0.29      | 1.82      | 0.14       |
| mug88_25   | 88  | 78   | 0.30      | 1.40      | 0.13       |
| mulsol.i.1 | 197 | 138  | -         | 26.28     | 15.67      |
| mulsol.i.2 | 188 | 173  | -         | 14.38     | 6.35       |
| mulsol.i.3 | 184 | 174  | -         | 14.43     | 6.40       |
| mulsol.i.4 | 185 | 175  | -         | 14.31     | 6.54       |
| mulsol.i.5 | 186 | 176  | -         | 14.76     | 6.64       |
| myciel3    | 11  | 10   | 0.10      | 0.10      | 0.09       |
| myciel4    | 23  | 23   | 0.38      | 0.12      | 0.10       |
| myciel5    | 47  | 47   | 2.67      | 1737.30   | 0.17       |
| qg.order30 | 900 | 900  | -         | -         | 280.04     |
| queen5_5   | 25  | 25   | 1.93      | 0.19      | 0.13       |
| queen6_6   | 36  | 35   | 198.53    | 1118.50   | 1.12       |
| queen7_7   | 49  | 47   | 216.83    | 47.78     | 0.92       |
| queen8_12  | 96  | 96   | -         | 700.99    | 6.28       |
| queen8_8   | 64  | 64   | 7205.63†  | 81.14     | 10.64      |
| queen9_9   | 81  | 81   | -         | 3375.17†  | 5288.50†   |
| r1000.1    | 1000| 1000 | -         | 4454.87   | 40.14      |
| r125.1     | 125 | 122  | 1.16      | 0.20      | 0.20       |
| r125.1c    | 125 | -    | -         | -         | -          |
| r125.5     | 125 | 125  | -         | 593.47    | 25.07      |
| r250.1     | 250 | 250  | 37.37     | 0.70      | 0.64       |
| r250.1c    | 250 | -    | -         | -         | -          |
| school1    | 385 | -    | -         | -         | -          |
| will199GPIA| 701 | 701  | -         | -         | 16.01      |
| zeroin.i.1 | 211 | 126  | -         | 33.66     | 16.97      |
| zeroin.i.2 | 211 | 157  | -         | 15.91     | 5.78       |
| zeroin.i.3 | 206 | 157  | -         | 14.66     | 5.73       |
8 Computational results

Table 16: Performance of the solution methods against $NP-m$ instances

| Instance          | $|V|$ | Opt. | 1-VAR [s] | 2-VAR [s] | DECOMP [s] |
|-------------------|-----|------|-----------|-----------|------------|
| 1-FullIns_4       | 93  | 93   | 3.38      | 0.34      | 0.25       |
| 2-FullIns_4       | 212 | 212  | 28.38     | 2100.79   | 0.72       |
| 3-FullIns_3       | 80  | 80   | 4.38      | 0.73      | 0.22       |
| 4-FullIns_3       | 114 | 114  | 8.72      | 12.28     | 0.29       |
| 4-Insertions_3    | 79  | 77   | 0.33      | 0.48      | 0.15       |
| 5-FullIns_3       | 154 | 154  | 61.07     | 0.61      | 0.48       |
| ash608GPIA        | 1216| 688  | 84.91     | 227.84    | 11.53      |
| ash958GPIA        | 1916| 1916 | 2970.73   | -         | 32.42      |
| le450_15a         | 450 | -    | -         | -         | -          |
| mug100_1          | 100 | 89   | 0.33      | 2.34      | 0.15       |
| mug100_25         | 100 | 89   | 0.33      | 2.31      | 0.14       |
| myciel6           | 95  | 95   | 14.44     | -         | 0.34       |
| qg.order40        | 1600| 1600 | -         | -         | 2084.32    |
| school1_nsh       | 352 | -    | -         | -         | -          |
| wapo5a            | 905 | 905  | -         | -         | 4080.97    |

Analysis of the results

First of all, we see that the approaches studied found solutions to 54 out of 57 $NP-s$ instances and 13 out of 15 $NP-m$ instances. The results obtained allow us to state that DECOMP is clearly the fastest of the approaches considered (53 out of 54 solved $NP-s$ instances and all of the solved $NP-m$ instances). For the one case where DECOMP is not the fastest approach, the difference between itself and TWO-VAR (the best one) can be attributed to one, disproportionately long run, in which the solver spent excessive time in branch and bound. Additionally, 2 $NP-s$ instances and 2 $NP-m$ could only be solved by DECOMP within the allotted time.

Further, we can notice that ONE-VAR outperformed TWO-VAR for 7 out of 54 solved $NP-s$ instances and – somewhat surprisingly – 8 out of 13 solved $NP-m$ instances. The converse is true for 45 out of 54 solved $NP-s$ instances.
8.3 Results and their interpretation

and 3 out of 13 solved $NP-m$ instances. For two $NP-s$ instances and two $NP-m$ instances, which were solved by DECOMP, neither ONE-VAR nor TWO-VAR returned a solution within the allotted time.

Finally, we notice that only 3 out of 57 $NP-s$ instances and 2 out of 15 $NP-m$ instances were not solved by any of the solution approaches. We also point out to the fact that for some instances, not all the nodes were assigned two colours (29 of the 54 solved $NP-s$ instances and 7 out of the 15 solved $NP-m$ instances). This is due to the fact that we limited the cardinality of node-compatible colours to 70% of all the colours of a given family (see Section 8.2 above), and – as a consequence – no feasible colour could be found for some nodes.

Overall, we can conclude that – for the instances studied – the decomposition-based approach beats the two MIP formulations both in terms of solution speed and the number of instances which could be solved. For the MIP approaches, the two-variable formulation generally performed better than the one-variable formulation for the $NP-s$ instances. We also note that – somewhat surprisingly – the one-variable formulation turned out to outperform the two-variable formulation for more than a half of the $NP-m$ instances.
Summary and outlook

In this thesis we have studied the integrated locomotive scheduling and driver assignment problem. We devoted Chapters 1–4 to this task. We then generalized the problem to come up with a graph 2-list-colouring with colour compatibilities, and explored its properties. This was done in Chapters 5–8. In the following, we will summarize the most important findings included in this thesis. We will also give indications regarding directions of future research.

Initially, we will consider the most important findings included in Part I of this thesis and discuss them by chapters in which they appeared.

In Chapter 1, we sketched the background for the integrated locomotive scheduling and driver assignment problem. To this end we showed that a demand for such a study exists, driven principally by the low modal share of rail freight in land transport in the EU and the resulting excessive CO2 emissions which could be avoided if the cargo were transported by rail. We also reviewed the existing literature covering the applications of MO in the railway industry. An interesting extension of this chapter would be to thoroughly examine how the integrated planning approaches could help improve the economic and commercial position of the individual railway carriers. Since such analyses would fall out of the scope of this thesis and would require additional data to be disclosed by the industrial partner, we decided not to perform them.

Chapter 2 was devoted to modelling the integrated vehicle scheduling and crew assignment problem. In particular, we reviewed the approaches to integrated treatment of locomotives and drivers in railway context. We also discussed the exact planning requirements and restrictions extensively. Then, we introduced the optimization model, which served as a basis of our further studies. Additionally, we presented a formulation strengthening technique – clique tightening. As an avenue for the future research, we highlight the possibility to incorporate more complex duty scheduling rules, characteristic for rail passenger transport. As we focused on the case of a rail freight company, these were not required and hence out of scope for this thesis. Moreover, in the interest of space we decided not to incorporate other formulations we tested. This decision was motivated by their poor performance even against the smallest of instances.

In Chapter 3, we concentrated on developing the solution algorithm for the integrated vehicle scheduling and crew assignment problem. We first showed the scheme of decomposition into locomotive master problem and
Summary and outlook

driver subproblem. We then introduced four classes of valid inequalities for the optimization model presented in Chapter 3. We also discussed a preprocessing scheme for the driver subproblem. Additionally, we suggested an IP-based presolve heuristic. In the future, it would be interesting to explore further classes of valid inequalities, which could be generated in a fashion similar to the ones already introduced. Also, a more detailed study of the presolve heuristic could be conducted, in which the change in the performance of the solution algorithm dependent on the heuristic parameters could be measured. In case of this thesis, these additional analyses were not required, since the instances submitted by our industrial partner could be solved using the methods we present.

Chapter 4 focused on the presentation of the case study based on instances made available to us by our industrial partner, DB Cargo Polska. We presented the implementation details and discussed the computational results. The methods introduced allowed for a generation of locomotive schedules and driver assignments in less than two hours, which exceeded the expectations of the industrial partner. Additionally, several realistic instances were derived to simulate less favourable resource availabilities, and the performance of the solution algorithm introduced in Chapter 3 against them was discussed. In the future, one could try testing the algorithm on smaller machines or using a different solver. Its performance against instances submitted by other carriers could also be tested.

Let us now turn our attention to Part II of this thesis, in which the graph 2-list-colouring with colour compatibilities problem was studied. Again, we will discuss the contributions made in this part going chapter by chapter.

In Chapter 5, we introduced the problem of graph 2-list-colouring with colour compatibilities. To this end, the basic definitions were introduced and the problem was stated. Additionally, we presented a brief literature survey about different types of graph colouring problems. We also introduced several other concepts, intending to use them later in the thesis.

Chapter 6 was devoted to introducing the formulations of the graph 2-list-colouring problem with colour compatibilities. In particular, we presented two formulations, and discussed some of their properties. We also included a method to tighten their formulation. Furthermore, we found an interesting special case, for which we could prove that a complete polyhedral description may be given. Future research could focus on testing additional formulations to this problem. Additionally, more cases could be found in which the complete polyhedral description can be given, or some other interesting polyhedral results presented.
Chapter 7 focused on the solution algorithm to the graph 2-list-colouring problem with colour compatibilities. It is an adaptation of the algorithm introduced in Chapter 4. In a similar fashion, a decomposition scheme was presented, followed by the valid inequality class, and a preprocessing scheme for one of the subproblems.

Finally, in Chapter 8, a number of computational results for the graph 2-list-colouring problem with colour compatibilities were given. To this end, the two formulations and the decomposition-based solution algorithm introduced in Chapter 9 were tested against a set of problems derived from popular benchmark graph colouring instances. Overall, one could conclude that the decomposition-based approach outperforms the integrated formulations in nearly all the cases. In the future, different parameters could be used to generate the test bed. Additionally, one could devote longer computation times and more powerful computational resources to test the solution approaches against more difficult instances.

Overall, the thesis objectives were to model and solve the integrated locomotive scheduling and driver assignment problem as well as its generalization – \((G_2LC-CC)\). The author hopes that the thesis has accomplished them. We presented mathematical methods for the integrated optimization of locomotive scheduling and driver assignment in rail freight transport. Our aim was to serve as many of the scheduled trains as possible without making use of additional empty runs. To the best of our knowledge, we introduce the first comprehensive model for this integrated planning task, representing in particular the working time requirements of the drivers in a very complete fashion. We are able to solve the problem for large, countrywide instances supplied by a major player in the Polish market, DB Cargo Polska. This was possible by strengthening the initial formulation of the model, together with developing an exact decomposition approach which allows for a sequential solution of the problem. Problem-specific as well as general valid inequality classes ensure the feasibility of solutions to the decomposed subproblems. To further decrease computation times, we devised an additional presolve heuristic. Our results show that the optimal solutions we obtained in many cases allow to run all of the scheduled trains. Further, these solutions can be obtained for planning horizons spanning up to one month in comparably short time (overnight).

Additionally, we identified an interesting theoretical problem within the mathematical model constructed to solve the integrated locomotive scheduling and driver assignment problem. We called it a graph 2-list-colouring problem with colour compatibilities. Two formulations for this problem were suggested. We also presented an interesting polyhedral result for a special
Summary and outlook

case of chordal graph with an additional property. We also introduced a custom solution algorithm and – on the basis of instances generated from popular benchmark datasets – showed its superior performance over the integrated models.

Altogether, the obtained results point to a very beneficial use of our methods in practice, in particular to significantly simplify the current planning process at rail freight carriers. We also believe that this thesis will serve as a starting point for more research in the future. We hope that in these ways, this thesis will enrich the railway sector and its more than two-hundred-years-long history.
Appendix A: Example shift reports

In this section, we present two exemplary monthly train assignments – one for a locomotive and another one for a driver. For both, we present a tabular summary as well as a chart which visualize an individual monthly assignment of trains. Each of the tables includes the individual number of each train assigned, its origin and destination stations as well as the departure and arrival times. In the chart, one can see a number of bars. Each of them corresponds to one train listed in the corresponding table. The trains are plotted against a grid resembling a monthly calendar. On the top of the grid, the names of the weekdays are mentioned.
Appendix A: Example shift reports

Locomotive monthly plan

Loco type and item: Heavy Diesel Locomotive
Number of shifts: 61

Table 17: A tabular presentation of the locomotive monthly plan

<table>
<thead>
<tr>
<th>Train no.</th>
<th>From</th>
<th>To</th>
<th>Departure time</th>
<th>Arrival time</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>PL035</td>
<td>PL020</td>
<td>2020-02-01 06:54:59</td>
<td>2020-02-01 13:18:59</td>
</tr>
<tr>
<td>149</td>
<td>PL020</td>
<td>PL035</td>
<td>2020-02-02 19:12:00</td>
<td>2020-02-02 21:54:59</td>
</tr>
<tr>
<td>206</td>
<td>PL035</td>
<td>PL085</td>
<td>2020-02-03 13:45:00</td>
<td>2020-02-03 14:39:59</td>
</tr>
<tr>
<td>226</td>
<td>PL085</td>
<td>PL035</td>
<td>2020-02-03 19:59:59</td>
<td>2020-02-03 20:46:00</td>
</tr>
<tr>
<td>290</td>
<td>PL035</td>
<td>PL020</td>
<td>2020-02-04 11:55:00</td>
<td>2020-02-04 14:15:00</td>
</tr>
<tr>
<td>352</td>
<td>PL020</td>
<td>PL035</td>
<td>2020-02-05 03:40:59</td>
<td>2020-02-05 07:14:59</td>
</tr>
<tr>
<td>459</td>
<td>PL035</td>
<td>PL020</td>
<td>2020-02-06 06:54:59</td>
<td>2020-02-06 13:18:59</td>
</tr>
<tr>
<td>521</td>
<td>PL020</td>
<td>PL035</td>
<td>2020-02-06 19:59:59</td>
<td>2020-02-06 23:06:00</td>
</tr>
<tr>
<td>587</td>
<td>PL035</td>
<td>PL046</td>
<td>2020-02-07 10:59:59</td>
<td>2020-02-07 12:00:00</td>
</tr>
<tr>
<td>600</td>
<td>PL046</td>
<td>PL035</td>
<td>2020-02-07 15:00:00</td>
<td>2020-02-07 16:00:00</td>
</tr>
<tr>
<td>667</td>
<td>PL035</td>
<td>PL020</td>
<td>2020-02-08 09:22:00</td>
<td>2020-02-08 12:45:00</td>
</tr>
<tr>
<td>777</td>
<td>PL020</td>
<td>PL015</td>
<td>2020-02-09 19:12:00</td>
<td>2020-02-10 03:54:59</td>
</tr>
<tr>
<td>850</td>
<td>PL015</td>
<td>PL010</td>
<td>2020-02-10 18:45:59</td>
<td>2020-02-10 23:35:59</td>
</tr>
<tr>
<td>873</td>
<td>PL010</td>
<td>PL045</td>
<td>2020-02-11 00:00:59</td>
<td>2020-02-11 06:11:00</td>
</tr>
<tr>
<td>917</td>
<td>PL045</td>
<td>PL003</td>
<td>2020-02-11 12:00:00</td>
<td>2020-02-11 12:29:00</td>
</tr>
<tr>
<td>1002</td>
<td>PL003</td>
<td>PL022</td>
<td>2020-02-12 10:50:00</td>
<td>2020-02-12 14:17:00</td>
</tr>
</tbody>
</table>

Continued on next page
## Table 17 – continued from previous page

<table>
<thead>
<tr>
<th>Train no.</th>
<th>From</th>
<th>To</th>
<th>Departure time</th>
<th>Arrival time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1027</td>
<td>PL022</td>
<td>PL003</td>
<td>2020-02-12 17:30:00</td>
<td>2020-02-12 21:22:00</td>
</tr>
<tr>
<td>1098</td>
<td>PL003</td>
<td>PL022</td>
<td>2020-02-13 10:50:00</td>
<td>2020-02-13 14:17:00</td>
</tr>
<tr>
<td>1122</td>
<td>PL022</td>
<td>PL003</td>
<td>2020-02-13 17:30:00</td>
<td>2020-02-13 21:22:00</td>
</tr>
<tr>
<td>1144</td>
<td>PL003</td>
<td>PL033</td>
<td>2020-02-13 22:30:59</td>
<td>2020-02-13 23:10:00</td>
</tr>
<tr>
<td>1165</td>
<td>PL033</td>
<td>PL003</td>
<td>2020-02-14 05:10:00</td>
<td>2020-02-14 05:46:00</td>
</tr>
<tr>
<td>1178</td>
<td>PL003</td>
<td>PL066</td>
<td>2020-02-14 09:20:59</td>
<td>2020-02-14 10:20:00</td>
</tr>
<tr>
<td>1198</td>
<td>PL066</td>
<td>PL003</td>
<td>2020-02-14 13:20:00</td>
<td>2020-02-14 14:23:00</td>
</tr>
<tr>
<td>1206</td>
<td>PL003</td>
<td>PL070</td>
<td>2020-02-14 15:35:00</td>
<td>2020-02-14 15:51:59</td>
</tr>
<tr>
<td>1221</td>
<td>PL070</td>
<td>PL003</td>
<td>2020-02-14 19:36:59</td>
<td>2020-02-14 19:54:00</td>
</tr>
<tr>
<td>1235</td>
<td>PL003</td>
<td>PL033</td>
<td>2020-02-14 22:30:59</td>
<td>2020-02-14 23:10:00</td>
</tr>
<tr>
<td>1257</td>
<td>PL033</td>
<td>PL003</td>
<td>2020-02-15 05:10:00</td>
<td>2020-02-15 05:46:00</td>
</tr>
<tr>
<td>1264</td>
<td>PL003</td>
<td>PL026</td>
<td>2020-02-15 06:27:59</td>
<td>2020-02-15 08:01:00</td>
</tr>
<tr>
<td>1327</td>
<td>PL026</td>
<td>PL003</td>
<td>2020-02-16 01:36:00</td>
<td>2020-02-16 03:13:59</td>
</tr>
<tr>
<td>1334</td>
<td>PL003</td>
<td>PL047</td>
<td>2020-02-16 04:54:00</td>
<td>2020-02-16 11:35:00</td>
</tr>
<tr>
<td>1361</td>
<td>PL047</td>
<td>PL012</td>
<td>2020-02-16 12:18:59</td>
<td>2020-02-16 12:35:00</td>
</tr>
<tr>
<td>1371</td>
<td>PL012</td>
<td>PL020</td>
<td>2020-02-16 17:35:00</td>
<td>2020-02-16 17:48:00</td>
</tr>
<tr>
<td>1375</td>
<td>PL020</td>
<td>PL026</td>
<td>2020-02-16 19:00:00</td>
<td>2020-02-17 04:50:00</td>
</tr>
<tr>
<td>1499</td>
<td>PL026</td>
<td>PL003</td>
<td>2020-02-18 08:12:00</td>
<td>2020-02-18 10:12:00</td>
</tr>
<tr>
<td>1517</td>
<td>PL003</td>
<td>PL033</td>
<td>2020-02-18 11:17:00</td>
<td>2020-02-18 11:48:00</td>
</tr>
<tr>
<td>1535</td>
<td>PL033</td>
<td>PL003</td>
<td>2020-02-18 16:11:00</td>
<td>2020-02-18 16:41:59</td>
</tr>
<tr>
<td>1586</td>
<td>PL026</td>
<td>PL003</td>
<td>2020-02-19 04:43:59</td>
<td>2020-02-19 06:33:00</td>
</tr>
<tr>
<td>1605</td>
<td>PL003</td>
<td>PL026</td>
<td>2020-02-19 09:45:00</td>
<td>2020-02-19 11:44:59</td>
</tr>
<tr>
<td>1629</td>
<td>PL026</td>
<td>PL003</td>
<td>2020-02-19 16:00:00</td>
<td>2020-02-19 19:59:59</td>
</tr>
<tr>
<td>1696</td>
<td>PL003</td>
<td>PL080</td>
<td>2020-02-20 09:13:00</td>
<td>2020-02-20 11:41:00</td>
</tr>
<tr>
<td>1806</td>
<td>PL003</td>
<td>PL022</td>
<td>2020-02-21 10:50:00</td>
<td>2020-02-21 14:17:00</td>
</tr>
<tr>
<td>1829</td>
<td>PL022</td>
<td>PL003</td>
<td>2020-02-21 17:30:00</td>
<td>2020-02-21 21:22:00</td>
</tr>
<tr>
<td>1850</td>
<td>PL003</td>
<td>PL033</td>
<td>2020-02-21 22:30:59</td>
<td>2020-02-21 23:10:00</td>
</tr>
</tbody>
</table>

Continued on next page
### Appendix A: Example shift reports

<table>
<thead>
<tr>
<th>Train no.</th>
<th>From</th>
<th>To</th>
<th>Departure time</th>
<th>Arrival time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1873</td>
<td>PL033</td>
<td>PL003</td>
<td>2020-02-22 05:10:00</td>
<td>2020-02-22 05:46:00</td>
</tr>
<tr>
<td>1880</td>
<td>PL003</td>
<td>PL026</td>
<td>2020-02-22 06:27:59</td>
<td>2020-02-22 08:01:00</td>
</tr>
<tr>
<td>1885</td>
<td>PL026</td>
<td>PL003</td>
<td>2020-02-22 08:12:00</td>
<td>2020-02-22 10:12:00</td>
</tr>
<tr>
<td>1901</td>
<td>PL003</td>
<td>PL033</td>
<td>2020-02-22 11:17:00</td>
<td>2020-02-22 11:48:00</td>
</tr>
<tr>
<td>1916</td>
<td>PL033</td>
<td>PL003</td>
<td>2020-02-22 16:11:00</td>
<td>2020-02-22 16:41:59</td>
</tr>
<tr>
<td>2107</td>
<td>PL026</td>
<td>PL003</td>
<td>2020-02-25 04:43:59</td>
<td>2020-02-25 06:33:00</td>
</tr>
<tr>
<td>2123</td>
<td>PL003</td>
<td>PL026</td>
<td>2020-02-25 09:45:00</td>
<td>2020-02-25 11:44:59</td>
</tr>
<tr>
<td>2164</td>
<td>PL026</td>
<td>PL003</td>
<td>2020-02-25 18:53:00</td>
<td>2020-02-25 20:35:00</td>
</tr>
<tr>
<td>2221</td>
<td>PL003</td>
<td>PL026</td>
<td>2020-02-26 09:45:00</td>
<td>2020-02-26 11:44:59</td>
</tr>
<tr>
<td>2296</td>
<td>PL026</td>
<td>PL003</td>
<td>2020-02-27 04:43:59</td>
<td>2020-02-27 06:33:00</td>
</tr>
<tr>
<td>2316</td>
<td>PL003</td>
<td>PL026</td>
<td>2020-02-27 09:45:00</td>
<td>2020-02-27 11:44:59</td>
</tr>
<tr>
<td>2388</td>
<td>PL026</td>
<td>PL003</td>
<td>2020-02-28 04:43:59</td>
<td>2020-02-28 06:33:00</td>
</tr>
<tr>
<td>2405</td>
<td>PL003</td>
<td>PL026</td>
<td>2020-02-28 09:09:59</td>
<td>2020-02-28 11:10:59</td>
</tr>
<tr>
<td>2484</td>
<td>PL026</td>
<td>PL003</td>
<td>2020-02-29 04:43:59</td>
<td>2020-02-29 06:33:00</td>
</tr>
<tr>
<td>2501</td>
<td>PL003</td>
<td>PL026</td>
<td>2020-02-29 09:09:59</td>
<td>2020-02-29 11:10:59</td>
</tr>
</tbody>
</table>
Appendix A: Example shift reports

Driver weekly plan

Name and surname: Driver045
Number of shifts: 24
Number of Sundays off: 1
Number of 35-hour breaks: 4

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
<th>Saturday</th>
<th>Sunday</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.02</td>
<td>4.02</td>
<td>5.02</td>
<td>6.02</td>
<td>7.02</td>
<td>8.02</td>
<td>9.02</td>
</tr>
<tr>
<td>10.02</td>
<td>11.02</td>
<td>12.02</td>
<td>13.02</td>
<td>14.02</td>
<td>15.02</td>
<td>16.02</td>
</tr>
<tr>
<td>17.02</td>
<td>18.02</td>
<td>19.02</td>
<td>20.02</td>
<td>21.02</td>
<td>22.02</td>
<td>23.02</td>
</tr>
<tr>
<td>24.02</td>
<td>25.02</td>
<td>26.02</td>
<td>27.02</td>
<td>28.02</td>
<td>29.02</td>
<td></td>
</tr>
</tbody>
</table>

Figure 22: An illustration of a driver monthly plan

Table 18: A tabular presentation of the driver monthly plan

<table>
<thead>
<tr>
<th>Train no.</th>
<th>From</th>
<th>To</th>
<th>Departure time</th>
<th>Arrival time</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>PL032</td>
<td>PL050</td>
<td>2020-02-01 02:30:00</td>
<td>2020-02-01 08:23:00</td>
</tr>
<tr>
<td>113</td>
<td>PL032</td>
<td>PL050</td>
<td>2020-02-02 08:06:00</td>
<td>2020-02-02 14:19:00</td>
</tr>
<tr>
<td>305</td>
<td>PL050</td>
<td>PL032</td>
<td>2020-02-04 15:29:59</td>
<td>2020-02-04 22:16:00</td>
</tr>
<tr>
<td>437</td>
<td>PL032</td>
<td>PL094</td>
<td>2020-02-06 01:11:00</td>
<td>2020-02-06 04:41:59</td>
</tr>
<tr>
<td>507</td>
<td>PL011</td>
<td>PL094</td>
<td>2020-02-06 17:10:59</td>
<td>2020-02-06 17:19:00</td>
</tr>
<tr>
<td>568</td>
<td>PL032</td>
<td>PL050</td>
<td>2020-02-07 08:06:00</td>
<td>2020-02-07 14:19:00</td>
</tr>
<tr>
<td>606</td>
<td>PL011</td>
<td>PL094</td>
<td>2020-02-07 17:10:59</td>
<td>2020-02-07 17:19:00</td>
</tr>
<tr>
<td>659</td>
<td>PL050</td>
<td>PL032</td>
<td>2020-02-08 06:56:00</td>
<td>2020-02-08 13:34:00</td>
</tr>
<tr>
<td>743</td>
<td>PL032</td>
<td>PL050</td>
<td>2020-02-09 08:06:00</td>
<td>2020-02-09 14:19:00</td>
</tr>
<tr>
<td>803</td>
<td>PL033</td>
<td>PL003</td>
<td>2020-02-10 05:10:00</td>
<td>2020-02-10 05:46:00</td>
</tr>
<tr>
<td>868</td>
<td>PL003</td>
<td>PL033</td>
<td>2020-02-10 22:30:59</td>
<td>2020-02-10 23:10:00</td>
</tr>
<tr>
<td>932</td>
<td>PL011</td>
<td>PL094</td>
<td>2020-02-11 17:19:59</td>
<td>2020-02-11 17:28:00</td>
</tr>
<tr>
<td>958</td>
<td>PL003</td>
<td>PL033</td>
<td>2020-02-11 22:30:59</td>
<td>2020-02-11 23:10:00</td>
</tr>
<tr>
<td>1024</td>
<td>PL011</td>
<td>PL094</td>
<td>2020-02-12 16:25:00</td>
<td>2020-02-12 16:34:00</td>
</tr>
<tr>
<td>1052</td>
<td>PL032</td>
<td>PL050</td>
<td>2020-02-12 22:03:59</td>
<td>2020-02-13 03:11:00</td>
</tr>
</tbody>
</table>

Continued on next page
### Table 18 – continued from previous page

<table>
<thead>
<tr>
<th>Train no.</th>
<th>From</th>
<th>To</th>
<th>Departure time</th>
<th>Arrival time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1223</td>
<td>PL094</td>
<td>PL011</td>
<td>2020-02-14 20:10:00</td>
<td>2020-02-14 20:21:00</td>
</tr>
<tr>
<td>1243</td>
<td>PL032</td>
<td>PL094</td>
<td>2020-02-15 01:11:00</td>
<td>2020-02-15 04:41:59</td>
</tr>
<tr>
<td>1320</td>
<td>PL003</td>
<td>PL033</td>
<td>2020-02-15 22:30:59</td>
<td>2020-02-15 23:10:00</td>
</tr>
<tr>
<td>1324</td>
<td>PL050</td>
<td>PL032</td>
<td>2020-02-16 00:15:00</td>
<td>2020-02-16 06:51:00</td>
</tr>
<tr>
<td>1485</td>
<td>PL094</td>
<td>PL005</td>
<td>2020-02-18 04:14:00</td>
<td>2020-02-18 06:42:59</td>
</tr>
<tr>
<td>1659</td>
<td>PL032</td>
<td>PL050</td>
<td>2020-02-19 22:03:59</td>
<td>2020-02-20 03:11:00</td>
</tr>
<tr>
<td>1726</td>
<td>PL033</td>
<td>PL003</td>
<td>2020-02-20 16:11:00</td>
<td>2020-02-20 16:41:59</td>
</tr>
<tr>
<td>1786</td>
<td>PL050</td>
<td>PL032</td>
<td>2020-02-21 06:56:00</td>
<td>2020-02-21 13:34:00</td>
</tr>
<tr>
<td>1883</td>
<td>PL050</td>
<td>PL032</td>
<td>2020-02-22 06:56:00</td>
<td>2020-02-22 13:34:00</td>
</tr>
<tr>
<td>2085</td>
<td>PL032</td>
<td>PL050</td>
<td>2020-02-24 22:03:59</td>
<td>2020-02-25 03:13:00</td>
</tr>
<tr>
<td>2206</td>
<td>PL094</td>
<td>PL032</td>
<td>2020-02-26 05:39:00</td>
<td>2020-02-26 08:26:59</td>
</tr>
<tr>
<td>2277</td>
<td>PL032</td>
<td>PL050</td>
<td>2020-02-26 22:03:59</td>
<td>2020-02-27 04:12:00</td>
</tr>
<tr>
<td>2380</td>
<td>PL032</td>
<td>PL050</td>
<td>2020-02-28 02:30:00</td>
<td>2020-02-28 08:23:00</td>
</tr>
<tr>
<td>2531</td>
<td>PL011</td>
<td>PL094</td>
<td>2020-02-29 17:19:59</td>
<td>2020-02-29 17:28:00</td>
</tr>
</tbody>
</table>
Appendix B: Algorithm for reassignment of trains to actual locomotives

In this appendix, we will introduce an algorithm, which converts the solution to the commodity-aggregated version of the problem into assignments for individual locomotives. For this end, recall that individual locomotives $l \in \mathcal{L}$ are grouped into locomotive classes $L \in \mathcal{L}$. Further, recall that the locomotive assignments are modelled on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, such that

$$\mathcal{V} := T \cup \{\Sigma, \Theta\}$$

and

$$\mathcal{A} := \bigcup_{L \in \mathcal{L}} \mathcal{A}^L,$$

with

$$\mathcal{A}^L := \{(t_1, t_2) : t_1 \in T_L \land t_2 \in T_{t_1,L}^{\text{next}}\},$$

for all $L \in \mathcal{L}$. Additionally, recall that defined $\Sigma, \Theta \in \mathcal{V}$ as the source and sink nodes of $\mathcal{G}$ respectively. Additionally, recall that the assignment of locomotive classes to trains is modelled by variables $f^L_{t_1, t_2}$, which assume the value 1 if locomotive of class $L \in \mathcal{L}$ servers trains $t_1, t_2$ consecutively and 0 otherwise. Let $f^L_{t_1, t_2}$ denote the value of the variable $f^L_{t_1, t_2}$ in the optimal solution.

The algorithm is given in pseudocode below.
Appendix B: Algorithm for reassignment of trains to actual locomotives

**Data:** Set of sets of locomotives grouped by locomotive class $\mathcal{L}$  
**Result:** Exact attribution of locomotives to trains $A : l \rightarrow T^l$ for $L \in \mathcal{L}$

```
for b \in \text{beginnings} do
    T = \{b\};
    \text{prev} \gets b;
    \text{next} \gets 0;
    \textbf{while} \text{next} \neq \emptyset \textbf{do}
    \text{next} \gets t_1 : f_{L, L}^{\text{prev}, t_1} = 1;
    \textbf{if} \text{next} \neq \emptyset \textbf{then}
        T \gets T \cup \{\text{next}\};
    \textbf{end}
    \text{prev} \gets \text{next};
\textbf{end}
\text{take} \ l \in L;
L \gets L \setminus \{l\};
A(l) \gets T;
end
```

**Algorithm 1:** Assigning individual locomotives to trains based on the solution of the commodity-aggregated version of the model.
Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


Bibliography


The integrated treatment of planning problems which are usually considered separately and sequentially has been studied for a long time, due to the better solutions one may find in the extended decision space of an integrated problem. This thesis considers the integrated locomotive scheduling and driver assignment problem in rail freight transport. We also consider the generalization of this problem, which we call graph 2-list-colouring with compatibility constraints. The motivation to study this problem originates from our collaboration with DB Cargo Polska within the ROMSOC Programme. The thesis consists of two parts. Part I focuses on modelling and solving the integrated locomotive scheduling and driver assignment problem in rail freight transport. After a literature review, we present a novel optimization model for the problem studied and a way to improve its formulation. Next, we introduce the decomposition-based solution approach we derive for the problem. To ensure the global feasibility of the solutions to the decomposed subproblems, we devise four classes of valid inequalities. We also develop a presolve heuristic. We then test our algorithm against two sets of instances. In general, the methods presented enabled the creation of locomotive timetables and driver assignments in less than two hours. In Part II, we study a generalization of the integrated locomotive scheduling and driver assignment problem, which we call graph 2-list-colouring with compatibility constraints. We begin by defining the problem studied and putting it in the context of other, more famous combinatorial problems. Then we present two formulations for the problem and discuss how they may be tightened. We also study a case for which we may find a complete polyhedral description. Next, we present a decomposition-based solution approach which adapts the algorithm introduced in Part I. We then test the performance of our method against a set of standard instances drawn from the literature, which were appropriately modified. Altogether, our work is a practical contribution to the solvability of the integrated locomotive scheduling and driver assignment problem. We also show how the developed method may be extended to successful use in more general graph-theoretic contexts.