Mathematical Optimization of Rolling Stock Rotations

vorgelegt von
Dipl.-Math.
Markus Reuther
geb. in Marienberg (Sachsen)

von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Volker Mehrmann
Gutachter:	Prof. Dr. Ralf Borndörfer
Gutachter:	Prof. Dr. Dr. h.c. mult. Martin Grötschel

Tag der wissenschaftlichen Aussprache: 5. Juli 2016

Berlin 2017

Zusammenfassung

Wir zeigen wie man Eisenbahnumläufe optimiert. Diese sind unerlässlich für die Produktion eines Fahrplans für den Personenverkehr. Das zugrundeliegende mathematische Optimierungsproblem ist das Rolling Stock Rotation Problem (RSRP) und RotOR – ein stark integrierter Optmierungsalgorithmus für das RSRP – bildet den roten Faden der Arbeit. RotOR wird bei der DB Fernverkehr AG (DBF) eingesetzt um Intercity-Express (ICE) Umläufe für das europäische Hochgeschwindigkeitsnetz zu optimieren. Diese Anwendung erfordert das Lösen von Szenarien in denen (A) viele verschiedene Anforderungen simultan, (B) typischer Weise große RSRP Instanzen und (C) Bedingungen mit einer schwierigen kombinatorischen Struktur zu betrachten sind. In dieser Arbeit schlagen wir die folgenden Konzepte für diese Gegebenheiten vor:

(A) Das Gesamtmodell von RotOR basiert auf einem Hypergraphen. Dieser Hypergraph ermöglicht einen einfachen aber dennoch sehr detailierten Umgang mit vielfältigen Anforderungen des Eisenbahnbetriebs. Derartige Anforderungen sind die Fahrzeugkomposition und die noch relativ unerforschte Gleichförmigkeit. Gleichzeitig führt der Hypergraph direkt zu einem gemischt-ganzzahligen Program für das RSRP.

(B) Unser algorithmischer Hauptansatz ist ein Coarse-to-fine (C2F) Spaltengenerierungsverfahren. Dazu wird der Hypergraph in feine und gröbere Schichten, sogenannte Layer, unterteilt. Die Layer entsprechen verschiedenen Detailierungsgraden. Gröbere Layer werden algorithmisch ausgenutzt um Spalten für den feinsten Layer zu erzeugen bis ein Optimalitätsbeweis gefunden ist. Die C2F Methode wird zunächst autonom für lineare Programme erklärt um eine Schnittstelle für andere Probleme bereit zu stellen.

(C) Eisenbahnumläufe müssen sogenannten Ressourcenbedingungen genügen, sodass z.B. ausreichend viele Wartungsmaßnahmen überdeckt sind. Derartige Bedingungen sind schwierig, allerdings vom Vehicle Routing Problem (VRP) in der Literatur bekannt. Wir definieren eine Schnittstelle zwischen RSRP und VRP und entwickeln aus deren Gemeinsamkeiten und vorallem Unterschieden ein geradliniges algorithmisches Konzept das wir Regionale Suche (RS) nennen. Unsere RS Algorithmen zeigen vielversprechende Ergebnisse für klassische VRP und RSRP Instanzen.

Diese drei Konzepte bilden den mathematischen Hauptbeitrag und den ersten Teil der Arbeit. Der zweite Teil erklärt alle Modellierungs- und Lösungskomponenten welche für die Optimierung von ICE Umläufen bei der DBF notwendig sind. Die Arbeit schließt mit einem komplexen Anwendungsfall bei dem RotOR von der DBF benutzt wurde um ICE Umläufe für eine Sperrung der Hochgeschwindigkeitstrecke zwischen Frankfurt (Main) und Köln zu berechnen. Aufgrund dieser Baustelle mussten nahezu alle Fahrzeuge der Kategorie ICE-W umgeleitet werden.

Abstract

We show how to optimize rolling stock rotations that are required for the operation of a passenger timetable. The underlying mathematical optimization problem is called rolling stock rotation problem (RSRP) and the leitmotiv of the thesis is RotOR, i.e., a highly integrated optimization algorithm for the RSRP. RotOR is used by DB Fernverkehr AG (DBF) in order to optimize intercity express (ICE) rotations for the European high-speed network. In this application, RSRPs have to be solved which (A) require many different aspects to be simultaneously considered, (B) are typically of large scale, and (C) include constraints that have a difficult combinatorial structure. This thesis suggests answers to these issues via the following concepts.

(A) The main model, which RotOR uses, relies on a hypergraph. The hypergraph provides an easy way to model manifold industrial railway requirements in great detail. This includes well known vehicle composition requirements as well as relatively unexplored regularity stipulations. At the same time, the hypergraph directly leads to a mixed-integer programming (MIP) model for the RSRP.

(B) The main algorithmic ingredient to solve industrial instances of the RSRP is a coarse-to-fine (C2F) column generation procedure. In this approach, the hypergraph is layered into coarse and fine layers that distinguish different levels of detail of the RSRP. The coarse layers are algorithmically utilized while pricing fine columns until proven optimality. Initially, the C2F approach is presented in terms of pure linear programming in order to provide an interface for other applications.

(C) Rolling stock rotations have to comply to resource constraints in order to ensure, e.g., enough maintenance inspections along the rotations. These constraints are computationally hard, but are well known in the literature on the vehicle routing problem (VRP). We define an interface problem in order to bridge between the RSRP and the VRP and derive a straightforward algorithmic concept, namely regional search (RS), from their common features and, moreover, differences. Our RS algorithms show promising results for classical VRPs and RSRPs.

In the first part of the thesis we present these concepts, which encompass its main mathematical contribution. The second part explains all modeling and solving components of RotOR that turn out to be essential in its industrial application. The thesis concludes with a solution to a complex re-optimization RSRP that RotOR has computed successfully for DBF. In this application all ICE vehicles of the ICE-W fleets of DBF had to be redirected past a construction site on a high-speed line in the heart of Germany.

Acknowledgments

My research position at Zuse Institute Berlin (ZIB) was initiated in a conversation during a coffee break of some of my former professors, including a supervisor of my diploma thesis Peter Tittmann, and myself. The discussion was about where I should apply for my internship semester as a student of the Hochschule Mittweida (HM). This was 10.5 years ago and my professors suggested: “versuchen Sie es doch mal am” ZIB. Now I know that this was one of many excellent advices and I would like to thank for all of them.

Surprisingly, my unsolicited application was taken for serious by Volker Kaibel, who employed me as a student in spring 2006 at ZIB. Thanks for that and also thanks to Marc Pfetsch for supervising me during the first months of my great ZIB experience.

After a couple of weeks I met Ralf Borndörfer (RB) who became my supervisor during two internships and also during my diploma thesis at ZIB in 2008. After I finished the diploma thesis I was kindly asked again by RB to further work at ZIB (there is always a lot of work in Berlin). Since I already had a “Dipl.-Math. (FH)” from HM, the only little formality that remained to be done was to redo my studies at the TU Berlin in order to drop the “(FH)” from my academic degree (with the “(FH)” I was overqualified for the ZIB as I understood).

After I passed the final examination under Martin Grötschel at TU Berlin I became a research assistant at ZIB in May 2009. I would like to thank Martin Grötschel for employing me at ZIB, for the excellent scientific infrastructure that he established and shared at ZIB, for his scientific criticism, as well as for his constancy in telling nice stories during many lunch breaks.

The task RB gave me as a member of ZIB was to work in close cooperation with DBF on the optimization of rolling stock rotations for ICE vehicles. This sounded very spectacular in the beginning, but after the first visits in Frankfurt (Main) I was pretty sure that this will never ever work. Fortunately, I am sometimes wrong.

I have been the main developer of RotOR and I was greatly supported by the extensive experience of Steffen Weider during the first two years, by the excellent management skills of Kerstin Waas during the first five years, by the great marketing and distinguished motivation skills of Thomas Schlechte (TS) during the past four years, and, finally, by the outstanding mind, patience, and, moreover, incredible optimism of RB all the time. Thank you all!

Additionally, special thanks go to RB for allowing me to enjoy numerous conferences, e.g., in Aachen, Bonn, Copenhagen, Hong Kong, Lisbon, Mittweida, Oslo, Patras, Rome, Rotterdam, Saarbrücken, Santiago de Chile, Tokyo, Vienna, Vilnius, Warsaw, and Wrocław, and for being also the supervisor during my PhD time, and for the paid leave that I got in order to write this thesis (note that it is not cumulative) in my idyllic single office at ZIB.

My position at ZIB was funded for six years by DBF and I also would like to thank for that. But I have no idea whom exactly to thank for the funding because DBF is not a person. Therefore, I would like to thank all the approximately 16,000 members of DBF (for sharing their income with me).

I also would like to thank Markus Dod, Ricardo Euler, Gerwin Gamrath, Boris Grimm, Julika Mehrgardt, Stanley Schade, TS, and Alexander Tesch for their ZIB-sided code contributions to RotOR as well as Isabel Beckenbach, Matthias Breuer, Boris Grimm, Marika Karbstein, Jonad Pulaj, Stanley Schade, TS, and Patrick Siebeneicher for proofreading parts of this thesis. To the best of my knowledge, the DBF-sided contributions to the project behind RotOR, for which I am also very grateful, have been accomplished by Matthias Breuer, Sabine Ifland-Richter, Anita Kirchner, Andreas Lenssen, Matthias Schork, Patrick Siebeneicher, Kerstin Waas, and by members of GSV. In addition, I would like to thank my co-authors of former publications RB, Boris Grimm, Marika Karbstein, Torsten Klug, Leonardo Lamorgese, Andreas Löbel, Carlo Mannino, Julika Mehrgardt, TS, Christof Schulz, Elmar Swarat, Kerstin Wass, and Steffen Weider for their scientific commitment.

As a member of the ZIB optimization department it is obviously mandatory to also thank for the numerous coffee breaks (which I have joined for one or two times as I remember). In fact, these meetings helped a lot in order to bring me *up* to earth and back to reality when I was captured by my project and this thesis. Therefore, I would like to seriously thank all colleagues and guests of our department who have contributed to these meetings.

I had serious problems with optimization, motivation, as well as health during working on this and I would like to thank my family for their support.

Finally and above all, I have to say that I could always count strongly on my wife, who took and takes care of me. Judith, Dir möchte ich am allermeisten und von ganzem Herzen danken!

Markus Reuther
Berlin, April 2016

The German words are more serious than the English (in parentheses).

Contents

Introduction **1**
Notation . . . 5
Organization . . . 6
Publications . . . 8

I. Concepts **11**

1. Rolling Stock Rotation Optimization by Hypergraphs **13**
1.1. Introduction . . . 13
1.2. Related Literature . . . 15
1.3. Graph-Based Hypergraphs . . . 18
1.4. The Rolling Stock Rotation Problem . . . 19
1.5. The Basic Model . . . 21
1.6. The Complexity Tree . . . 26
1.6.1. Standard Assignments . . . 28
1.6.2. Hypergraph Assignments . . . 29
1.6.3. Resource-Constrained Assignments . . . 30
1.6.4. Single-Commodity RSRP . . . 31
1.6.5. Non-Maintenance RSRP . . . 32
1.6.6. Multi-Depot Vehicle Scheduling . . . 33
1.6.7. Multi-Depot Vehicle Routing . . . 34
1.7. Summarizing Comments . . . 36

2. The Coarse-to-Fine Approach **37**
2.1. Motivation . . . 37
2.2. Literature on Projective C2F Ideas . . . 38
2.3. C2F Column Generation for Linear Programs . . . 43
2.3.1. Column Generation . . . 43
2.3.2. Column Selection by Layers . . . 44
2.3.3. C2F Column Generation Algorithm . . . 46
2.4. Layers for Rolling Stock Rotation Optimization . . . 48
2.4.1. Composition Layer . . . 51
2.4.2. Configuration Layer . . . 52
2.4.3. Vehicle Layer . . . 55
2.5. C2F Hyperarc Generation . . . 57
2.6. Computational Study . . . 59
2.7. Conclusion . . . 63

3. Resource-Constrained Assignments by Regional Search **67**
3.1. Introduction . . . 68
3.2. Literature . . . 69
3.3. Resource-Constrained Assignments . . . 72
3.4. Local Search . . . 74
3.5. Regional Search . . . 80
3.5.1. Regional Search for Binary Programs . . . 81
3.5.2. Regional Search for the RCAP . . . 82
3.5.2.1. Branching-Based RS . . . 84
3.5.2.2. Move-Based RS . . . 86
3.5.2.3. Dual Diversification . . . 89
3.6. Global Search . . . 91
3.6.1. Branching Scheme . . . 92
3.6.2. Assignment Reduction . . . 95
3.6.3. Shortest-Path Reduction . . . 96
3.6.4. Bin-Packing Reduction . . . 97
3.6.5. Symmetry Reduction . . . 98
3.7. Computational Results . . . 99
3.7.1. Rolling Stock Rotation Instances . . . 100
3.7.2. TSP, ATSP, CVRP, and ACVRP Instances . . . 101
3.8. Conclusion . . . 104

II. RotOR **111**

4. RotOR's Hypergraph Model **113**
4.1. Vehicle Configuration . . . 114
4.2. Turn Duration Rules . . . 118
4.3. Railway Topology . . . 122
4.4. Deadhead Trips . . . 124
4.5. Service Paths . . . 126
4.6. Vehicle Orientation . . . 129
4.7. Vehicle Composition . . . 132
4.8. Trip Sequences . . . 136
4.9. Coupling and Decoupling . . . 140
4.10. Regularity Patterns . . . 142
4.10.1. Regular Trips . . . 145
4.10.2. Regular Turns . . . 149
4.10.3. Regular Handouts . . . 151
4.11. Re-optimization Templates . . . 156
4.12. Objective Function . . . 162
4.12.1. Greenfield Optimization . . . 162
4.12.2. Re-optimization . . . 164

5. RotOR's Mixed-Integer Programming Model **167**
5.1. The Model . . . 168
5.2. Maintenance Constraints . . . 170
5.3. Capacity Constraints . . . 173
5.3.1. Fleet Capacity . . . 173
5.3.2. Infrastructure Capacity . . . 175
5.4. Trunk Constraints . . . 176

6. RotOR's Algorithms **181**
6.1. General Workflow . . . 181
6.2. C2F Constraint Matrix Generation . . . 184
6.3. Rapid Branching . . . 189
6.4. Cut Date Heuristic . . . 191
6.5. Coupling Components . . . 194
6.5.1. Infeasible Couplings . . . 194
6.5.2. Cutting Planes . . . 197
6.5.3. Coupling Components . . . 198
6.5.4. Coupling Component Heuristic . . . 199
6.6. Fractional Maintenance Relaxation . . . 201
6.7. Handout Optimization . . . 204

7. RotOR's Computations and Applications **209**
7.1. Implementation Details . . . 210
7.2. How to Compare? . . . 212
7.3. Industrial Test Set . . . 213
7.3.1. Problem-oriented View . . . 215
7.3.2. Algorithmic View . . . 221
7.3.3. Solution-oriented View . . . 226
7.3.3.1. Greenfield Objectives . . . 227
7.3.3.2. Re-optimization Objectives . . . 231
7.4. The Price of Regularity . . . 234
7.5. The Köln-Rhein-Main Construction Site (**Conclusion of the thesis**) . . . 237

Abbreviations **243**

List of Figures **245**

List of Tables **247**

List of Algorithms **249**

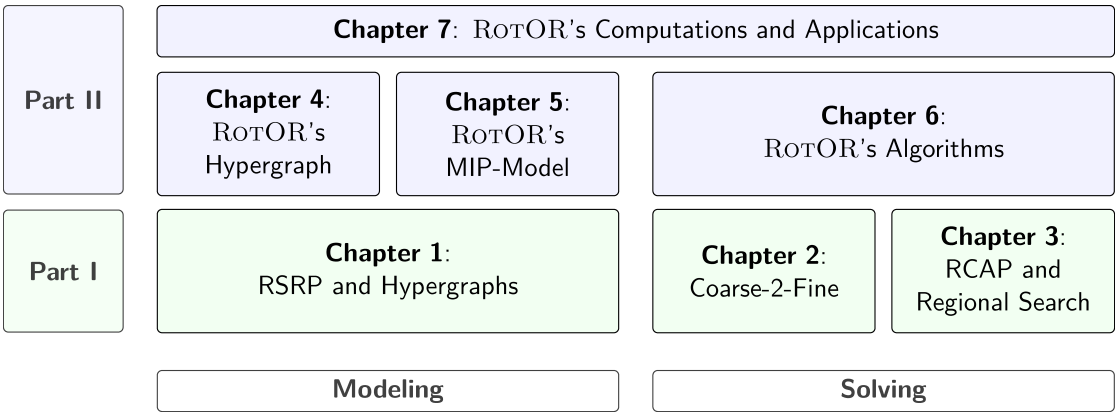
Bibliography **251**

Introduction

Believe it or not, yet another thesis which deals with discrete applied mathematics! This one is about an optimization algorithm that we call Rotation Optimizer for Railways (RotOR). RotOR's input is a railway timetable and its output is a set of cycles. One such cycle is called rolling stock rotation and it determines the operation of timetabled trips by railway vehicles.

As is well known, the operation of a railway system is very expensive. Thus, the good news is that we can expect millions of Euros in savings if a RotOR is used by a railway operator. The even better news is that the mathematical optimization of rolling stock rotations is still challenging simply because railway systems are typically large, complex, and tend to become even more complicated as technology progresses. Both points above prove beyond doubt that RotOR's implementation is an ideal starting point for a doctoral thesis as this document is supposed to be.

However, RotOR's implementation is far from being the primary outcome of the thesis. Implementations come and go as rolling stock and even staff does. Also useful ideas come, but do not vanish if someone denotes them. To this end, this thesis, whose chapters are subdivided into two parts and whose organization is illustrated by the following figure, has been written.



Organization of the thesis.

The first part of the thesis is dedicated to three novel ideas that we claim to be simple enough to give a direct contribution to the optimization of industrial rolling stock rotations. Consequently, they are implemented in RotOR and build its mathematical base. In fact, we find these ideas autonomous in the sense that we strongly believe in their portability to other applications, i.e., also outside railways.

We summarize the ideas including their assembly into the chapters of Part I as follows:

A Graph-Based Hypergraph as Basic Modeling Component:
Chapter 1 serves to define the rolling stock rotation problem (RSRP) that we focus on in this thesis. The RSRP is an abstract problem that defines the boundary for our optimization efforts. In spite of its formality the RSRP anticipates the main modeling idea of the thesis, i.e., to model rolling stock rotations under a hypergraph which is based on a standard graph. The hypergraph is the core of RotOR's integer programming approach and provides an elegant way to handle sophisticated railway requirements. In fact, many, if not most, of the industrial aspects for the rolling stock rotations of our application can be easily modeled by the hypergraph. In particular, the fact that railway vehicles can be coupled together to form vehicle compositions is completely captured by the hypergraph in our application. Note that, we mean by vehicle composition real-world aspects, e.g., the size of a vehicle composition, the orientation as well as position of individual vehicles, and the fleets that are used.

A Coarse-to-Fine Approach for Layered and Large Problems:
Chapter 2 explains a *coarse-to-fine* approach for the solution of large linear programs. The purpose of this approach is to gain computational benefits for problems that have different levels of detail. We call these levels *layers* and derive *coarse* layers by an aggregation of the rows of the *fine* layer (i.e., the original linear program). The technology behind our coarse-to-fine approach is dynamic column generation that we extend by algorithmic components utilizing the coarse layers. The coarse-to-fine approach is applied to the linear relaxation of RotOR's mixed-integer programming model for rolling stock rotation optimization. This coarse-to-fine implementation makes it possible to solve problems with up to 80 million hyperarc variables, see page 216.

Regional Search as a Straightforward Algorithmic Concept:
Chapter 3 is the most autonomous component of the thesis. In fact, the chapter provides an interface between rolling stock rotation optimization and classical vehicle routing. To this end, we define the resource-constrained assignment problem (RCAP) as a prototype problem. The RCAP allows for an algorithmic comparison of maintenance constraints for rolling stock rotations with structurally identical constraints for vehicle routes. Interestingly, the structural equality of these *resource constraints* does not carry over to computational tractability, i.e., railway and vehicle routing problems have large differences in hardness. A common approach in this situation is to solve the easy instances by a global search and to tackle the hard ones by a local search. Our suggestion is to develop a hybrid of local and global search, namely a *regional search*. By standalone local, regional, and global search implementations we investigate railway instances as well as vehicle routing instances. A highlight of Chapter 3 is that we compute solutions for $200 - 2$ classical vehicle routing problems from the literature which are "almost optimal on average" by a regional search algorithm.

RotOR is based on these three ideas but its model and algorithms are much more sophisticated than presented in Part I. Therefore, Part II has been written. The reason for the increased level of detail is that RotOR has been developed in a very close cooperation of DB Fernverkehr AG (DBF) and Zuse Institute Berlin (ZIB). During the cooperation relevant requirements and data for the automatic creation of rolling stock rotations within the production process of DBF have been identified, discussed, declared, defined, prepared, (re)-implemented, tested, debugged, and finally applied. The main development at ZIB took more than five years, ended with the deployment of RotOR into operation, and has strongly influenced the content of this thesis, in particular Part II.

The first two chapters of Part II explain the model that is implemented in RotOR. The model distinguishes industrial requirements that can be completely modeled by RotOR's hypergraph. These requirements are explained in Chapter 4. The remaining requirements for which we additionally need a mixed-integer programming (MIP) model (that is based on the hypergraph) are presented in Chapter 5 where we also denote RotOR's complete model that DBF uses.

The list of requirements that RotOR can handle is too long and too manifold to be introduced here. However, we like to emphasize two dedicated aspects that particularly enable RotOR to take part in industrial competitions.

The first major functionality is RotOR's ability to deal with *greenfield* as well as with *re-optimization* scenarios. Re-optimization settings exclusively distinguish from (traditional) greenfield optimization by already existing reference rotations. The reference rotations are already completely or partially implemented in the production process but have become infeasible or inefficient. On the occasion of unpredictable disruptions as well as planned construction sites the aim in re-optimization is to find new rolling stock rotations such that the differences to the reference rotations are as small as possible. Re-optimization is of more short-term, i.e., tactical nature than the more strategic greenfield optimization. Both settings have important purposes in the railway industry and are handled within RotOR in one go: A few effective but structurally slight modifications to RotOR's hypergraph completely provide the re-optimization functionality on top of greenfield optimization without increasing complexity in terms of modeling and solving.

Another very distinguished feature of RotOR is the generation of handouts for rolling stock rotations. In fact, even when RotOR finds rolling stock rotations that can be proved to be optimal RotOR does not stop computing! That's not a bug, that's a feature. In reality, it turns out that it is just not enough to compute a solution. We also have to think about how to make it easy to manage. In simple words, we simply ask how to print a rolling stock rotation on a physical paper? Our advise to people that find this question negligible or even nonsense is: If you have to find a modification in a rolling stock rotation that traverses Monday to Sunday for 50 times you will be happy to have a good handout. This will become even more important for you if you have to modify, check, and apply rotations every day, i.e., if you are a railway planner. In this sense, RotOR also provides an advise to hand out each rolling stock rotation that has been computed.

The attentive reader will wonder what kind of algorithm RotOR actually calls in order to compute the proper rolling stock rotations; or, expressed as a non-specialist:

Will RotOR compute "the" optimal solution for the problem?

The answer is neither no nor yes, it is precisely: perhaps. In fact, RotOR's algorithm for the solution of industrial RSRPs is (more or less completely) heuristic. At this point, we like to briefly defend why we have programmed a heuristic (only):

From our experience it definitely will take a long time to solve a $\mathcal{NP}$-hard optimization problem as the RSRP, more precisely various instances of it, to proven optimality. This is particularly the case in complex industrial applications where it is only a matter of time before someone comes up with an instance that can not be globally solved (i.e., including optimality proof) within a reasonable amount of computation time. Nonetheless, often the challenging applications involve optimization problems that are "resolved" every day, without knowing the meaning of $\mathcal{NP}$-completeness and, moreover, without having optimality proofs as provided by global solvers–real-life is a heuristic. Therefore, we find it worth to consider if a problem is $\mathcal{NP}$-hard and if so reasonable to prepare by good heuristics for the worst case, e.g., a combinatorially exploding branching tree without any primal solution at hand. Nevertheless, RotOR also relies on global solvers when calling them inside in sub-routines. Those calls are installed very carefully because it looks much like $\mathcal{P} \neq \mathcal{NP}$.

RotOR's algorithm for industrial RSRPs is introduced in Chapter 6 and its heuristic attitude is as follows. We model everything needed in industry by a single integrated MIP model, solve its linear programming (LP) relaxation including optimality proof, and apply various heuristics in order to overcome integer infeasible solutions. The heuristics are arranged in a rapid branching search tree, which is heuristic in itself. When RotOR's algorithms successfully terminate the best primal solution found, appropriate rotation handouts, and the lower bound from the linear programming relaxation are reported. In this way, we declare RotOR as a highly integrated solution approach for rolling stock rotation optimization that is installed and used by our cooperation partner DBF.

This thesis concludes with the last section of Chapter 7 where we illustrate a solution to a complex industrial RSRP that has been computed by RotOR and which has, indeed, been used for the operation of intercity express (ICE) vehicles of DBF. To the best of our knowledge, this is the first documented case where ICE rotations have been created by a mathematical optimization algorithm and, primary, have also been successfully implemented in operation. Thus, today, it is save to say that the mathematical optimization of rolling stock rotations can work very well, i.e., it helps!

We close the introduction with a few remarks on notation, reading order, and published papers that are associated with this thesis.

Notation

The word "train" is an often, if not the most often, used word in terms of railways. At the same time, it has a lot of meanings: A "train" can be a physical railway vehicle (e.g., a locomotive), a type of railway vehicles (e.g., the ICE of DBF considered as a train), a repeating or single entry of a railway timetable, a means of transportation, a toy for playing, and even "to train" is possible to say. In our work we use it very rarely and distinguished, i.e., we define a train in terms of the regularity requirement for the RSRP lately in Section 4.10. For the entries of a railway timetable we frequently use the word "(timetabled) trips" while we call a set of physical railway vehicles "rolling stock".

The mathematical notation that we use in this thesis is commonly known with the following slight particularities:

- ▷ We assume that all variables, coefficients, and parameters take values from the set of rational numbers (i.e., $\mathbb{Q}$) because a discussion in terms of real numbers (i.e., $\mathbb{R}$) is simply not necessary for all our purposes.

- ▷ For a directed graph (V, A) we denote the set of arcs that come into the node $v \in V$ by $A(v)^{\text{in}} = \{\, a \in A \,|\, a = (u, v) \in A\} \subseteq A$ and the set of arcs that go out of v by $A(v)^{\text{out}} = \{\, a \in A \,|\, a = (v, w) \in A\} \subseteq A$. Under this notation we always refer explicitly to A, which is made because we deal with *graph-based hypergraphs* and the notation allows to distinguish incident hyperarcs from incident standard arcs as we need it later.

- ▷ In Chapter 2 we use the symbol $[\cdot]$ to denote *coarsening projections*. The same symbol is used differently in order to denote the subset of natural numbers, i.e., $[\mathfrak{v}] := \{1, \dots, \mathfrak{v}\} \subset \mathbb{N}$ with $\mathfrak{v} \in \mathbb{N}$ in Chapter 4.

- ▷ Except for Chapter 3 the set H (T) denotes a set of hyperarcs (timetabled trips). In Chapter 3 hyperarcs (timetabled trips) do not appear and the set H (T) denotes a set of head (tail) nodes of a bipartite graph.

- ▷ The symbol $\mathcal{S}$ is used to denote a set of stations in Chapter 4 in terms of the railway infrastructure, while $\mathcal{S}$ denotes a set of segments in terms of handout optimization in Chapters 4 and 6.

- ▷ In Chapter 6 z-variables are redefined section-wise for different purposes. Of course, this is mentioned when we do this.

Even if some symbols are "recycled" among the thesis, we do not expect any danger of confusion because at least the sections associated with the different usages do not intersect.

Our practice in denoting algorithms is slightly different from other operations research publications. All algorithms are outlined by using the `listings` LaTeX package. To this end, a (free) style that is inspired by the C++ syntax, see Stroustrup (2013, [98]), has been evolved during writing. The application of the C++ syntax is very basic, i.e., it

is limited to the meaning of functions, parenthesis, and a few standard key-words (e.g., if, while, for). We feel that this notation is more useful and slightly more stylish than completely artificial "pseudo code" figures. The following basic functions are used within algorithms. They can be seen as abbreviations in order to denote standard procedures.

SOLVE $(\cdot)$: This is a wildcard function that is called in order to solve a program that is denoted as argument of the function. Which algorithm is used is mentioned in the text nearby.

COMPUTE $(\cdot)$: This is also a wildcard function which is used to indicate very basic computations. What is computed from what and how obviously derives from the parameters, context, and comments when it is used.

CHOOSE (S): This function is used to denote to arbitrarily choose one element e from the set S. The element $e \in S$ that is chosen is considered to be the return value of the function.

Note that we do not provide a complete outline for all functions that are mentioned in algorithmic figures. Some functions are just used to indicate that something needs to be done. What is meant derives from the name of the function (and the text around).

The introduced algorithmic notation is used in Chapters 2, 3, and 6. All algorithms have been implemented and computational results are provided for all of them during the thesis. Note that, all computational experiments that deal with instances of the RSRP are independently accomplished. The corresponding test sets have been chosen carefully for their respective purpose from the set of confidential instances that were provided by our industrial partner DBF at the time when the experiment has been performed. We do not see any scientific gain by unifying the test-sets unless a test-set is publicity available as it is the case for the vehicle routing instances that we investigate in Chapter 3.

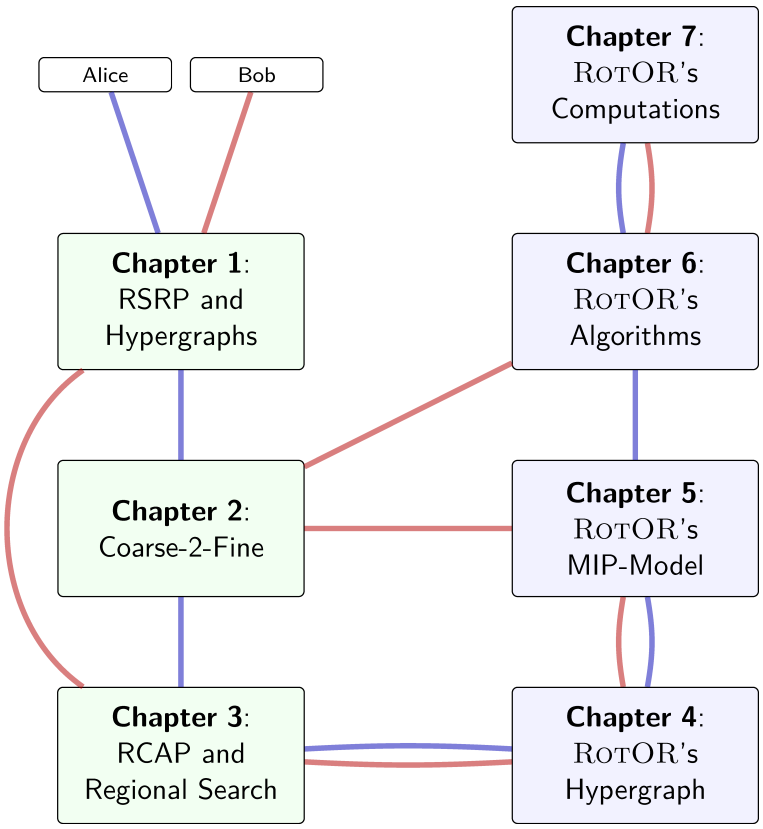
Literature is reviewed in all three second sections of the three chapters of Part I. These sections cite papers directly associated with the dedicated idea of the respective chapter. But, we also cite papers in other sections when we feel appropriateness.

Organization

Finally, we give the following recommendations in order to organize reading, i.e., if the reader can identify with Alice or Bob (i.e., the actors introduced by Rivest, Shamir, and Adleman (1978, [91])) we suggest to read the thesis according to the corresponding path in the associated figure.

Alice wants to know everything. She likes to experience the major ideas of the thesis in Part I first in order to see how these ideas are implemented in RotOR in Part II.

Bob, of course, also wants to know everything that has been written. But his primary objective is to minimize reading time. Therefore, he decides to apply a 3-opt move to Alice's approach in order to read the chapters in an order in that the ideas are slightly more based on each other.



Recommendation for the reading flow.

Publications

Wide parts of this thesis are already published in the following papers. The matching between these papers (among themselves) and the content of the thesis is mentioned during the chapters.

The *main author* of a contribution of a publication is a person who develops, implements, and writes the contribution, and finally receives and includes feedback from the other authors of the publication. In this way, the main author of this thesis is the main author of [1], [2], [3], [4], [5], [6], [7, only Section 5], [8, except Section 1, which was joint work of all authors], [9], [10, except computational results], and [11, only Section 3].

[1] Ralf Borndörfer and Markus Reuther. "Regional Search for the Resource Constrained Assignment Problem". In: *15th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2015)*. Ed. by Giuseppe F. Italiano and Marie Schmidt. Vol. 48. OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2015, pp. 111–129. ISBN: 978-3-939897-99-6. DOI: 10.4230/OASIcs.ATMOS.2015.111 (cited on pages 8, 30, 67, 100, 209).

[2] Markus Reuther. "Local Search for the Resource Constrained Assignment Problem". In: *14th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2014)*. Ed. by Stefan Funke and Matúš Mihalák. Vol. 42. OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2014, pp. 62–78. ISBN: 978-3-939897-75-0. DOI: 10.4230/OASIcs.ATMOS.2014.62 (cited on pages 8, 30, 67, 72, 86, 100, 102).

[3] Ralf Borndörfer, Markus Reuther, and Thomas Schlechte. "A Coarse-To-Fine Approach to the Railway Rolling Stock Rotation Problem". In: *14th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2014)*. Ed. by Stefan Funke and Matúš Mihalák. Vol. 42. OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2014, pp. 79–91. ISBN: 978-3-939897-75-0. DOI: 10.4230/OASIcs.ATMOS.2014.79 (cited on pages 8, 37, 39, 44, 59).

[4] Ralf Borndörfer, Markus Reuther, Thomas Schlechte, and Steffen Weider. "A Hypergraph Model for Railway Vehicle Rotation Planning". In: *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2011)*. Ed. by Alberto Caprara and Spyros Kontogiannis. Vol. 20. OpenAccess Series in Informatics (OASIcs). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2011, pp. 146–155. ISBN: 978-3-939897-33-0. DOI: 10.4230/OASIcs.ATMOS.2011.146 (cited on pages 8, 13, 18, 114, 167).

[5] Ralf Borndörfer, Markus Reuther, Thomas Schlechte, and Steffen Weider. "Vehicle Rotation Planning for Intercity Railways". In: *Proceedings of Conference on Advanced Systems for Public Transport (CASPT 2012)*. Ed. by J.C. Munoz and S. Voss. (URN links to ZIB report). 2012. URN: urn:nbn:de:0297-zib-14731 (cited on pages 8, 13, 167).

[6] Ralf Borndörfer, Markus Reuther, Thomas Schlechte, Kerstin Waas, and Steffen Weider. "Integrated Optimization of Rolling Stock Rotations for Intercity Railways". English. In: *Transportation Science* (2015). (URN links to ZIB report). DOI: 10.1287/trsc.2015.0633. URN: urn:nbn:de:0297-zib-16424 (cited on pages 8, 114, 167, 181, 210, 228).

[7] Ralf Borndörfer, Andreas Löbel, Markus Reuther, Thomas Schlechte, and Steffen Weider. "Rapid Branching". English. In: *Public Transport* (2013). (URN links to ZIB report), pp. 1–21. ISSN: 1866-749X. DOI: 10.1007/s12469-013-0066-8. URN: urn:nbn:de:0297-zib-14728 (cited on pages 8, 181, 189).

[8] Ralf Borndörfer, Marika Karbstein, Julika Mehrgardt, Markus Reuther, and Thomas Schlechte. "The Cycle Embedding Problem". In: *Operations Research Proceedings 2014*. (URN links to ZIB report). 2014. URN: urn:nbn:de:0297-zib-52788 (cited on pages 8, 32, 37, 39–41, 51, 228).

[9] Ralf Borndörfer, Julika Mehrgardt, Markus Reuther, Thomas Schlechte, and Kerstin Waas. "Re-optimization of Rolling Stock Rotations". In: *Operations Research Proceedings 2013*. (URN links to ZIB report). 2013, pp. 49–55. DOI: 10.1007/978-3-319-07001-8. URN: urn:nbn:de:0297-zib-42569 (cited on pages 8, 114, 156).

[10] Ralf Borndörfer, Boris Grimm, Markus Reuther, and Thomas Schlechte. "Template Based Re-Optimization of Rolling Stock Rotations". In: *Proceedings of Conference on Advanced Systems for Public Transport (CASPT 2015)*. 2015 (cited on pages 8, 114, 156, 161, 209).

[11] Ralf Borndörfer, Leonardo Lamorgese, Torsten Klug, Carlo Mannino, Markus Reuther, and Thomas Schlechte. "Recent Success Stories on Optimization of Railway Systems". In: *Proceedings of the IAROR conference RailTokyo*. (URN links to ZIB report). 2015. URN: urn:nbn:de:0297-zib-53726 (cited on pages 8, 209).

Part I.

Concepts

Chapter 1.

Rolling Stock Rotation Optimization by Hypergraphs

This chapter is dedicated to the rolling stock rotation problem (RSRP) that is the root of all considerations of the thesis. It is a mathematical optimization problem, although, strongly inspired by its application in the railway industry (especially at DB Fernverkehr AG (DBF)), but it has a rather formal appearance in this chapter. In fact, exactly this formality is a contribution of the chapter.

The chapter is organized as follows. We start with a high-level introduction to rolling stock rotation optimization in Section 1.1, which is followed by a review of publications from the literature in Section 1.2. In Section 1.3 we introduce graph-based hypergraphs, which we utilize for the definition of the RSRP in Section 1.4. This is followed by a constraint integer programming model in Section 1.5, which is tailored to the hypergraph idea. Finally, we show relations of the RSRP to other partially well known problems from the literature and, ultimately, summarize the outcome of the chapter.

Parts of the contributions of this chapter have been published under the following titles:

- "A Hypergraph Model for Railway Vehicle Rotation Planning" [4], and
- "Vehicle Rotation Planning for Intercity Railways" [5].

1.1. Introduction

The rolling stock, i.e., railway vehicles, is among the most expensive and limited assets of a railway operator. The rolling stock is required to operate a timetable. The implementation of a timetable by a rolling stock fleet must be done in a most efficient way to be in the black.

In order to have a master plan in operation, *rolling stock rotations* are created in advance. This is done within the production process of a railway operator. A rolling stock rotation is a cycle that covers timetabled trips. It is built in order to plan the succeeding operation of timetabled trips by railway vehicles.

A railway timetable forms the majority of the input data for rolling stock rotation optimization. The timetable consists of timetabled trips, which were created in order to provide a reliable transportation offer for freight or, in our application, for passengers of intercity express (ICE) vehicles. The creation of a railway timetable is a highly

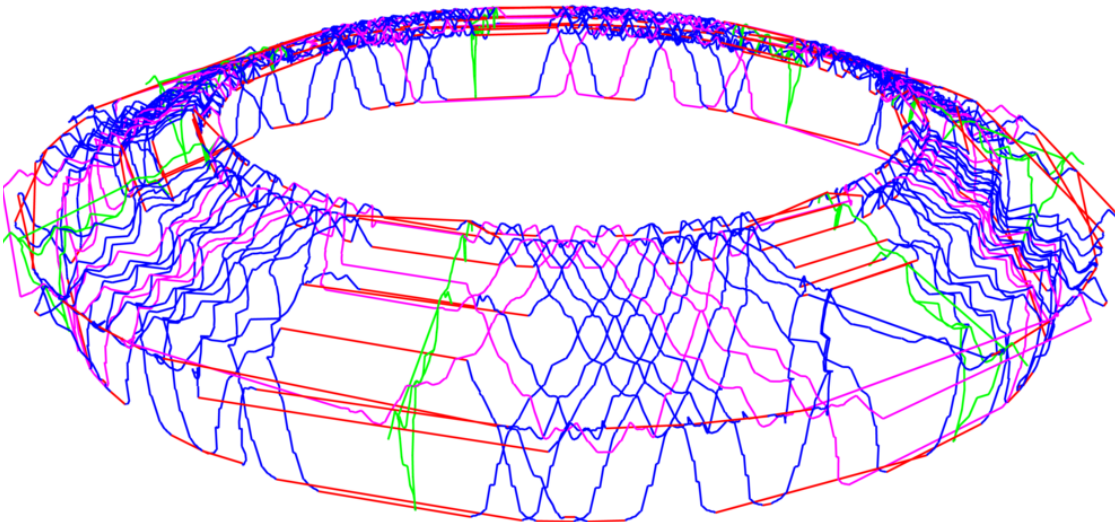


Figure 1.1.: A (small) railway timetable arranged in a torus for a cyclic standard week. The trips of the timetable are covered by two rolling stock rotations: One cycle alternates between blue paths (i.e., the trips of the timetable) and red connections. The second rolling stock rotation is purple. The seven (not six) green graphs indicate the railway infrastructure, which is utilized by railway vehicles on seven days of operation. The picture has been created using Hydraw [45].

complex and substantial task for its own, see Schlechte (2012, [94]) for a recent thesis on timetabling for railways.

Note that we consider rolling stock rotations as cycles. Indeed, this is a constraint of the so called *standard week* (concept). The standard week is a rich planning concept, which imposes the general assumption that everything will repeat after one week of operation. By this assumption we do not have to deal with precise calendar dates. Therefore, the standard week is applicable for long term (i.e., strategical considerations) as well as for mid-term (e.g., one year before operation), and is also used for tactical plans, which are executed in, say, six weeks. Thus, the standard week is an overarching planning concept for the preparation of operations. Almost each decision made in operation w.r.t. rolling stock is derived from, or at least affected by, a cyclic rolling stock rotation at DBF.

The reason for the huge amount of preparation, which railway operators spend, is simply that railway operation nowadays is and always was complicated. For that matter, our cooperation partner DBF is not an exception since it provides the largest intercity passenger railway service in Europe. In fact, the proper complexity in railway planning is only partially caused by aspects of size (e.g., the number of timetabled trips to be covered by rolling stock rotations). The reason why it is complicated and, moreover, expensive originates from the diversity of different requirements, which need to be respected entirely.

The major requirements can be summarized as *vehicle composition rules*, *maintenance constraints*, *capacity constraints*, and *regularity stipulations*. Each of these requirements

is already complex in its own right. Moreover, it can be almost impossible to treat these requirements sequentially, i.e., a step by step approach could lead to infeasibilities or inefficient results. These most important requirements can be informally described as follows:

- Figure 1.1 shows a cyclic passenger timetable that is valid on seven days of operation. For each day of operation all given passenger trips are plotted as paths arranged in a torus, in which time proceeds counterclockwise. A profile at a specific time of this torus represents the current location of all vehicles operating the timetable. As one can see in this picture, a common structure of railway timetables is that they are almost periodic, i.e., only a few of the given passenger trips differ from day to day. This implies a first requirement for the rolling stock rotations: They should reflect the periodicity of the timetable. This objective is called *regularity*.

- Another main characteristic of nearly all railway systems is that railway vehicles can be combined to form *vehicle compositions*. Among other things, the expected passenger demand on a timetabled trip leads to an individual and reasonable set of feasible vehicle compositions for each trip of the timetable. This gives many degrees of freedom for the rolling stock rotations, which have to be considered.

- The rolling stock has to be maintained frequently. This leads to several maintenance constraints with different technical backgrounds. We consider cumulative time and distance resources which are constrained by upper bounds. In order to comply to those bounds railway vehicles are required to be maintained in periodic intervals.

- Maintenance and also parking activities usually consume infrastructure and crew capacity. Both types of capacity are limited. Moreover, also the number of railway vehicles of a dedicated type (i.e., of a fleet) is not infinite as is well known. Infrastructure as well as fleet capacity constraints have to be considered explicitly in our application. Crew capacity is considered in succeeding planning steps at DBF.

This set of requirements is minimal (w.r.t. set inclusion) to be considered in the application at DBF. Moreover, their presentation here is very sparse. In reality, all of these requirements branch into many details. We address these details in Chapters 4 and 5 where we explain RotOR's integrated model.

We proceed with an overview of literature that is related to the RSRP as we consider it in this thesis.

1.2. Related Literature

In this section we give a brief but effective overview on literature that is closely related to the RSRP as we consider it. The idea of the section is not to provide an extensive survey, but to arrange the contributions of this thesis in terms of the existing literature on rolling stock optimization.

Vehicle scheduling is extensively discussed in the literature, see Löbel (1997, [74]) for a survey. Ahuja et al. (2005, [15]) present a mixed-integer programming (MIP) formulation for a locomotive scheduling problem. The model is solved by a very large-scale neighborhood search technique but does not include any maintenance constraints. In Ziarati et al. (1997, [109]) a large-scale non-linear integer programming formulation for the integrated optimization of locomotive schedules including maintenance constraints is developed. Cordeau, Soumis, and Desrosiers (2001, [38]) propose an integer programming model based on a multi-commodity flow formulation for the integrated assignment of locomotives and passenger cars to passenger trips. A three stage heuristic approach to incorporate maintenance tasks in precomputed rolling stock rosters is described in Anderegg et al. (2003, [17]). Furthermore, Maróti and Kroon (2005, [76]) and Maróti and Kroon (2007, [77]) present two integer programming formulations and alter optimized rolling stock rosters to incorporate maintenance tasks.

In Fioole et al. (2006, [47]) a MIP model for vehicle composition requirements is given. It considers detailed rules for individual positions of vehicles in compositions. The most important difference to our vehicle composition approach is that almost every timetabled trip has predefined successor trips.

An integer programming model for vehicle composition requirements, but without considering regularity or maintenance requirements, is given in Mellouli and Suhl (2007, [80]). The results presented in [80] have also been developed in cooperation with our industrial partner DBF.

Constraints that are very similar to the maintenance constraints in the RSRP come up in duty rostering problems, e.g., constraints on the maximal working time per week and the maximal number of successive working days of the drivers. Behrendt (2008, [23]) introduced several MIP formulations for the duty rostering problem including constraints on cumulative resources such as working time. We will present an adaption of one of these models for the treatment of maintenance constraints for the RSRP in Section 5.2.

Our adaptation of the model for the maintenance requirements proposed in Behrendt (2008, [23]) is mathematically equivalent to the model developed independently by Giacco, D'Ariano, and Pacciarelli (2011, [55]) for the optimization of rolling stock rosters[1]. Giacco, D'Ariano, and Pacciarelli (2011, [55]) reported computational results for scenarios of the Italian railway company Trenitalia. They assume that the railway timetables are repeated on every day of operation and they consider only the minimization of the number of vehicles as objective function. Moreover, they do not integrate vehicle composition, regularity, and infrastructure capacity in their model.

In Giacco et al. (2014, [54]) a two-step approach is presented for rolling stock rostering and maintenance scheduling by extending the contribution of [55] by another MIP model that runs on top of the rolling stock rosters. This two-step approach is also part of the thesis by Giacco (2014, [53]).

Cacchiani, Caprara, and Toth (2012, [34]) present a fast heuristic for the train unit assignment problem (which is similar to the RSRP).

Haahr et al. (2014, [59]) present a path-based model for timetables involving an acyclic

[1] A rolling stock rotation plan (as we deal with) can be seen as a rolling stock roster.

time horizon. They claim that their model can easily take maintenance constraints into account. This model and a branch-and-price algorithm are tested for re-optimization instances of the suburban railway operator in Copenhagen (DSB S-tog) without maintenance constraints. The paper [59] is part of the thesis by Haahr (2015, [58]).

Wagenaar, Kroon, and Schmidt (2016, [106]) present a model for railway rolling stock rescheduling. In their approach so called *maintenance appointments* are taken into account. The maintenance appointments are already scheduled for certain railway vehicles and are particularly required to be respected after a disruption in their application.

A comparison of the approaches presented in [106] and [59] is made by Haahr et al. (2015, [60]). Another literature comparison in terms of requirements for rolling stock optimization can be found in Thorlacius, Larsen, and Laumanns (2015, [100]). A literature overview on re-optimization can be found in Nielsen (2011, [83]) and Ahuja, Möhring, and Zaroliagis (2009, [16]).

Conclusion. At this point it must be said that the literature on rolling stock rotation optimization is rather fragmented in the sense that it is hard to identify a significant common line (as we find it in comparison to, e.g., vehicle routing, see Toth and Vigo (2014, [101])). Our explanation for this issue is that rolling stock rotation optimization is a planning problem whose concrete shape strongly depends on several side conditions. Examples for those side conditions are:

- ▷ Which railway operator of which country is involved in the optimization approach?
- ▷ What time horizon is to be considered?
- ▷ How has the timetable been constructed?
- ▷ Which railway vehicles (e.g., locomotives, carriages, or complete ICE vehicles) are to be optimized?

From that we could argue that our approach, namely RotOR, must be new because implementable rolling stock rotations for ICE vehicles have not been successfully computed before by a mathematical optimization algorithm. On the one hand, we find this argument true (but rather weak). On the other hand, the argument is not really needed here because there are indeed particularities of RotOR's model that clearly distinguish from all other publications that we mentioned:

- ▷ A time horizon of seven *individual* days of operation is considered within RotOR.
- ▷ Regularity and handout optimization are not at all investigated in the literature (to the best of our knowledge).
- ▷ RotOR's vehicle composition model is very detailed, i.e., even the orientation of railway vehicles is considered.
- ▷ RotOR's model is fully integrated.

In addition, we believe that it is save to say that the algorithmic ideas of Chapters 2, 3, and 6 are, indeed, new.

A slightly common line in the literature about rolling stock optimization is that almost all publications agree that the maintenance constraints for railway vehicles are important and, moreover, non-trivial to be tackled. In fact, there seems to be no standard approach to tackle those constraints. This is a particular motivation for Chapter 3 of this thesis.

1.3. Graph-Based Hypergraphs

This section addresses our "key technology" in modeling, i.e., to deal with hypergraphs. In fact, it kills three huge birds with one small stone: The hypergraph can be easily constructed in a way such that it completely expresses almost all requirements for vehicle configuration as well as vehicle composition and also for regularity, see Sections 4.1, 4.7, and 4.10 of Chapter 4. These requirements are essential in industry.

To the best of our knowledge, a hypergraph based approach has never been used before to solve the RSRP, not even for variants of it. We allow ourselves to mention the reference [4] to our first publication on it.

Note that the thesis by Heismann (2014, [62]) is also based on our modeling idea. It separates completely from our contributions by a theoretical consideration of a special version of the RSRP, namely the hypergraph assignment problem, see Section 1.6.

The presentation in this section is made in order to provide a high-level illustration of why a hypergraph-based model is useful. How we really use it, is the content of the next sections.

Let (V, A) be a directed graph with node set $V \neq \emptyset$ and a non-empty set of directed arcs $A \subseteq V \times V$. In order to explain the hypergraph idea with a minimized level of sophistication we consider a very simple selection problem under the assumption $|A| \geq 2$:

$$\text{Create an arbitrary non-empty subset } \mathcal{A} \subseteq A.$$

This is easy: Take an $a \in A$ and set $\mathcal{A} := \{a\}$. We now extend the selection problem by a further constraint:

$$a_1 \in \mathcal{A} \quad \Leftrightarrow \quad a_2 \in \mathcal{A} \quad \text{for} \quad a_1, a_2 \in A. \tag{1.1}$$

The situation has become more complicated under constraint (1.1). Suppose we have chosen $a \in A$ as a new element for $\mathcal{A}$. First of all, we have to check if $a \in \{a_1, a_2\}$. If not, we can finalize our selection procedure, i.e., $\mathcal{A} := \{a\}$. Otherwise (i.e., if $a \in \{a_1, a_2\}$) we can choose another $a_3 \in \mathcal{A} \setminus \{a_1, a_2\}$ and set $\mathcal{A} := \{a_3\}$ if a_3 exists. Alternatively, we can set $\mathcal{A} := \{a_1\}$ and, additionally, $\mathcal{A} := \mathcal{A} \cup \{a_2\}$.

We admit that we made an issue of constraint (1.1). But in comparison to the situation without constraint (1.1), we clearly have to spend much more effort in order to find a solution to the selection problem. In fact, the RSRP is composed of an enormous amount of constraints of type (1.1). Therefore, we spend special attention to them.

Our idea is to introduce the sets $h := \{a_1, a_2\}$ and $H := \{\{a\} \,|\, a \in A \setminus h\} \cup \{h\}$. If we now directly select from H, the constraint (1.1) is *automatically* fulfilled. We generalize this idea as follows.

Graph-based hypergraph. We define the triple (V, A, H) over the standard directed graph (V, A) with nodes V and standard directed arcs $A \subseteq V \times V$ to be a *(graph-based) hypergraph* with *(arc-based) hyperarcs* $H \subseteq 2^A$, where 2^A denotes the powerset of A.

This hypergraph definition is different from other definitions in the literature:

▷ Definition from Gallo et al. (1993, [51]):

> "A *hypergraph* is a pair H = (V, E), where V = $\{v_1, v_2, \ldots, v_n\}$ is the set of vertices (or nodes), [...]. A directed hyperedge or *hyperarc* is an ordered pair, $E = (X, Y)$, of (possibly empty) disjoint subsets of vertices; X is the tail of E while Y is its head."

▷ Definition from Cambini, Gallo, and Scutellà (1997, [35]):

> "A *directed hypergraph* is a pair $H = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, and $E = \{e_1, e_2, \ldots, e_m\}$ is the set of *hyperarcs*. A hyperarc e is a pair (T_e, h_e), where $T_e \subseteq V$ is the *tail* of e and $h_e \in V \setminus T_e$ is its *head*."

▷ Definition from Heismann (2014, [62]):

> "A *hypergraph* $G = (V, E)$ is a pair of a vertex set V and a set $E \subseteq 2^V \setminus \{\emptyset\}$ of non-empty subsets of V called *hyperedges*."

All those definitions can not directly handle constraints of type (1.1) that require us to consider hyperarcs as sets of standard arcs. This justifies our own definition.

Let $\{(u_1, v_1), \ldots, (u_k, v_k)\} \in H$ be an arc-based hyperarc according to our definition. We can interpret $\left(\bigcup_{i=1}^{k} u_i, \bigcup_{i=1}^{k} v_i\right)$ as a directed hyperedge (X, Y) as defined in [51]. In this way, our definition is more general compared to [51]. On the other hand, we especially do not consider directed hyperedges with $Y = \emptyset$ or $X = \emptyset$, namely *headless* and *tailless* hyperarcs, which is a specialization. As a result, our hypergraph definition is closely related to some other definitions introduced in the literature, but not directly comparable w.r.t. generality.

We proceed with a formal mathematical description of the problem that we aim to solve in this thesis.

1.4. The Rolling Stock Rotation Problem

The RSRP has its roots in a complex railway application. In this application it is not always completely sharp which responsibilities belong to rolling stock rotation planning and which do not. Therefore, we give a mathematical description in this section in order to provide a clearly arranged frame for the responsibilities that we take into account in this thesis. This description is rather short and formal and we refer to Chapters 4 and 5 for all the details of the RSRP.

The input data for the RSRP consists of a timetable that is composed of timetabled trips, so called *services*, a graph-based hypergraph, *maintenance constraints*, and *capacity constraints*:

Timetable. The set of timetabled passenger trips is denoted by T. A trip $t \in T$ has a departure and an arrival date in the standard week as well as a departure and arrival location.

Services. The set of *services* is denoted by S. A service $s \in S$ represents a dedicated activity performed on a vehicle, e.g., refuel and maintain a railway vehicle at Berlin.

Hypergraph. Let V be a set of *nodes* representing departures and arrivals of railway vehicles operating passenger trips of T and let $A \subseteq (V \cup S)^2$ be a set of directed standard arcs. The set of *hyperarcs* $H \subseteq 2^A$ is a subset of the power set of the standard arcs A. Thus, an element of H is a set of standard arcs. The RSRP *hypergraph* is denoted by

$$(V \cup S, A, H).$$

The standard arc $a = (u, v) \in A$ *operates* a trip $t \in T$ if $u \in V$ represents the departure of t and $v \in V$ represents the arrival of T. We say that the hyperarc $h \in H$ *covers* (or *operates*) the trip $t \in T$, if an arc $a \in h \subseteq A$ operates t. We denote the set of hyperarcs that cover the timetabled trip $t \in T$ by $H(t) \subset H$.

Maintenance constraints. The set of maintenance constraints is denoted by L. A *maintenance constraint* $l \in L$ is represented by a resource function $r_l : S \cup A \mapsto \mathbb{Q}_+$ stating the resource consumption of arcs and maintenance services, a *resource upper bound* $U_l \in \mathbb{Q}_+$, and a set of associated maintenance services $S_l \subseteq S$. A *resource path* $P \subseteq A$ is a simple path starting and ending at nodes of S_l and it is feasible w.r.t. the maintenance constraint l if P fulfills

$$\sum_{a \in P} r_l(a) + \sum_{s \in S(P) \setminus S_l} r_l(s) \leq U_l. \tag{1.2}$$

Here, $S(P) \subseteq V \cup S$ denotes the set of nodes that is covered by the path P. Note that even a service of $S(P) \setminus S_l$ can consume resources, e.g., the time needed to refuel a vehicle must also be considered in another maintenance constraint that constrains the maximal amount of time between two succeeding maintenance services. We say that each maintenance service of S_l *resets* the maintenance constraint l.

Capacity constraints. A *capacity constraint* $b \in B$ consists of a resource function $r_b : H \mapsto \mathbb{Q}$ and a *capacity bound* $U_b \in \mathbb{Q}$. Capacity constraints are used to model, e.g., infrastructure capacity, see Section 5.3. We say that the set of hyperarcs $H^\star \subseteq H$ *fulfills* the capacity constraint $b \in B$ if

$$\sum_{h \in H^\star} r_b(h) \leq U_b. \tag{1.3}$$

The output of the RSRP is a set of *feasible (rolling stock) rotations*:

Rolling stock rotation. A cycle $C \subseteq A$ is a *feasible (rolling stock) rotation* w.r.t. the maintenance constraint l if the resource consumption $\sum_{a \in C} r_l(a)$ of the whole cycle

is zero or if each node of $V \cup S$ that is covered by C is contained in a resource path that is feasible w.r.t. l. We denote by $V(C) \subseteq V \cup S$ the nodes that the cycle $C \subseteq A$ covers, i.e., $V(C) := \bigcup_{(u,v)\in A}\{u, v\}$.

We denote the RSRP as follows:

Rolling stock rotation problem (RSRP). We are given a set T of timetabled passenger trips and a hypergraph $(V \cup S, A, H)$ with a cost function $c : H \mapsto \mathbb{Q}_+$, a set L of maintenance constraints, and a set B of capacity constraints. The RSRP is

$$\min_{H^\star \subseteq H} \left\{ \sum_{h\in H^\star} c(h) \;\middle|\; \begin{array}{lll} \exists\, C_1, \ldots, C_n \subseteq A, & n \in \mathbb{N} : & \\ V(C_i) \cap V(C_j) = \emptyset & \forall\, i, j \in \{1, \ldots, n\} & \text{and} \\ C_i \text{ is a feasible rotation} & \forall\, i \in \{1, \ldots, n\},\, l \in L & \text{and} \\ \bigcup\limits_{h\in H^\star} h = \bigcup\limits_{i=1}^{n} C_i & & \text{and} \\ |H^\star \cap H(t)| = 1 & \forall\, t \in T & \text{and} \\ \sum\limits_{h\in H^\star} r_b(h) \le U_b & \forall\, b \in B & \end{array} \right\}. \quad \text{(RSRP)}$$

That is, the RSRP is to find a node-disjoint set of rolling stock rotations (which are feasible w.r.t. all maintenance constraints) that form a set of hyperarcs such that each timetabled trip is covered exactly once, all capacity constraints are fulfilled, and the cost function is minimized.

Note that any solution to the RSRP is composed of node-disjoint cycles. Therefore, a hyperarc of the form $\{(u, v), (w, v)\} \subseteq A$ can not be part of a solution because the node v would be covered twice. Therefore, we assume that each hyperarc $h \in H$ defines a perfect matching between its tail nodes $\{u \in V \mid (u, v) \in h\}$ and head nodes $\{v \in V \mid (u, v) \in h\}$.

We proceed with a mathematical program for the RSRP.

1.5. The Basic Model

In this section we introduce the basic model for the RSRP that we use as main reference in this thesis. But before we are able to denote the basic model, we need to introduce some notation and definitions.

Notation and definitions. We denote the subset of all hyperarcs covering the timetabled trip $t \in T$ as $H(t) \subset H$ and we denote by $H(a) \subseteq H$ the subset of hyperarcs that contain the standard arc $a \in A$. W.l.o.g. we assume that $H(a) \neq \emptyset$ for each arc $a \in A$. If $H(a) = \emptyset$, the standard arc a can never be contained in a feasible rotation because we are restricted to select hyperarcs for a solution. For a node $v \in V \cup S$ we denote sets of incoming as well as outgoing hyperarcs (standard arcs) of v as $H(v)^{\text{in}}$ and $H(v)^{\text{out}}$

($A(v)^{\text{in}}$ and $A(v)^{\text{out}}$), respectively. These notations are formalized by definitions (1.4):

$$\begin{array}{lcll} H(t) & := & \{h \in H \mid h \text{ covers } t\} & \text{for trip } t \in T, \\ H(a) & := & \{h \in H \mid a \in h\} & \text{for arc } a \in A, \\ H(v)^{\text{in}} & := & \{h \in H \mid \exists\, a \in h \,:\, a = (u,v)\} & \text{for node } v \in V \cup S, \\ H(v)^{\text{out}} & := & \{h \in H \mid \exists\, a \in h \,:\, a = (v,u)\} & \text{for node } v \in V \cup S, \\ A(v)^{\text{in}} & := & \{(u,v) \in A\} & \text{for arc } a \in A, \\ A(v)^{\text{out}} & := & \{(v,u) \in A\} & \text{for arc } a \in A. \end{array} \tag{1.4}$$

A constraint integer program as basic model. RotOR's *overall model* for the RSRP that we use in our industrial application is a MIP model, which is presented in Section 5.1 of Chapter 5 in its full shape. Here, we consider a more compact *basic model* for the RSRP in order to have a convenient reference in the proceeding sections. To this end, we use a constraint programming notation for the maintenance constraints. Obviously, the constraint notation does not help in solving RSRP instances, i.e., it is abstract and can be seen as a placeholder.

The reason why we denote them in this way is that we do not want to anticipate a dedicated modeling idea for these constraints. In fact, we develop two different ways to tackle the maintenance constraints: In our industrial application we use an extra MIP part. This part is sophisticated, customized for the dedicated application at DBF, and, therefore, explained later in Section 5.2 of Chapter 5 in terms of RotOR's overall industrial model. In Chapter 3 we develop an alternative, which is also based on a constraint programming formulation.

Integer programming (IP). Many approaches for combinatorial optimization problems use an IP formulation as basis. Our approach for the RSRP is not an exception. The major gain that we obtain from an IP model is the direct access to its linear programming (LP) relaxation. This is advantageous for two reasons. First, the solution of LP models of enormous size is a manageable task because a great algorithmic progress achieved during the last 70 years meets fast computers with much memory, nowadays. Second, beside the solution of LP models, the "only" remaining task in solving IP models is to force that all integer variables have to take integer values. This is also a standardized as well as computationally extensively investigated problem.

Basic model. As explained in Section 1.4 an instance of the RSRP is defined through the following data: A hypergraph $(V \cup S, A, H)$ with a cost function $c : H \mapsto \mathbb{Q}_+$, a set of maintenance constraints L with resource functions $r_l : S \cup A \mapsto \mathbb{Q}_+$ and resource upper bounds $U_l \in \mathbb{Q}_+$, and a set of capacity constraints B with resource functions $r_b : H \mapsto \mathbb{Q}_+$ and capacity bounds $U_b \in \mathbb{Q}_+$. We are now ready to present the basic model for the RSRP. We introduce a binary decision variable x_h for each hyperarc $h \in H$ that is equal to one if and only if $h \in H$ is part of the solution and denote the constraint integer programming model (BM):

$$\min \sum_{h \in H} c_h x_h, \tag{BM}$$

$$\sum_{h \in H(t)} x_h = 1 \qquad \forall\, t \in T, \tag{covering}$$

$$\sum_{h \in H(v)^{\text{in}}} x_h = \sum_{h \in H(v)^{\text{out}}} x_h \qquad \forall\, v \in V \cup S, \tag{vehicle-flow}$$

$$\sum_{h \in H} r_b(h)\, x_h \leq U_b \qquad \forall\, b \in B, \tag{rail capacity}$$

$$\textsc{Maintenance}(\, l, x\,) \qquad \forall\, l \in L,$$

$$x_h \in \{0,1\} \qquad \forall\, h \in H. \tag{x-domain}$$

Program (BM) is composed of a linear objective function, a *(hyper-) flow part*, i.e., equalities (covering) and (vehicle-flow), capacity constraints (rail capacity), and maintenance constraints as well as the integrallity constraints for the x-variables (x-domain).

Linear objective function. The objective function of model (BM) minimizes the total cost of the hyperarcs associated with x-variables that take value one. Note that the objective function is simply linear but, however, has still the ability to take distinguished costs for correlated standard arcs into account. For example, we can easily apply the model if a cost structure in which $c(\{a_1\}) + c(\{a_2\}) > c(\{a_1, a_2\})$ for standard arcs $a_1, a_2 \in A$ and hyperarcs $\{a_1\}, \{a_2\}, \{a_1, a_2\} \in H$ is required. This situation is not as easy to handle by using decision variables for standard arcs and has an important industrial application, see Section 4.4.

Flow part. The *flow part* (i.e., equalities (covering) and (vehicle-flow)) is a distinguished feature of the basic model. It is responsible to force that the solution is a set of cycles that cover the timetabled trips. Hereby, the *covering constraints* (covering) assign exactly one hyperarc to each timetabled trip $t \in T$, while the *flow conservation constraints* (vehicle-flow) in combination with the integrality constraints (x-domain) force the solution to be a set of cycles. Note that the flow part is notationally identical in terms of a standard graph but its column vectors are significantly different, i.e., they belong to hyperarcs instead of standard arcs.

Note that model (BM) allows cycles that are not node-disjoint, i.e., the constraints $\sum_{h \in H(v)^{\text{out}}} x_h \leq 1$ for all $v \in V \cup S$ is not implied (but can be included of course). In fact, these constraints are automatically fulfilled by the hypergraph construction as described in Chapter 4. There, any hyperarc either connects arrival with departure nodes (possibly via individual service nodes, see Section 4.5) or it connects departure with arrival nodes. By this construction it is not possible to cover any node more than once.

Reformulation by using standard arc variables. In order to make the modeling of cycles more clearer we reformulate the flow conservation constraints in the following way. We additionally introduce binary decision variables for standard arcs, i.e., the variable y_a takes value one if and only if the standard arc $a \in A$ is part of a solution. We couple these new arc variables to the hyperarc variables by:

$$y_a = \sum_{h \in H(a)} x_h \quad \forall\, a \in A \tag{1.5}$$

and replace the flow conservation constraints of model (BM) by equalities:

$$\sum_{a \in A(v)^{\text{in}}} y_a = \sum_{a \in A(v)^{\text{out}}} y_a \quad \forall\, v \in V \cup S. \tag{1.6}$$

By substituting all y-variables in (1.6) according to equality (1.5) we notice that we still deal with exactly the same solution space (in terms of the x-variables) of the basic model. Obviously, the standard flow conservation constraints (1.6) together with the integrality constraints for the y-variables force the solution to be a set of cycles which is exactly what we wanted to illustrate by the reformulation. In addition, we observe that the reformulation does not promise a gain because we additionally would have to deal with a new type of variables, namely the y-variables, and a large set of additional coupling constraints (1.5).

Inequalities for capacity constraints. Inequalities (1.3) are the natural formulation for the capacity constraints of the RSRP. They are latter used to constrain the capacity of fleets (i.e., the number of available railway vehicles) as well as maintenance and parking facilities, see Sections 5.3.1 and 5.3.2.

Note that the definition of the capacity constraints is very general. In order to illustrate this, we temporarily assume that $L = \emptyset$ and that each hyperarc $h \in H$ has the form $h = \{(u, v), (v, u)\}$ where the nodes $u \in V$ and $v \in V$ are associated with the departure and the arrival node of a dedicated timetabled trip. In this way, each x_h associated with $h \in H$ covers a dedicated timetabled trip and, at the same time, forms a cycle, i.e., a rolling stock rotation. Let now the resource functions and resource upper bounds of the capacity constraints act as rows of a rational matrix A and rational right-hand side vector b, respectively (such that A and b have suitable dimensions). Hence, our model appears as general binary program (BP), i.e., $\min\{c^T x \mid Ax = b,\ x \text{ binary}\}$.

This is far from what we wanted to say here, i.e., the basic model is more special than a general BP. We only use the general notation for the capacity constraints in order to mention that they exist but without spotlighting them.

Maintenance constraints. The maintenance constraints denoted in the basic model (BM) play a central role in the RSRP. In fact, they are responsible for the need of graph-based hypergraphs such that the perfect matching between the tail and head nodes of a hyperarc is precisely defined. The constraint notation of a maintenance constraint $l \in L$ in

model (BM) "only" states that the hyperarcs associated with x-variables with value one have to feasible w.r.t. l.

In the following we illustrate that an especially graph-based hypergraph is necessary to handle the maintenance constraints and illustrate how they can be exemplarily modeled by linear inequalities.

To this end, we consider a single maintenance constraint $l \in L$ with a resource function $r_l : S \cup A \mapsto \mathbb{Q}_+$ stating the resource consumption of arcs and maintenance services, a *resource upper bound* $U_l \in \mathbb{Q}_+$, and a set of maintenance services $S_l \subseteq S$ dedicated to l.

As already defined in Section 1.4, a path $P \subseteq A$ is feasible w.r.t. l if P is a simple path starting and ending at nodes of S_l such that inequality (1.2) holds. In easy words: The resource consumption cumulated through P must to not exceed U_l. Otherwise, we call P *infeasible path* w.r.t. l. According to the definition of a feasible rotation w.r.t. the maintenance constraint l given in Section 1.4, we define an *infeasible cycle* $C \subseteq A$ as a cycle that does not cover any service nodes of S_l but has a proper resource consumption, i.e., $\sum_{a \in C} r_l^a > 0$.

Let $\mathfrak{I}_l \subseteq 2^A$ be the family of all paths and cycles the are infeasible w.r.t. the maintenance constraint $l \in L$. An obvious way to implement the constraint $\textsc{Maintenance}(\, l, x\,)$ for $l \in L$ by a set of linear constraints is to use the following simple inequalities:

$$\sum_{a \in I} \sum_{h \in H(a)} x_h \leq |I| - 1 \quad \forall\, I \in \mathfrak{I}_l. \tag{1.7}$$

Inequalities (1.7) clearly show that the hypergraph indeed needs to be graph-based because the maintenance constraints require to follow paths and cycles composed of standard directed arcs, i.e., elements of the family $\mathfrak{I}_l$. Otherwise, i.e., when hyperarcs do not define the perfect matching between their tail and head nodes, those paths and cycles are not well defined.

In terms of inequalities (1.7) the RSRP can be formulated as a pure IP model. However, we do not tackle the maintenance constraints in this way. Instead, we use additional non-negative continuous variables in our industrial application. These variables are coupled to the proper hyperarc variables in a way such that the maintenance constraints are enforced, see Section 5.2.

We have seen how to derive an IP formulation for the RSRP by using hypergraphs. Indeed, the usage of hypergraphs is a relatively new modeling technology for combinatorial optimization problems that arise during the planning of transportation systems. For example, hypergraphs (although undirected) play a role in terms of line planning, see Karbstein (2013, [69]). While their hypergraph application is rather far from ours an application in terms of timetabling is, indeed, very close:

Hypergraphs in timetabling. Timetabling is the planning phase that directly precedes the planning of the rolling stock rotations. In timetabling a *railway network* $(\mathcal{S}, \mathcal{T})$ is considered as a pair of a set of *stations* $\mathcal{S}$, which are connected by a set of *(railway) tracks* $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{S}$. The railway network is used to realize a set of train movements. A *conflict* occurs if a track is occupied by more than one train movements at a certain

time. A *request* demands for a train movement through a set of stations in a railway network. Given a set of requests to be realized within a dedicated time horizon, the track allocation problem, also known as the train timetabling problem (TTP), is to find a conflict free (track) allocation, i.e., to find departure and arrival times for each request, such that the track allocation is optimal w.r.t. an objective function.

In the solution approach considered in Schlechte (2012, [94, Model PCP, page 118]) the originally continuous time horizon is discretized through a set $\{1, \dots, n\}$ of n discrete time steps. The railway network $(\mathcal{S}, \mathcal{T})$ is expanded to an acyclic directed graph (V, A) with $V \subseteq \mathcal{S} \times \{1, \dots, n\}$ and $A \subseteq V \times V$. In this standard directed graph, the set $H \subseteq 2^A$ (H is denoted as $\mathcal{Q}$ in [94]) of conflict free allocations for each single track of $\mathcal{T}$ is considered. More precisely, a $h \in H$ is associated with a $t \in \mathcal{T}$ and is a conflict free allocation of t. Even more precisely, h is a conflict free subset of standard directed arcs associated with a track $t \in \mathcal{T}$. Thus, $h \in 2^A$, i.e., h is a hyperarc of the graph-based hypergraph (V, A, H). And, indeed, in Schlechte (2012, [94]) binary decision variables y_h associated with conflict free allocations of tracks are generated within a column generation approach in order to solve the TTP. The interpretation of hyperarcs in timetabling has also been considered in the proposed way in Harrod and Schlechte (2013, [61]).

1.6. The Complexity Tree

In this section we basically give proofs for Proposition 1:

Proposition 1. *The RSRP is $\mathcal{NP}$-hard.*

To this end, we "branch" on requirements of the RSRP. Some branches end in known problems from the literature that are proven to be $\mathcal{NP}$-hard and that can be solved with an algorithm for the RSRP, while other branches illustrate relations between requirements of the RSRP.

First, we will explain what relations between problems we mean. To this end, we we define a wording that is more special compared to the existing literature, see Garey and Johnson (1979, [52]). Note that we do not want to reinvent the wheel in terms of complexity theory here. Our terminology is compatible to the existing literature (even if the following is rather informal) but is used with a stronger focus on concrete algorithms than on complexity classes.

(Linear) Problem extensions. We say that an optimization problem P_{EXT} is an *extension* of (or alternatively *extends*) another optimization problem P_{RED} if an algorithm A that can solve all instances of P_{EXT} is also "applicable" to solve all instances of P_{RED}[2]. In fact, the crux in this definition is the adjective applicable. In terms of traditional complexity theory [52], where the word "reduction" is used instead of extension, applicable means that there exists a polynomial procedure to prepare an instance I of P_{RED} for the algorithm A. There, polynomial is meant w.r.t. the encoding length $|I|$ of I. What

[2]The subscript of P_{EXT} (P_{RED}) obviously refers to extended (reduced). Unfortunately, the word "reduction" is already defined differently among the standard literature, see [52].

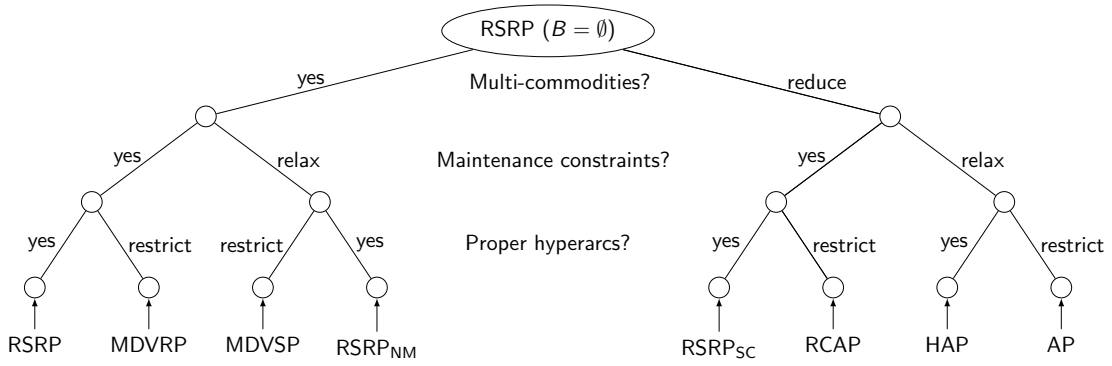


Figure 1.2.: Problems that the RSRP (without capacity constraints) extends.

we mean by "applicable" in our definition (of linear problem extensions) is that it is guaranteed that the computational effort of the preparation routine is *linear* w.r.t. the encoding length of an instance of P_{RED}.

Why do we define in this way? Suppose that we found out that P_{EXT} extends P_{RED}. If now an (even heuristic) algorithm A_H for P_{EXT} is at hand we immediately know that A_H can be used for P_{RED} where the computational effort of A_H will only slightly (i.e., linearly) increase. In this way, we avoid to consider preparation routines with a high complexity of, e.g., $\mathcal{O}(|I|^{10})$ for an instance I of P_{RED}.

Please carefully note that we use the word extension in our terminology always w.r.t. the requirements of the RSRP, i.e., we imagine to reduce, relax, or modify something (with linear computational effort) of the RSRP in order to obtain "the less complicated" P_{RED} when we consider the RSRP as an extension of some problem P_{RED}. In fact, we describe the way how to obtain P_{RED} from the RSRP in the following sections (the other way around is always obvious).

The compatibility of our terminology with the existing literature is as follows. The statement

$$P_{\text{EXT}} \text{ extends } P_{\text{RED}}.$$

reads

"P_{EXT} is a generalization of P_{RED}." or "P_{RED} is a specialization of P_{EXT}".

in many publications of the literature.

Complexity tree. The complexity tree in Figure 1.2 gives an overview of problem extensions that we like to emphasize in terms of the RSRP. It is composed of three branching layers that correspond to requirements of the RSRP. All problems denoted in the leafs of the tree are the problems that we want to consider and that the RSRP extends.

Requirement multi-commodities. The requirement *multi-commodities*[3] refers to the sets $H(t)$ for $t \in T$, namely the set of hyperarcs that can be used to cover the timetabled trip $t \in T$. If $|H(t)| > 1$ for one $t \in T$, i.e., there is more than one choice for the operation of at least one trip $t \in T$, the requirement *multi-commodity* needs to be taken into account. Otherwise we declare the multi-commodities requirement of the RSRP to be reduced, i.e., $|H(t)| = 1$ is assumed for all trips $t \in T$.

Requirement maintenance constraints. By assuming $L = \emptyset$ we relax the requirement *maintenance constraints* from the RSRP, i.e., we consider the case if there are no maintenance constraints. If we only relax the maintenance constrains from the RSRP we obtain a relaxation, which we call *non-maintenance relaxation*.

Requirement proper hyperarcs. Reducing the requirement *proper hyperarcs* from the RSRP means to restrict our considerations exclusively to hyperarcs $h \in H$ with $|h| = 1$, i.e., we consider the case where all hyperarcs are actually standard arcs. If this requirement is reduced we call the remaining problem *non-hyper restriction*.

In order to illustrate that it is indeed a restriction (and not a relaxation) we note that the non-hyper restriction of the RSRP can always be obtained for a certain instance by forcing *constraints* $x_h = 0$ for all $h \in H$ with $|h| > 1$ in program (BM).

In the non-hyper restriction of the RSRP each hyperarc $h \in H$ can be denoted as $h = \{a\}$ for $a \in A$. This notation as a trivial set is unnecessary in this case and we, therefore, assume that the sets A and H coincide, i.e., we define $H := A$ and translate the objective function $c : H \mapsto \mathbb{Q}_+$ into $c : A \mapsto \mathbb{Q}_+$.

Requirement capacity constraints. It is possible to perform further branching in the tree of Figure 1.2 by discussing if the set of capacity constraints B is empty or is not empty. But we do not expect further insights from this discussion and we always assume $B = \emptyset$ in the remaining part of this section.

We proceed with the discussion of the reduction leafs of the complexity tree in Figure 1.2 from the right to the left.

1.6.1. Standard Assignment Problem

The standard assignment problem (AP) is to find a perfect matching in a complete, bipartite, and undirected graph that minimizes some linear objective function defined for the edges.

We now show how the RSRP can be modified in order to obtain a AP (which simultaneously shows how the AP is extended by the RSRP). First of all, we force that we exclusively deal with departure and arrival nodes, i.e., the hypergraph of the RSRP does not contain service nodes.

[3] Constraints (covering) of model (BM) have the interpretation of choosing a commodity in the sense of a multi-commodity flow approach. This is where the name originates from.

By following Figure 1.2 we assume $|H(t)| = 1$ for all $t \in T$ (i.e., the operation of each timetabled trip is without alternatives) and $|h| = 1$ for all $h \in H$ (i.e., we deal exclusively with standard arcs). Thus, we can identify an unique standard arc $a_t = (u, w) \in A$ with unique $u \in V$ and unique $w \in V$ such that a_t operates the timetabled trip $t \in T$ by departing at u and arriving at w. Hence, equalities (covering) reduce to

$$x_{a_t} = 1 \quad \forall\, t \in T. \tag{1.8}$$

Thus, x_{a_t} becomes a constant for all $t \in T$ which we remember but do not denote anymore. Let the sets U and W be the two disjoint sets of departure and arrival nodes, i.e., $U := \{u \in V \,|\, a_t = (u, v) \,:\, t \in T\}$, $W := \{v \in V \,|\, a_t = (u, v) \,:\, t \in T\}$ and $V = U \cup W$. Under these assumptions program (BM) reduces to:

$$\begin{aligned}
\min \sum_{a \in A} c_a x_a & \\
\sum_{a \in A(u)^{\text{in}}} x_a = 1 & \qquad \forall\, u \in U, \\
\sum_{a \in A(w)^{\text{out}}} x_a = 1 & \qquad \forall\, w \in W, \\
x_a \in \{0, 1\} & \qquad \forall\, a \in A.
\end{aligned}$$

This is the standard integer programming formulation for the standard assignment problem over the bipartite graph $(U \cup W, A)$. In our context, it states to find a minimal cost perfect matching, i.e., an assignment, of all arrivals to all departures. This assignment forms a set of cycles, i.e., rotations that partition the nodes of the graph (V, A).

The standard formulation of the AP has the nice characteristics that each basic solution associated with its LP relaxation is automatically integral and that it can be solved by the Hungarian method, which is a $\mathcal{O}(|V|^3)$ algorithm, see Kuhn (1955, [70]).

1.6.2. Hypergraph Assignment Problem

The hypergraph assignment problem (HAP) extends the AP from the previous section in the sense that it is generalized for bipartite hypergraphs, see Heismann (2014, [62]).

In order to show how to obtain the HAP from the RSRP (i.e., how the RSRP extends the HAP), we follow the same argumentation as we did for the AP. We force that we exclusively deal with departure and arrival nodes. Again, we identify $h_t \in H$ as unique hyperarc that covers $t \in T$ following the restriction $|H(t)| = 1$ from Figure 1.2 and see that equality (covering) of program (BM) reduces to $x_{h_t} = 1$ for all $t \in T$.

In the HAP case we deal with proper hyperarcs that are possibly composed of more than one standard arc. To this end, the definitions for the sets U and W denoting the departure and arrival nodes become a very little more complicated in comparison to the AP case:

$$\begin{aligned}
U &:= \{u \in V \,|\, (u, v) \in h_t \,:\, t \in T\}, \\
W &:= \{v \in V \,|\, (u, v) \in h_t \,:\, t \in T\}.
\end{aligned}$$

Program (BM) reduces to:

$$\begin{aligned} \min \sum_{h \in H} c_h x_h & \\ \sum_{h \in H(u)^{\text{in}}} x_h = 1 & \qquad \forall\, u \in U, \\ \sum_{h \in H(w)^{\text{out}}} x_h = 1 & \qquad \forall\, w \in W, \\ x_h \in \{0,1\} & \qquad \forall\, h \in H. \end{aligned}$$

This program seems to derive from the standard integer programming formulation of the standard assignment problem by simply replacing each a by a h and each A by a H. In fact, the meaning has substantially changed: It is now a hypergraph assignment problem as introduced by Heismann (2014, [62]) with the slight notational difference that we deal with hyperarcs here instead of *hyperedges* as introduced by Heismann (2014, [62])).

The relation of hyperarcs and hyperedges is as follows. Let $h \in H$ be a hyperarc and let $U_h = \{u \in U \mid (u,v) \in h\}$ be the *tails* or *tail nodes* of h and let $W_h = \{v \in W \mid (u,v) \in h\}$ be the *heads* or *head nodes* of h. The set $U_h \cup W_h$ is the *hyperedge* that we associate with h. According to Heismann (2014, [62]) a hyperedge is defined to be an element of the set $E \subseteq 2^{U \cup W} \setminus \emptyset$ such that every hyperedge $e \in E$ has the same number $|e \cap U| = |e \cap W|$ of nodes in U and W. Thus, every hyperarc can be interpreted as a hyperedge. The reversal is not true because a hyperarc perfectly matches its tail and head nodes, in addition. However, this difference does not cause any consequence for the integer programming formulation of the HAP that we denoted above.

The HAP is proven to be $\mathcal{NP}$-hard by a reduction to the three dimensional matching problem, see Heismann (2014, [62, Theorem 3.1.1.]). However, the computational behavior of instances of the HAP is very promising since all instances of the HAP that were considered so far can be solved to proven optimality by using standard IP methods, e.g., the branch and cut (B&C) algorithm of Cplex [66]. To this end, a moderate computational effort for the separation of clique inequalities has proven useful, see Heismann and Borndörfer (2012, [63]).

1.6.3. Resource-Constrained Assignment Problem

The resource-constrained assignment problem (RCAP) is an extension of the AP and is introduced in our papers Reuther (2014, [2]) and Borndörfer and Reuther (2015, [1]). In the first instance, the RCAP is extended by the RSRP as the AP is. But the RCAP additionally takes a single maintenance constraint into account, i.e., $|L| = 1$ is assumed. The single maintenance constraint is renamed as *resource constraint* since it generalizes also other constraints, e.g., the subtour elimination constraint of the symmetric traveling salesman problem (TSP) is a resource constraint as we show in Chapter 3.

As illustrated in Section 1.6.1 a solution to the standard assignment problem is a partition of the nodes of the directed graph (V, A) into directed cycles. These cycles alternate between arcs that operate timetabled trips and arcs that connect arrival nodes to departure nodes.

The arcs that operate timetabled trips are without alternatives, i.e., each arc $a = (u, v) \in A$ that operates a timetabled trip $t \in T$ by connecting its departure node u to its associated arrival node v will definitely be part of a feasible solution. In order to streamline the notation, we merge all those departure nodes with their arrival nodes and forget all constant arc variables. Then, a constraint integer program for the RCAP reads:

$$\begin{aligned}
&\min \sum_{a \in A} c_a x_a \\
&\sum_{a \in A(v)^{\text{in}}} x_a = 1 \qquad \forall\, v \in V, \\
&\sum_{a \in A(v)^{\text{out}}} x_a = 1 \qquad \forall\, v \in V, \\
&\textsc{ResourceConstraint}(\, x \,), \\
&x_a \in \{0, 1\} \qquad \forall\, a \in A.
\end{aligned}$$

Chapter 3 is dedicated to the solution of the RCAP exactly by the program above. It will be shown that the RCAP extends several variants of the vehicle routing problem (VRP) from the literature. Here, we briefly describe how the RCAP extends the asymmetric capacitated vehicle routing problem (ACVRP). The ACVRP as considered by Fischetti, Toth, and Vigo (1994, [49]) is to find a minimal set of $k > 0$ cycles, namely *tours*, in a complete directed graph $(N \cup \{d\}, A)$ with node demands $r_v \in \mathbb{Q}$ for all $v \in N$ such that each node of N is covered exactly once by one tour, each tour covers d exactly once, $\sum_{v \in N \cap N(T)} r_v \leq B$ holds for each tour $T \subseteq A$ covering the nodes $N(T) \subseteq V$ of the solution, and the solution minimizes some linear objective function $c : A \mapsto \mathbb{Q}$. The minimal number of tours k is considered to be given in advance and the constant B is the capacity of a vehicle.

An instance of the ACVRP can be modeled as an instance of the RCAP by introducing k copies of the depot node d and by using the resource constraint to exclude infeasible cycles, i.e., tours that exceed the vehicles' capacity. More precisely, the resource function (i.e., $r_l : S \cup A \mapsto \mathbb{Q}_+$ in terms of the RSRP) of the outgoing arcs of a node is declared to model the demand of the node and the outgoing or incoming arcs of the depot copies are defined to be *replenishment arcs*. Replenishment arcs are defined in Section 3.3 and take over the role of reseting the resource constraint while traversing the arc.

The extension of the ACVRP to the RCAP shows particularly that the RCAP is $\mathcal{NP}$-hard as well. The computational behavior of instances of the RCAP varies from easily computable to completely intractable by using recent approaches. This and other objects will be shown in Chapter 3.

1.6.4. Single-Commodity RSRP

The single-commodity RSRP (RSRP_{SC}) derives from the RSRP "only" by assuming $H(t) = 1$ for all $t \in T$. Thus, constraints (covering) of model (BM) become trivially fulfilled. As a slight consequence a RSRP_{SC} may decompose into smaller RSRP_{SC} since

constraints (covering) typically couple alternative vehicle types (i.e., fleets, see Section 4.1) in our industrial application of the RSRP at DBF.

The RSRP_{SC} instances that arise in those decompositions may be HAPs or RCAPs for themselves. This shows that the RSRP_{SC} is still $\mathcal{NP}$-hard. But the reduced size (assuming a decomposing RSRP_{SC}) of these sub-problems induces a natural and useful branching rule: We first try to enforce all x-variables that are associated with hyperarcs of the set $\bigcup_{t \in T} H(t)$ within RotOR's rapid branching implementation, see Section 6.3. Therefore, the RSRP_{SC} is worth to mention even if we can not give a prominent literature citation for it.

Beside, the algorithmic value of the RSRP_{SC} we do not have any direct instances of the RSRP_{SC} arising in our railway application. At least the orientation of railway vehicles (see Section 4.6) at the departure of timetabled trips is alternative and has to be decided by RotOR. This leads to $|H(t)| \geq 2$ at least for a few timetabled trips $t \in T$ in all industrial scenarios that we have investigated during our cooperation with DBF so far.

The only situation where we might have to directly solve a RSRP_{SC} appears during the coarse phase of the cycle embedding approach (see Section 2.2 and our paper Borndörfer et al. (2014, [8])). But even there, a pure RSRP_{SC} is a rather exceptional case.

1.6.5. Non-Maintenance RSRP

The non-maintenance RSRP (RSRP_{NM}) is that problem that arises if we drop all maintenance constraints, i.e., $L = \emptyset$. Beside the relaxation of the capacity constraints, the RSRP_{NM} is the only "reduction" of the RSRP that we describe here that is simultaneously also a relaxation of the RSRP, i.e., the optimal objective value of a RSRP is always equal to or greater than the optimal objective value of its corresponding RSRP_{NM} relaxation. Of course, also other problems, e.g., the assignment problem can serve as a relaxation. But, such relaxations must be explicitly derived by defining appropriate graphs with objective functions and do not automatically appear by reducing a requirement of the RSRP.

An algorithm that assumes that there is at least one maintenance constraint for some reason can be started with an artificial maintenance constraint, i.e., by defining the resource function $r_l : A \mapsto \mathbb{Q}_+$ with $l \in L$ as $r_l(a) = 0$ for all $a \in A$ and by declaring all arcs as replenishment arcs. This suggest that it can not be assumed in general that a maintenance constraint is intrusive, i.e., the optimal objective value of the RSRP and its RSRP_{NM} relaxation may be equal or almost equal. But even if the objective function values are equal the RSRP can need significantly more computational effort than its RSRP_{NM} relaxation (simply because a feasible solution needs to be computed as well).

Form our experience we definitely can say that the RSRP_{NM} relaxation of an instance of the RSRP can be expected to be computationally much easier. The case that a maintenance constraint computationally helps (e.g., by directly implying an integer feasible solution in terms of RotOR's overall model presented in Section 5.1) has not occurred so far (to us).

In contrast to the RSRP_{SC}, the RSRP_{NM} does have proper applications in industry. In long term studies, say a scenario that will become operational in 20 years at the earliest,

the detailed maintenance constraints may not be known or already a rough estimation on the number of needed railway vehicles is sufficient. Those cases have been considered seriously during our cooperation with DBF.

It is also always interesting to see what the price of the maintenance constraints is. This is simply the difference of the optimal objective function value of a RSRP and its RSRP_{NM}. In railways this price is strove to be low. This is comprehensive because the rolling stock is expensive for its own. A rolling stock type whose operation depends on a maintenance constraint that increases strongly the operational cost is not cost efficient. Therefore the lower bound provided by the RSRP_{NM} can be expected to be good. But, of course, exceptions exist.

Since the RSRP_{NM} is an extension of the multiple depot vehicle scheduling problem, which we explain in the next section, it is $\mathcal{NP}$-hard. Nevertheless, we use RSRP_{NM} relaxations computationally within the cut date heuristic, which we explain in Section 6.4. There, it is used for a construction heuristic in which a certain RSRP_{NM} under a perturbated objective function is solved.

1.6.6. Multiple Depot Vehicle Scheduling Problem

The multiple depot vehicle scheduling problem (MDVSP) is extensively studied in the thesis of Löbel (1997, [74]) in the context of operating a set of timetabled trips T by a set of (road) vehicles, e.g., buses. It is assumed that each vehicle is initially located at a depot, whereas multiple depots D exist and the available number of vehicles in a depot $d \in D$ is given by $\kappa_d \in \mathbb{N}$. In order to denote the MDVSP a standard directed graph (V, A) is used. The node set of V is assumed to decompose by $V = D^- \cup D^+ \cup T^- \cup T^+$ such that T^+ (T^-) represents departures (arrivals) of timetabled trips and each node of D^+ (D^-) represents a starting (ending) node per depot. A vehicle schedule is defined as an elementary path in (V, A) that starts at a node of $s \in D^+$ and ends at a node of $t \in D^-$ such that s and t are associated with the same depot. An important assumption is imposed on the structure of (V, A). By construction, each elementary cycle $C \subseteq A$ in (V, A) contains exactly one so called *backward arc* $(d^-, d^+) \in A$ with $d^- \in D^-$ and $d^+ \in D^+$ and such that $C \setminus \{(d^-, d^+)\} \subset A$ is a vehicle schedule.

The MDVSP is to find a minimal cost set of feasible vehicle schedules in (V, A) such that each timetabled trip is covered by exactly one vehicle schedule under some objective function $c : A \mapsto \mathbb{Q}_+$.

In order to show how the RSRP extends the MDVSP we simply assume (see Figure 1.2) $|h| = 1$ for all $h \in H$ and $L = \emptyset$. Then, program (BM) reads:

$$
\begin{aligned}
\min \sum_{a \in A} c_a x_a, & \\
\sum_{a \in A(t)} x_a = 1 & \qquad \forall\, t \in T, \\
\sum_{a \in A(v)^{\text{in}}} x_a = \sum_{a \in A(v)^{\text{out}}} x_a & \qquad \forall\, v \in V, \\
x_a \in \{0, 1\} & \qquad \forall\, a \in A.
\end{aligned}
$$

An instance of the MDVSP can be modeled by this program as follows. We assume that (V, A) is built in such a way that a dedicated timetabled trip $t \in T$ is represented by as many pairs of departure and arrival nodes as depots with for t appropriate vehicles exist. In addition, the arcs of A exclusively connect nodes associated with equal depots.

Special attention has to be given to the decision variables $x_{(d^-,d^+)}$ that are associated with backward arcs $(d^-, d^+) \in A$. In Löbel (1997, [74]) these variables are considered to be integer. They always take the value of the number of vehicle schedules associated with a depot $d \in D$. These variables would be bounded to take value one at most by strictly applying our notation. To this end, we assume that (V, A) always contains exactly κ_d starting nodes in D^+ as well as exactly κ_d ending nodes in D^- for each depot $d \in D$. Then, all arc variables can be declared as binary. In addition, the constraints on the maximal number of vehicles per depot are also implied by the maximal number of variables associated with backward arcs that can be one at the same time by construction.

The MDVSP is proven to be $\mathcal{NP}$-hard but instances with up to $25 \cdot 10^3$ timetabled trips and $70 \cdot 10^6$ decision variables have been solved to proven optimality by an algorithm that is based on a Lagrangian relaxation of the program above, see Löbel (1997, [74, Chapters 3 and 12]).

As a side product, the relation of the RSRP to the MDVSP shows how we could deal with acyclic time horizons since the graph for the MDVSP becomes acyclic if we delete all backward arcs. In this way, on the one hand, a problem with a cyclic time horizon can be seen as an extension of a problem with an acyclic one. On the other hand, we expect that the acyclic case of the RSRP has some special industrial requirements. E.g., the situations at the beginning and at the end of the time horizon may be constrained and the duration of the time horizon could be (much) larger than the seven days of operation of the standard week. Therefore, we carefully note that our model can be applied to acyclic time horizons as well. But we do not claim any exploration of the acyclic case here.

1.6.7. Multi-Depot Vehicle Routing Problem

We refer to the multi-depot vehicle routing problem (MDVRP) according to the definition of Renaud, Laporte, and Boctor (1996, [90]). There, it is defined as follows. Given a graph (N, A_N) for that the node set N is composed of a set of customers $N_c \subset N$ ($n_c := |N_c|$) and depots $N_d \subset N$ ($n_d := |N_d|$), i.e., $N = N_c \cup N_d$. For each arc a travel time $t_a \in \mathbb{Q}_+$ is given and each customer $v \in N_c$ is associated with a demand $r_v \in \mathbb{Q}_+$ and a service time $s_v \in \mathbb{Q}_+$. It is assumed that n_c identical vehicles of capacity $Q \in \mathbb{Q}_+$, which do not have to be used at all of course, are based at each depot $d \in N_d$. The MDVRP is to find a set of cycles, called *vehicle routes*, such that: (1) each cycle contains exactly one depot of N_d, (2) each customer of N_c is visited by exactly one cycle, (3) the total demand of each cycle does not exceed the vehicle capacity, (4) the total duration of each route (including travel and service time) does not exceed a limit $L \in \mathbb{Q}_+$ and (5) the total travel time is minimized.

The MDVRP extends the ACVRP (see Section 1.6.3) by (4) and the requirement for multiple depots, which we address first. In Section 1.6.6 for the MDVSP, we already

have seen how to deal with multiple depots, i.e., we simply create copies. To this end, we create the directed graph (V, A), which we use in order to show how the MDVRP is extended by the RSRP, such that it contains exactly n_c copies for each depot of N_d. The copies are created in order to express the availability of vehicles through depots. Further, we assume that each customer $k \in N$ is represented by n_d nodes in (V, A). Analog to the consideration for the MDVSP, we assume that the graph (V, A) does not contain arcs that connect nodes associated with different depots. The objective function $c : A \mapsto \mathbb{Q}_+$ for a certain arc $a = (u, v) \in A$ can be directly derived from the MDVRP input data by defining it as the travel time between the (customer or depot) nodes u and v.

The additional constraint (4) is structurally just another resource constraint as (3) is for the ACVRP, again see Section 1.6.3. In order to distinguish (3) and (4) we simply denote them by DemandConstraint and DurationConstraint, respectively. The definitions of these two constraints directly follow the discussion for the ACVRP, i.e., customer demands, service times, as well as travel times are modeled by assigning appropriate resource functions to the arcs and by defining arcs that go into a node associated with a depot as replenishment arcs.

We define $V(k) \subseteq V$ as the set of nodes that represent the customer $k \in N_c$ and $A(k) := A(V(k))^{\text{out}}$ as set of arcs that can be traversed after customer $k \in N_c$ is visited by a vehicle that comes from a dedicated depot. Under this notation, program (BM) reduces to a formulation for the MDVRP:

$$
\begin{aligned}
\min \sum_{a \in A} c_a x_a, & \\
\sum_{a \in A(k)} x_a = 1 & \qquad \forall\, k \in N_c, \\
\sum_{a \in A(v)^{\text{in}}} x_a = \sum_{a \in A(v)^{\text{out}}} x_a & \qquad \forall\, v \in V, \\
\textsc{DemandConstraint}(\, x\,), & \\
\textsc{DurationConstraint}(\, x\,), & \\
x_a \in \{0, 1\} & \qquad \forall\, a \in A.
\end{aligned}
$$

This program corresponds to the reduction of requirements of the RSRP according to Figure 1.2. The MDVRP and variants of it are known to be computational very hard in particular in terms of optimality proofs in the literature. At the same time, modern meta-heuristics are able to compute high-quality solutions to the MDVRP, see Vidal (2013, [105, Section 3.7.2]).

1.7. Summarizing Comments

From Chapter 1 we suggest to remember the following statements for the next chapters:

- ▷ The RSRP is the optimization problem that RotOR solves.
- ▷ The RSRP is formal but it summarizes exactly all major structural requirements needed in the industrial application at DBF.
- ▷ The RSRP is recent, complicated, and has not been investigated in its full shape in the literature so far.
- ▷ The RSRP extends (i.e., generalizes) many well known problems from the literature.
- ▷ The RSRP anticipates that a hypergraph is very useful in rolling stock rotation optimization.

In fact, the proof for the last statement, which we declare as the main contribution of the chapter, is actually not at all part of Chapter 1. We address this issue by Table 1.1, which summarizes the industrial details of the RSRP that we explain in Chapters 4 and 5.

Requirement	Section	feasibility	optimality	modeling
Vehicle Configuration	4.1	x	x	hypergraph
Turn Duration Rules	4.2	x	x	hypergraph
Railway Topology	4.3	x		hypergraph
Deadhead Trips	4.4	x	x	hypergraph
Service Paths	4.5	x	x	hypergraph
Vehicle Orientation	4.6	x	x	hypergraph
Vehicle Composition	4.7	x		hypergraph
Trip Sequences	4.8	x		hypergraph
Coupling and Decoupling	4.9	x		hypergraph
Regularity Patterns	4.10		x	hypergraph
Re-optimization Templates	4.11		x	hypergraph
Objective Function	4.12		x	hypergraph
Maintenance Constraints	5.2	x		MIP
Capacity Constraints	5.3	x		MIP
Trunk Constraints	5.4	x		MIP

Table 1.1.: Industrial requirements of rolling stock rotations for ICE vehicles.

Table 1.1 lists all requirements that we consider in our industrial application. By column "feasibility" ("optimality") we indicate if the corresponding requirement has to be "primarily" considered in order avoid infeasible (suboptimal) rolling stock rotations when solving RSRP instances. But those columns are rather secondary.

The last column of Table 1.1 denotes our argument for the usefulness of the hypergraph, i.e., it denotes if the implementation of the corresponding requirement is mainly accomplished by RotOR's hypergraph or by RotOR's MIP model.

Chapter 2.

The Coarse-to-Fine Approach

In this chapter we propose a coarse-to-fine (C2F) approach to solve certain linear programs by column generation. The problems that we address contain layers corresponding to different levels of detail, i.e., coarse layers as well as fine layers. These layers are utilized to design efficient pricing rules. In a nutshell, the method shifts the pricing of a fine linear program to a coarse counterpart. In this way, major decisions are taken in the coarse layer, while minor details are tackled within the fine layer. We elucidate our methodology by applying it to the rolling stock rotation problem (RSRP). To this end, we provide comprehensive computational results that demonstrate the benefit of this new technique for the solution of large scale problems.

The chapter is organized as follows. After a motivation of our C2F idea in Section 2.1, we provide a literature review including a classification of related ideas in Section 2.2. In Section 2.3 we describe our general C2F column generation approach for linear programming. Section 2.4 introduces three layers for the RSRP that are motivated by combinatorial aspects of vehicle composition requirements (see Chapter 4) for rolling stock rotations. Our instantiation, namely the *C2F hyperarc generation algorithm*, of the C2F method for the RSRP is presented in Section 2.5. Finally, we provide comprehensive computational results for real-world instances of the RSRP given by our industrial partner DB Fernverkehr AG (DBF). We assume that the reader is familiar with column generation methods for linear programming (LP) models, see Lübbecke and Desrosiers (2005, [75]) for an introduction.

Parts of the contribution of this chapter are published under the following titles:

- ▷ "The Cycle Embedding Problem" [8], and, primary,
- ▷ "A Coarse-To-Fine Approach to the Railway Rolling Stock Rotation Problem" [3].

2.1. Motivation

The C2F approach is motivated by the RSRP as we introduced it in Chapter 1 and as we refine it for industrial application in Chapters 4 and 5. The RSRP consists of several "layers" that address different levels of detail. The major decisions of the RSRP deal with covering timetabled trips by rolling stock rotations. This is a coarse layer of the problem. At the same time minor decisions, for example, about the detailed arrival of a multi-traction vehicle composition at some station, must be considered for technical reasons. This defines a fine layer.

Suppose there is a solution for the coarse layer that has been found by ignoring the details of the fine layer. Then, it is often possible to extend this coarse solution to a solution for the fine layer, but not always. In this situation one can try to refine the coarse model locally at the critical parts. This leads to an iterative refinement approach with a model that mixes coarse and fine parts and is therefore difficult to handle.

The idea of this chapter is different. We propose to work with a version of the fine model that is restricted to a small subset of variables. This restricted model is iteratively extended by using information from the coarse model. In other words, the coarse model is used to identify the relevant decisions (i.e., variables) of the fine model by (hopefully) focusing the attention exactly to where it is needed.

The variable selection process is handled by column generation. Our idea is to work with two LP models, one for the coarse and one for the fine layer. The coarse LP model is constructed by aggregating suitable rows of the fine LP model and, hopefully, turns out to be a combinatorial optimization problem of low complexity, e.g., a network flow problem. Variables for the fine LP model are generated using the coarse LP model until convergence. This method aims at a rapid solution progress and at a complete elimination of stalling and tailing-off effects that are due to the fine layer.

Technically, our C2F idea is a row aggregation technique that is embedded into the column generation algorithm (CGA). Row aggregation is a topical research area which also has been combined with the CGA in the literature. Therefore, we compare our C2F idea to other related C2F ideas in the next section. The most distinguished feature of our approach is its flexibility: In order to define a coarse layer an arbitrary aggregation of the rows of a linear program can be used. While the computational benefit strongly depends on the concrete definition of the layers, the convergence proof of our C2F CGA is independent from it. Of course, the layers have to be defined in a meaningful way and the success of the method depends on the quality of the layering. While we offer no general theory how to do this, in many applications the layers are evident, e.g., for the RSRP. These are the applications that we have in mind.

We remark that a somewhat similar idea of separated layers is used by multi-grid methods to solve linear equation systems, see Tang et al. (2010, [99]). There, a preconditioner, which is emerged from a coarse grid, takes over the role of the coarse layer. The preconditioner improves the computational tractability.

2.2. Literature on Projective C2F Ideas

This section has three objectives. First, we aim at reviewing publications that are related to our C2F approach. Second, we particularly mention the cycle embedding approach, which was our "first shot" in the coarse-to-fine direction for the RSRP. Finally, we introduce a basic coarse-to-fine characteristic that we call *projective*.

To this end, we select mathematical programming approaches that follow the *C2F idea*. The C2F idea is to gain an algorithmic advantage by separating (i.e., coarsening) an optimization problem into coarse aspects and fine details. The coarse aspects are primarily investigated and completed by fine details when necessary. The overall goal of

C2F idea	year		exact	projective	static	decreases
Rogers et al.	1991	[92]			x	size
Schlechte et al.	2011	[95]			x	size
Borndörfer et al.	2014	[8]		x	x	size
Raphael	2001	[88]	x	x		time
Elhallaoui et al.	2005	[44]	x	x		degeneracy
Bärmann et al.	2015	[22]	x	x		size
Sections 2.3-2.5	2014	[3]	x	x	x	size

Table 2.1.: Comparison of different C2F ideas.

coarsening is to *decrease* the search space that is traversed by the algorithm. Therefore, we especially do not refer to techniques that "only" tackle dedicated constraints of a problem by, e.g., classical separation of linear inequalities, Benders decomposition, or Lagrangian relaxation.

Our distinction of the C2F idea (as denoted above) is not completely sharp. Therefore, we note that our selection might not be complete. In particular, we are not aware of a C2F idea that dominates others from the current literature. All ideas require a high level of specific problem knowledge for a proper implementation. Nevertheless, we compare the concepts by abstract characteristics as denoted in Table 2.1.

In this table column "exact" denotes whether the original optimization problem is solved to proven optimality (a no cross entry points to a heuristic). Most of all reviewed approaches more or less deal with fine and coarse solutions. The fine solutions are dedicated to the original problem while coarse "solutions", which do not respect all fine details, appear as intermediate results during the procedure. We classify a relation between the fine and coarse solutions by column "projective" of Table 2.1:

Definition 2.2.1. *We call a C2F idea* projective *if it is (in principle) possible to find for each fine solution at least one corresponding coarse solution.*

In other words, by following a projective C2F idea we do not eliminate any fine solutions, i.e., nothing is lost. The fourth column of Table 2.1 indicates whether the coarsening is created at the beginning of the procedure and remains (a non-cross means that the coarsening changes during the procedure). By the last column of Table 2.1 we denote the *primary* desired effect of coarsening. Of course, from a decreased problem size or degeneracy it can be expected that the computation time also decreases, but this is a secondary result. We proceed with the review of particular publications listed in Table 2.1. Please note that we do not understand the number of crosses per row in Table 2.1 as a quality measure.

Aggregation and disaggregation for optimization problems. Rogers et al. (1991, [92]) propose a terminological framework in which they arrange approximately 200 former

publications. A simplified version of their Aggregation and Disaggregation (AaD) framework can be summarized as (1) aggregate variables and/or constraints of an original optimization problem, (2) solve another (appropriate and related) optimization problem on the aggregated data, and (3) disaggregate the solution (for the aggregated problem) in order to obtain a solution for the original optimization problem. With a few exceptions among the 200 surveyed papers the sequence (1), (2), (3) is not iterated. Also, heuristics are typical for the aggregation and disaggregation procedures and there is usually no formal relation between the aggregated and original optimization problem. Aggregation and disaggregation (AaD) is the classical and obvious C2F idea which is very frequently applied even if the computer hardware has significantly improved a lot since 1991.

Micro-Macro transformation for railway timetabling. Schlechte et al. (2011, [95]) propose an AaD procedure for the timetabling problem, which is the preceding railway planning step for rolling stock rotations. There, the coarse optimization problem and the original fine problem are called *macroscopic* and *microscopic model*, respectively. This naming is common for the aggregation and disaggregation of railway infrastructure characteristics in publications about railway optimization and railway simulation, in particular.

The main idea is to round (i.e., to aggregate) the driving times that rolling stock requires to traverse a railway track in order to decrease the size of a time-expanded graph. In addition, aggregation routines are applied in order to aggregate block segments that are expected to be successively traversed. The (macroscopic) graph, which results from the aggregation procedures, is used to compute a railway timetable. In particular the rounding procedure for the driving times can lead to a solution for the macroscopic model that can theoretically not be realized in the microscopic model. The transition from the macroscopic to the microscopic model (i.e., the disaggregation) is usually made by a simulation model. It is interesting that it can happen that a microscopic timetable becomes infeasible w.r.t. the macroscopic model by the rounding procedure of the driving times, i.e., their idea is not projective, see Blanco and Schlechte (2014, [26]).

Cycle embedding for the RSRP. We also evaluated an AaD approach for the RSRP that we published in the paper [8], which is based on the masters thesis by Mehrgardt (2013, [79]). We call this AaD procedure *cycle embedding approach*. It is related to the C2F approach presented in this chapter and is motivated by the Micro-Macro idea for railway timetabling. The cycle embedding approach operates on the composition and configuration layer that we use as basic components of the C2F column generation procedure for the RSRP too, see Section 2.4.

But the workflow of the AaD [8] is easier and completely different. The basic C2F idea is to coarsen (i.e., to aggregate) the RSRP hypergraph (V, A, H) to a coarser hypergraph $([V], [A], [H])$, which is exactly the configuration layer as we introduce it in Section 2.4. Then, an appropriate coarse RSRP over $([V], [A], [H])$ is solved. The coarse RSRP is a relaxation of the original RSRP and its (coarse) solution does not distinguish orientations and positions of railway vehicles. In particular, the coarse solution may underestimate

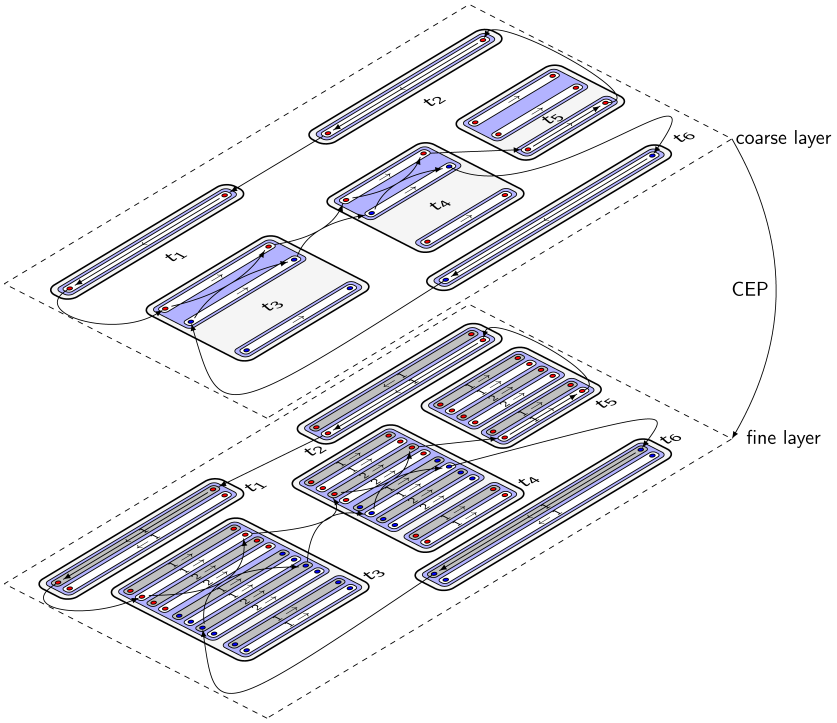


Figure 2.1.: Cycle embedding problem. The figure corresponds to Figure 2.2.

additional turn around trips, see Section 4.4.

The disaggregation of the coarse solution is the crucial part of the idea and leads to the cycle embedding problem (CEP). The CEP asks the question whether it is possible to embed a set of (coarse) hyperarcs $[H^\star] \subseteq [H]$ that form rolling stock rotations (i.e., cycles) in the coarse configuration layer into the fine composition hypergraph (V, A, H), see Figure 2.1 (which will become clear when the proper layers for the RSRP are presented in Section 2.4).

On the one hand, the CEP is $\mathcal{NP}$-hard for proper hypergraphs and polynomial when a standard directed graph is considered, see [79, 8]. On the other hand, the CEP appears computationally as a very easy problem where almost all instances that we considered turned out to be feasible. Even under aggressively provoked infeasibilities (which have been achieved by artificially increasing durations for additional turn around trips) we were also able to find feasible cycle embeddings by a cutting plane approach, see Borndörfer et al. (2014, [8]). Nevertheless, we reject the cycle embedding approach for our industrial application. The main reason is that important parts of the objective function can not be expressed in the coarse hypergraph and, therefore, can not be controlled in the cycle embedding approach. This leads to fine solutions that are far from being optimal. In particular, the requirement for regular trips (see Section 4.10.1) puts an end to the cycle embedding approach. Regularity for positions and orientations of railway vehicles at the departure of timetabled trips (i.e., the requirement that we later call regular trips, see

Section 4.10.1) is uncontrollable by only using the configuration layer because it does not distinguish orientations and positions. The cycle embedding approach is a projective C2F idea by definition.

Coarse-to-Fine dynamic programming. Raphael (2001, [88]) introduces a technique called *coarse-to-fine dynamic programming*. The methodology is demonstrated for the problem of finding a shortest path in a *trellis graph*. A trellis graph is a weighted directed graph such that "each node has an associated level and arcs can only connect nodes at adjacent levels" [88]. The idea, which is obviously projective, is to partition the nodes within levels into aggregated "superstates" that lead to a coarsened trellis graph. Each shortest path in the coarse graph underestimates a shortest path in the original graph. The proposed algorithm first computes a shortest path in the coarse graph. On the basis of this path, the superstates are iteratively refined until a shortest path in the original graph has been detected.

Dynamic constraint aggregation for set-partitioning problems. Elhallaoui et al. (2005, [44]) propose an exact algorithm for the set-partitioning problem that is based on column generation. The restricted master problem (RMP), which arises during the column generation iterations, is solved with an implementation of the primal simplex algorithm. They observe that the arising LP models are highly degenerate. This leads to high computation times of the primal simplex algorithm. They state "The number of constraints in a LP model has a greater impact on the computing performance of the simplex algorithm than its number of variables." [44]. Consequently, their idea is to reduce the number of rows in the restricted master problem in order to decrease degeneracy and, ultimately, to reduce the computation times.

Although, their approach can be applied to set-partitioning problems in general, they demonstrate it for the solution of LP relaxations that arise in an application which deals with the simultaneous vehicle and crew-scheduling problem in urban mass transit. The desired reduction of the number of rows is achieved by a partition of the original rows into clusters. In their application a cluster can be seen as an additional constraint that states that the duty elements of a cluster appear in a duty all together. The clusters are motivated by the observation that a bus driver follows a vehicle schedule for a longer period of time in optimal solutions (with high probability).

All rows of a cluster are replaced by a representative row in the RMP. This leads to a dual solution vector that is smaller than needed for the pricing algorithm. This issue is tackled by a procedure that disaggregates the dual variables for the clusters to dual variables for the original duty elements.

In this approach it can happen that a dedicated solution (also an optimal) is incompatible with a clustering. Indeed, such incompatibilities are recognized by their algorithm and used to change the clustering if appropriate. Therefore, Elhallaoui et al. (2005, [44]) present an exact C2F approach that is not static and is projective.

Iterative aggregation for network design. Bärmann et al. (2015, [22]) consider a basic network design problem that can be seen as a single-commodity flow problem with zero-objective function plus fixed charges for an extra flow capacity of an arc. This problem is tackled as a mixed-integer programming (MIP) model. Their idea is to partition the flow conservation constraints for the flow problem into clusters. Then, the equalities of each cluster are replaced with a single equality that is the sum over the original equalities of the cluster. This yields a relaxation of the original problem, which is still a MIP model. The crucial problem specific property of the approach seems to be that the flow variables have zero as objective coefficients. This makes it possible to decide if a solution to the relaxation can be extended to an *optimal* solution for the original problem by solving a simple flow problem. If such an extension is not possible, the aggregated clusters are refined in further iterations until an extension to an optimal solution is possible.

Conclusion. We reviewed several C2F publications that follow ideas that are different from the ones presented in the remaining part of this chapter. All of them deal with particular optimization problems and, especially, no generic C2F approach exists. Even if these ideas differ clearly from each other, we like to highlight the projective characteristic, which all exact approaches that we reviewed share. It is a natural property that is worth to be emphasized.

2.3. C2F Column Generation for Linear Programs

In this section, we present our C2F idea for general LP models. Our C2F idea is embedded in the CGA that we briefly recall in Section 2.3.1. For a survey on column generation techniques see Lübbecke and Desrosiers (2005, [75]). In Section 2.3.2 we explain our idea of coarsening in terms of layers. The algorithmic features are assembled to the C2F CGA, which we present in Section 2.3.3.

2.3.1. Column Generation

Given index sets $I = \{1, \dots, m\}$ and $J = \{1, \dots, n\}$, a matrix $A \in \mathbb{Q}^{I \times J}$, and vectors $b \in \mathbb{Q}^I$ and $c \in \mathbb{Q}^J$, consider a linear program

$$\min\ c^T x \quad \text{s.t.} \quad Ax = b, \quad x \in \mathbb{Q}^J_+ \tag{MP}$$

and its dual

$$\max\ b^T \pi \quad \text{s.t.} \quad A^T \pi \leq c, \quad \pi \in \mathbb{Q}^I.$$

We call program (MP) the master problem. If $|J|$ is very large, the CGA is the method of choice to solve the master problem (MP). By using the CGA one restricts J to a subset $J' \subseteq J$ of columns to solve the RMP, i.e.,

$$\min\ c^T x \quad \text{s.t.} \quad Ax = b, \quad x \in \mathbb{Q}^{J'}_+. \tag{RMP}$$

We assume x_j to be zero for $j \in J \setminus J'$ in a vector $x \in \mathbb{Q}^J$, which makes x compatible for both the MP and RMP.

In each iteration of the CGA we try to *price columns* (i.e., to find at least one column that is added to program (RMP)) $j \in J \setminus J'$ by solving the *pricing problem*. The pricing problem is to solve

$$\bar{c} := \min \left\{ c_j - \pi^T a_j \mid j \in J \right\} \tag{PP}$$

where $a_j \in \mathbb{Q}^m$ is the column vector of A for column $j \in J$ and $c_j \in \mathbb{Q}$ is the objective coefficient of column j. If $\bar{c} \geq 0$, we have a proof that an optimal solution $x^\star$ for program (RMP) is also an optimal solution for program (MP). Otherwise, we select a set of columns $J^\star \subseteq J$ such that at least one $j \in J^\star$ has negative *reduced cost* $d_j := c_j - \pi^T a_j$, add the columns associated with $J^\star$ to program (RMP), and continue with re-optimizing program (RMP).

2.3.2. Column Selection by Layers

We are free in selecting columns for the set $J^\star$ by a *column selection rule* as long as at least one element of $J^\star$ has negative reduced cost. But, it is obvious that a better column selection rule improves the efficiency of the CGA. In particular, it can be beneficial to add also columns with positive reduced cost as we will see. We address applications where J is enumerated to check every $j \in J$ whether d_j is negative, e.g., the simplex method. We call this enumeration *pricing loop*.

Our main idea is to introduce *layers* (precise definition follows) that are utilized to improve two aspects of the CGA. The first one is to speed-up the pricing loop in each iteration of the CGA. The second one is to refine the column selection rule. The latter, aims at reducing the total number of iterations performed by the CGA and to reduce the total number of columns generated.

We restrict our considerations for general LP models to two layers, namely the *coarse layer* and the *fine layer*. The *fine layer* is equal to program (MP). The coarse layer appears by the following considerations.

Let $[\cdot] : I \mapsto [I]$ be a *coarsening projection* that maps the index set I of the equations of program (MP) to a smaller *coarse index set* $[I]$ of size $|\,[I]\,| \leq |I|$. We use this notation because the projection $[\cdot]$ induces an equivalence relation on the row indices I, namely, $i \sim j \iff [i] = [j]$. Let $v \in \mathbb{Q}^I$ be a column vector with index set I and let v_i be the element of v with index $i \in I$. We define $\tau(\, v,\, i\,)$ to be the number of non-zero coefficients in v supported by rows equivalent to row i, i.e.,

$$\tau(\, v,\, i\,) := |\{ v_k \neq 0 \mid [k] = [i] \}| \,.$$

We denote the *coarse vector* or *coarsening* of v by $[v] \in \mathbb{Q}^{[I]}$. The coarse vector $[v]$ is composed of the following *coarse coefficients*

$$[v]_{[i]} := ([v]_{[i]1}\,, [v]_{[i]2}) := \\ (\,\min\{v_k \neq 0 \,|\, k \in I \,:\, [k] = [i]\},\, \max\{v_k \neq 0 \,|\, k \in I \,:\, [k] = [i]\}\,) \cdot \tau(v, i)$$

where we define[1] $[v]_{[i]} := (0, 0)$ if $\{v_k \neq 0 \,|\, k \in I \,:\, [k] = [i]\} = \emptyset$. Note that $[v]_{[i]}$ is a pair of numbers, namely, the minimal and the maximal non-zero coefficient in the set of rows

[1] This definition here is a correction of the one given in [3].

equivalent to row i, multiplied by the number of non-zero entries. Let $([A_{\cdot j}])_{j=1,\dots,|J|}$ be the bimatrix of coarse column vectors of A. Typically, this bimatrix contains identical columns caused by the coarsening projection, see Example 2.1 on page 47 and Figure 2.3. Later, in an implementation, we choose exactly one representative for a set of identical columns but we denote the resulting bimatrix by $[A]$ with columns $[J]$ and assume that $[A]$ has as many columns as A has in order to not leave the vector space of the columns of A. Further, we define the coarse objective coefficient $[c_j] := \min_{i \in J}\{c_i \mid [i] = [j]\}$ for column $j \in J$.

Let $\pi \in \mathbb{Q}^I$ be an optimal dual solution vector of program (MP) and let a_j, $j \in J$, be a column vector with objective coefficient c_j. For the ease of notation, the *coarse reduced cost* $[d]$ is defined via coefficients $[d_j] := [c_j] - [\pi]^T \cdot [a_j]$, $j \in J$, where we define the multiplication of pairs as $(a_1, b_1) \cdot (a_2, b_2) := \max\{a_1b_1,\, a_1b_2,\, a_2b_1,\, a_2b_2\}$ for two pairs $(a_1, a_2) \in \mathbb{Q}^2$ and $(b_1, b_2) \in \mathbb{Q}^2$. Note that the coarse reduced cost is *not* the coarsening of the reduced cost vector d. The *coarse reduction* (R) of the master problem (MP) is

$$\min\ [d]^T x \quad \text{s.t.}\, [A]\, x\, [=]\, [b]\,, x \in \mathbb{Q}_+^{[J]}, \tag{R}$$

where we define

$$[A]\, x\, [=]\, [b] \quad :\Leftrightarrow \quad [b]_{[i]1} \le \sum_{j \in J} [A_{\cdot j}]_{[i]2}\, x_j, \quad \sum_{j \in J} [A_{\cdot j}]_{[i]1} x_j \le [b]_{[i]2} \quad \forall\, [i] \in [I].$$

That is, the coarse reduction (R) approximates equations of the MP by two extreme case constraints arising from the minimum and maximum non-zero coefficients in equivalent rows. Note that the objective function of the coarse reduction is to minimize $[d]$ (and not c); the reason for this will become clear in the sequel. We also address the coarse reduction as *coarse LP model* and the MP as *fine LP model*.

The polytopes associated with programs (MP) and (R) are denoted by $P_{(\text{MP})}$ and $P_{(\text{R})}$, respectively. Coarsening has the following simple but important properties.

Lemma 1. *The coarse polytope associated with program* (R) *includes the fine polytope associated with program* (MP)*, i.e.,* $P_{(\text{R})} \supseteq P_{(\text{MP})}$.

Proof. Let $\sum_{i \in I:[i]=[k]} \sum_{j \in J} A_{ij} x_j = \sum_{i \in I:[i]=[k]} b_i$ for all $[k] \in [I]$ be the system of sums of equivalent rows of program (MP). Every row of (R) is a relaxation of this (already relaxed) system because each coefficient is respectively over- or underestimated. □

Lemma 2. *The coarse reduced cost always underestimate the (original) reduced cost, i.e.,*

$$[d_j] = [c_j] - [\pi]^T \cdot [a_j] \le c_j - \pi^T \cdot a_j = d_j \quad \forall\, j \in J.$$

Proof. By definition we have $[c_j] \le c_j$ and each summand in $\pi^T \cdot a_j$ is overestimated by a summand of $[\pi]^T \cdot [a_j]$. □

Lemma 1 shows that the coarse reduction (R) provides an approximation of the fine MP which has fewer rows and, thus, is probably easier to solve. We want to take advantage of

this approximation in a CGA for the fine LP model by shifting the pricing to the coarse reduction. A naive way to do this is to solve the coarse reduction w.r.t. the original objective function c (and not $[d]$ for that it is defined) in a first step, producing a set of columns

$$J^\star := \left\{ j \in J \;\middle|\; [x^\star]_{[j]} > 0 \right\} \subseteq J,$$

and then to solve the fine master LP model in a second step, starting from program (MP) restricted to the columns $J^\star$. However, this simplistic procedure is unlikely to work well because of a lack of information exchange between the coarse and the fine LP models.

2.3.3. C2F Column Generation Algorithm

Lemma 2 proposes an alternative (w.r.t. the simple approach mentioned at the end of the last section), i.e., to use the coarse reduced cost as objective function for the coarse reduction and, in addition, to prune the pricing loop by the coarse reduced cost. These generic ideas are formalized in Algorithm 2.1 that illustrates one single iteration within a CGA.

```
1  C2FColumnGeneration()
2  {
3    π* := solve((RMP)); // π* ∈ ℚ^m is an optimal dual solution of (RMP)
4
5    [π*] := compute(π*); // coarsen dual solution according to [·]
6
7    [d*] := compute([π*]); // compute coarse reduced cost
8
9    [x*] := solve((R)); // [x*] ∈ ℚ^[J] solves the coarse reduction (R) w.r.t. [d]
10
11   J* := {j ∈ J | [x*]_[j] > 0}; // select new columns for (RMP) by [x*]
12
13   if( {j ∈ J* | d_j < 0} = ∅ )
14   {
15     [J*] := {[j] ∈ [J] | [d_j] < 0}; // "pricing loop" in coarse layer
16
17     Q := {j ∈ J | [j] ∈ [J*], d_j < 0}; // "pricing loop" in fine layer
18
19     if( Q ≠ ∅ ) // Are there still columns with negative reduced cost?
20     {
21       J* := J* ∪ {choose(Q)}; // add at least one more column
22     }
23   }
24
25   return J*;
26 }
```

Algorithm 2.1: C2F column generation for linear programs (only a single iteration is outlined).

The C2F CGA, as outline by Algorithm 2.1, is based on an optimal dual solution of the fine RMP, see line 3. Then, the fine dual solution is coarsened according to the coarse projection in line 5 and the coarse reduced cost are computed in line 7.

The C2F CGA solves the fine MP by using the coarse reduction as a primary column selection rule, see lines 9 and 11 of Algorithm 2.1. The intention behind this rule is to compute a reasonable combination of (hopefully) improving columns by solving the coarse reduction. Using the coarse reduced cost as an objective aims at a "good combination" of improving columns of negative reduced cost and further columns of positive reduced cost that are "necessary" to complete the construction of the solution. Column selection by the solution of the coarse reduction is crucial for the performance of our C2F approach. Note that a similar idea (without coarsening and in the context of Lagrangian relaxation) has been introduced by Löbel (1997, [74, Section 7.1.2]) published under the name "Lagrangian pricing".

After columns have been selected through the coarse reduction, the algorithm iterates through a coarse pricing loop, see line 15. Note that this is only necessary (for the proof of convergence) if it turns out that no columns with negative reduced cost have been selected through the coarse reduction. This is indicated by the if-statement in line 13 of Algorithm 2.1. On the basis of the set $[J^\star]$ of coarse columns with negative coarse reduced cost it is straightforward to check if there are fine columns with negative reduced cost. Note that this check is pruned by the results of the coarse pricing loop. By Lemma 2 we can not miss any columns in the fine layer with negative reduced cost. That shows that the preselection by $[J^\star]$ is exact. Finally, if the set Q is not empty it is sufficient to select one element of Q in order to ensure global convergence.

Algorithm 2.1 is iteratively called until convergence. This is the general method that we propose. It works particularly well if the coarse reduction appears (better to say is arranged) as a simple combinatorial optimization problem such as a standard assignment problem (AP). We will discuss an example of this type in the context of the RSRP application in Section 2.4.3.

Consider the following matrix and coarsening projection:

$$A = \begin{pmatrix} 1 & 0 & 0 & -4 \\ 0 & 1 & 2 & 1 \end{pmatrix} \text{ and } [i] := \left\lceil \frac{i}{2} \right\rceil . \tag{2.1}$$

Then, we have $[A] = ((1,1)\,(2,2)\,(-4,1))$.

Example 2.1 shows that coarsening typically produces many identical columns, in particular, for matrices arising from combinatorial optimization problems. As already mentioned, identical columns are reduced, keeping only the copy with the smallest objective coefficient. This is a desirable effect that can produce a substantial speed-up of the coarse-to-fine pricing loop.

Coarsening rows by coarsening nodes. In this and the previous sections, we proposed to coarsen rows of an LP model such that the coarsening projection is "meaningful" w.r.t. the concrete underlying optimization problem. Even if this is the basic idea, we follow an equivalent but more indirect approach when we come the RSRP-specific approach in the

next section: We define the coarsening projections in terms of the objects of RotOR's hypergraph that are associated with RotOR's MIP model. This is simple to denote and easy to motivate (as we find it). In addition, this was the application for that the C2F idea was developed originally.

2.4. Layers for Rolling Stock Rotation Optimization

From now on, we describe the C2F approach specialized to the RSRP.

A compromise. The final goal of this section is to solve the LP relaxation of RotOR's overall MIP formulation (RMIP) for the RSRP. This model is presented later in Section 5.1 on page 169. The C2F approach is motivated by industrial vehicle composition requirements that we also explain relatively late in Chapter 4 in detail.

In order to keep the presentation compact and understandable at the same time, we apply the following compromise in this early chapter of the thesis. We explain in a way by that the reader is not required to know all details of RotOR's overall model (RMIP), even if we sometimes refer to it. Just keep in mind that this model is a large MIP formulation for that we introduce two additional coarse layers in the following.

In fact, it is enough to remember how the basis model (BM) for the non-maintenance relaxation (also without capacity constraints) of the RSRP as introduced on page 23 works in order to follow the considerations of this section. We briefly define only the required combinatorial details of the model, namely vehicle composition aspects, that we need here.

We find this "forward declaring" approach sufficient to transport the ideas and much better than assuming Chapters 4 and 5 to be already read, even if we induce a few redundancies. At the same time, this approach automatically allows to distinguish aspects of the RSRP that are important for the C2F idea from others that do not.

Vehicle composition in a nutshell. We derive the coarse layers for the RSRP from detailed combinatorial aspects of vehicle composition. Therefore, we explain what is meant by *fleet*, *vehicle orientation*, *position*, *vehicle composition*, and *vehicle configuration* in the following.

A *fleet* is a basic type of railway vehicles (e.g., the approximately 40 vehicles of the intercity express (ICE)'s first generation owned by DBF form a fleet). The set F denotes the set of fleets.

An *orientation* is an element of the set $O = \{\text{Tick}, \text{Tack}\}$. The orientation describes the two options of how vehicles can be placed on a railway track. This is distinguished at DBF by the position of the first class carriage of the vehicle w.r.t. the driving direction. Tick (Tack) means that the first class carriage is located at the head (tail) of the vehicle w.r.t. the driving direction, see Section 4.6.

A *(vehicle) composition* c of size $n \in \mathbb{N}_+$ is an n-tuple of the form

$$c = ((f_1, o_1), (f_2, o_2), \ldots, (f_n, o_n)) \in (F \times O)^n,$$

i.e., a vehicle composition defines detailed positions, orientations, and fleets of railway vehicles that are coupled together.

A *(vehicle) configuration* is a multiset of fleets and can be seen as a main characteristic of a vehicle composition. We say that the vehicle configuration k is *realized* by the vehicle composition $c = ((f_1, o_1), (f_2, o_2), \ldots, (f_n, o_n))$ if $k = \{f_1, \ldots, f_n\}$, i.e., if the multiset of fleets used in the composition c is equal to the vehicle configuration k.

In fact, the relation and, moreover, the difference between vehicle composition and vehicle configuration is important for rolling stock rotation optimization and of particularly interest for the C2F implementation.

Vehicle composition by RotOR's hypergraph. Each node $v \in V$ of RotOR's hypergraph (V, A, H) (the set of service nodes S is not necessary to be considered in the following) is constructed in such a way that it defines the fleet, the vehicle orientation, the position, and the vehicle configuration (not to be confused with vehicle composition) of a railway vehicle that traverses the node v. Each hyperarc $h \in H$ of RotOR's hypergraph is created in order to model the movement of a vehicle composition (not to be confused with vehicle configuration). The vehicle composition derives from the nodes that h connects by its standard arcs.

In fact, the main structure of model (RMIP) is implied by the underlying hypergraph (V, A, H). Therefore, it is natural to first define projections for (V, A, H) (precise definition follows) and to apply the C2F CGA afterwards. In addition, our description gains a lot of convenience from the discussion in terms of nodes, arcs, and hyperarcs. We do not need to discuss every constraint of model (RMIP) separately (even if this is necessary withing RotOR). Instead, all details (in particular the coarse coefficient matrix) as required in an implementation derive from the hypergraph projections in a straightforward way.

The starting point for a hypergraph projection is the definition of how the nodes of V are projected onto coarser nodes. Based on this definition we expose further details in the next three subsections. To this end, we frequently consult Figure 2.2, which is composed of Figures 4.3, 4.15, and 4.17 from Chapter 4. Figure 2.2 serves as a running example and shows the relation of three layers for the RSRP plus an additional figure that illustrates the vehicle configurations that are allowed to be chosen for the trips of the input timetable at the very top.

The layers, namely, a *composition layer* $G = (V, A, H)$, a *configuration layer* $[G] = ([V], [A], [H])$, and a *vehicle layer* $[[G]] = ([[V]], [[A]], [[H]])$ induce hypergraphs themselves. They are strongly motivated by the vehicle composition requirements of the RSRP.

In our application at DBF the RSRP must be solved for the composition layer, but many technical rules only apply to the configuration layer, which is much smaller w.r.t. the size of the set of hyperarcs. In addition, we define the vehicle layer to set up a supercoarse RSRP that provides a reasonable description of the major problem characteristics (e.g., the total number of railway vehicles used in a solution) and for that we design the coarse reduction such that it is combinatorially solvable in polynomial time.

We proceed with the discussion of RotOR's three layers, namely, the composition

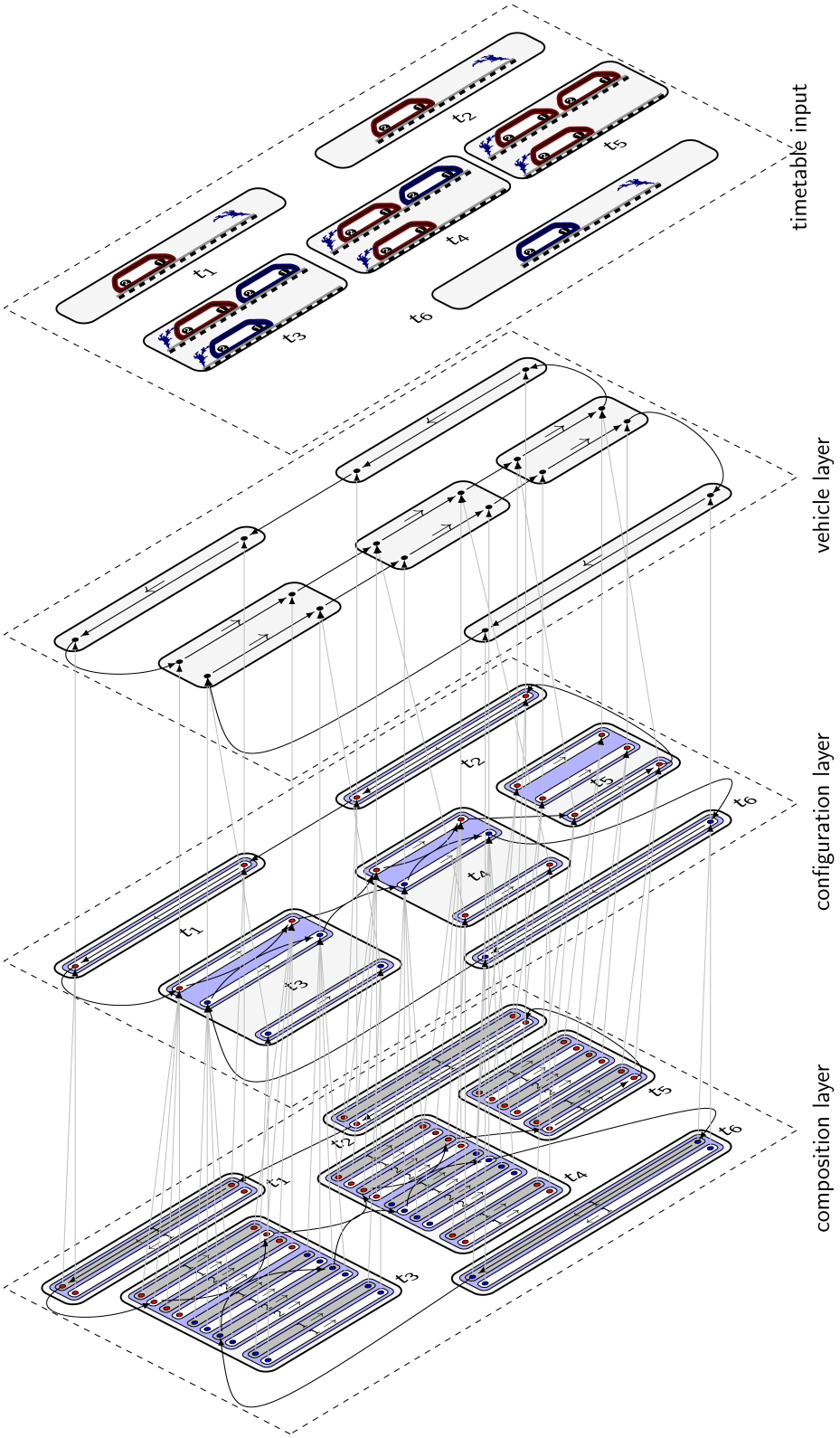


Figure 2.2.: Layers for the RSRP.

layer, the configuration layer, and the vehicle layer.

2.4.1. Composition Layer

We define the *composition layer* simply as the hypergraph (V, A, H) of the RSRP. It is illustrated on the very bottom of Figure 2.2 (which is identical to Figure 4.17). The composition layer represents the most fine layer on that the RSRP and, in particular, the LP relaxation of model (RMIP) needs to be solved.

The hypergraph (V, A, H) is constructed in such a way that each hyperarc $h \in H$ identifies a vehicle composition. This is introduced in Section 4.7 in detail. In order to allow for this construction the nodes V are particularly assembled. Since this assembly is essential for the coarsening projections we describe it in the following.

We define an *event*[2] as a triple (e, t, p) with $e \in \{d, a\}$ defining the departure $(e = d)$ or the arrival $(e = a)$ of an individual railway vehicle at position $p \in \mathbb{Z}_+$ in a vehicle composition while operating the timetabled trip $t \in T$.

A node $v \in V$ of the composition layer is of the form of a four-tuple:

$$v = ((e, t, p), k, f, o) \tag{2.2}$$

where we assume the following notation:

- ▷ (e, t, p) identifies an event,
- ▷ k denotes the vehicle configuration (see Section 4.1),
- ▷ $f \in F$ declares the fleet, and
- ▷ $o \in O = \{\text{Tick}, \text{Tack}\}$ states the vehicle orientation (see Section 4.6).

For example, let $v \in V$ be the node in the lowest line[3] in Figure 2.2. The node v can be interpreted in terms of (2.2) as follows. The event of v is $(d, t_3, 1)$, i.e., the departure of a railway vehicle at position 1 of trip t_3 (t_3 is the label given in Figure 4.17). The vehicle configuration and fleet in that a vehicle through v departs are $k = \{\text{Blue}\}$ and $f = \text{Blue}$ (this is indicated by the blue color). Finally, the orientation while departing is $o = \text{Tick}$, which is indicated by the white box that surrounds v and its corresponding arrival node.

By having nodes that are composed as denoted in (2.2) the construction of the hyperarcs of (V, A, H) is straightforward in our implementation even under detailed vehicle composition requirements. Moreover, this form is the basis for the coarsening projections that we introduce in the next sections.

At this point we want to bridge between (V, A, H) and the rows of program (RMIP). The rows of program (RMIP) that we coarsen in our implementation are:

- ▷ (vehicle-flow),

[2] We use the term "event" in the same way as we do it in the cycle embedding approach, see Section 2.2 of this chapter and the paper [8].

[3] In order to decrease confusion: The blue circle with the lowest y-coordinate in a natural Cartesian system of coordinates for Figure 2.2 is meant.

▷ (res-coupling),

▷ (res-flow), and (res-reset).

All of these rows have an associated node of V or standard arc of A. The coarsening of the rows directly follows from the coarsening (i.e., aggregation) of nodes and standard arcs that we present in the next section. More precisely, assume that $a_1 \in A$ and $a_2 \in A$ become aggregated (i.e., $[a_1] = [a_2]$ in the notation of the next section). Then, that one equalities of type (res-coupling) are aggregated that correspond to a_1 as well as a_2 and that are associated with the same maintenance constraint of L. The same distinction of maintenance constraints is made for rows of type (res-flow) and (res-reset)). The flow conservation equalities (vehicle-flow) are aggregated exactly as their corresponding nodes are aggregated.

We proceed with the configuration layer, which is a coarsening of the composition layer.

2.4.2. Configuration Layer

We create the configuration layer mainly in order to reduce the size for the RSRP hypergraph. The reduced size is achieved by simplifying vehicle compositions to vehicle configurations as we will see. The configuration layer is mainly used as a tool for pruning during our hyperarc generation algorithm, see Section 2.5. In addition, it prepares for the implementation of the coarse reduction, which is not directly derived from the configuration layer. Instead, we introduce an even more coarse layer in the next section that is based on the configuration layer.

Let (V, A, H) be the hypergraph of the composition layer. Our basic idea is to omit the orientation and position of railway vehicles from (V, A, H) in order to coarsen it. To this end, we introduce the following coarsening projections[4]:

$$\begin{aligned} [v] &:= ((e,t,p),k) && \text{for} && v = ((e,t,p),k,f,o) \in V, \\ [a] &:= ([v],[w]) && \text{for} && a = (v,w) \in A, \text{ and} \\ [h] &:= \{[a] \mid a \in h\} && \text{for} && h \in H. \end{aligned}$$

Again, let $v \in V$ be the node at the lowest line in Figure 2.2 which operates $t_3 \in T$. As illustrated the node v is projected onto a coarser node of the configuration layer for that we can not distinguish its orientation anymore. Also the eight red nodes that belong to the operation of t_5 by the vehicle configuration $\{\text{Red}, \text{Red}\}$ are coarsened. Note that

[4]Note that, from a pure formal point of view, $[v]$ means to omit the orientation o and the fleet f (instead of the position p as mentioned above) from the node $v = ((e,t,p),k,f,o) \in V$. The reason for this difference between notation and explanation is a notational sophistication: Consider $v_1 = ((d,t,1),k,f,o) \in V$ and $v_2 = ((d,t,2),k,f,o) \in V$, i.e., v_1 and v_2 "only" distinguish by the departure position of t for two vehicles of the same fleet f. Under the definition $[v] := ((e,t),k,f,p)$ (as one might expect) we have $[v_1] = [v_2]$, which would formally reduce the number of vehicles that depart at t when configuration k is used. This is not our objective. Therefore, we denote to omit the fleet instead of the position and identify the fleets of the configuration k by the positions in our implementation.

we still have four corresponding nodes in the configuration layer for them. This ensures that the number of railway vehicles that cover the timetabled trip t_5 is equal in the configuration and composition layer.

We denote the hypergraph of the configuration layer as $([V], [A], [H])$ with the canonical definitions:

$$\begin{aligned} [V] &:= \{[v] \mid v \in V\}, \\ [A] &:= \{[a] \mid a \in A\}, \text{ and} \\ [H] &:= \{[h] \mid h \in H\}. \end{aligned}$$

In this way, the configuration layer is also a graph-based hypergraph that completely determines the coarsening of rows and columns in terms of model (RMIP). Figure 2.3 illustrates this for the flow conservation constraints (vehicle-flow). To this end, some flow conservation constraints are shown on the bottom of this figure. This indicates the fine layer. Above of that, the coarse coefficient bimatrix $[A]$ of the coarse layer is illustrated (it shows only a part of $[A]$). In particular, the desired effect on the number of variables of coarsening can be seen. By coarsening rows (nodes) many columns (hyperarcs) become identical. Sets of identical coarse columns (rows) are indicated as red (gray) blocks in the coarse layer of Figure 2.3. Each red block is associated with a single coarse hyperarc.

Since the projections of the composition layer onto the configuration layer omit the orientation and the position of railway vehicles, the hyperarcs of $[H]$ can be interpreted as connections of timetabled trips with vehicle configurations (not to be confused with vehicle compositions). Indeed, this is the original motivation for defining the configuration layer. Vehicle configuration plays a much more essential role in our application than vehicle composition. Most of the time-dependent constraints, e.g., the minimal time needed for cleaning or refueling, refer "only" to the configuration layer, i.e., they are independent of the concrete vehicle composition that is realized. Of course, vehicle composition is important and necessary but it mainly causes large hypergraphs. Vehicle configuration is much more responsible for the hardness of the RSRP. An example for a particular computational hardness of the RSRP, which only results from vehicle configuration, is discussed in Section 6.5.

Finally, we close this subsection with a comparison of the size of the composition and configuration layer. To this end, we consider a vehicle configuration k (k is a multiset of fleets, see Section 4.1) that consists of the fleets $\{f_1, \ldots, f_l\} \subseteq F$ such that fleet f_i appears $m_i \in \mathbb{Z}_+$ times in k. Let C be the set of all possible vehicle compositions (see Section 4.7) that realize k. Each composition of $c \in C$ is of size $n := \sum_{i=1}^{l} m_i$. Thus, we have 2^n possibilities of different combinations of orientations in C. Furthermore, we have $n!$ possible permutations of fleets in a vehicle composition of size n. A fleet f_i that appears m_i times in k reduces this number by $m_i!$ permutations that are equal. In summary we have

$$|C| = \frac{2^n \cdot n!}{\prod_{i=1}^{n} m_i!}$$

possible vehicle compositions that realize the vehicle configuration k. For one fleet we have $|C| = 4$, for two different fleets we get $|C| = 8$, for three different fleets we get

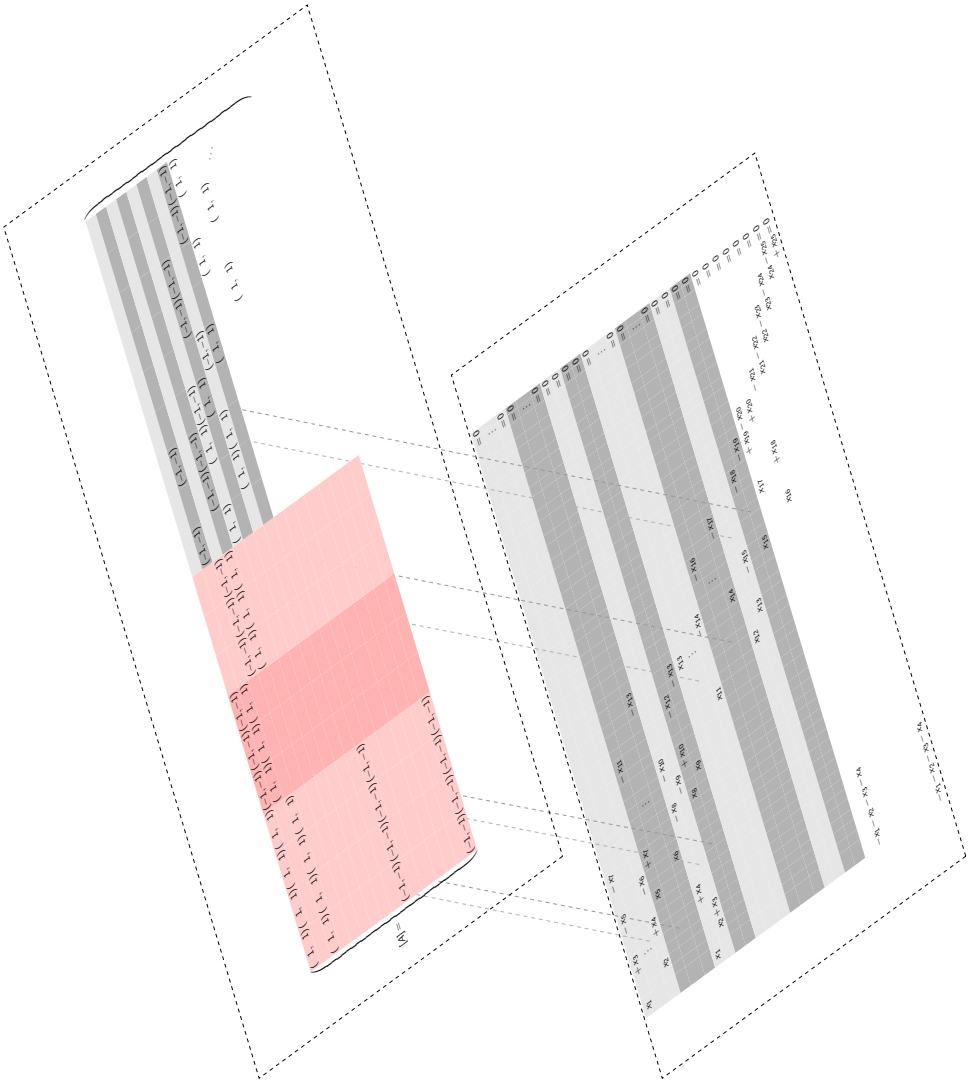


Figure 2.3.: Coarsening of flow conservation constraints (vehicle-flow) of model (RMIP) for the composition layer (V, A, H) to the configuration layer $([V], [A], [H])$.

$|C| = 48$. In this sense, the cardinality of the set of hyperarcs in the composition layer (V, A, H), which hyperarcs are associated with vehicle compositions, is exponential in the size of the set of hyperarcs in the configuration layer $([V], [A], [H])$, which hyperarcs are associated with vehicle configurations.

2.4.3. Vehicle Layer

The vehicle layer is the coarsest layer that we create for the RSRP. Its purpose is to derive an implementation of the coarse reduction, which we introduced in Section 2.3 for general LP models.

Of course, a coarse reduction could already be derived from the configuration layer. This would lead to a (more or less) general LP model which is smaller than the original master problem and can be solved with a generic LP algorithm. This may work well in other applications, but we expect that the computational benefit of this straightforward implementation of the coarse reduction is too low in our application.

Recall, that the coarse reduction is mainly solved to identify the "right" columns while generating and we are, fortunately, not strictly bound to the generic procedure proposed in Section 2.3. Therefore, our implementation of the coarse reduction is strongly related but formally not completely equal to the generic version presented in Section 2.3. More precisely, it is based on the coarse reduced cost of the configuration layer and the underlying graph of the vehicle layer is a further coarsening of the configuration layer $([V], [A], [H])$. But some details (e.g., the objective function of the coarse reduction) follow a slightly different setup.

The main motivation for our concrete implementation comes from our experience in solving RSRP instances. In these instances, a main characteristic is the number of railway vehicles that are used in a solution. Our observation is that this number is often already very well approximated if we solve a very simplified version of the original problem, namely a AP. In the following we describe how we setup this AP on the basis of the configuration layer.

Let $([V], [A], [H])$ be the hypergraph of the configuration layer and consider the following further coarsening projections:

$$\begin{aligned} [[v]] &:= (e,t,p) && \text{for} \quad [v] = ((e,t,p),k) \in [V], \text{ and} \\ [[a]] &:= ([[v]],[[w]]) && \text{for} \quad [a] = ([v],[w]) \in [A]. \end{aligned}$$

Given a configuration layer with $([V], [A], [H])$, we define the *vehicle layer* as the (graph-based hyper-) graph $([[V]], [[A]], [[H]])$ with

$$\begin{aligned} [[V]] &:= \{[[v]] \mid [v] \in [V]\}, \\ [[A]] &:= \{[[a]] \mid [a] \in [A]\}, \text{ and} \\ [[H]] &:= \{\{[[a]]\} \mid [[a]] \in [[A]]\}. \end{aligned}$$

The set $[[H]]$ is only defined for the sake of an uniform notation. It is not a set of proper hyperarcs (anymore). In fact, we disassemble the hyperarcs to standard arcs in the

vehicle layer in order to gain a coarse reduction which can be solved very quickly. This disassembly can be seen as a relaxation of a straightforward (in terms of program (R) from Section 2.3.2) coarse reduction in the sense that requiring proper hyperarcs instead of "only" standard arcs is a constraint, see Section 1.3.

Since we want to price on the basis of the standard arcs of an AP, we have to migrate the coarse reduced cost, which are still associated with hyperarcs, to them. Our idea to do this comes from the consideration that a column vector associated with a hyperarc in model (RMIP) is simply the sum of the column vectors of the standard arcs of which it consists. For example, consider the non-zero coefficients w.r.t. the flow conservation constraints (vehicle-flow) of the binary variable x_h for the hyperarc $h = \{a_1, a_2\} \subseteq A$:

$$\underbrace{\begin{pmatrix} \cdots \\ 1 \\ \cdots \\ -1 \\ \cdots \\ 1 \\ \cdots \\ -1 \\ \cdots \end{pmatrix}}_{\text{rows (vehicle-flow) w.r.t. } h} = \underbrace{\begin{pmatrix} \cdots \\ 0 \\ \cdots \\ 0 \\ \cdots \\ 1 \\ \cdots \\ -1 \\ \cdots \end{pmatrix}}_{\text{rows (vehicle-flow) w.r.t. } a_1} + \underbrace{\begin{pmatrix} \cdots \\ 1 \\ \cdots \\ -1 \\ \cdots \\ 0 \\ \cdots \\ 0 \\ \cdots \end{pmatrix}}_{\text{rows (vehicle-flow) w.r.t. } a_2}.$$

From this observation, we derive the idea to uniformly partition the contribution of the coarse reduced cost of a hyperarc pf $[H]$ to its standard arcs. In this way, we define the *partial coarse reduced cost* $[[c]] : [[A]] \mapsto \mathbb{Q}$ as:

$$[[c]]\left([[a]]\right) := \min\left\{ \frac{[d_h]}{|h|} \;\middle|\; [a] \in [h] \in [H] \right\}$$

for each arc $[[a]] \in [[A]]$. Note that this formula is based on the coarse reduced cost of the configuration layer, but $[[c]]\left([[a]]\right)$ for $[[a]] \in [[A]]$ does formally not denote reduced cost in terms of LP. Its purpose, namely to transfer the coarse reduced cost associated with hyperarcs of the configuration layer to the standard arcs of the vehicle layer, is heuristic.

As denoted, to further coarsen the node v from $[v]$ to $[[v]]$ means to also omit the vehicle configuration k from v. This has the following essential effect. In the vehicle layer $\left([[V]], [[A]], [[H]]\right)$ there is only a single option of how to cover a timetabled trip. This follows from the construction of the layers, i.e., everything that can produce alternatives (e.g., the fleet, orientation, position, or configuration of the operating railway vehicles) has been omitted by coarsening, i.e., has been projected out.

This is illustrated in Figure 2.2. For example, in the composition layer there are many options for the operation of the timetabled trip t_3, i.e., $|H(t_3)|$ is large. This is reduced to two possible vehicle configurations in the configuration layer and further simplified to two departure and arrival pairs in the vehicle layer.

Finally, we are ready to denote the *assignment pricing problem* (APP) that serves as coarse reduction in our specialized C2F CGA for the solution of the LP relaxation of

model (RMIP) for the RSRP. It reads:

$$\min \sum_{[[a]]\in[[A]]} [[c]]\left([[a]]\right) x_{[[a]]}, \tag{APP}$$

$$\sum_{[[a]]\in[[A]]([[v]])^{\text{in}}} x_{[[a]]} = 1 \qquad \forall\, [[v]] \in [[V]]\,,$$

$$\sum_{[[a]]\in[[A]]([[v]])^{\text{out}}} x_{[[a]]} = 1 \qquad \forall\, [[v]] \in [[V]]\,,$$

$$x_{[[a]]} \in \{0,1\} \qquad \forall\, [[a]] \in [[A]]\,.$$

The decision variables in program (APP) correspond to the standard arcs of the vehicle layer. A solution of the assignment pricing problem (APP) is a set of cycles that cover all the departure and arrival pairs of the vehicle layer.

Note that program (APP) forces the timetabled trip t_3 in Figure 2.2 to be covered two times. This is not necessarily optimal in a solution to the original master problem since we can choose between the vehicle configurations {Red, Blue} and {Blue} for the operation of t_3. We avoid this issue, which would lead to a C2F approach that is not projective in the sense of Section 2.2, as follows. Let $[[u]]$ be the departure node of t_3 that is associated with position two in the vehicle layer and let $[[v]]$ by the corresponding arrival node. We simply (re-)define $[[c]]\left(\left([[v]]\,,[[u]]\right)\right) := 0$ in order to allow a solution of the assignment pricing problem to artificially cover the second position of t_3 by $x_{([[v]],[[u]])} = 1$.

The assignment pricing problem (APP) is an AP, which we solve by the Hungarian method [70] in RotOR's implementation. In addition, since we formulated this problem over the set of standard arcs, the problem (APP) typically decomposes into several independent smaller sub-problems that distinguish by the fleets. That is because there is no standard arc in A that connects different fleets. This carries over to $[[A]]$.

2.5. C2F Hyperarc Generation

Algorithm 2.2 summarizes our specialization of the general C2F method for model (RMIP) of the RSRP. We call this algorithm *C2F hyperarc generation algorithm*. The notation for the RMP that is used in Algorithm 2.2 is explained and motivated later in Section 6.1. We denote the RMP associated with model (RMIP) by $\mathrm{LP}\left(\emptyset, \underline{H}\right)$ where the set $\underline{H} \subseteq H$ identifies the set of hyperarcs (of the composition layer) that have *not been generated yet*.

The Algorithm 2.2 states a *single* column generation iteration. It computes the set $H^\star \subseteq H$ of new hyperarc variables. Therefore, Algorithm 2.2 is called until convergence (i.e., $H^\star = \emptyset$) in our implementation[5]. After an optimal dual solution vector of the RMP has been computed (which can be achieved using standard LP solvers) we coarsen the dual solution vector and compute the coarse reduced cost according to the configuration layer. Then, the proper generation of hyperarcs is accomplished by two strategies.

[5]Note that we call the C2F hyperarc generation algorithm as a sub-routine in Algorithm 6.2. There, we extend the C2F hyperarc generation with the simultaneous generation of the coupling inequalities (res-coupling).

```
1  C2FHyperarcGeneration ( π⋆ )
2  {
3    // The procedure assumes:
4    //
5    // ▷ π⋆ ∈ ℚ^m as an optimal dual solution vector for LP(∅, H̲)
6    //
7    // ▷ (V, A, H \ H̲), as restriced composition layer
8    // ▷ ([V], [A], [H]) as configuration layer
9    // ▷ ([[V]], [[A]], [[H]]) as vehicle layer
10
11   [π⋆] := compute ( π⋆ ); // coarsen duals for configuration layer
12
13   // coarsen reduced cost according to configuration layer
14   [d] := compute ( [π⋆] ); // [d] ∈ ℚ^|[H]|
15
16   // solve the assignment problem in vehicle layer
17   [[x]]⋆ := solve ( (APP) ); // [[x]]⋆ ∈ ℚ^|[[A]]|
18
19   // "price" according to [[x]]⋆
20   H⋆ := { h ∈ H | ∃ a ∈ h : [[x]]⋆_[[a]] = 1 };
21
22   // price by enumeration in composition layer and
23   // prune by configuration layer
24   for ( v ∈ V )
25   {
26     // sort hyperarcs with negative reduced cost that
27     // go out of v by reduced cost;
28     // this can be pruned by [d] of configuration layer
29     {h_1, h_2, ..., h_n} := { h | h ∈ H(v)^out and
30                                   [d_h] < 0 and
31                           d_{h_i} ≤ d_{h_j} < 0 for i < j ≤ n };
32
33     if ( n > 0 ) { H⋆ := H⋆ ∪ { h_1, ..., h_⌈∛n⌉ }; }
34   }
35
36   return H⋆; // return generated hyperarcs
37 }
```

Algorithm 2.2: C2F hyperarc generation for the RSRP.

By the first strategy, which is denoted in lines 15 to 21, we solve the assignment pricing problem (see Section 2.4.3) defined by the vehicle layer with the partial coarse reduced cost function $[[c]]$. Then, we add all hyperarcs to $H^\star$ that correspond to an arc of the optimal solution of the assignment pricing problem (APP). Note that the solution to program (APP) is integral (i.e., provides a discrete selection) and that $H^\star$ may also contain columns that have positive reduced cost, i.e., for that the corresponding dual constraints are not (yet) violated in terms of the current column generation round. This acts as an efficient column selection strategy and is our implementation of the coarse reduction (R) as we introduce it in Section 2.3.

By the second strategy (which starts at line 21 of Algorithm 2.2) we perform a node wise enumeration of hyperarcs of the composition layer with negative reduced cost. If a node has n outgoing hyperarcs with negative reduced cost, we add the $\lceil \sqrt[3]{n} \rceil$ "best" ones to $H^\star$. This enumeration, i.e., the pricing loop, is performed by using the configuration layer as a pruning criterion, see line 30 of Algorithm 2.2. More precisely, we only have to consider hyperarcs $h \in H$ of the composition layer that have negative coarse reduced cost $[d_h]$ in the configuration layer, see Lemma 2.

2.6. Computational Study

In this section we present our computational study of the C2F approach for the RSRP. The computations that we present here are identical with the experiments that we published in the paper [3]. However, our discussion of these experiments and results in this section is more comprehensive than in [3] where a page limit had to be respected.

Our C2F hyperarc generation algorithm is a procedure to solve the LP relaxation of model (RMIP), i.e., it is an LP method. Therefore, it is natural to compare to standard LP methods. To this end, we consider the implementation of the interior point algorithm (IPA) for LP models of the standard software CPLEX[6] version 12.5.0.1. We observed that implementations of the simplex algorithm are not appropriate for ROTOR's model. Those methods seem to suffer dramatically from a high level of degeneracy. The performance loss when using simplex algorithms is such high that even their warm start capabilities become useless in our situation. Consequently, we do not use a crossover algorithm (which is usually followed by simplex iterations) to compute a basic solution on the basis of an interior point solution.

As is well known, solving LP relaxations of integer programming (IP) models is an intermediate step to obtain integer feasible solutions. In this respect our approach is not an exception, i.e., the solution of the C2F hyperarc generation algorithm might be fractional. Based on such possibly fractional solutions, we compute integer feasible solutions by an IP method. This IP method is a *heuristic algorithm* that is specialized for industrial RSRP instances, see Chapter 6. The initial motivation for the development of the C2F hyperarc generation algorithm originates from this IP method. We expect to gain smaller computation times and, primarily, to be even able to compute RSRP instances that are too large for the memory of a conventional computer by reducing the size of the arising LP models.

Running a heuristic IP algorithm on top of different exact LP algorithms can lead to dubious results. Especially the rapid branching scheme (see Section 6.3) where we apply an objective perturbation is ill-conditioned w.r.t. slightly different fractional solutions. More precisely, it is generally possible that a very slow LP method can lead to a very fast overall IP procedure. An exemplary reason for this is simply because a particular fractional solution can immediately lead to a "good" branching node within the branching tree if one has luck.

[6] We made a similar test with the IPA implementation of GUROBI [57] version 5.0 that showed almost equal results.

In order to avoid such blurring effects on the computational results, we perform an isolated investigation of the C2F hyperarc generation algorithm in this section. This means that we exclusively solve the LP models that appear in the root nodes of the branching trees for RSRP instances. In other words, we analyze our C2F idea purely in terms of a solving method for LP models.

Our argumentation to demonstrate the positive effect on the computation times within our IP method of Chapter 6 works as follows. In addition to the computation times for the solution of the linear programs we also measure the computation time that is needed to *resolve* the "generated" RMP. More precisely, after the last iteration of the C2F hyperarc generation algorithm we take the columns that have been priced, solve the RMP by the IPA [66] over these columns again (from scratch and without any column generation), and refer the time needed for this as *resolving time*.

In this way, the resolving time approach is a tool to simulate the application of our C2F hyperarc generation algorithm within rapid branching in order to avoid blurring effects. Moreover, the resolving time approach perfectly fits with the proper application of the C2F algorithm within our rapid branching implementation for two reasons. First, we do not generate any new columns after branching, see Section 6.1. Therefore, the LP models within the rapid branching tree do not become larger. Second, we do not know a way to warm start the LP solvers (that call the IPA). Thus, we find it reasonable and reliable to measure the time for resolving from scratch.

A notable implementation detail is how we handle the hypergraphs of the composition and configuration layer. We only store the hypergraph associated with the configuration layer, namely $([V], [A], [H])$ in the memory of the computer. Given a hyperarc $[h] \in [H]$ we can enumerate all fine hyperarcs that map to $[h]$ by an iterator routine for the composition layer on the stack of the computer program, i.e., RotOR. This can be seen as a dynamic graph generation approach since by using our pruning strategy, we do not have to handle or enumerate the whole fine hypergraph at any time (but we do this once in order to count the total number of hyperarcs).

All our computations were performed with RotOR version 2.2 on computers of the optimization department of Zuse Institute Berlin (ZIB) with an Intel(R) Xeon(R) CPU X5672 with 3.20 GHz, 12 MB cache, and 48 GB of RAM in single thread mode.

We define four different algorithmic variants in order to show the relevance of all algorithmic ingredients that we have introduced:

	$\|T\|$	$\|V\|$	$\|H\|$	maint.	columns	rows	non-zeros	memory	solving	resolving
RSRP_076	4194	13222	48530881		2101712	306563	11568458	5487	00:08:43:22	00:00:21:57
RSRP_078	4216	13354	48182502	x	1993105	294444	10816502	5346	00:10:16:59	00:00:24:46
RSRP_079	4216	13354	70055918		2740552	542018	19993611	7893	00:23:07:53	00:00:56:48
RSRP_081	4216	13354	49069226	x	2035475	297927	11246994	5435	00:09:12:58	00:00:20:26
RSRP_082	4216	13354	70942642		2758776	541800	20281403	8040	00:23:42:09	00:01:12:16
RSRP_123	5600	14278	39649900	x	1930201	302840	11487017	5271	00:03:10:40	00:00:07:42
RSRP_124	5600	14278	57480718		2749024	566930	21739579	7810	00:06:20:10	00:00:17:14
RSRP_138	805	9810	29049302		1570890	87038	8904560	4934	00:12:22:26	00:00:33:23
RSRP_140	987	16790	75274348		3639659	103609	31758855	8721	01:17:04:26	00:02:50:51
RSRP_141	987	16790	48378460		3377672	35114	16993798	6519	00:12:59:52	00:00:56:15

Table 2.2.: Computational results of the C2F hyperarc generation for the large RSRP instances.

Variant 1: The first variant is exactly as described by Algorithm 2.2.

Variant 2: This variant is defined by Algorithm 2.2 excluding lines 15 to 21 (and setting $H^\star := \emptyset$ instead), i.e., we omit our column selection strategy.

Variant 3: This variant is defined by excluding lines 15 to 21 and line 30 from Algorithm 2.2, i.e., without our column selection strategy and without our pruning strategy by $[d]$.

Variant 4: We solve the RSRP for the composition layer from scratch, i.e., without any column generation as a static LP model by the IPA of Cplex.

Our test set consists of 147 RSRP instances[7]. In all tables the columns $|T|$, $|V|$, and $|H|$ report the number of trips to cover, the number of nodes, and the number of hyperarcs, respectively. A cross in the column "maint." indicates that there is a single maintenance constraint for each fleet in the corresponding instance of the RSRP (it is the regular inspection, see Section 5.2). Instances that do not distinguish by columns two to five have, e.g., different objective functions.

Naturally, a test set consists of many small, some medium, and a few large instances where the terms small, medium, and large associate limiting resources as computation time our memory. In fact, we distinguish small, medium, and large instances after the analysis[8] of the results as follows:

Small: We declare all instances that could be solved with variant 4 as *small*.

Medium: All instances that could not be solved with variant 4 but with variants 2 and 3 are considered as *medium*.

[7] This were almost all instances that our cooperation partner DBF provided (by the time when we made these computations).

[8] Usually, the classification into different instance classes is made a priori. However, we classify on the basis of the results, which allows a more straightforward and appropriate presentation.

	variant	$\|T\|$	$\|V\|$	$\|H\|$	columns	rows	non-zeros	memory	solving	resolving
RSRP_036	1	2030	4910	23413966	1066358	214189	7592519	2983	00:03:40:24	00:00:08:34
	2				2937722	505270	21429752	6926	01:20:28:17	00:00:24:19
	3				2937722	505270	21429752	6765	02:00:29:56	00:00:24:54
RSRP_039	1	2030	4910	23413966	1093157	219206	7770222	2984	00:04:35:03	00:00:10:29
	2				2982385	516429	21689739	7086	01:16:27:16	00:00:33:40
	3				2982385	516429	21689739	6931	01:18:52:52	00:00:34:09
RSRP_050	1	1126	4696	20963280	1002933	180804	7369383	2605	00:05:25:32	00:00:13:56
	2				2981197	440050	21510641	6561	01:19:15:26	00:00:43:16
	3				3232294	463437	23487090	7263	02:00:47:44	00:00:42:45
RSRP_080	1	4216	13354	25521577	1303150	34288	4305082	2838	00:01:01:08	00:00:03:19
	2				3910682	34288	12948542	5716	00:05:27:09	00:00:06:13
	3				3910682	34288	12948542	5687	00:07:48:10	00:00:05:54
RSRP_083	1	4216	13354	26408301	1233754	34309	4332590	2860	00:00:54:05	00:00:03:21
	2				3566001	34309	12261824	5459	00:05:38:45	00:00:07:31
	3				3566001	34309	12261824	5429	00:07:20:07	00:00:06:29
RSRP_133	1	1806	4610	20209896	1056449	196397	8004125	2993	00:04:22:19	00:00:11:51
	2				3718902	538010	27889234	7898	01:16:44:40	00:00:32:21
	3				3718902	538010	27889234	8179	01:18:27:16	00:00:33:19

Table 2.3.: Computational results of the C2F hyperarc generation algorithm for the medium RSRP instances.

Large: All instance that could only be solved with our proper C2F hyperarc generation algorithm are referred as *large* instances.

We recognize that ten of the 147 instances run out of memory with variants 2 to 4 and six instances run out of memory only for variant 4. Thus, indeed, all 147 completely partition into the three classes of instances that we separately present in Tables 2.2, 2.3, and 2.4.

Table 2.2 reports about the large instances that could only be solved by using our C2F hyperarc generation algorithm. The columns six to the last denote the number of columns, rows, and non-zeros that were generated as well as the maximal memory usage (in Megabytes) that was allocated by the executing process of the algorithm. The last two columns report the running time of the algorithm and resolving time in the format dd:hh:mm:ss. These result give an impression of what can be computed with our C2F idea. In particular, one can see that a relatively huge amount of computation time for the MP can be decreased to a resolving time with that one can work with within an IP approach.

The next Table 2.3 is similar to Table 2.2. The difference is that we also report the results for the variants 2 and 3 for the medium instances. The results for variant 2 and variant 3 clearly demonstrate the positive effects of the additional layers for the RSRP. We see that the pruning rule associated with the configuration layer significantly speeds up the computations by comparing the solving times between variants 2 and 3 (while this has no effect on the generated restricted master problem). The most positive impact is clearly caused by our column selection strategy, i.e., to solve the coarse reduction associated with the vehicle layer. This impact can be seen by comparing the size of the

generated RMPs and by the run time consumption of variant 1 compared to variant 2.

Finally, Table 2.4 presents the results for the small instances. For these 131 instances we report slightly more aggregated figures in order to conclude this section with statements about the average behavior of RotOR's implementation. The columns 6 to 17 report the fractions of variants 1, 2, and 3 w.r.t. variant 4. For example, for the instance `RSRP_001` we measured a memory consumption of 476 MB and 1206 MB when we ran variant 3 and variant 4, respectively. Therefore, we denote $0.39 = {}^{476}/_{1206}$ in the respective cell of Table 2.4. The last row of Table 2.4 reports the arithmetic means over these fractions.

2.7. Conclusion

In Chapter 2 we developed a new algorithm components for the solution of very large LP models in terms of column generation. The concept that we propose is called (projective) coarse-to-fine (C2F) column generation and is introduced as a general framework in that specific applications can be embedded.

The C2F framework is useless unless meaningful problem specific layers are developed. Indeed, the layers for the RSRP including their utilization is the original contribution of the chapter for rolling stock rotation optimization. The C2F idea has been deployed to and tested by the railway industry (i.e., DBF) through the sophisticated implementation within RotOR.

The computational results show that the running time of the C2F algorithm, namely variant 1, is on average 1.5 (see the last row of Table 2.4) times higher than the static LP approach that is used for comparison. This shows that our algorithm can have a slight overhead for small instances but does not lose competitiveness. The more essential result for our desires is that the resolving time has decreased by a factor of 0.05, again see the last row of Table 2.4. In other much more optimistic words, the C2F hyperarc generation algorithm may (slightly) slow down the solution of the LP relaxation in the root node of a branching tree but it generates a RMP that can be solved 20 times faster on average within an IP method on top.

The original intention behind the C2F approach was to significantly increase the order of magnitude for that we can compute RSRP instances, which we particularly demonstrated by the results for the large and medium instances of the RSRP.

Table 2.4.: Computational results of C2F hyperarc generation for small RSRP instances.

					variant 3				variant 2				variant 1			
instance	$\|T\|$	$\|V\|$	$\|H\|$	maintenance	solving	columns	memory	resolving	solving	columns	memory	resolving	solving	columns	memory	resolving
RSRP_001	310	620	805482	x	13.5	0.29	0.39	0.15	10.5	0.29	0.39	0.16	2.1	0.09	0.16	0.06
RSRP_002	310	620	1163370		18.9	0.25	0.36	0.20	16.2	0.25	0.37	0.18	2.0	0.09	0.14	0.06
RSRP_003	310	620	436534		7.2	0.32	0.47	0.03	4.5	0.32	0.47	0.03	1.3	0.11	0.23	0.00
RSRP_004	884	1830	4085542	x	15.2	0.25	0.39	0.16	12.6	0.25	0.39	0.16	1.6	0.06	0.12	0.05
RSRP_005	884	1830	5868584		10.1	0.18	0.27	0.13	9.5	0.18	0.27	0.15	1.0	0.06	0.10	0.04
RSRP_006	884	1830	2212292		10.7	0.26	0.38	0.13	6.4	0.26	0.37	0.11	1.2	0.07	0.20	0.06
RSRP_007	884	1830	4085386	x	6.0	0.12	0.20	0.10	4.5	0.12	0.19	0.11	2.4	0.08	0.15	0.07
RSRP_008	884	1830	5868428		3.0	0.10	0.16	0.05	2.8	0.10	0.16	0.06	1.1	0.08	0.12	0.03
RSRP_009	884	1830	2212136		6.1	0.15	0.26	0.09	3.4	0.15	0.26	0.09	1.0	0.08	0.19	0.06
RSRP_010	884	1768	6508938	x	19.6	0.20	0.30	0.21	17.3	0.20	0.31	0.24	1.4	0.04	0.11	0.07
RSRP_011	884	1768	9385210		16.4	0.16	0.25	0.18	14.9	0.16	0.25	0.20	1.5	0.04	0.09	0.04
RSRP_012	884	1768	3521040		9.9	0.22	0.31	0.13	6.3	0.22	0.32	0.12	0.9	0.05	0.18	0.04
RSRP_013	884	1830	4100410		15.8	0.24	0.35	0.19	12.6	0.24	0.36	0.16	1.4	0.06	0.12	0.05
RSRP_014	884	1830	2212292		10.8	0.26	0.38	0.12	6.4	0.26	0.38	0.12	1.2	0.07	0.20	0.06
RSRP_015	277	1464	757340	x	16.5	0.39	0.58	0.25	17.1	0.39	0.56	0.30	2.2	0.12	0.21	0.11
RSRP_016	277	1464	1110078		13.8	0.35	0.49	0.22	12.8	0.35	0.45	0.22	2.4	0.11	0.18	0.10
RSRP_017	277	1464	389980		5.0	0.35	0.54	0.07	3.5	0.35	0.53	0.07	1.3	0.13	0.26	0.02
RSRP_018	277	1464	757514	x	13.1	0.38	0.53	0.24	12.6	0.38	0.55	0.28	2.1	0.13	0.22	0.10
RSRP_019	277	1464	1110208		13.6	0.32	0.44	0.25	13.0	0.32	0.43	0.23	2.0	0.11	0.19	0.10
RSRP_020	277	1464	390110		5.2	0.33	0.49	0.05	3.8	0.33	0.49	0.08	1.5	0.13	0.25	0.03
RSRP_021	277	980	549264		5.6	0.24	0.36	0.05	3.7	0.24	0.37	0.07	1.6	0.12	0.20	0.02
RSRP_022	277	980	1079526	x	9.9	0.23	0.36	0.14	8.8	0.23	0.34	0.16	1.4	0.08	0.14	0.06
RSRP_023	277	980	1589436		8.8	0.20	0.30	0.17	7.6	0.20	0.30	0.15	1.4	0.08	0.12	0.06
RSRP_024	277	980	549442		5.2	0.23	0.35	0.04	3.5	0.23	0.36	0.06	1.5	0.12	0.20	0.02
RSRP_025	174	898	139442		2.2	0.33	0.50	0.00	1.7	0.33	0.51	0.00	1.4	0.16	0.44	0.00
RSRP_026	146	828	114058		1.9	0.31	0.57	0.00	1.6	0.29	0.57	0.00	1.3	0.12	0.57	0.00
RSRP_027	277	980	1079526	x	11.8	0.24	0.37	0.18	10.0	0.24	0.37	0.16	1.6	0.09	0.14	0.06
RSRP_028	277	980	1589436		8.5	0.19	0.29	0.13	7.4	0.19	0.30	0.14	1.2	0.07	0.12	0.06
RSRP_029	277	980	549442		6.6	0.25	0.39	0.05	4.2	0.25	0.41	0.05	1.9	0.12	0.22	0.02
RSRP_030	310	620	805482	x	15.5	0.30	0.42	0.17	13.5	0.30	0.44	0.18	1.9	0.10	0.15	0.04
RSRP_031	310	620	1163370		19.0	0.27	0.37	0.27	16.2	0.27	0.35	0.24	2.3	0.09	0.14	0.07
RSRP_032	310	620	436534		5.9	0.30	0.44	0.02	3.5	0.30	0.46	0.02	1.2	0.09	0.22	0.00
RSRP_033	310	620	805482	x	9.6	0.26	0.36	0.16	7.6	0.26	0.36	0.15	1.9	0.10	0.16	0.05
RSRP_034	310	620	436534		5.2	0.30	0.43	0.02	3.1	0.30	0.42	0.02	1.1	0.09	0.21	0.00
RSRP_035	2030	4910	16101438	x	9.2	0.15	0.25	0.08	8.6	0.15	0.25	0.10	0.6	0.05	0.09	0.03
RSRP_037	2030	4910	8464864		15.7	0.19	0.31	0.19	10.3	0.19	0.31	0.18	1.4	0.05	0.14	0.09
RSRP_038	2030	4910	16101438	x	8.9	0.15	0.25	0.08	8.0	0.15	0.25	0.07	0.7	0.05	0.09	0.02
RSRP_040	2030	4910	8464864		13.5	0.19	0.31	0.17	9.2	0.19	0.30	0.16	1.2	0.05	0.14	0.07
RSRP_041	1126	4696	14335774	x	10.6	0.16	0.26	0.16	9.5	0.15	0.25	0.14	1.3	0.05	0.09	0.06
RSRP_042	1126	4696	7507040		9.6	0.16	0.25	0.19	9.6	0.17	0.27	0.25	1.4	0.06	0.11	0.11
RSRP_043	1126	4696	14335774	x	9.4	0.16	0.26	0.14	7.9	0.15	0.25	0.14	1.1	0.05	0.09	0.05
RSRP_044	1126	4696	7507040		8.3	0.16	0.24	0.18	5.4	0.15	0.24	0.19	1.4	0.06	0.12	0.09
RSRP_045	1126	4696	14335774	x	6.6	0.15	0.24	0.12	7.0	0.15	0.24	0.14	1.0	0.05	0.09	0.04
RSRP_046	1126	4696	7507040		9.9	0.16	0.26	0.19	7.1	0.16	0.26	0.19	1.5	0.06	0.12	0.11
RSRP_047	1126	4696	14335774	x	9.1	0.16	0.26	0.19	8.6	0.16	0.25	0.20	1.1	0.05	0.09	0.05
RSRP_048	1126	4696	7507040		9.7	0.15	0.25	0.21	7.1	0.15	0.26	0.19	1.6	0.06	0.12	0.11
RSRP_049	1126	4696	14335774	x	7.2	0.14	0.23	0.13	6.9	0.14	0.23	0.12	1.1	0.05	0.09	0.05
RSRP_051	1126	4696	7507040		9.7	0.15	0.24	0.19	6.8	0.16	0.24	0.20	1.5	0.06	0.12	0.11
RSRP_052	1126	4696	14335774	x	11.2	0.16	0.27	0.17	9.5	0.15	0.25	0.15	1.2	0.04	0.09	0.06
RSRP_053	1126	4696	7507040		10.1	0.15	0.25	0.21	7.5	0.15	0.24	0.20	1.5	0.05	0.11	0.11
RSRP_054	277	1464	757340	x	12.7	0.33	0.46	0.26	11.0	0.32	0.48	0.25	2.3	0.10	0.19	0.11
RSRP_055	277	1464	1110078		9.0	0.25	0.35	0.19	8.1	0.24	0.34	0.24	1.6	0.08	0.15	0.09
RSRP_056	277	1464	389980		4.0	0.24	0.40	0.05	3.0	0.26	0.42	0.05	1.2	0.09	0.20	0.02
RSRP_057	277	980	1079104	x	8.7	0.23	0.36	0.14	7.4	0.23	0.36	0.12	1.2	0.08	0.13	0.05
RSRP_058	277	980	1588922		9.3	0.19	0.29	0.14	9.1	0.19	0.28	0.18	1.2	0.07	0.11	0.05
RSRP_059	277	980	549264		5.0	0.22	0.36	0.05	3.3	0.22	0.35	0.05	1.6	0.11	0.20	0.02
RSRP_060	174	898	271794	x	5.9	0.38	0.51	0.12	5.2	0.38	0.49	0.12	2.1	0.15	0.26	0.06
RSRP_061	174	898	398192		8.0	0.37	0.51	0.21	7.6	0.37	0.52	0.22	1.8	0.12	0.22	0.11
RSRP_062	174	898	139442		2.0	0.30	0.48	0.00	1.7	0.30	0.48	0.00	1.2	0.14	0.44	0.00
RSRP_063	146	828	222958	x	5.5	0.39	0.54	0.13	5.2	0.39	0.54	0.13	1.6	0.15	0.25	0.04
RSRP_064	146	828	326288		6.2	0.35	0.47	0.20	6.1	0.35	0.48	0.20	1.6	0.12	0.22	0.08
RSRP_065	146	828	114058		1.9	0.28	0.55	0.00	1.5	0.28	0.55	0.00	1.2	0.14	0.55	0.00
RSRP_066	1126	4593	13572721	x	11.1	0.16	0.27	0.16	11.5	0.16	0.27	0.18	1.0	0.04	0.09	0.05
RSRP_067	1126	4593	7102630		12.7	0.17	0.28	0.22	9.5	0.17	0.28	0.23	2.0	0.06	0.12	0.10
RSRP_068	277	1443	732152	x	14.6	0.39	0.56	0.31	10.3	0.33	0.48	0.27	2.2	0.11	0.20	0.11
RSRP_069	277	1443	1072828		13.5	0.37	0.49	0.29	11.6	0.33	0.43	0.26	2.1	0.11	0.19	0.11
RSRP_070	277	1443	377056		5.1	0.32	0.48	0.05	3.6	0.32	0.47	0.07	1.5	0.13	0.25	0.03
RSRP_071	2186	8630	17525972		9.3	0.13	0.21	0.17	7.1	0.13	0.21	0.17	1.5	0.05	0.11	0.10
RSRP_072	1060	3934	18612816	x	7.9	0.10	0.17	0.12	6.5	0.10	0.17	0.11	1.2	0.03	0.07	0.05
RSRP_073	1060	3934	10018932		12.3	0.12	0.19	0.15	7.1	0.11	0.18	0.16	1.5	0.04	0.10	0.10
RSRP_074	1126	4696	14335774	x	8.6	0.15	0.25	0.15	9.2	0.15	0.26	0.12	1.2	0.04	0.09	0.05

Continued on next page

Table 2.4 – continued from previous page

					variant 3				variant 2				variant 1			
instance	$\|T\|$	$\|V\|$	$\|H\|$	maintenance	solving	columns	memory	resolving	solving	columns	memory	resolving	solving	columns	memory	resolving
RSRP_075	1126	4696	7507040		9.0	0.16	0.26	0.19	6.8	0.15	0.25	0.19	1.5	0.06	0.11	0.11
RSRP_077	4194	13222	26124420		6.9	0.13	0.21	0.10	5.1	0.13	0.20	0.12	1.0	0.05	0.11	0.06
RSRP_084	277	1464	994532	x	14.2	0.33	0.46	0.18	11.2	0.33	0.44	0.20	2.2	0.10	0.17	0.08
RSRP_085	277	1464	1347270		14.1	0.27	0.37	0.22	12.2	0.27	0.36	0.21	2.1	0.09	0.15	0.09
RSRP_086	277	1464	627172		6.5	0.20	0.31	0.05	3.5	0.20	0.30	0.03	1.2	0.07	0.14	0.02
RSRP_087	277	1464	1347270		11.7	0.25	0.33	0.25	9.9	0.25	0.34	0.24	2.1	0.08	0.13	0.08
RSRP_088	277	1464	627172		7.0	0.20	0.30	0.05	3.6	0.20	0.28	0.05	1.2	0.07	0.13	0.02
RSRP_089	277	1464	994532	x	12.9	0.31	0.42	0.20	9.8	0.31	0.43	0.19	1.8	0.08	0.14	0.07
RSRP_090	277	1464	1347270		10.2	0.25	0.33	0.20	9.0	0.25	0.34	0.23	1.9	0.08	0.13	0.08
RSRP_091	277	1464	627172		6.1	0.20	0.30	0.05	3.3	0.20	0.28	0.05	1.1	0.07	0.14	0.02
RSRP_092	277	1464	993868	x	14.8	0.32	0.45	0.24	11.8	0.32	0.46	0.23	1.8	0.10	0.17	0.09
RSRP_093	277	1464	1346606		10.3	0.27	0.34	0.21	9.1	0.27	0.35	0.23	1.8	0.09	0.14	0.08
RSRP_094	277	1464	626508		6.3	0.21	0.31	0.05	3.2	0.21	0.30	0.05	1.3	0.08	0.14	0.02
RSRP_095	277	1464	994532	x	10.4	0.26	0.36	0.17	9.9	0.30	0.41	0.19	1.8	0.09	0.16	0.08
RSRP_096	277	1464	1347270		11.0	0.22	0.30	0.22	11.7	0.25	0.33	0.24	1.7	0.08	0.15	0.10
RSRP_097	277	1464	627172		6.5	0.20	0.29	0.05	3.4	0.20	0.29	0.05	1.3	0.08	0.15	0.03
RSRP_098	1126	4696	19234394	x	10.2	0.13	0.21	0.16	7.6	0.13	0.20	0.14	1.1	0.04	0.07	0.05
RSRP_099	1126	4696	12405660		10.0	0.10	0.16	0.14	6.0	0.11	0.17	0.11	1.4	0.04	0.08	0.06
RSRP_100	1126	4696	19234364	x	8.9	0.13	0.21	0.12	8.8	0.14	0.22	0.15	0.9	0.04	0.07	0.04
RSRP_101	1126	4696	12405642		10.3	0.11	0.16	0.11	5.6	0.11	0.16	0.11	1.3	0.04	0.08	0.05
RSRP_102	77	154	47214	x	2.1	0.55	0.83	0.00	1.8	0.55	0.82	0.00	1.2	0.32	0.78	0.00
RSRP_103	77	154	69594		1.9	0.45	0.59	0.02	1.9	0.45	0.59	0.02	1.2	0.31	0.46	0.00
RSRP_104	77	154	24370		1.4	0.56	1.01	0.00	1.3	0.56	1.01	0.00	1.1	0.33	1.01	0.00
RSRP_105	77	154	47926	x	2.0	0.58	0.81	0.00	1.8	0.58	0.80	0.00	1.2	0.31	0.74	0.00
RSRP_106	77	154	70880		2.3	0.48	0.61	0.03	2.1	0.48	0.61	0.03	1.4	0.33	0.48	0.00
RSRP_107	77	154	24746		1.4	0.57	1.00	0.00	1.2	0.57	1.00	0.00	1.1	0.31	1.00	0.00
RSRP_108	73	146	42692	x	2.0	0.60	0.82	0.00	1.9	0.60	0.85	0.00	1.2	0.33	0.80	0.00
RSRP_109	73	146	63412		2.4	0.49	0.64	0.03	2.1	0.49	0.63	0.03	1.5	0.33	0.48	0.00
RSRP_110	73	146	21796		0.9	0.59	1.00	0.00	1.1	0.59	1.00	0.00	1.0	0.33	1.00	0.00
RSRP_111	75	150	44718	x	2.0	0.59	0.85	0.00	1.8	0.59	0.86	0.00	1.2	0.34	0.79	0.00
RSRP_112	75	150	66506		3.2	0.56	0.75	0.03	2.8	0.56	0.75	0.03	1.6	0.33	0.51	0.00
RSRP_113	75	150	22730		1.3	0.58	1.00	0.00	1.1	0.58	1.00	0.00	1.1	0.35	1.00	0.00
RSRP_114	336	672	934018	x	19.3	0.35	0.53	0.15	17.4	0.35	0.54	0.20	2.8	0.12	0.18	0.07
RSRP_115	336	672	1363482		35.2	0.36	0.52	0.29	31.1	0.36	0.52	0.30	3.1	0.13	0.18	0.11
RSRP_116	336	672	498576		11.4	0.31	0.53	0.05	6.6	0.31	0.57	0.05	1.5	0.12	0.26	0.02
RSRP_117	854	1708	3597182	x	16.7	0.22	0.33	0.20	14.8	0.22	0.33	0.20	2.2	0.07	0.13	0.07
RSRP_118	854	1708	5155568		9.5	0.17	0.24	0.14	8.6	0.17	0.25	0.12	1.5	0.07	0.11	0.05
RSRP_119	854	1708	1961760		9.3	0.27	0.40	0.11	5.7	0.27	0.39	0.11	1.0	0.07	0.19	0.04
RSRP_120	1033	3106	8407556	x	5.6	0.11	0.20	0.12	5.0	0.11	0.20	0.13	1.1	0.05	0.09	0.07
RSRP_121	1033	3106	12213320		7.4	0.11	0.17	0.12	7.5	0.11	0.17	0.14	1.1	0.04	0.08	0.05
RSRP_122	1033	3106	4487388		8.6	0.16	0.24	0.12	6.4	0.17	0.26	0.16	1.1	0.06	0.12	0.07
RSRP_125	5600	14278	21205982		21.4	0.17	0.28	0.18	12.2	0.17	0.28	0.16	1.9	0.05	0.16	0.07
RSRP_126	5593	14250	21218266		18.4	0.17	0.27	0.17	10.8	0.17	0.28	0.15	1.7	0.05	0.16	0.06
RSRP_127	1488	2976	8049988	x	21.5	0.25	0.39	0.30	18.3	0.25	0.36	0.30	1.4	0.06	0.12	0.06
RSRP_128	1488	2976	4362772		13.0	0.26	0.41	0.16	8.8	0.26	0.40	0.18	0.8	0.06	0.18	0.05
RSRP_129	1488	2976	8050512	x	20.4	0.24	0.38	0.32	17.7	0.24	0.34	0.25	1.5	0.06	0.12	0.06
RSRP_130	1488	2976	11670716		12.2	0.17	0.25	0.21	11.5	0.17	0.25	0.21	1.1	0.06	0.10	0.06
RSRP_131	1488	2976	4363296		12.0	0.27	0.40	0.18	8.3	0.27	0.40	0.19	0.9	0.06	0.18	0.05
RSRP_132	1806	4610	13937042	x	15.0	0.21	0.32	0.20	13.8	0.21	0.33	0.20	1.2	0.05	0.11	0.05
RSRP_134	1806	4610	7421090		11.6	0.24	0.39	0.23	8.8	0.24	0.38	0.25	1.3	0.06	0.16	0.09
RSRP_135	167	486	355108	x	5.1	0.21	0.34	0.07	4.5	0.21	0.32	0.07	1.9	0.13	0.22	0.05
RSRP_136	167	486	459034		4.4	0.18	0.25	0.11	4.0	0.18	0.26	0.13	2.1	0.14	0.20	0.09
RSRP_137	167	486	191534		2.3	0.28	0.45	0.00	1.9	0.28	0.46	0.00	1.2	0.16	0.38	0.00
RSRP_139	805	9810	15819548		10.4	0.17	0.26	0.13	6.1	0.17	0.27	0.13	1.4	0.05	0.19	0.07
RSRP_142	805	2928	9081282		10.9	0.16	0.25	0.16	10.0	0.16	0.24	0.16	2.8	0.05	0.12	0.08
RSRP_143	805	2928	4944614		9.7	0.20	0.30	0.18	6.6	0.20	0.30	0.19	1.0	0.05	0.18	0.09
RSRP_144	21	126	11292	x	1.2	0.70	1.01	0.00	1.2	0.70	1.01	0.00	1.1	0.44	1.01	0.00
RSRP_145	21	126	5720	x	1.0	0.67	1.00	0.00	1.0	0.67	1.00	0.00	1.1	0.48	1.01	0.00
RSRP_146	183	586	328922	x	5.5	0.28	0.41	0.10	4.3	0.28	0.42	0.10	2.6	0.19	0.31	0.06
RSRP_147	183	586	175398	x	2.2	0.33	0.50	0.00	1.6	0.33	0.49	0.00	1.1	0.21	0.37	0.00
mean					9.3	0.27	0.40	0.13	7.5	0.27	0.40	0.13	**1.5**	**0.11**	**0.23**	**0.05**

Chapter 3.

Resource-Constrained Assignments by Regional Search

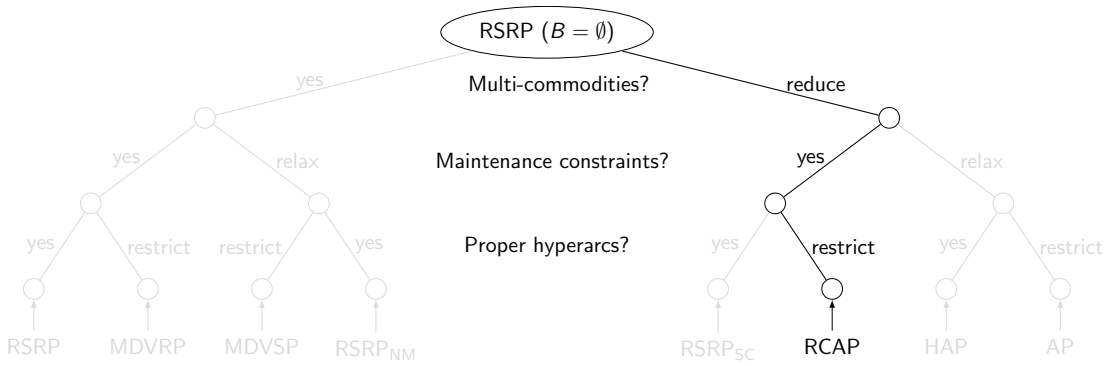


Figure 3.1.: Extension of the RCAP to the RSRP, see Section 1.6 with Figure 1.2 for other problems that the RSRP extends.

This chapter is dedicated to the resource-constrained assignment problem (RCAP). The RCAP derives from the main subject of this thesis, namely the rolling stock rotation problem (RSRP), by restricting to pure standard graphs and by reducing the multi-commodity requirement of the RSRP. The relation of the RCAP and the RSRP is illustrated in Figure 3.1, which is based on the investigation of different problems that the RSRP extends (see Section 1.6).

The following facts motivate that it is worth to investigate the RCAP. The RCAP

- ▷ allows for an isolated investigation of the maintenance constraints of the RSRP,
- ▷ simultaneously generalizes classical variants of the vehicle routing problem (VRP),
- ▷ and, primary, leads to an algorithmic approach that we call *regional search*.

The organization of the chapter is outlined in Section 3.1 and its contributions have been published under the following titles:

- ▷ "Local Search for the Resource Constrained Assignment Problem" [2], and
- ▷ "Regional Search for the Resource Constrained Assignment Problem" [1].

3.1. Introduction

In this section we motivate our particular attention to the RCAP from a high-level point of view and explain the organization of the chapter.

Generally speaking, the RCAP, which we formally introduce in Section 3.3, is to find a minimal cost cycle partition of the nodes of a directed graph such that a *resource constraint* is fulfilled. The resource constraint has its roots in maintenance constraints for railway vehicles in the RSRP. These constraints play an essential role in our industrial application for the RSRP but are less investigated in the railway optimization literature so far. In contrast to this, however, similar constraints appear in almost all variants of the VRP which is one of the most extensively investigated combinatorial optimization problems. In fact, the RCAP extends variants of the VRP. In this way, the RCAP provides access to the recent RSRP and classical problems, i.e., it bridges between vehicle routing and railway optimization.

The variability in computational effort needed to solve an RSRP instance to proven optimality is huge. We will see that the maintenance constraints in our railway application have a completely different computational flavor than similar constraints in vehicle routing have. Exactly this characteristic leads to the idea of regional search (RS), which is the major contribution of this chapter. RS is a straightforward algorithmic concept. As a symbiosis of a local and a global search algorithm, the result of an RS is a local optimum for a combinatorial optimization problem. In addition, the local optimum must be globally optimal as well if an instance of a problem relaxation[1] is computed. In Section 3.2 we refer to algorithms from the literature that are already regional search algorithms.

We implement our RS idea in Section 3.5 in three different ways:

- ▷ We present an RS algorithm for general binary programs in order to provide the idea for a standardized setting, see Section 3.5.1.

- ▷ In Section 3.5.2.1 we develop a branching-based RS that is derived from LP arguments of the Hungarian method and which is dedicated to the RCAP. Within this procedure we call an exact branch and bound (B&B) algorithm that is based on a "regional" constraint integer program (CIP) for the RCAP.

- ▷ The third move-based RS described in Section 3.5.2.2 is also derived from the Hungarian method. Instead of solving a CIP we call a k-opt heuristic, exchanging k arcs of the incumbent solution in each improvement step. For particular instances this RS is faster than the branching-based version which has higher chances of finding an optimal solution instead.

[1] We assume that an RS can compute instances of the original optimization problem and it can compute instances of the relaxation as well (e.g., a mixed-integer programming (MIP) solver should also be able to solve pure linear programming (LP) models). The only assumption for the relaxation that we consider by an RS is that a feasible solution of the original problem needs to be feasible for the relaxation as well.

As already mentioned, our RS approach for the RCAP is based on the Hungarian method for the classical standard assignment problem. The original version of this algorithm as presented by Kuhn (1955, [70]) constructs a proven optimal primal solution by iterating feasible dual solutions. It accomplishes primal feasibility only at termination. For our purposes it is more appropriate if the primal and dual sides change their roles. In fact, this is what *we* call *primal Hungarian method* and has been published by Balinski and Gomory (1964, [20]). The primal Hungarian method is an exact algorithm for the standard assignment problem that iteratively improves primal solutions until optimality is proven. In this way, the primal Hungarian method can be seen as a local search procedure. We use it as basis for our RS algorithms and, therefore, present it in Section 3.4 in detail.

Another contribution of the chapter is a global search algorithm, namely a B&B procedure for the RCAP. This is the subject of Section 3.6. The B&B approach is completely combinatorial (i.e., no general LP or integer programming (IP) framework is used) and customized to be called as a sub-routine within the branching-based RS from Section 3.5.2.1. In addition, we use this B&B algorithm as standalone exact method in order to solve RCAP instances to proven optimality.

In the second last section we present computational results for both the regional and global search. The test set is composed of simplified RSRP instances from the railway application as well as classical (a)symmetric traveling salesman and capacitated vehicle routing instances.

3.2. Literature

As already mentioned, the RSRP extends the RCAP. For the RSRP we review the literature in Section 1.2. At the same time, the RCAP is an extension of variants of the well known VRP. In this extension the resource constraint (formal definition follows in Section 3.3) of the RCAP plays an essential role. We informally explain the resource constraint on the basis of the classical asymmetric traveling salesman problem (ATSP), which is extended by the VRP.

In terms of the RCAP, the ATSP can be formulated as: Find a minimal cost partition of the nodes of a directed graph (V, A) into cycles under the additional side constraint that there is no subtour (i.e., exactly one cycle has to be computed). The no subtour constraint can be seen as resource constraint of the RCAP, i.e., the traveling salesman has to collect a flower at each city, he can load at most $|V| - 1$ flowers and he has to drop the flowers at some special depot node[2]. For other VRPs the resource constraint appears as the maximal vehicle capacity. For the remaining part of this section it is sufficient to imagine the resource constraint of the RCAP as a constraint that restricts the cumulative distance of cycles or paths that are part of a solution, i.e., a set of cycles that cover the nodes of a directed graph.

Reviewing the VRP literature is a mammoth task. It is realized by the excellent book of Toth and Vigo (2014, [101]), which we advertise as the main reference for the VRP.

[2]This modeling idea is used in MIP formulations for the symmetric traveling salesman problem (TSP), see Miller, Tucker, and Zemlin (1960, [81]).

The equivalent for the special case of the TSP is the book of Applegate et al. (2007, [18]).

For problems that have similar resource constraints but consider an acyclic graph a path-based model in combination with column generation and dynamic programming is the method of choice. In these approaches a resource-constrained shortest path problem (RCSPP) appears as pricing problem. For the RCSPP in acyclic graphs there exist a huge variety of powerful pseudo-polynomial time algorithms, see Dumitrescu (2002, [42]). In cyclic graphs one has to deal with node repetitions in dynamic programming approaches, which is a rather hard task. By today, there is no method of choice to tackle these resource constraints in graphs that contain cycles, see Pugliese and Guerriero (2013, [86]). Therefore, the RCAP is still an interesting candidate for operations research.

In this chapter, we propose the concept of regional search for the solution of the RCAP. Regional search is strongly connected to local search, i.e., a regional search is a local search (but not the other way around). Almost all algorithms for the VRP, ATSP, or RSRP use some kind of local search procedures. They are either used as working horse to prune the search space within exact algorithms or as standalone algorithms.

MIP based local search heuristics, e.g., relaxation enforced neighborhood search (RENS) [25], relaxation induced neighborhood search (RINS) [39], local branching [48], and crossover [93] restrict the original problem to a much smaller MIP model by introducing bound changes and additional constraints. The remaining problems are solved by using the LP relaxation as a pruning argument. Apart from such methods there exists a huge set of combinatorial motivated local search algorithms. They are designed to take use of one or more elementary local move operations and to quickly explore a large neighborhood. The papers of Lin and Kernighan (1973, [72]), Kanellakis and Papadimitriou (1980, [68]), and Clarke and Wright (1964, [37]) are classical seminal references in the case of the TSP, ATSP, and VRP.

A closer look reveals that local search algorithms can be distinguished between those that make use of linear programs and others which do not. This observation brings the following idea into play: Why not just making some modifications to an LP algorithm in order to gain a heuristic for some hard combinatorial optimization problem? Indeed, this is the idea of regional search to some extend.

In fact, there exist a variety of papers that exactly follow this idea (without calling it regional search):

The **modulo network simplex algorithm** as proposed in Nachtigall and Opitz (2008, [82]) is a heuristic for the periodic event scheduling problem (PESP). This algorithm can be summarized as a local search procedure that improves the current incumbent solution (i.e., a periodic timetable) by move operations emerged from a modified network simplex algorithm. The modulo network simplex algorithm is a regional search algorithm according to our definition 3.5.1 (see page 80) because it will compute an optional solution for each PESP instance for that the periodicity constraint is redundant (by proof).

The **integral simplex using decomposition (ISUD)** algorithm presented in Zaghrouti, Soumis, and Hallaoui (2014, [108]) is also a regional search algorithm. Primary, the ISUD is a heuristic for the set-partitioning problem that “simulates”

pivot operations of the primal simplex algorithm for set-partitioning problems. It will compute the optimal solution of a problem instance if its polyhedron is integral. This is the case, e.g., for standard assignment problems and set-partitioning problems with a balanced constraint matrix, see Borndörfer (1998, [27, Section 1.7]).

Almost all **MIP-heuristics** (e.g., RENS proposed by Berthold (2013, [25]) or RINS introduced by Danna, Rothberg, and Pape (2005, [39])) will compute a feasible optimal solution for a MIP model if "the" optimal solution to its LP relaxation is already integer feasible. Thus, formally, many MIP-heuristics are RS algorithms. But we see those algorithms as rather artificial examples for RS algorithms because they are called on top of an LP solver and do not interfere its behavior.

The presented list of regional search algorithms of the current literature is far from being complete. Although, we do not intend to complete this list but we like to emphasis that there are many local search algorithms that are indeed not regional search procedures. An obvious example is the 2-opt algorithm for the TSP, see Lin (1965, [73]). A Hamiltonian tour that can not be improved by exchanging not more than two edges (i.e., a 2-optimal tour) might not be globally optimal even if all optimal solutions to the assignment relaxation of the TSP instance do not contain any subtours. It is also mentionable that most of all meta-heuristics (see [101] for a survey) are not regional search algorithms.

Note that we do not claim that regional search is superior to local search. The currently best performing heuristic for the symmetric TSP, namely the Lin-Kernighan heuristic [72, 64] is not a regional search algorithm. But who knows whether there exists a regional search algorithm for the TSP that outperforms the Lin-Kernighan heuristic?

We use the algorithm proposed by Balinski and Gomory (1964, [20]) as basis for the regional search for the RCAP. Balinski and Gomory describe an exact method to solve the assignment problem which we call the *primal Hungarian method* in this chapter. The primal Hungarian method is similar to the famous (standard) Hungarian method, see Kuhn (1955, [70]). The most important distinguishing feature of [20] is that a perfect matching, i.e, a feasible primal incumbent solution is always at hand. The incumbent solution is iteratively improved by applying cycles that alternate between arcs to delete and arcs to add. Our idea is to use the alternating cycles emerged from this algorithm for an improvement heuristic for the RCAP. Using alternating cycles as neighborhood structure has been extensively studied in the literature. The main difference to our approach is that we find the alternating cycles by arguments from linear programming. Thus, it is not necessary to define neighborhood graphs in order to derive candidate lists.

We close our excursion to the literature related to this chapter by citing the book of Burkard, Dell'Amico, and Martello (2009, [33]). This book is an excellent and extensive companion for the standard assignment problem and variants of it.

3.3. Resource-Constrained Assignments

Let (V, A) be a directed graph with dedicated *events* taking place at every arc. We distinguish *replenishment events* from other events and call arcs with replenishment events *replenishment arcs*. Let $r : A \mapsto \mathbb{Q}_+ \times \mathbb{Q}_+$ be a *resource function* that assigns a pair of nonnegative rational numbers (r_a^1, r_a^2) to every arc denoting a resource consumption before and after the event, respectively, and define $r_a := r_a^1 + r_a^2$. A *resource path* is an elementary path in (V, A) of the form $P = (a_0, a_1, \ldots, a_m, a_{m+1}) \subseteq A$ such that a_0 and a_{m+1} are replenishment arcs and $a_1, \ldots, a_m$ are not replenishment arcs. Let $\mathbb{P}(A)$ be the set of all resource paths and $B \in \mathbb{Q}_+$ be a *resource bound*. We call a resource path $P = (a_0, a_1, \ldots, a_m, a_{m+1}) \in \mathbb{P}(A)$ *feasible* if the following *resource constraint* is fulfilled (otherwise P is *infeasible*):

$$r_{a_0}^2 + \sum_{i=1}^{m} r_{a_i} + r_{a_{m+1}}^1 \leq B. \tag{3.1}$$

Finally, let $c : A \mapsto \mathbb{Q}$ be some objective function associated with the arcs of (V, A).

Definition 3.3.1 (Resource-constrained assignment problem (RCAP))**.** *Given a directed graph (V, A), a resource function r, an objective function c, and a resource bound B. The RCAP is to find a set of directed cycles $C_1, \ldots, C_n \subseteq A$ in (V, A) such that every node is contained in exactly one cycle, every cycle contains at least one replenishment arc, all resource paths in $\mathbb{P}(\bigcup_{i=1}^{n} C_i)$ are feasible, and $c\,(\bigcup_{i=1}^{n} C_i)$ is minimal.*

Figure 3.2 illustrates the RCAP by showing a set of nodes covered by three cycles. The dashed arcs are replenishment arcs. A resource path fulfilling constraint (3.1) is highlighted in red where the two crosses indicate replenishment events.

In our first paper [2] about the RCAP, we additionally defined all cycles $C \subseteq A$ with $\sum_{a \in A} r_a = 0$ to be feasible. In order to streamline the presentation we assume that (V, A) does not contain such cycles in this chapter[3]. We also assume that (V, A) does not contain multiple arcs between two nodes. Moreover, w.l.o.g. we also assume that (V, A) is complete. Graphs that are not complete can be made complete by introducing arcs with sufficiently high cost.

We remark that the treatment of replenishment events "in the middle of the arcs" could be replaced by a consideration of replenishment nodes (or service nodes as we call them in the RSRP case, see Section 4.5). This would blow up the RCAP instances that we are interested in. In addition, the use of replenishment nodes is no more possible if multiple resource constraint are considered, which we like to keep open.

The RCAP is a specialization of the rolling stock rotation problem and has its roots in it. In rolling stock rotation planning the resource constraint models for example a maintenance constraint for railway vehicles (e.g., refueling), see Section 5.2. To model

[3] Our implementation can deal with such cycles, i.e., a cycle $C \subseteq A$ that does not contain any replenishment arc is considered as infeasible except for the case when $\sum_{a \in A} r_a = 0$. There is one exceptional case when cycles without any resource consumption and without any replenishment arcs appear: In rolling stock rotations there are sometimes constraints that a certain idle time has to be covered. A cycle that covers exclusively an idle time is feasible and does clearly not cover any driven distance.

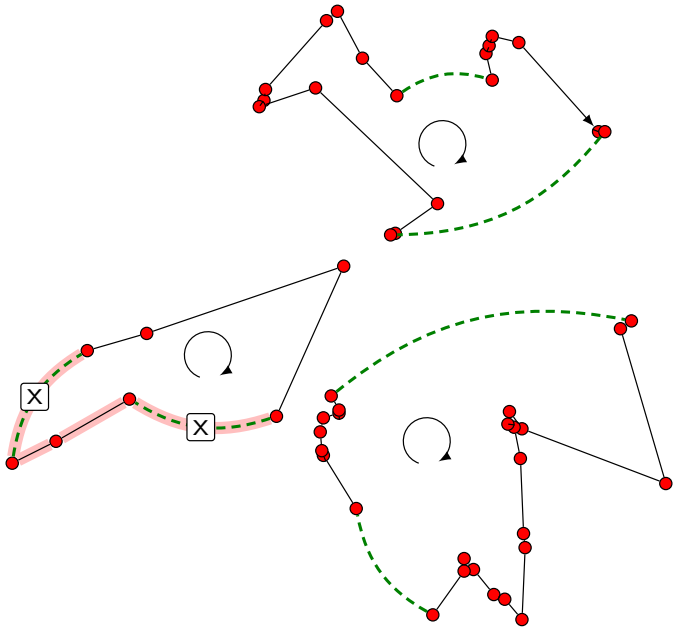


Figure 3.2.: RCAP solution: A cycle partition of a directed graph. The dashed arcs indicate replenishment arcs. The red path is a resource path where the replenishment events are indicated by the crosses.

time or distance consumptions directly before or after replenishment events at the arc $a \in A$ one can use the pair (r_a^1, r_a^2).

As already explained, the RCAP generalizes the symmetric and asymmetric traveling salesman problem. In the ATSP example from Section 3.2 all the arcs that come into the depot node would perform the replenishment event "drop the flowers" and all other arcs the event "take the flower of the past city".

Moreover, the RCAP extends variants for the VRP. For example, the symmetric capacitated vehicle routing problem (CVRP) [37] is to find a minimal set of cycles, namely *tours*, in a complete undirected graph $G = (V \cup \{d\}, E)$ with node demands $r_v \in \mathbb{Q}$ for all $v \in V$ such that each node of V is covered exactly once by one cycle, each cycle covers d exactly once, $\sum_{v \in V \cap C} r_v \leq B$ holds for each cycle C of the solution, and the solution minimizes some linear objective function $c : E \mapsto \mathbb{Q}$. For the CVRP the minimal number of tours $t \in \{1, \ldots, |V|\}$ is sufficiently approximated by a lower bound that is called *L1* (see Section 3.6.4) in the literature, i.e.,

$$t = \left\lceil \frac{\sum_{v \in V} r_v}{B} \right\rceil$$

holds for most of all instances that we consider. An instance of the CVRP can be modeled as an instance of the RCAP by introducing t copies of d, using the resource function of the outgoing arcs of a node to model the demand of the node, and declaring the incoming arcs of the depot node as replenishment arcs.

We use the proposed transformations for the TSP, ATSP, and CVRP for our computational study and, moreover, they show that:

Proposition 2. *The RCAP is $\mathcal{NP}$-hard.*

Let RCAP' be the problem if we relax the resource constraint in the RCAP with the graph (V, A), i.e., if any $a \in A$ is a replenishment arc and fulfills $(r_a^1, r_a^2) = (0, 0)$. The standard standard assignment problem (AP) [70] is to find a cost minimal perfect matching in a complete bipartite undirected graph $(T \cup H, E)$ with $T \cap H = \emptyset$ for some linear objective function.

The following lemma results from the definition of the RCAP and particularly motivates its name.

Lemma 3. *The RCAP' over the directed graph $D = (V, A)$ is equivalent to the AP over the undirected bipartite graph $G = (T \cup H, E)$.*

Proof. Let $t : V \mapsto T$ and $h : V \mapsto H$ be two bijective functions. An arc $a = (u, v) \in A$ with $u, v \in V$ of (V, A) can be bijectively transformed to an edge $e = \{t(u), h(v)\}$ of G such that $c(a) = c(e)$. Therefore any solution $A_0 \subseteq A$ of the RCAP' can be uniquely transformed to a solution $E_0 \subseteq E$ of the AP and vice versa. □

In the remaining part of the chapter we assume that a node of V is associated with a tail and a head node of T and H as it is introduced in the proof of Lemma 3. Since edges in $(T \cup H, E)$ and arcs in (V, A) have an one-to-one correspondence, we identify arcs of the directed graph (V, A) for the RCAP and edges of the undirected graph $(T \cup H, E)$ of the AP. We either use (V, A) or $(T \cup H, A)$ as notation for the considered graphs depending on what is appropriate. During Section 3.4 we use both graphs because the primal Hungarian method is originally an algorithm for the standard assignment problem that we modify in order to use it heuristically for the RCAP as well.

3.4. Local Search: The Primal Hungarian Method

The algorithm proposed by Balinski and Gomory (1964, [20]), which we call *primal Hungarian method* in this thesis, is the basis for the RS approaches of the next sections. We introduce this algorithm in this section[4].

The presentation here is very close to the implementation that we made. In fact, the closeness to an implementation is exactly what we need for our regional search approach and distinguishes this section from the original paper [20]. Even if not all details of this section are required to understand the remaining part of the chapter we give a complete presentation of the primal Hungarian method in order to have a standalone reference. The essentials that are really needed to understand the remaining part of the chapter are summarized at the end of this section.

[4]The beginning of the section title, i.e., "Local Search: ..." was a conscious decision for two reasons. First, the primal Hungarian method can indeed be seen as a local search algorithm. The second more important reason is to gain the three succeeding section titles "local, regional, and global search" in the thesis' table of contents.

Let $(T \cup H, A)$ be a complete bipartite graph with $|T| = |H|$ and $c : A \mapsto \mathbb{Q}$. Further, let $x_a \in \{0, 1\}$ be a binary decision variable that is equal to one if the arc $a \in A$ belongs to a solution and zero otherwise. Denote by $\pi_u^t \in \mathbb{Q}$ a free dual variable for each tail node $u \in T$ and by $\pi_v^h \in \mathbb{Q}$ a free dual variable for each head node $v \in H$. We denote the set of incoming and outgoing arcs of $v \in V$ by $A(v)^{\text{in}} := \{a \in A \,|\, a = (u, v)\}$ and $A(v)^{\text{out}} := \{a \in A \,|\, a = (v, w)\}$, respectively. The standard assignment problem can be formulated by the following linear program:

$$\begin{aligned} \min \ & \textstyle\sum_{a \in A} c_a x_a \\ \text{s.t.} \quad & \sum_{a \in A(v)^{\text{in}}} x_a = 1, \quad \forall\, v \in T \\ & \sum_{a \in A(v)^{\text{out}}} x_a = 1, \quad \forall\, v \in H \\ & x_a \geq 0, \quad \forall\, a \in A. \end{aligned} \tag{$\text{AP}_{\text{PRIMAL}}$}$$

The dual linear program of program ($\text{AP}_{\text{PRIMAL}}$) is:

$$\begin{aligned} \max \ & \textstyle\sum_{u \in T} \pi_u^t + \sum_{v \in H} \pi_v^h \\ \text{s.t.} \quad & \pi_u^t + \pi_v^h \leq c_a, \quad \forall\, a = (u, v) \in A \\ & \pi_u^t \in \mathbb{Q}, \quad \forall\, u \in T \\ & \pi_v^h \in \mathbb{Q}, \quad \forall\, v \in H. \end{aligned} \tag{AP_{DUAL}}$$

In each basic solution of program ($\text{AP}_{\text{PRIMAL}}$) the x-variables are all binary, see [20]. Thus, the integrality constraints for them can be relaxed if one solves program ($\text{AP}_{\text{PRIMAL}}$) with a simplex method. Let $d_a := c_a - \pi_u^t - \pi_v^h$ be the *reduced cost* of the arc $a = (u, v) \in A$. By the strong duality theorem the x- and π-variables have optimal values if and only if they are feasible for program ($\text{AP}_{\text{PRIMAL}}$) and ($\text{AP}_{\text{DUAL}}$) and the reduced cost of all "active" x-variables are zero, i.e.:

$$x_a \cdot d_a = 0, \quad \forall a \in A. \tag{3.2}$$

The original Hungarian method proposed by Kuhn (1955, [70]) can be briefly summarized as follows. Start with a feasible solution for program (AP_{DUAL}) and choose an initial (possibly empty) matching in $(T \cup H, A)$ with corresponding x-variables for program ($\text{AP}_{\text{PRIMAL}}$) such that (3.2) is fulfilled. In each iteration of the (dual) Hungarian method the size of the current incumbent matching is increased under preserving (3.2) and dual feasibility. The iteration stops if the current incumbent matching becomes perfect. In fact, the original procedure provides dual feasibility at every stage of the algorithm and can therefore be seen as a dual method.

In the *primal* Hungarian method programs ($\text{AP}_{\text{PRIMAL}}$) and ($\text{AP}_{\text{DUAL}}$) change their roles. It starts with a perfect matching in $(T \cup H, A)$ and an (almost arbitrary) configuration of the π-variables that need not be feasible for program (AP_{DUAL}) but have to

satisfy (3.2). In each iteration of the primal Hungarian method the perfect matching in $(T \cup H, A)$ is improved to another perfect matching until all arcs have positive reduced cost, i.e., the π-variables are feasible for program (AP_{DUAL}). The same distinction of the primal and (dual) Hungarian method holds for the primal and dual simplex algorithm.

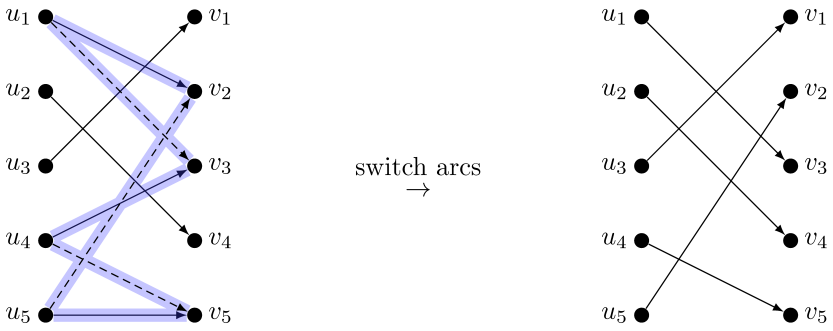


Figure 3.3.: Alternating cycle applied to a perfect matching in a bipartite graph $(T \cup H, A)$ with $\{u_1, u_2, u_3, u_4, u_5\} \subseteq T$ and $\{v_1, v_2, v_3, v_4, v_5\} \subseteq H$. The dashed arcs on the left are the new arcs.

The improvements found by the primal Hungarian method have a dedicated structure. Given a perfect matching $M \subseteq A$, an *alternating cycle* is a cycle in (the underlying undirected graph of) $(T \cup H, A)$ that alternates between arcs of M and $A \backslash M$. This is illustrated in Figure 3.3. On the left there is a bipartite graph with five nodes. The perfect matching is represented by the black arcs. The dashed arcs do not belong to the matching but to the alternating cycle (which is blue). It is easy to see, that if we delete all arcs of the matching that are contained in the alternating cycle and add all dashed arcs we get another perfect matching, which is illustrated on the right of Figure 3.3.

```
1  primalHungarianMethod( T ∪ H, A )
2  {
3     initializePrimals();
4     initializeDuals();
5
6     for( a* ∈ { a ∈ A | d_a < 0 } ) // pricing loop
7     {
8        if( findAlternatingCycle( a* ) )
9        {
10          applyAlternatingCycle( a* );
11       }
12    }
13 }
```

Algorithm 3.1: General workflow of the primal Hungarian method.

Algorithm 3.1 describes the general workflow of the primal Hungarian method. We start with the construction of a perfect matching in $(T \cup H, A)$. This is indicated by the

function INITIALIZEPRIMALS(), which can be arbitrarily implemented. Afterwards, we initialize the dual variables as illustrated in Algorithm 3.2. It is easy to see, that (3.2) is fulfilled under this initialization.

Suppose that we found an arc $a^\star \in A$ with negative reduced cost, i.e., $d_a < 0$ during the pricing loop. An iteration of the primal Hungarian method is a single call of Algorithm 3.3, which takes $a^\star$ as input argument. If this function returns `true`, an alternating cycle that leads to an improvement has been found and is applied to the current incumbent solution as indicated by the function APPLYALTERNATINGCYCLE($a^\star$), which is trivially to implement. Otherwise, if the function returns `false` no alternating cycle has been found. But in this case, the dual variables have been modified such that $d_{a^\star} \geq 0$.

```
1  INITIALIZEDUALS()
2  {
3    for( v ∈ H )
4    {
5      u := TAIL(v); // tail of v in current matching
6      π_u^t := c_(u,v);
7      π_v^h := 0;
8    }
9  }
```

Algorithm 3.2: Initialization of the dual variables.

We start the search for an improving alternating cycle in line 5 of Algorithm 3.3 on page 79 with the current tail node of $v^\star$ and perform a breadth-first search (BFS). Whenever a tail or head node is processed during this BFS we label these nodes with the predecessor nodes in a sub-routine, which is illustrated by Algorithm 3.4. The BFS is applied to the *equality set* $A^0 = \{a \in A \,|\, d_a = 0\}$ restricted to all tail and head nodes that have not become labeled yet, see lines 9 to 22 of Algorithm 3.3.

If the tail node $u^\star$ of $a^\star$ could not be reached, which would close an alternating cycle, we modify the dual variables in order to "make them more feasible". The following property directly derives from the shape of the dual linear programs ($\text{AP}_{\text{PRIMAL}}$) and ($\text{AP}_{\text{DUAL}}$) and is the core property on that the primal Hungarian method is based on[5]:

We can choose any $\epsilon \in \mathbb{Q}$ and increase the dual variables (by ϵ) of all tail nodes that have an outgoing arc in A^0 if we simultaneously decrease the duals (by ϵ) of all head nodes that have an incoming arc in A^0.

Fortunately, under such an ϵ-modification the condition (3.2) is clearly preserved because for all $a = (u, v) \in A$ with $x_a = 1$ we either decrease the dual for the tail node u and increase the dual for the head node v *by the same value* or we do not modify both duals. Hence, $d_a = 0$ still holds for all $a \in A$ with $x_a = 1$.

In order to ensure that the primal Hungarian method terminates, we choose the value for ϵ as small as possible but positive, see lines 24 to 27 of Algorithm 3.3. This particular

[5]The same holds for the original dual Hungarian method and the primal and dual version of the network simplex algorithm.

choice ensures that once an arc has positive reduced cost it will not get negative reduced cost again. More precisely, let d'_a of any arc $a \in A$ denote the new reduced cost that appear after an ϵ-modification is performed. The primal Hungarian method ensures:

$$d_a \geq 0 \quad \Rightarrow \quad d'_a \geq 0. \qquad \text{(no cycling)}$$

Due to the choice of the ϵ in the ϵ-modification we either extend A^0 such that we can still hope to reach the tail node $u^\star$ of $a^\star$ by continuing the BFS in lines 32 to 42 of Algorithm 3.3 or we modify the dual variables such that

$$d_{a^\star} < 0 \quad \Rightarrow \quad d'_{a^\star} \geq 0. \qquad \text{(dual improvement)}$$

Consequently, we stop the BFS if we reach the tail node $u^\star$ of $a^\star$. The alternating cycle $C \in A$ is then defined by the (predecessor-) labels and alternates between arcs of the current incumbent perfect matching and arcs of $A^0 \cup \{a^\star\}$.

The function APPLYALTERNATINGCYCLE($a^\star$) could be implemented as follows: Set $x_a = 0$ for all arcs $a \in A$ of the alternating cycle with $x_a = 1$ in the current matching and set $x_a = 1$ for the other arcs $a \in A$ of the alternating cycle with $x_a = 0$ in the current primal solution. All new arcs $a \in A \setminus \{a^\star\}$ of the alternating cycle were chosen such that $d_a = 0$. In order to ensure (3.2) also for $a^\star$ we finalize the improvement step by setting $\pi^h_{v^\star} := \pi^h_{v^\star} - d_{a^\star}$. An alternative for this is to call the function INITIALIZEDUALS() after each primal improvement.

```
1  isAlternatingCycle( (u, v) ∈ A, u* ∈ T )
2  {
3    w := tail(v); // tail of v in current matching
4    Q := Q ∪ {w};
5
6    L_w := v; // set label of tail node w
7    L_v := u; // set label of head node v
8
9    if( w == u* ) { return true; }
10   else          { return false; }
11 }
```

Algorithm 3.4: Check for an alternating cycle.

Let $M \subseteq A$ be a perfect matching and $M' \subseteq A$ be the resulting perfect matching if we apply the alternating cycle C that has been found for $a^\star \in A$ with $d_{a^\star} < 0$. We have an improvement of the objective function, i.e., $c(M') - c(M) < 0$ because we can subtract all dual variables associated with nodes covered by C on both sides of the inequality and get $d(M') - d(M) < 0$. All arcs of M and M' have zero reduced cost except for $a^\star$:

$$c(M') - c(M) = d(M') - d(M) = d_{a^\star} < 0. \qquad \text{(primal improvement)}$$

The properties (primal improvement), (dual improvement), and (no cycling) provide the correctness of the primal Hungarian method.

```
1  findAlternatingCycle( a* = (u*, v*) ∈ A )
2  {
3    L_{v*} := u*; // set label of head node v*
4
5    Q := { tail( v* ) }; // start BFS with tail of v* in current matching
6
7    while( d_{a*} < 0 )
8    {
9      while( Q ≠ ∅ )
10     {
11       u := choose( Q );
12
13       Q := Q\{u};
14
15       for( (u, v) ∈ {a = (u, v) ∈ A | v is not labeled, d_a = 0} )
16       {
17         if( isAlternatingCycle( (u, v), u* ) )
18         {
19           return true; // alternating cycle has been found
20         }
21       }
22     }
23
24     J := {a = (u, v) ∈ A | u is labeled, v is not labeled, d_a ≥ 0};
25
26     if( J ≠ ∅ ) { ε := min{d_a | a ∈ J}; }
27     else       { ε := −d_{a*}; }
28
29     for( u ∈ {u ∈ T | u is labeled} ) { π^t_u := π^t_u + ε; }
30     for( v ∈ {v ∈ H | v is labeled} ) { π^h_v := π^h_v − ε; }
31
32     J := {a = (u, v) ∈ A | u is labeled, v is not labeled, d_a = 0};
33
34     for( v ∈ {w ∈ H | ∃ a = (u, w) ∈ J} )
35     {
36       (u, v) := choose( J );
37
38       if( isAlternatingCycle( (u, v), u* ) )
39       {
40         return true; // alternating cycle has been found
41       }
42     }
43   }
44
45   // no alternating cycle has been found, but
46   // d_{a*} ≥ 0 is accomplished by ε←modifications
47   return false;
48 }
```

Algorithm 3.3: Search for an alternating cycle.

Since our explanation of the primal Hungarian method in this section is very close to an implementation we refer to the original paper [20] for a formal proof of the correctness of the presented algorithms.

The main modifications to the primal Hungarian method in the next sections can be briefly summarized as follows. In Sections 3.5.2.1 and 3.5.2.2 we essentially develop replacements for the function APPLYALTERNATINGCYCLE($a^\star$). In Section 3.5.2.3 we show an extension for the choice of the initial dual variables, i.e., Algorithm 3.2. This extension is made in order to diversify the regional searches of Sections 3.5.2.1 and 3.5.2.2. For the remaining part of the chapter it is enough to remember the Algorithms 3.1 and 3.2 without all the very interesting details that we presented in sub-routines.

3.5. Regional Search

The RCAP is a multifaceted combinatorial optimization problem in the sense that the variability in computational effort needed to solve an instance to proven optimality is huge. On the one hand, a small instance can be computational hard to solve, e.g., CVRPs. On the other hand, large problem instances, in which the resource constraint is less restrictive, might be solved with little computational effort. We aim at utilizing this characteristic for our algorithmic design. The idea is that the algorithm should automatically allot less computation time to easy instances and more computation time to hard ones. We call this behavior *self-calibration*. Note that this desirable property is not evident for local search algorithms or metaheuristics in general.

Our idea to implement this design is referred to as *regional search (RS)*. It works as follows. Let P be a combinatorial optimization problem and let P' be a relaxation of P. Consider a feasible solution S for P, interpret S as a solution S' for P' for the moment, and consider a *local search* algorithm A' that *exactly solves* P'. In order to turn A' into an algorithm A that searches for improvements of S we "lift" the neighborhoods that are roamed by A' for S' back to the original problem P. In other words, the relaxation induces a neighborhood w.r.t. S. The lifted neighborhoods are called *regions* in order to highlight that they are exact for P', i.e., A is automatically exact if an instance of P' is considered. Exactly this algorithmic behavior is our characterization of an RS:

Definition 3.5.1 (Regional search (RS))**.** *Let P be a combinatorial optimization problem and let A be a primal heuristic algorithm for P. Further, let P' be a relaxation of P. The algorithm A is a regional search if A is exact for any instance of P'.*

In this way, the computational effort of A is related to the difference in tractability between P and P', i.e., A can be expected to be self-calibrating.

Our first attempt to instantiate a concrete RS algorithm is presented in the next Section 3.5.1. We have not implemented this algorithm because its purpose is to present the idea of a standardized setting: We explain an RS for binary programs by using the simplex algorithm. The remaining part of the section is dedicate to two proper RS algorithms for the RCAP, which can be seen as specializations of the simplex based RS of Section 3.5.1.

In Section 3.5.2 we present two alternative RS algorithms for the RCAP. The first one is based on a branching scheme for a CIP[6], while the second RS algorithm is based on k-opt moves. Both RS algorithms for the RCAP are presented here because it turns out that no one of them dominates the other one in terms of computation time *and* solution quality.

3.5.1. Regional Search for Binary Programs by using the primal Simplex Algorithm

Given a rational matrix A and vectors b and c of suitable dimensions, we consider a binary program (BP) as

$$\begin{array}{ll} \min\{c^T x \,|\, Ax = b,\ x \text{ binary}\} & \text{with its LP relaxation} \\ \min\{c^T x \,|\, Ax = b,\ 0 \leq x \leq 1\} & \end{array}$$

that we call LPR. Our RS for BPs assumes that a feasible starting solution $x^\star$ is at hand, i.e., all values of $x^\star$ are binary and $Ax^\star = b$. We now interpret $x^\star$ as a basic solution of LPR and try to improve $x^\star$ by using the well known primal simplex algorithm. The primal simplex algorithm iteratively improves a basic incumbent solution by searching through the *simplex neighborhood*. The simplex neighborhood of a basic solution $x^\star$ of LPR is defined as the set of all basic solutions of LPR that share an edge with $x^\star$ in the polytope associated with LPR. We denote $\tilde{x} \sim x^\star$ if the basic solutions $\tilde{x}$ and $x^\star$ of LP share such an edge.

We now perform an improvement step of the primal simplex algorithm and end up with another basic solution $\tilde{x}$ for LPR with $\tilde{x} \sim x^\star$ and $c^T\tilde{x} < c^T x^\star$ (assuming a non-degenerate simplex operation). In general, $\tilde{x}$ will not be binary, i.e., feasible for the BP. In order to improve the chances to reach an improving binary vector we "lift" the simplex neighborhood as follows. If $\tilde{x} \sim x^\star$ and $c^T\tilde{x} < c^T x^\star$ we solve the following program:

$$\min\{c^T x \,|\, Ax = b,\ x \text{ binary},\ x_j = 1\,\forall \text{ column indices } j \,:\, x^\star_j = \tilde{x}_j = 1\}. \quad (\mathrm{BP}_{\mathrm{REGION}})$$

Program ($\mathrm{BP}_{\mathrm{REGION}}$) is to solve the BP under the additional constraint that all variables that agree to be one in both solutions of $\tilde{x} \sim x^\star$ are fixed. Note that $x^\star$ is always a feasible solution to program ($\mathrm{BP}_{\mathrm{REGION}}$) and $\tilde{x}$ is always a feasible solution to the LP relaxation of program ($\mathrm{BP}_{\mathrm{REGION}}$). The motivation behind this setup is to gain a computational compromise between the goals (1) improvement of the objective function value while (2) preserving feasibility and (3) solving small sub-problems in order to be fast. Goal (1) is promised by the simplex algorithm through $c^T\tilde{x} < c^T x^\star$ and goal (2) is meet by solving a restricted version of the original BP in which the current incumbent solution is always feasible. Goal (3) is achieved if the difference of successive basic solutions within the simplex algorithm is small. In this case, a large number of variables that agree to be one lead to a huge simplification of program ($\mathrm{BP}_{\mathrm{REGION}}$) compared to the original problem.

[6]Note that the CIP presented in Section 3.5.2.1 is also the basis for the global B&B procedure, see Section 3.6.

We suggest to solve program (BP_{REGION}) whenever $\tilde{x} \sim x^\star$ and $c^T\tilde{x} < c^T x^\star$. Thus, we investigate all solutions that the simplex algorithm would investigate, which shows that the algorithm above is an RS for BPs according to Definition 3.5.1 (on page 80). It will always exactly solve BPs for which the LP relaxation has an integral optimal solution. In this way, every global search algorithm, i.e., exact algorithm, for problems that have an LP relaxation is an RS, but not every local search algorithm is regional. Note that the proposed algorithm can also be seen as an iterated variable neighborhood search algorithm, see Berthold (2014, [24]) for an overview in the recent context of mixed integer non-linear programming.

We conclude this section by the argument that the main algorithmic ingredients for an RS approach are at hand if one comes up with an exact algorithm and a LP relaxation for an optimization problem.

3.5.2. Regional Search for the RCAP

Algorithm 3.5 outlines the main loop of our RS algorithms for the RCAP. Within this loop Algorithm 3.6 is called as long as improvements of the current incumbent solution are found. We do not explicitly distinguish current and improved solutions but assume that the current solution is updated as soon as an improvement has been found. This loop is the same as in local search algorithms. Here, we decide to replace the term "local" by "regional" whenever we feel that this is appropriate.

Before the regional search loop can be executed, we have to find an initial cycle partition in (V, A) that is feasible w.r.t. the resource constraint. How to find an initial feasible solution is described in Section 3.7 when we come to concrete computations. We initialize the dual variables as described in the original Algorithm 3.2.

```
1 {
2   while( isRegionallyOptimal() == false ) { }
3 }
```

Algorithm 3.5: Regional search loop.

Algorithm 3.6 outlines the overall procedure to "prove regional optimality". This procedure is the same for both, the branching-based and move-based RS version, which we describe in Sections 3.5.2.1 and 3.5.2.2, respectively.

In lines 5 up to 9 of Algorithm 3.6 we exactly perform the same steps as in the original primal Hungarian method. Note that each arc with negative reduced cost potentially leads to an improvement. Consequently, we test every arc (with negative reduced cost) in order to prove regional optimality.

We essentially apply the following two modifications to the original primal Hungarian method in order to gain RS algorithms for the RCAP:

Modification I: Function tryAlternatingCycle($a^\star$);

It is very likely that the resource constraint is violated if we just directly apply the alternating cycles because we did not spend any attention to this constraint so far. Therefore, we try to apply them but simultaneously test some alternatives in both of our RS versions. This is indicated by the function tryAlternatingCycle($a^\star$) in Algorithm 3.6. The implementation of this function distinguishes the branching-based and move-based RS.

Modification II: Outer loop of Algo. 3.6 & function orthogonalizeDuals();

We additionally diversify the search in terms of the dual variables. This strongly distinguishes our approach from diversification methods that are used in metaheuristics where often a diversification in terms of primal solutions is performed. Our dual diversification is indicated in Algorithm 3.6 by the loop that starts in line 3 and ends in line 18. The corresponding procedure is described in Section 3.5.2.3.

```
1  isRegionallyOptimal()
2  {
3    do
4    {
5      for( a* ∈ {a ∈ A | d_a < 0, a is enabled} ) // pricing loop
6      {
7        if( findAlternatingCycle(a*) )
8        {
9          if( tryAlternatingCycle(a*) )
10         {
11           for( a ∈ A ) { enable(a); }
12
13           return false; // report an improvement
14         }
15         disable(a*);
16       }
17     }
18   } while( orthogonalizeDuals() );
19
20   // report regional optimality, i.e.,
21   // no improvement has been found
22   return true;
23 }
```

Algorithm 3.6: Proof of regional optimality.

In order to avoid cycling of the algorithm we have to exclude arcs that were already tried from the pricing loop. This is indicated by function disable($a^\star$) in line 15 of Algorithm 3.6. Otherwise, arcs that are associated with alternating cycles that do not lead to an improvement will be priced again, which leads to an endless pricing loop. In the original version of the primal Hungarian method it can not happen that an arc

is priced more than one time, see condition (no cycling). In our heuristic version this is possible because it can happen that only a part of an alternating cycle, which may not even involve the priced arc, is applied. Therefore, we enable all arcs again if an improvement has been found, see function ENABLE($a^\star$), which is called in line 11.

3.5.2.1. Regional Search based on Branching

In our branching-based RS we install the alternating cycles found by the primal Hungarian method by a branching scheme. This scheme is based on a CIP for the RCAP. Based on an alternating cycle we setup a (hopefully) small (i.e., restricted) CIP in the implementation of the function TRYALTERNATINGCYCLE($a^\star$) that is called in Algorithm 3.6 for the priced arc $a^\star \in A$ associated with the alternating cycle. We call the small CIP *alternating cycle region*[7] and solve it in order to find an improvement under the resource constraint instead of directly applying the alternating cycle.

Before we come to further details we consider the constraint integer programming formulation for the RCAP from that we derive the alternating cycle region. To this end, we denote by $x_a \in \{0,1\}$ a variable that is equal to one if $a \in A$ belongs to a solution and zero otherwise. By using the constraint notation of Achterberg (2009, [12, Example 3.2])), the RCAP can be formulated as a CIP as follows:

$$\begin{aligned} \min\ & \textstyle\sum_{a\in A} c_a x_a \\ \text{s.t.}\ & \textstyle\sum_{a\in A(v)^{\text{in}}} x_a = 1 \quad \forall\, v \in V, \\ & \textstyle\sum_{a\in A(v)^{\text{out}}} x_a = 1 \quad \forall\, v \in V, \\ & \textsc{ResourceConstraint}(\, x\,), \\ & x_a \in \{0,1\} \quad \forall\, a \in A \end{aligned} \tag{RCAP$_{\text{CIP}}$}$$

with

$$\textsc{ResourceConstraint}(x) \ \Leftrightarrow \nexists P \in \mathbb{P}(\text{supp}(x)) \,:\, P \text{ is an infeasible path.}$$

By deleting the RESOURCECONSTRAINT from program (RCAP$_{\text{CIP}}$) we obtain its assignment relaxation: For every node $v \in V$ there must be exactly one integral incoming and outgoing arc variable which forces $x \in \mathbb{Q}^{|A|}$ to define a cycle partition of the nodes of (V, A). The resource constraint of program (RCAP$_{\text{CIP}}$) forces the cycle partition to not contain any infeasible paths. Program (RCAP$_{\text{CIP}}$) serves as a basis for the branching-based RS as well as for the branch-and-bound algorithm of Section 3.6.

We derive the alternating cycle region from program (RCAP$_{\text{CIP}}$) as follows. Let

$$C = \{a_1^+, a_1^-, \ldots, a_n^+, a_n^-\} \subseteq A$$

be the alternating cycle that is associated with $a^\star$ (arcs with a + as superscipt are the new ones). Further, let $x^\star \in \{0,1\}^{|A|}$ be the current incumbent solution to program (RCAP$_{\text{CIP}}$) that corresponds to the cycle partition $M \subseteq A$ and let $\tilde{x} \in \{0,1\}^{|A|}$

[7] The alternating cycle region can be seen as a neighborhood in terms of traditional local search.

correspond to the cycle partition that we obtain if we directly apply C to M. Analogous to the considerations for the simplex-based RS of Section 3.5.1 it is very unlike that $\tilde{x}$ is feasible again since we did not spend any attention to the resource constraint so far. Therefore, we "lift" the direct application of the alternating cycle C onto the solution of the alternating cycle region, i.e., we solve program ($\text{RCAP}_{\text{REGION}}$):

$$
\begin{aligned}
&\min \textstyle\sum_{a\in A} c_a x_a \\
&\text{s.t.} \quad \sum_{a\in A(v)^{\text{in}}} x_a \quad = 1, \quad \forall\, v \in V \\
&\qquad \sum_{a\in A(v)^{\text{out}}} x_a \quad = 1, \quad \forall\, v \in V \\
&\qquad\qquad \textsc{ResourceConstraint}(\, x\,), \\
&\qquad\qquad x_a \quad = 1 \quad \forall\, a \in M \setminus \{a_1^-, \ldots, a_n^-\}, \\
&\qquad\qquad x_a \quad \in \{0,1\}, \quad \forall\, a \in A.
\end{aligned}
\tag{$\text{RCAP}_{\text{REGION}}$}
$$

Solving this program increases the chances of finding an improved cycle partition under a resource constraint.

An evident interpretation of solving program ($\text{RCAP}_{\text{REGION}}$) is that the primal Hungarian method suggest to apply the cycle C associated with the priced arc $a^\star$ in order to improve the current value of the objective function. But this is too naive. In order to compensate the resource constraint, we only take the arcs that the cycle proposes to delete seriously.

In terms of the original program (RCAP_{CIP}) we fix all arc variables to value one that agree to be of value one before and after the application of the alternating cycle. Note that alternating cycles would also appear if we use the primal simplex algorithm instead of the primal Hungarian method because the primal simplex follows exactly the same duality arguments and the symmetric difference of two vertices $\tilde{x}$ and $x^\star$ of the assignment polytope with $\tilde{x} \sim x^\star$ is exactly an alternating cycle, see Balinski and Russakoff (1974, [21]). Therefore, the idea of the alternating cycle region is exactly the same idea as the one introduced for BPs in Section 3.5.1: In the branching-based RS the CIP (RCAP_{CIP}) for the RCAP and the primal Hungarian method take over the roles of the BP and the simplex algorithm, respectively.

Program ($\text{RCAP}_{\text{REGION}}$) can be easily turned into a plain RCAP by replacing all constant arc variables associated with arcs of $\bigcup_{a=(u,v)\in M\setminus\{a_1^-,\ldots,a_n^-\}} (A(u)^{\text{in}} \setminus \{a\})$. Thus, we are allowed to and, indeed do, solve program ($\text{RCAP}_{\text{REGION}}$) by the B&B algorithm presented in Section 3.6. Figure 3.4 illustrates the idea of the alternating cycle region.

Obviously, this method is an RS for the RCAP w.r.t. its assignment relaxation in terms of Definition 3.5.1 because it investigates at least all solutions, i.e., all solutions that can be reached by improving alternating cycles, that the primal Hungarian method would consider. By loading a standard assignment problem into the branching-based RS we obtain one of its optimal solutions including an optimality proof (i.e., dual variables

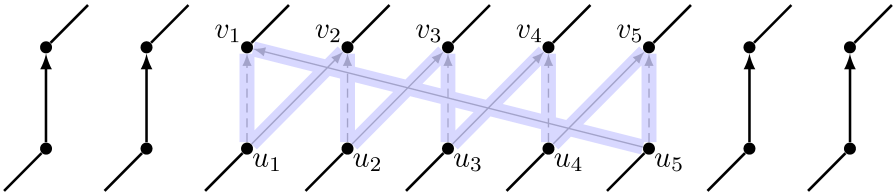


Figure 3.4.: Alternating cycle region: The arcs that are not dashed form a part of the current incumbent cycle partition. The alternating cycle $C = \{a_1^+, a_1^-, a_2^+, a_2^-, a_3^+, a_3^-, a_4^+, a_4^-, a_5^+, a_5^-\} = \{(u_1, v_1), (u_5, v_1), (u_5, v_5), (u_4, v_5), (u_4, v_4), (u_3, v_4), (u_3, v_3), (u_2, v_3), (u_2, v_2), (u_1, v_2)\}$ is "suggested" by the primal Hungarian method and illustrated in gray. The alternating cycle region is to solve a restricted CIP (i.e., a smaller RCAP) that derives by fixing all variables that are associated with black arcs to value one.

that fulfill program (AP_{DUAL})).

The proposed implementation of function tryAlternatingCycle($a^\star$) within the RS loop (i.e., Algorithm 3.5) has an exponential worst case complexity: We still have to solve program ($\text{RCAP}_{\text{REGION}}$), which is a hopefully sufficiently small RCAP. In fact, it turns out that it is computationally too short-sighted to always search for an optimal solution of program ($\text{RCAP}_{\text{REGION}}$). It can happen that the arising problems are almost as hard as the original instance if the alternating cycle is large enough. Those cases are exceptional but not impossible. We resolve this issue by setting a limit of 10^3 branching nodes during depth-first-search [12] for model ($\text{RCAP}_{\text{REGION}}$). Note that we still perform an RS even under this node limit.

3.5.2.2. Regional Search based on k-opt Moves

Given a set of arcs A', an operation that deletes k arcs from A' and simultaneously adds k new arcs to A' is widely known as a *k-opt move* in the literature. In fact, the concept of k-opt moves was our original starting point for the development of the RS algorithms. In particular, the move-based RS, which we explain in this section, was our first RS version published in [2].

Many combinatorial motivated local search algorithms are based on dedicated k-opt moves. For example the k-opt algorithm for the TSP over the undirected graph (V, E) checks for an incumbent Hamiltonian tour $T \subseteq E$ if there exist two subsets $K \subset E$ and $X \subset T$ with $|K| = k$ and $|X| = k$ such that $(T \backslash X) \cup K$ is an improved tour.

An important special class of k-opt moves are the *sequential k-opt moves*, see Irnich, Funke, and Grünert (2006, [67]), that are characterized by a cycle that alternates between arcs to delete and new arcs. Indeed, the most successful heuristics for the TSP are based mainly on sequential k-opt moves, see Helsgaun (2009, [64]).

If we compute an optimal partition of cycles in the graph (V, A) with objective function $c : A \mapsto \mathbb{Q}_+$ by the primal Hungarian method we observe that this algorithm also performs sequential k-opt moves. A sequential k-opt move is illustrated in Figure 3.5. In fact, Figure 3.5 shows the equivalent operation in the graph (V, A) that is defined by the

Figure 3.5.: k-opt move defined by an alternating cycle in the graph (V, A) with $\{v_1, v_2, v_3, v_4, v_5\} \subseteq V$. The k-opt move with $k = 3$ corresponds to the alternating cycle in the undirected graph shown in Figure 3.3.

alternating cycle in the undirected bipartite graph $(T \cup H, A)$ of Figure 3.3. The nodes u_i and v_i in Figure 3.3 are the tail and head nodes of the node v_i in Figure 3.5.

Moreover, the k-opt moves found by the primal Hungarian method have the following properties:

Lemma 4. *Consider an instance of the RCAP with relaxed resource constraint, i.e., an instance of the AP with the directed graph (V, A) and objective function $c : A \mapsto \mathbb{Q}_+$. Let $C_0 \subseteq A$ be some cycle partition in (V, A).*

There is always a sequence of $m \leq |V|$ sequential k-opt moves such that the sequence of cycle partitions $C_0, C_1, \ldots, C_m$ with $C_i \subseteq A$ for $i \in \{1, \ldots, m\}$ that result from applying the k-opt moves fulfills $c(C_0) > c(C_1) > \ldots > c(C_m)$ and C_m is optimal w.r.t. c.

Proof. The primal Hungarian method produces exactly such a sequence of sequential k-opt moves for an arbitrary initial primal solution $C_0 \in A$. □

Clearly, Lemma 4 does not provide a deep new insight, but it emphasizes an important fact. Since the objective value decreases properly by applying the k-opt moves, the method does not suffer from degeneracy. Note that Lemma 4 does not hold for the TSP, e.g., the so called *quad-change* with $k = 4$ can not be decomposed into sequential k-opt moves, see Lin and Kernighan (1973, [72]).

In terms of Lemma 4, the primal Hungarian method can be seen as an exact local search procedure for the assignment problem since a feasible primal solution is always at hand. As already mentioned, in the case of the RCAP clearly not all sequential k-opt moves found by the primal Hungarian method lead to another feasible solution. Our idea is to use the original moves as a suggestion and to only apply parts of those moves that lead to a feasible improved solution. To this end, we implement the sub-routine TRYALTERNATINGCYCLE($a^\star$) that is called within Algorithm 3.6 within the move-based RS as follows.

Let $a_1^+ = (u_1, v_2)$ be an arc that is not contained in the current cycle partition. Further, let $a_1^- = (u_1, v_1)$ be the arc that goes out of the node u_1 w.r.t. the current cycle partition and let $a_2^- = (u_2, v_2)$ be the arc that comes into the node v_2. If we decide to insert a_1^+ in the current matching we have to delete a_1^- and a_2^- at least and the least onerous operation to close the cycle partition is to insert the *closing arc* $a_1^+ = (u_2, v_1)$. We call this 2-opt move defined by the alternating cycle $C = \{a_1^+, a_1^-, a_2^+, a_2^-\}$ FLIP(a_1^+).

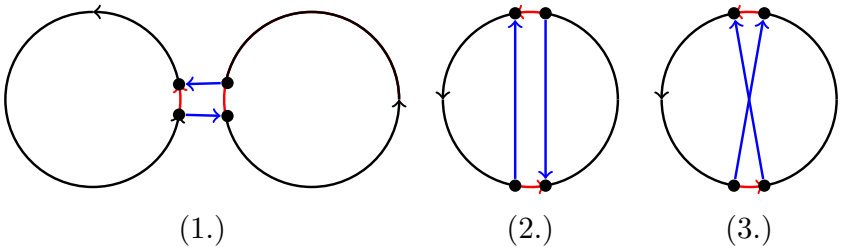


Figure 3.6.: Possible flip operations.

Figure 3.6 illustrates the possible operations performed by FLIP. The red arcs are the ones that we delete and the blue arcs are the ones that we add. In Sub-figures (1.) and (2.) the only two possibilities that arise if we exchange two arcs are shown: We either merge two cycles to a new one or split on cycle into two new cycles. The third sub-figure shows the relation to the usual 2-opt move for the symmetric TSP. In fact, the usual 2-opt move for the TSP [73] exchanges more than two arcs: It also inverts a segment of the current Hamiltonian cycle, which is clearly very different from the modifications performed by a flip.

In our data structure for the current matching the flip operation has complexity $\mathcal{O}(|V|)$ because in the latter we take use of a function of the data structure that evaluates if the current matching is feasible w.r.t. the resource constraint or not. To provide such a function one needs to know if two nodes are in the same cycle or not and thus we always have to do bookkeeping for the cycles of the nodes at least.

The following insight leads to our move-based RS implementation.

Lemma 5. *Let $C = \{a_1^+, a_1^-, \ldots, a_n^+, a_n^-\} \subseteq A$ be the alternating cycle found by Algorithm 3.6 where a_i^+ and a_i^- are the arcs to add and the arcs to delete, respectively. Further, let $M \subseteq A$ be the assignment that arises if we directly apply C to the current assignment.*

By applying any $n-1$ of the flips $\text{FLIP}(a_1^+), \ldots, \text{FLIP}(a_n^+)$ imposed by the plus-arcs of C in an arbitrary sequence we exactly obtain M.

Proof. By applying $n-1$ flips the assignment clearly contains $n-1$ of the a_i^+ arcs and each flip inserts a closing arc that is deleted by another flip because C is an alternating cycle. Thus, the assignment also contains the last of the n a_i^+ arcs. Otherwise it is not a matching what is obviously the case after apply a set of flip operations. □

We use this property in the following way. We perform two search strategies, namely a greedy strategy and an "anti-greedy" strategy, in the implementation of the function TRYALTERNATINGCYCLE($a^\star$).

Each of the two strategies consists of exactly $n-1$ succeeding flips. In the greedy strategy we always apply the flip that leads to the best objective function value w.r.t. the current assignment. In the anti-greedy search strategy we always apply that move that leads to the worst objective function value. By performing this two procedures we

iterate $2 \cdot (n-1) - 1$ different assignments and in particular also the assignment that results from directly applying the alternating cycle as emphasized by Lemma 5.

Finally, we continue with the best feasible assignment that appeared during the two search strategies. Note that it can also happen that this procedure can lead to non-sequential k-opt moves. This situation arises if we apply less than $n-1$ flips.

The implementation of the move-based RS within tryAlternatingCycle($a^\star$) performs $2 \cdot (n-1)$ greedy and anti-greedy iterations. In each iteration a flip operation with a complexity of $\mathcal{O}(|V|)$ is executed. Thus, the procedure has a run time complexity of

$$\mathcal{O}\left(|C| \cdot |V|\right),$$

which turns out to be fast when computing.

In terms of our branching-based RS from the previous section the move-based RS can be seen as a fast diving heuristic for the alternating cycle region, i.e., program ($\text{RCAP}_{\text{REGION}}$). In fact, in the move-based RS we basically perform two "dives" (i.e., one for greedy and another one for the anti-greedy strategy) when we interpret the succeeding 2-opt moves in a branching scheme for program ($\text{RCAP}_{\text{REGION}}$).

We close this section with the observation that our move-based RS is indeed a regional search according to Definition 3.5.1 because by Lemma 5 we can not lose any improvement when computing an instance of the standard assignment problem.

3.5.2.3. Dual Diversification

Algorithm 3.6 restricted to lines 5 to 17 results in a locally, namely regionally, optimal solution. In this situation it is worth to develop procedures that help to escape from local optimal solutions. In the literature there are many methods known for this purpose. Our approach strongly differs from those in the sense that we do not restart our RS algorithms for different primal starting solutions. Instead, we diversify the search in terms of the dual starting solutions. We call this approach *dual diversification*, which is performed by the outer loop (i.e., lines 3 and 18) of Algorithm 3.6.

The original motivation behind the dual diversification procedure is the observation that the initial dual variables (see Algorithm 3.2) for the primal Hungarian method can be chosen almost arbitrarily:

Lemma 6. *Let $M \subseteq A$ be a perfect matching in the complete bipartite graph $(T \cup H, A)$ with $c : A \mapsto \mathbb{Q}$. There are $|M|$ dual variables that can be chosen arbitrarily in order to derive a valid dual starting point for the primal Hungarian method.*

Proof. For each $a = (u, v) \in M$ we either choose the dual variable π_u^t or π_v^h arbitrarily. Then, we set the other such that $c_a = \pi_u^t + \pi_v^h$ for each $a = (u, v) \in M$. In this way, the reduced cost of all $a \in M$ are zero and (3.2) is fulfilled as well. Thus, these dual variables are a valid starting point for the primal Hungarian method. □

Lemma 6 provides the insight that we can choose an unlimited number of dual solution vectors as starting points. On the one hand, the particular choice of the dual starting point for the original primal Hungarian method has an insignificant effect (assuming

a computation from scratch, i.e., without "warm-start"). On the other hand, the dual solution determines which alternating cycles are tried by the RS versions and this, of course, has a significant effect on the solution quality.

Our diversification idea is to execute the pricing loop on top of $|V|$ dual starting points that are "sufficiently different". To this end, we (heuristically) consider $|V|$ vertices of the vector space $\mathbb{Q}^{|V|}$ to be sufficiently different if they form an orthogonal basis of $\mathbb{Q}^{|V|}$.

```
1  orthogonalizeDuals()
2  {
3    if( |B| == |V| )
4    {
5      // report that |V| dual starting solutions were tried
6      return false;
7    }
8
9    J := { i | B_{i,1} ≠ 0 }; // all non-zero indices in first dual vector
10
11   if( J == ∅ ) { return false; }
12
13   k := (min{ i | i ∈ J } + |B| − 1) mod |V|;
14
15   b := e_k; // e_k ∈ Q^{|V|} is an unit vector
16
17   // Gram-Schmidt orthogonalization loop
18   for( a ∈ B )
19   {
20     b := b − (a_k / <a,a>) a;
21   }
22
23   B := B ∪ {b}; // append new column b to matrix B
24
25   for( v ∈ H ) // set new dual variables as starting point
26   {
27     u := tail(v); // tail of v in current cycle partition
28     π^t_u := b_v;
29     π^h_v := c_(u,v) − π^t_u; // see proof of Lemma 6
30   }
31   return true;
32 }
```

Algorithm 3.7: Computation of a new dual starting point by a Gram-Schmidt orthogonalization.

Our implementation of this idea is outlined in Algorithm 3.7. Before the algorithm starts we initialize the matrix $B \in \mathbb{Q}^{|V|\times 1}$ that consists of a single column vector of dimension $|V|$ at the beginning. This initial column vector is defined by the values of all π^t dual variables (i.e., the tail dual variables) that we initialize exactly as described in the original Algorithm 3.2.

Then, the matrix B is iteratively extended by further column vectors until it forms

an orthogonal basis of the vector space $\mathbb{Q}^{|V|}$, i.e., until $B \in \mathbb{Q}^{|V|\times|V|}$ and for any two different column vectors $a \in \mathbb{Q}^{|V|}$ and $b \in \mathbb{Q}^{|V|}$ of B we have $< a, b >= 0$. One iteration (i.e., one extension) is accomplished by an iteration of a standard Gram-Schmidt process in lines 13 (in that we find the "next" dimension k) to 21. Note that we can not find a basis of $\mathbb{Q}^{|V|}$ if all arcs of the initial cycle partition have zero cost, because then the first column in B is the zero vector (as we initialize it according to Algorithm 3.2). But this is an exceptional case, which is caught in line 11 of Algorithm 3.7.

After each extension of B (i.e., one call of Algorithm 3.7 within the outer loop of Algorithm 3.6) we try to find and apply alternating cycles w.r.t. the new dual solution. Our hope associated with this Gram-Schmidt strategy is that we consider enough different dual starting points in order to try a reasonable set of alternating cycles by the RS versions. The "enough different" is claimed to be fulfilled by $|V|$ orthogonal vectors. Also randomly initialized dual starting points have been considered in preliminary computational experiments. But this idea has been discarded because we have not seen a significantly increased solution quality by using random vectors[8]. Moreover, the Gram-Schmidt procedure has the benefit of being deterministic: Two independent calls to the algorithm produce exactly the same result for equal input data (this holds for both of our RS versions).

This section concludes our regional search compilation except for computational results, see Section 3.7. In the next section we describe a global search approach for the RCAP that we particularly use as a sub-routine within the branching-based RS.

3.6. Global Search: A Branch and Bound Scheme for the RCAP

We present a standalone[9] B&B algorithm for the RCAP that is based on the constraint integer program (RCAP_{CIP}) presented in Section 3.5.2.1.

An alternative formulation for the RCAP in terms of a pure IP model can be derived by replacing the ResourceConstraint in model (RCAP_{CIP}) with the *infeasible path constraints*

$$\sum_{a\in P} x_a \leq |P| - 1, \forall \text{ infeasible paths } P \in \mathbb{P}(A). \tag{3.3}$$

However, we do not expect that this IP model will produce useful results. Another alternative way to model the RCAP as MIP is given in Section 5.2 in terms of the more general RSRP. We also do not use this model here because we expect that its is much less appropriate for our regional and global search purposes for the RCAP than program (RCAP_{CIP}).

Indeed, a vast number of papers – the most successful by now is Pecin et al. (2014, [85]) – consider much stronger formulations for the exact solution of the CVRP and the TSP, see the book by Toth and Vigo (2014, [101]). Here, we do not aim to generalize or

[8] We conjecture that $|V|$ randomly generated vectors form a basis of $\mathbb{Q}^{|V|}$ with high probability.

[9] By "standalone" it is meant that the algorithm can be implemented in a reasonable time without having access to commercial third-party software.

adopt those approaches to the RCAP, even if this is an interesting research area. Instead, we pursue a much simpler approach that can solve lightly constrained easy problems fast, namely a B&B algorithm that does not generate any primal or dual cutting planes. We refer to Fischetti, Toth, and Vigo, Toth and Vigo (1994, 2014, [49, 101]) for similar algorithms developed for the VRP.

We like to point out, that the B&B algorithm has a very low level of sophistication (e.g., compared to [85]). In contrast to this, it is currently not possible to fully integrate our algorithm in all commercial MIP solvers. Not all commercial solvers provide callbacks for user written branching rules and most of them are based on the assumption that the optimization model can be completely expressed as a MIP model.

As already mentioned, our B&B algorithm is based on program (RCAP_{CIP}) and its assignment relaxation RCAP′, which we use for bounding. In each node, called *sub-problem*, of the branching tree the following steps are performed while branching according to the scheme of Section 3.6.1:

- ▷ solve the assignment relaxation of the current RCAP
- ▷ eliminate arcs using the assignment reduction, see Section 3.6.2
- ▷ eliminate arcs using the shortest path reduction, see Section 3.6.3
- ▷ eliminate arcs using the bin-packing reduction, see Section 3.6.4
- ▷ discard current branching node if
  - ▷ the optimal objective value of the node's assignment relaxation is not below the upper bound
  - ▷ the optimal solution of the node's assignment relaxation is feasible
  - ▷ there are no further branching candidates, see Section 3.6.5.

In each *reduction* procedure we try to find detachable arcs of the current sub-problem that fulfill the following criterion: Any solution to the current sub-problem containing a detachable arc is definitely not better than the incumbent solution. If a reduction procedure detects an arc $a \in A$ fulfilling this criterion, we *detach* the arc from the current sub-problem, i.e., we delete the arc from the arc set A. Note that a detached arc remains detached in all child nodes of the branching tree. In the following sections, we explain our branching scheme and the three reduction procedures. We do not use a special notation to distinguish sub-problems from the original RCAP. Instead, we consider each branching node as a new RCAP instance.

3.6.1. Branching Scheme

Our algorithm uses the assignment relaxation of the RCAP to solve the subproblems in the branching tree. Thus, the solution of the current node relaxation is always integral. In fact, it is composed of a set of cycles $C_1, \ldots, C_k \subseteq A$. If all cycles contain at least one replenishment arc and all resource paths of $\mathbb{P}(\bigcup_{i=1}^{k} C_i)$ are feasible, we do not have

to perform further branching. Otherwise, we branch on arc variables, i.e., for each branching candidate $a = (u, v) \in A$ we create two new sub-problems. The first arises from forcing $x_a = 1$ and in the other one the constraint $x_a = 0$ is imposed. The latter case is handled by detaching $a \in A$ from the current sub-problem, while the former is handled by detaching all arcs of $(A(u)^{\text{out}} \cup A(v)^{\text{in}}) \setminus \{a\}$.

The following two situations lead to further branching on a certain sub-problem:

- ▷ a cycle, called *infeasible cycle*, of $\{C_1, \ldots, C_k\}$ does not contain a replenishment arc
- ▷ a path of $\mathbb{P}\left(\bigcup_{i=1}^{k} C_i\right)$ is infeasible.

Let $I = \{I_1, \ldots, I_m\}$ with $I_i \subseteq A$ for $i = 1, \ldots, m$ be the family of cycles and paths fulfilling one of these two criteria. In general it is valid to branch on each arc $a \in A$ of the current sub-problem, but it is natural to only branch on arcs $a \in \bigcup_{i=1}^{m} I_i$.

The set $\bigcup_{i=1}^{m} I_i$ can be large and the concrete choice of the branching candidate can have a huge effect on the computational performance, see [12]. However, there is usually little data at hand to make a well-founded decision on the branching candidate.

Our expectations on a branching rule are: (1) It should remove "infeasibilities" as early as possible; (2) It should increase the lower bound as much as possible; (3) It should be computationally easy; and (4) It should be unique (i.e., break ties) in order to avoid random decisions (often a simple rule is already better than random).

Many rules have been studied in the TSP, ATSP, and CVRP literature. In particular, the paper of Turkensteen et al. (2008, [102]) provides a literature review for the ATSP case. This paper suggests the following two criteria to qualify the arc $a \in A$ for branching:

1. Let $P \subseteq A$ that one infeasible path or cycle with $a \in P$. The criterion is $\text{PL}(a) := |P|$.
2. The criterion is the optimal objective function value of the node relaxation s.t. $x_a = 0$.

The maximization of criterion 2 is known as strong branching in the literature, see Achterberg (2009, [12]). In Turkensteen et al. (2008, [102]) it is suggested to lexicographically combine (we also always combine criteria lexicographically here) strong branching with minimizing criterion 1. The argumentation for this rule is conclusive and matches expectations (1) to (3). But we observed the following issue w.r.t. expectation (4). Let $a' \in A$ be an arc contained in an infeasible path or cycle. Following Turkensteen et al. (2008, [102]) we have to compute the *strong branching bound* $\text{SB}(a')$:

$$\text{SB}(a') := \min \sum_{a \in A \setminus \{a'\}} c_a x_a$$

$$\text{s.t.} \sum_{a \in A(v)^{\text{in}} \setminus \{a'\}} x_a = 1 \text{ and } \sum_{a \in A(v)^{\text{out}} \setminus \{a'\}} x_a = 1 \, \forall \, v \in V, \quad x_a \in \{0, 1\} \, \forall \, a \in A. \tag{RCAP$_{\text{SB}}$}$$

Our observation is that the values $\text{SB}(a')$ do not distinguish particular arcs, i.e., many arcs of an infeasible path or cycle give the same strong branching bound. This is

comprehensible because if we force $x_a = 0$, it is unlikely that all other arcs of the corresponding infeasible path or cycle remain. Whenever at least two arcs have the same strong branching bound the choice is random and can be expected to be "wrong" in half of all cases.

Our idea to diversify the strong branching bound is to introduce an additional constraint into (RCAP$_{\text{SB}}$) in order to force that things change. The constraint reads:

$$\sum_{i=1}^{k} \sum_{a \in C_i} x_a \geq |V| - 2. \tag{3.4}$$

It forces us to change at most two arcs of the current cycle partition to end up with another cycle partition. Similar constraints are well-known in a MIP concept that is called *local branching*, see Fischetti and Lodi (2003, [48]) for a different application. Denoting the bound that is given by model (RCAP$_{\text{SB}}$) including inequality (3.4) as LB(a) for $a \in A$, the following lemma holds.

Lemma 7. *LB$((u, v))$ can be computed exactly by a local search over all 2-opt moves that insert one arc of $A(u)^{in}$ into the optimal solution of the current node relaxation.*

Proof. Inequality (3.4) and equality $x_a = 0$ miraculously constrain to 2-opt moves. □

Consequently, a natural suggestion is to consider an arc $a \in A$ maximizing LB(a) for branching.

We remark that LB does also not diversify completely (which is impossible, e.g., if $c_a = 0$ for all $a \in A$) but much better than SB. To break the remaining ties, we introduce another criterion that depends on the branching history, see Berthold (2014, [24, Section 10.2]) for an overview. Suppose that we just computed the optimal solution of the assignment relaxation of a branching node[10] $j \in \mathbb{N}$ and that the arc $a \in A$ appears in this solution, i.e., $x_a = 1$. Let $z^\star$ be the assignment relaxation's optimal objective value. We store the triple $(z^\star, j, a)$ in a set O and define the *average objective value* AO(a) of the arc $a \in A$ as:

$$\text{AO}(a) := \frac{\sum\limits_{(z,j,a') \in O \,:\, a' = a} z}{|\{(z, j, a') \in O \,|\, a' = a\}|}.$$

At this point we considered the following four criteria for choosing a branching candidate $a \in \bigcup_{i=1}^{m} I_i$: PL($a$), SB($a$), LB($a$), and AO($a$). Each of these criteria can be minimized as well as maximized. Also any lexicographic order (e.g., first select all arcs $a \in A$ minimizing PL(a), of these maximize LB(a), etc.) can be chosen. This gives rise to $2^4 \cdot 4! = 384$ possibilities which we implemented all in order to close the discussion about our branching rule. To this end, we prove the optimality of an already optimal solution for the instances: `br17` (ATSP), `gr17` (TSP), and `eil22` (CVRP) (the initial optimal solutions provide equal opportunities).

Most of the 384 rules are obviously not competitive. But twelve rules are not evidently dominated. For each of the twelve rules the total number of branching nodes per instance can be found in Table 3.1.

[10] We assume that our branching nodes are naturally numbered.

inst.	branching rule	nodes
br17	max SB min PL max AO max LB	76425
eil22	max SB min PL max AO max LB	65769
gr17	max SB min PL max AO max LB	765
br17	max SB min PL max LB max AO	78579
eil22	max SB min PL max LB max AO	59833
gr17	max SB min PL max LB max AO	785
br17	max SB max AO min PL max LB	52415
eil22	max SB max AO min PL max LB	61155
gr17	max SB max AO min PL max LB	781
br17	max SB max AO max LB min PL	44581
eil22	max SB max AO max LB min PL	61247
gr17	max SB max AO max LB min PL	781
br17	max SB max LB min PL max AO	35959
eil22	max SB max LB min PL max AO	57941
gr17	max SB max LB min PL max AO	795
br17	max SB max LB max AO min PL	32159
eil22	max SB max LB max AO min PL	57853
gr17	max SB max LB max AO min PL	795

inst.	branching rule	nodes
br17	max LB max SB min PL max AO	44233
eil22	max LB max SB min PL max AO	40961
gr17	max LB max SB min PL max AO	485
br17	max LB max SB max AO min PL	30103
eil22	max LB max SB max AO min PL	41193
gr17	max LB max SB max AO min PL	485
br17	max LB min PL max SB max AO	44505
eil22	max LB min PL max SB max AO	41199
gr17	max LB min PL max SB max AO	493
br17	max LB min PL max AO max SB	45355
eil22	max LB min PL max AO max SB	42747
gr17	max LB min PL max AO max SB	535
br17	max LB max AO max SB min PL	26821
eil22	max LB max AO max SB min PL	43383
gr17	max LB max AO max SB min PL	531
br17	max LB max AO min PL max SB	26847
eil22	max LB max AO min PL max SB	43391
gr17	max LB max AO min PL max SB	531

Table 3.1.: Computational evaluation of branching rules: The notation defines the lexicographic order of the criteria by that arcs are selected as branching candidates. The last column denotes the number of branching nodes needed to prove optimality for an already optimal incumbent solution.

Based on these results we declare the rule (i.e., the last rule denoted in Table 3.1)

$$\max \text{LB } \max \text{AO } \min \text{PL } \max \text{SB}$$

as (our) clear winner by considering that computing $\text{LB}(a)$ is much faster ($\mathcal{O}(|V|)$) than computing $\text{SB}(a)$ ($\mathcal{O}(|V|^2)$) with warm start and $\mathcal{O}(|V|^3)$ without).

3.6.2. Assignment Reduction

By deleting the ResourceConstraint from model (RCAP_{CIP}) the assignment relaxation RCAP′ derives. It is a valid relaxation, which we use for bounding within the B&B algorithm. The assignment problems are solved with an $\mathcal{O}(|V|^3)$ implementation of the (dual) Hungarian method described in the paper Kuhn (1955, [70]) that has celebrated its 60th birthday past year.

The Hungarian algorithm produces optimal dual variables π_u^t and π_v^h for each arc $a = (u, v) \in A$. Let z_{LB} be the optimal objective value of RCAP′ and z_{UB} an already known upper bound for the RCAP. Then an arc $a \in A$ can be detached if $z_{\text{LB}} + c_a - \pi_u^t - \pi_v^h \geq z_{\text{UB}}$, a rule which is known under the name *reduced cost presolving*, see Achterberg (2009, [12]).

Let $M = \{a \in A \mid x_a = 1\}$ be the solution of some assignment relaxation. It is easy to see that arcs can be detached by imposing $x_a = 0$ for an arc $a \in M$ and $x_a = 1$ for an arc $a \in A \setminus M$ if the corresponding sub-problems turn out to be infeasible or

dominated by the best known upper bound. However, solving all these sub-problems can be computationally expensive. This computational burden can be mitigated by performing a local optimization before solving the sub-problems. Namely, if we try to detach $a = (u, v) \in A$ from the current sub-problem, we can locally optimize in $O(|V|)$ over all 2-opt moves defined by $A(v)^{\text{in}} \setminus \{a\}$. If the best objective value during this local optimization is below the best known upper bound we do not have to solve the assignment problem that forces $x_a = 0$ (this can be done similarly for $a \in A \setminus M$).

3.6.3. Shortest-Path Reduction

In this section we develop a pruning rule that eliminates an arc $a \in A$ if it can be proven that no feasible path $P \subseteq A$ with $a \in P$ exists in the current sub-problem. To this end, we transform the directed graph $D = (V, A)$ into another directed graph D_{SP}. We introduce the node set $V_{\text{SP}} := V \cup \{s, t\}$ of D_{SP}, i.e., we extend D by a source s and a target t. For $a = (u, v) \in A$ we apply the following transformation:

$$A_{\text{SP}}(a) := \begin{cases} \{(u,t),(s,v)\}, \text{ if } a \text{ is a replenishment arc} & \left(\begin{array}{l} c^{\text{SP}}_{(u,t)} := r^1_a, \\ c^{\text{SP}}_{(s,v)} := r^2_a \end{array} \right), \\ \{(u,v)\}, \text{ otherwise} & \left(c^{\text{SP}}_{(u,v)} := r^1_a + r^2_a \right). \end{cases}$$

The transformed graph is $D_{\text{SP}} := (V_{\text{SP}}, A_{\text{SP}}) := (V \cup \{s,t\}, \bigcup_{a \in A} A_{\text{SP}}(a))$ with well defined objective coefficients c^{SP}_a for all $a \in A_{\text{SP}}$. Every feasible path must be elementary in a solution to the RCAP and every elementary resource path of $\mathbb{P}(A)$ corresponds to an elementary s-t-path P in D_{SP} by construction. Our elimination criterion for an arc $a \in A$ is as follows. If we can prove that a *shortest* elementary s-t-path P in D_{SP} *such that* $a \in P$ has cost $c(P) > B$ we are allowed to detach a. This elimination criterion is $\mathcal{NP}$-hard to compute, as stated in Lemma 8:

Lemma 8 (Elementary s-v-t-paths in directed graphs are $\mathcal{NP}$-hard to compute)**.** *Given a directed graph $D = (V, A)$ and three different nodes $s, v, t \in V$, it is $\mathcal{NP}$-complete to decide if D contains an elementary path that starts at s, traverses v, and ends at t.*

Proof. Given a directed graph $D = (V, A)$ with different nodes $v_1, u_1, v_2, u_2 \in V$ the disjoint path problem (DPP) is to find a v_1-u_1-path and a v_2-u_2-path in D such that the two paths are vertex-disjoint. The DPP is $\mathcal{NP}$-complete[11], see Fortune, Hopcroft, and Wyllie (1980, [50]). An instance of the DPP can be instantiated as an elementary s-v-t-path problem by setting $s = v_1$, $t = u_2$ and by introducing arcs (u_1, v) and (v, v_2). □

Fortunately, we can relax the criterion by computing non-elementary paths in D_{SP} and also obtain a valid elimination rule. It can be checked by first computing the shortest-paths from s to all nodes of V, followed by computing the shortest-paths from V

[11] The DPP for *undirected graphs* is solvable in polynomial time, see Shiloach (1978, [97]).

to t, and finished by iterating over all arcs of A and to evaluate the elimination criterion. This procedure has complexity $\mathcal{O}(|V|^2)$.

3.6.4. Bin-Packing Reduction

Let J be a set of items with associated weights $w_j \in \mathbb{Q}_+$ for $j \in J$ and a bin capacity $B \in \mathbb{Q}_+$. The standard bin-packing problem is to find a block partition $S_1, \dots, S_k$ of J with $\sum_{j \in S_k} w_j \leq B$ for all blocks $S_1, \dots, S_k$ such that k is minimal. In a solution of the RCAP the nodes are also assigned to capacitated bins, namely, to resource paths. This motivates a bin-packing relaxation of the RCAP that can be used for pruning in the B&B tree. To this purpose, we interpret the nodes of the graph as items and the feasible paths as bins. The pruning rule contributes if it can be proven that more bins are needed than available. A valid lower bound on the minimal resource consumption that the node (or item) $u \in V$ will contribute to a feasible path can be computed by solving the following assignment problem:

$$\begin{aligned} w_u := \min \quad & \sum_{a \in A(u)^{\text{in}}} r_a x_a \\ \text{s.t.} \quad & \sum_{a \in A(v)^{\text{in}}} x_a = 1 \text{ and } \sum_{a \in A(v)^{\text{out}}} x_a = 1 \ \ \forall\, v \in V, \quad x_a \geq 0 \ \ \forall\, a \in A. \end{aligned} \tag{RCAP$_{\text{ITEMS}}$}$$

These quantities are used as node weights. Moreover an obviously valid upper bound for the maximal number of feasible paths (or bins) can be computed by solving the following model (RCAP$_{\text{BINS}}$):

$$\begin{aligned} z_{\text{UB}} := \max \quad & \sum_{a \in \tilde{A}} x_a \\ \text{s.t.} \quad & \sum_{a \in A(v)^{\text{in}}} x_a = 1 \text{ and } \sum_{a \in A(v)^{\text{out}}} x_a = 1 \ \ \forall\, v \in V, \quad x_a \geq 0 \ \ \forall\, a \in A. \end{aligned} \tag{RCAP$_{\text{BINS}}$}$$

It maximizes the number of replenishment arcs $\tilde{A} \subseteq A$, which is equivalent to maximizing the number of resource paths. The following lemma summarizes the bin-packing pruning rule.

Lemma 9. *Let I be an instance of the RCAP. Let z_{LB} be any valid lower bound for the optimal solution of the bin-packing problem with item set V, weights w_u derived from model (RCAP$_{\text{ITEMS}}$) for all $u \in V$ and a bin capacity of B. Further let z_{UB} be the optimal objective value of model (RCAP$_{\text{BINS}}$). If $z_{LB} > z_{UB}$ it is proven that I is infeasible.*

Proof. Let $z_{LB} > z_{UB}$ and let I be a feasible instance. There must be a cycle partition $C_1, \dots, C_k$ containing feasible paths. The value z_{UB} is associated with an optimal solution of (RCAP$_{\text{BINS}}$), therefore $z_{UB} \geq |\mathbb{P}(\bigcup_{i=1}^{k} C_k)|$. Each path in $\mathbb{P}(\bigcup_{i=1}^{k} C_k)$ provides a feasible assignment of items to bins, i.e., an assignment of nodes to feasible paths, because the weight of each item $v \in V(P)$ is underestimated in a worst case by the optimal objective value w_v of model (RCAP$_{\text{ITEMS}}$), thus $z_{LB} \leq |\mathbb{P}(\bigcup_{i=1}^{k} C_k)|$. The contradiction is given by $z_{LB} \leq |\mathbb{P}(\bigcup_{i=1}^{k} C_k)|$ and $z_{UB} \geq |\mathbb{P}(\bigcup_{i=1}^{k} C_k)|$. □

Since the bin-packing problem is $\mathcal{NP}$-hard, we replace z_{LB} by the lower bounds $L2$ and $L3$ developed by Martello and Toth (1990, [78]). These bounds can be computed in $\mathcal{O}(|V|)$ for $L2$ and in $\mathcal{O}\left(|V|^3\right)$ for $L3$ and have a worst case quality of $\frac{2}{3}z_{\mathrm{BP}}$ and $\frac{3}{4}z_{\mathrm{BP}}$ where z_{BP} denotes the optimal objective value of the bin-packing problem.

3.6.5. Symmetry Reduction

In this section we collect some algorithmic insights found by solving symmetric TSP and CVRP instances with our algorithm. This type of problems can be characterized as having the property that *each resource path is a cycle*, and that the cost function is symmetric. Therefore, every cycle can be reversed, such that the cost and the resource consumption of the tour and the reversed tour are equal. This can be problematic in a B&B algorithm that has to search through many essentially identical alternatives.

The CVRP that has been mentioned for the first time in the paper Clarke and Wright (1964, [37]) is to find a minimal set of cycles, called *tours*, in a complete undirected graph $G = (V \cup \{d\}, E)$ with node demands $r_v \in \mathbb{Q}_+$ for all $v \in V$ such that each node of V is covered exactly once by a cycle, every cycle covers the depot node d exactly once, $\sum_{v \in V \cap C} r_v \leq B$ holds for every cycle C of the solution, and the solution minimizes some linear objective function $c : E \mapsto \mathbb{Q}$. We assume that the minimal number of tours t is known (as most of the articles of the CVRP literature do). An instance of the CVRP can be modeled as a RCAP by introducing t copies of d, using the resource function values of the outgoing arcs of a node to model the demands, and declaring the incoming arcs of d as replenishment arcs. For $t = 1$, TSP instances can be modeled directly as RCAPs. Our first observation is:

Lemma 10. *Consider a RCAP instance over the directed graph $D = (V, A)$ such that each resource path is a cycle and let $f : V \mapsto \{1, \ldots, |V|\}$ be some numbering of the nodes. We only have to consider arcs $a = (u, v) \in A$ with $f(u) < f(v)$ as branching candidates.*

Proof. Consider the set of cycles $C_1, \ldots, C_k$ of an infeasible solution of the current node relaxation. Then, an infeasible path $P \in \mathbb{P}(\bigcup_{i=1}^{n} C_i)$ exists. Since P is a cycle there is at least one arc $a = (u, v) \in P$ with $f(u) < f(v)$ that can be used as a branching candidate. □

Note that, although this attractive rule was originally developed to break symmetries, it can also be used in an ATSP context. However, we could not find an effective way to utilize it in our implementation. The concrete reason is unclear to us. We can only speculate that merely using arcs $a = (u, v)$ with $f(u) < f(v)$ as branching candidates destroys the performance of the B&B algorithm because in approximately half of the cases the one arc that increases the lower bound at most is not chosen. Nevertheless, we were able to verify by Lemma 10 that our implementation does not suffer from symmetric cost matrices.

Another symmetry issue refers to the depot copies in the CVRP case. We assume that (V, A) does not contain loops and arcs connecting depot nodes. Then, each cycle partition of (V, A) is symmetric to $t!$ cycle partitions that arise from interchanging the

depot nodes. This problem can be easily resolved by excluding all arcs incident to a depot node as branching candidates. If all other arcs, i.e., all arcs that are not incident to the depot, are fixed to one or zero we always obtain single-customer tours for which each $x_a = 0$ leads to an infeasible RCAP instance.

3.7. Computational Results

In this section we present computational results of the algorithms presented in this chapter. To this end, we investigate two problem classes, namely RCAP instances derived from our railway application as well as classical vehicle routing problems.

The algorithmic setup that we use here is arranged as follows. We derive two slightly different concrete algorithmic procedures from the local, regional, and global search algorithms of this chapter. Each of two procedures is dedicated to a problem class. The reason for this slightly heterogeneous setup is that we expect computational drawbacks on the solution quality or computation time for at least one problem class if we use a completely unified algorithm[12].

In the first procedure we slim down a few algorithmic ingredients for the railway application of the RCAP. In the second procedure, which is dedicated to vehicle routing instances, we arrange all components that we presented during the chapter in the following way:

Procedure I: Given an instance of the RCAP with the graph (V, A) we run the B&B algorithm of Section 3.6 in combination with a single run of an RS from Section 3.5 in the root node of the branching tree. Thereby, the following custom settings are made:

- ▷ We only use the move-based RS that we start exactly once with the initial solution $M_0 \subseteq A$:
$$M_0 := \{(v, v) \in A \,|\, v \in V\}$$
which is a feasible solution for all feasible instances that we consider.
- ▷ The bin-packing reduction from Section 3.6.4 is not used.

The procedure has a time limit of 16 hours.

Procedure II: For an RCAP instance with the graph (V, A) we also run the B&B algorithm of Section 3.6. But here we sequentially call both RS algorithms from Section 3.5 only in the root node of the branching tree for exactly one time. To this end, we construct a feasible starting solution as follows

- ▷ For (A)TSP instances we just take the order of the nodes in that they are arranged in the input data.

[12] It is well known that computational results can be cheated by implementing rules that are dedicated to specific instances of a test set. We do not see our two procedures as a cheating approach. Instead, we like to emphasize that our philosophy is exactly to create tailor-made algorithms for real-world problems.

▷ For (A)CVRP instances we run a poor variant of the savings heuristic, see [37].

Let $M_0 \subseteq A$ be the starting solution for the RCAP instance. We first run the moved-based RS with the starting solution M_0 and take its solution as starting solution for the branching-based RS. The motivation behind this setup is to gain the fast computation time from the move-based RS and the high solution quality of the branching-based RS at the same time. The global search procedures is either ran until optimality or manually stopped when we loose hope to completely close the gap between the lower and upper bound.

All procedures for the starting solutions of the RS algorithms are designed to be feasible and fast but of (very) poor solution quality. A starting solution with high quality can clearly lead to a good regionally optimal solution but without having anything achieved by the RS algorithms themselves. We hope that our computations are not affected by this effect.

All our computations presented in this section were performed on computers with an Intel(R) Xeon(R) CPU X5672 with 3.20 GHz, 12 MB cache, and 128 GB of RAM by using a single thread under the operating system Ubuntu 14.04. All implementations are written in the `C++` programming language [98] and compiled by the compiler `g++` 4.8.4 released by the Free Software Foundation.

3.7.1. Rolling Stock Rotation Instances

The computations that we present in this section are published in [1] (the former paper [2] includes a similar study that is restricted to the move-based RS).

The interpretation of the RCAP instances that we investigate from rolling stock rotation optimization is to cover a given set of timetabled passenger trips by a set of cycles, called rolling stock rotations. The resource constraint models a limit on the driven distance between two consecutive maintenance services. The main objective is to minimize the number of vehicles and the total distance of deadhead trips (needed to overcome different arrival and departure locations between two trips).

The idea behind the starting solution (as we explained it above) is to cover each timetabled trip by an own railway vehicle. This leads to $|V|$ initial rolling stock rotations, which are feasible because there is a time window of almost a whole week to perform a maintenance service (this can be expensive but is feasible).

We tested the regional and global search algorithms according to procedure I for 15 RCAP instances derived from industrial RSRP instances. Note that, by using the RS strictly as described in procedure II we get similar results for these 15 instances w.r.t. solution quality and computation time for less constrained instances (e.g., all with 8000 km and 6000 km and all with 97 nodes). But the computation times for the large and hard constrained instances (e.g., `RCAP_02`, see below) increase compared to the pure move-based RS.

Table 3.2 reports the results for the 15 instances that arise from RSRPs that are associated with three timetables (indicated by the number of nodes in column three) for different upper bounds of a dedicated maintenance constraint denoted in column two.

instance	B [km]	$\lvert V\rvert$	root gap	nodes	hh:mm:ss	proved
RCAP_01	1000	617	-	1	00:00:00	infeasibility
RCAP_02	2000	617	25.88	15105	15:50:56	9.81 % primal gap
RCAP_03	4000	617	3.95	51483	15:45:57	0.21 % primal gap
RCAP_04	6000	617	0.19	143	03:18:07	optimality
RCAP_05	8000	617	0.13	43	00:07:37	optimality
RCAP_06	1000	97	-	1	00:00:00	infeasibility
RCAP_07	2000	97	12.71	41	00:00:10	optimality
RCAP_08	4000	97	0.00	1	00:00:02	optimality
RCAP_09	6000	97	0.00	1	00:00:02	optimality
RCAP_10	8000	97	0.00	1	00:00:02	optimality
RCAP_11	1000	310	-	1	00:00:00	infeasibility
RCAP_12	2000	310	38.16	944551	16:07:46	16.71 % primal gap
RCAP_13	4000	310	16.70	119159	09:17:54	optimality
RCAP_14	6000	310	7.78	2053	00:08:33	optimality
RCAP_15	8000	310	7.78	87	00:21:40	optimality

Table 3.2.: Results for RCAP instances from the railway application.

The *root gap* in column four is defined as

$$\frac{(\mathrm{UB} - \mathrm{LB})}{\mathrm{UB}} \cdot 100, \qquad \text{(root gap)}$$

i.e., the worst case optimality gap in percent, where $\mathrm{UB} > 0$ is the objective value of the regionally optimal solution and LB the value of the lower bound obtained in the root node of the branching tree. Columns five, six, and seven contain the number of nodes of B&B tree, the computation time, and the solution status on termination of the B&B algorithm, see page 102 for the definition of the primal gap.

In the industrial application the instances associated with a maintenance constraint of 8000 km (i.e., the "easiest"!) are the ones of interest that could all be solved to proven optimality fast. Also tighter constrained instances are solved with very high solution quality. The most difficult instances RCAP_02, RCAP_03, and RCAP_12 display worst case optimality gaps. Nevertheless, we claim that they are also completely "resolved" from an applied point of view. In fact, the very large lower bound (which was mainly obtained by the reduction procedures and by branching) proves practical inefficiency of those solutions beyond doubt.

3.7.2. TSP, ATSP, CVRP, and ACVRP Instances

We also made experiments for a large number of instances taken from the literature, see Ralphs (2014, [87]) and Reinelt (1990, [89]). To this end, we use the regional search algorithm and the B&B algorithm as described by procedure II.

type	#instances	arithmetic mean		geometric mean	
ATSP	19	1.99	[15](1.70)	1.51	(1.21)
CVRP	106	0.89	(5.09)	0.63	(3.81)
ACVRP	8	1.84		1.34	
TSP	65	2.22	(2.60)	1.71	(1.97)
all	198	1.47	(3.91)	**1.04**	(2.78)

Table 3.3.: Summary of regional search for VRP Instances. For the geometric mean a shift of 1 has been chosen, see [12]. All numbers indicate means over primal gap values.

We present results for all ATSP instances from Reinelt (1990, [89]) and for the TSP instances with less than 500 nodes. From [87] we consider all CVRP and asymmetric capacitated vehicle routing problem (ACVRP)[13] instances from the test sets A, B, E, F, G, M, P, and V except for six instances for which we could not verify the objective values of the solutions provided in the library[14].

Table 3.3 provides mean values for the column "bk gap" (i.e., the deviation in percent to the objective value of a best known solution as published in the literature) of Table 3.4 that can be found at the very end of this chapter and which shows more details of the computational study for the 198 instances.

A similar summary as in Table 3.3 is made in our paper [2] where we restricted to the pure move-based RS. We provide the corresponding values of [2] in braces such that the gain of solution quality can be seen if we call the branching-based RS on top of the move-based RS.

Finally, we illustrate the solution quality (i.e., the primal gap) that is achieved by the RS implementations in terms of the quality of the bound from the assignment relaxation, i.e., the dual gap. To this end, Figure 3.7, which is dedicated to the move-based RS, and Figure 3.8, which is dedicated to the branching-based (on top of the move-based) RS, were created.

In these figures, each black dot corresponds to an RS for one of the 198 VRP instances from the literature where the x-coordinate is associated with the *primal gap* and the y-coordinate with the *dual gap*. More precisely, let LB be the objective value obtained from the assignment relaxation and let UB be the objective value associated with the regional optimal solution. We compute the primal gap as

$$\frac{\text{UB} - \text{OPT}}{\text{OPT}} \cdot 100 \qquad \text{(primal gap)}$$

[13]The ACVRP instances were not considered in our first paper [2].

[14]This verification is needed. Otherwise incomparable results would possibly appear.

[15]The value 1.7 is inconsistent, i.e., it should be 1.99 or more because the branching-based RS is started on top of the move-based RS. We have figured out the reason in our implementation and identified the issue to be insignificant for our computational study.

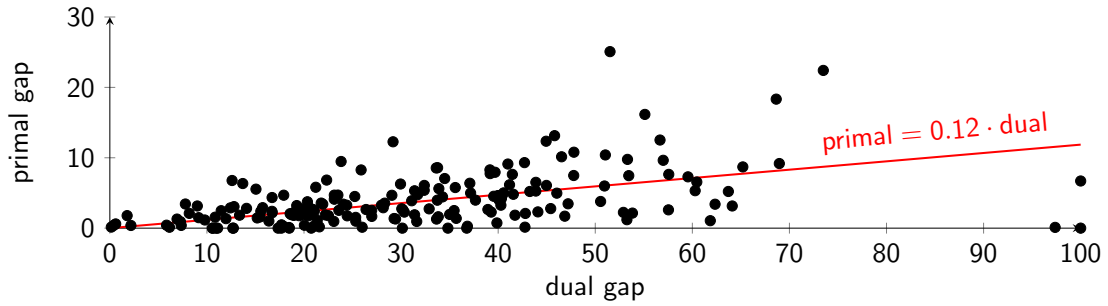


Figure 3.7.: Linear regression through $(0, 0)$ for the primal-dual gaps of the move-based RS.

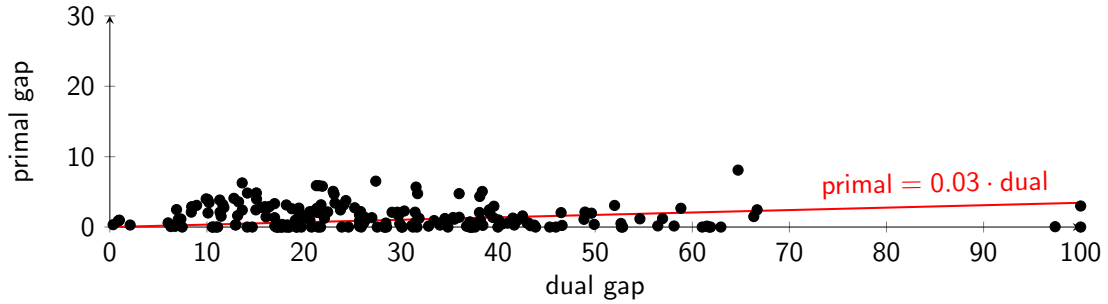


Figure 3.8.: Linear regression through $(0, 0)$ for the primal-dual gaps of the branching-based RS.

and the dual gap by

$$\frac{\text{OPT} - \text{LB}}{\text{LB}} \cdot 100 \qquad \text{(dual gap)}$$

where OPT denotes the optimal objective value of the corresponding instance. Note that OPT is typically not known in advance. In Figures 3.7 and 3.8 we assume that the best known objective values provided by the libraries from the literature are also the optimal ones (this is not a strong assumption, i.e., most of all are already proven to be optimal).

From our experience we speculate that the computational hardness of an instance is somehow related to its dual gap, i.e., the larger it is the harder the instance (while exceptions also have been seen). First of all, we have to note that the dots are not uniformly distributed w.r.t. the dual gap. This is just a given fact of the test-set and we accept it as it is. Moreover, we observe that our test-set contains rather easy and also (very) hard instances. It is now natural to ask for a relation between the primal and dual gap. This has been accomplished by a linear regression through the point sets. In this regression the additional constraint that the regression has to pass the origin of the coordinate system has been imposed because we find this obvious for RS algorithms.

From these regressions we allow to draw the following conclusions from the figures. Both plots give rise to suspect a relation between the dual gap (i.e., hardness of the

problems) and primal gap, i.e., solution quality of the algorithms. This is more obvious in the figure for the move-based RS while the branching-based RS seems to be able to compensate the hardness of some particular instances more efficiently. Note that this are heuristic conclusions, i.e., we do not even know if a relation between the primal and dual gap exists and if so, if it is indeed linear. We also strongly conjecture that the dual gap is only an indication of hardness and that there are other circumstances that may also not be known in advance.

Nevertheless, by assuming meaningfulness of the regressive relations of Figures 3.7 and 3.8 we exactly obtain what we expect from an RS, i.e., a relationship between hardness and solution quality of a single RS run, i.e., its self-calibrating computational effort.

3.8. Conclusion

In Chapter 3 we investigated the resource-constrained assignment problem and derived regional search as an algorithmic concept. We showed how to apply this heuristic idea in various ways to instances from the railway application as well as to instances from the VRP literature.

In comparison to other more problem specific heuristics (especially for the symmetric TSP, see Helsgaun (2009, [64])) our regional search is almost competitive w.r.t. solution quality. It is definitely competitive in solving asymmetric instances to proven optimality, as reported in the last three columns of Table 3.4: 18 of 19 ATSP instances from [89] and all ACVRP instances considered in [49] are solved to proven optimality! These results give evidence that our algorithms are powerful tools for a wide variety of resource-constrained assignment problems ranging from recent railway applications via VRPs to classical TSPs and ATSPs.

In terms of vehicle routing modern metaheuristics turn out be the state of the art heuristic procedures at the moment, see [101, Page 110]:

> "Even though we do not yet know the optimal solutions for most of the instances used in the computational comparison, it is safe to say that current metaheuristics are capable of producing high quality solutions for instances with up to 500 customers, and solution quality has improved steadily over the last decade. However, the gains in solution quality are new becoming marginal, an indication that the current solution methodologies my have reached a plateau."

Those methods, which are hybrids of many different search technologies, provide a superior solution quality compared to our RS algorithms in terms of the VRP instances that we investigated, see Vidal (2013, [105]). But, we do not at all see this as a downside of our ideas because the RS algorithms terminate by definition without setting a time limit, they are much easier to implement (we know that this is not a scientific argument), and, primary, can be even used to improve modern metaheuristics by hybridization, which is a trend that will never stop as we believe.

Coming back to the rolling stock rotation problem (RSRP): Clearly, the RSRP, from which the RCAP is derived, consists of many more aspects as vehicle composition, regularity, and infrastructure capacity. Ideas for the direct integration of the result of this chapter into RotOR's algorithm for industrial RSRP instances exist but have neither been tested nor implemented. In this way, we look towards promising further work.

At the beginning of our cooperation with DB Fernverkehr AG (DBF) we expected that the maintenance constraints of the RSRP are the ones that the most increase the complexity. Indeed, this is what we believe to disprove by this chapter: We could solve even instances with maintenance constraints that we believed to be very hard to proven optimality, see Table 3.2. And this, with a model (i.e., the constraint integer program (RCAP_{CIP})) that can not be much weaker in terms of its dual bound! On the one hand, this may partially be a result of the high-quality solutions provided by the regional search algorithm. On the other hand, increasing the lower bound is much harder than decreasing the primal bound, which is well known among the optimization community. In this way, we additionally gain a direct outcome for our industrial application: An even weak model can be sufficient in order to tackle the maintenance constraints of the RSRP. This is exactly what RotOR does in its proper industrial application at DBF, see Section 5.2.

Table 3.4.: Regional and global search for VRP instances: The third column gives the number of nodes of the considered RCAP instance; the fourth column is the deviation in percentage of the initial solution for our regional search (computed with a poor greedy heuristic) w.r.t. the best known objective value [56] (column five). The columns "bk gap" and "lb gap" give the deviation in percent of the regionally optimal objective value (column "RS [s]" denotes its computation seconds) w.r.t. column "best" and w.r.t. the lower bound obtained in the root node of the branching tree, respectively. The last two columns give the number of branching nodes and the computation time if the B&B approach is able to solve all remaining sub-problems, i.e., if we are able to find an optimal solution by proof.[16]

instance	type	$\|V\|$	root gap	best	lb gap	bk gap	RS [s]	GS nodes	dd:hh:mm:ss
A034-02f	ACVRP	35	48.91	1406	14.15	0.00	7.4	18093	00:00:00:18
A036-03f	ACVRP	38	46.43	1644	12.78	4.08	5.4	37037	00:00:00:38
A039-03f	ACVRP	41	55.16	1654	9.92	4.00	10.9	11043	00:00:00:25
A045-03f	ACVRP	47	58.19	1740	6.72	0.11	5.9	2025	00:00:00:10
A048-03f	ACVRP	50	63.05	1891	8.39	2.12	4.4	11865	00:00:00:19
A056-03f	ACVRP	58	65.61	1739	13.64	2.41	15.1	1192799	00:00:30:17
A065-03f	ACVRP	67	69.61	1974	7.45	0.00	32.7	83185	00:00:02:23
A071-03f	ACVRP	73	71.24	2054	10.11	2.00	10.1	121205	00:00:05:27
br17	ATSP	17	76.65	39	100.00	0.00	2.6	152825	00:00:00:23
ft53	ATSP	53	50.52	6905	16.97	3.33	23.1	441917	00:00:15:39
ft70	ATSP	70	31.04	38673	2.09	0.29	17.2	1462829	00:00:58:00
ftv170	ATSP	171	61.45	2755	6.87	2.48	37.2	6683339	01:03:08:19
ftv33	ATSP	34	42.56	1286	13.63	6.27	5.3	157	00:00:00:05
ftv35	ATSP	36	40.44	1473	7.32	1.14	4.0	1353	00:00:00:04
ftv38	ATSP	39	38.90	1530	7.05	1.10	4.5	5407	00:00:00:12
ftv44	ATSP	45	39.77	1613	8.43	2.89	4.7	2323	00:00:00:09
ftv47	ATSP	48	58.59	1776	10.22	3.48	4.6	26341	00:00:01:09
ftv55	ATSP	56	59.54	1608	15.09	4.85	4.5	209665	00:00:08:42
ftv64	ATSP	65	61.55	1839	10.08	3.92	12.3	46923	00:00:03:02
ftv70	ATSP	71	59.84	1950	11.35	2.11	5.9	452675	00:00:16:56
kro124p	ATSP	100	82.71	36230	6.28	0.07	166.6	14253731	01:23:08:01
p43	ATSP	43	8.77	5620	97.37	0.05	6.5		
rbg323	ATSP	323	79.37	1326	0.90	0.90	112.0	739	00:00:08:02
rbg358	ATSP	358	83.58	1163	0.34	0.34	113.2	663	00:00:02:46
rbg403	ATSP	403	69.02	2465	0.88	0.88	94.6	177	00:00:06:39
rbg443	ATSP	443	68.80	2720	0.98	0.98	121.3	43	00:00:06:32
ry48p	ATSP	48	73.42	14422	15.64	2.80	10.2	150917	00:00:05:24
A-n32-k5	CVRP	36	52.94	784	31.63	0.00	22.0		
A-n33-k5	CVRP	37	49.70	661	38.07	2.07	62.1		
A-n33-k6	CVRP	38	42.92	742	36.74	0.13	84.4		
A-n34-k5	CVRP	38	52.73	778	35.99	1.39	67.5		
A-n36-k5	CVRP	40	50.00	799	38.17	0.99	68.3		
A-n37-k5	CVRP	41	56.61	669	26.99	1.33	22.7		
A-n37-k6	CVRP	42	42.03	949	45.31	0.00	321.0		
A-n38-k5	CVRP	42	54.74	730	43.72	0.27	69.3		
A-n39-k5	CVRP	43	59.86	822	37.08	0.72	206.6		

Continued on next page

Table 3.4 – continued from previous page

instance	type	$\|V\|$	root gap	best	lb gap	bk gap	RS [s]	GS nodes	dd:hh:mm:ss
A-n39-k6	CVRP	44	55.06	831	38.42	0.24	112.0		
A-n44-k6	CVRP	49	56.03	937	31.10	0.21	75.2		
A-n45-k6	CVRP	50	55.00	944	37.18	0.00	118.7		
A-n45-k7	CVRP	51	51.89	1146	40.40	0.43	1025.8		
A-n46-k7	CVRP	52	58.68	914	37.31	0.00	187.9		
A-n48-k7	CVRP	54	52.35	1073	39.09	2.45	549.6		
A-n53-k7	CVRP	59	59.26	1010	39.45	1.37	677.7		
A-n54-k7	CVRP	60	54.68	1167	51.99	3.07	1700.3		
A-n55-k9	CVRP	63	54.78	1073	40.04	1.01	491.2		
A-n60-k9	CVRP	68	54.23	1354	54.60	1.17	4232.7		
A-n61-k9	CVRP	69	55.98	1034	41.66	0.29	1080.4		
A-n62-k8	CVRP	69	59.67	1288	48.94	2.13	3293.0		
A-n63-k10	CVRP	72	54.07	1314	49.89	0.38	3884.1		
A-n63-k9	CVRP	71	54.22	1616	48.84	1.10	4342.6		
A-n64-k9	CVRP	72	57.66	1401	41.51	1.27	3110.8		
A-n65-k9	CVRP	73	59.86	1174	37.05	0.00	1275.8		
A-n69-k9	CVRP	77	62.16	1159	37.10	0.69	1497.4		
A-n80-k10	CVRP	89	58.80	1763	41.97	0.96	6728.1		
att-n48-k4	CVRP	51	63.86	40002	26.12	0.52	83.3		
bayg-n29-k4	CVRP	32	55.70	2050	17.71	0.00	12.2	34154469	00:06:08:31
bays-n29-k5	CVRP	33	46.72	2963	25.89	0.00	29.2		
B-n31-k5	CVRP	35	29.56	672	30.06	0.00	41.0		
B-n34-k5	CVRP	38	44.35	788	32.83	0.13	76.2		
B-n35-k5	CVRP	39	53.30	955	37.28	0.00	130.4		
B-n38-k6	CVRP	43	57.34	805	43.85	0.00	174.0		
B-n39-k5	CVRP	43	63.20	549	52.82	0.00	61.3		
B-n41-k6	CVRP	46	54.90	829	61.88	0.00	176.5		
B-n43-k6	CVRP	48	58.38	742	52.70	0.00	425.0		
B-n44-k7	CVRP	50	50.81	909	61.72	0.00	336.1		
B-n45-k5	CVRP	49	53.06	751	45.94	0.00	184.2		
B-n45-k6	CVRP	50	56.23	678	43.11	0.59	349.7		
B-n50-k7	CVRP	56	67.11	741	34.82	0.00	314.5		
B-n50-k8	CVRP	57	50.13	1312	56.93	1.20	2457.7		
B-n51-k7	CVRP	57	53.78	1032	36.88	0.10	566.7		
B-n52-k7	CVRP	58	66.00	747	61.50	0.13	846.0		
B-n56-k7	CVRP	62	66.41	707	62.94	0.00	673.5		
B-n57-k7	CVRP	63	29.65	1153	66.67	2.45	1912.7		
B-n57-k9	CVRP	65	43.09	1598	34.31	0.68	1695.2		
B-n63-k10	CVRP	72	60.39	1496	58.82	2.67	3284.2		
B-n64-k9	CVRP	72	66.83	861	46.58	0.23	2814.0		
B-n66-k9	CVRP	74	52.97	1316	58.12	0.15	3445.1		
B-n67-k10	CVRP	76	65.36	1032	43.33	0.19	3108.3		
B-n68-k9	CVRP	76	60.09	1272	56.44	0.16	2461.7		
B-n78-k10	CVRP	87	62.37	1221	61.02	0.00	8131.7		
dantzig-n42-k4	CVRP	45	34.67	1142	49.61	1.97	76.3		
E-n101-k14	CVRP	114	64.54	1071	29.07	2.10	8541.6		
E-n101-k8	CVRP	108	65.83	817	20.61	0.97	2077.9		
E-n13-k4	CVRP	16	38.10	247	10.93	0.00	2.7	143	00:00:00:04
E-n22-k4	CVRP	25	38.73	375	30.13	0.00	4.5	74055	00:00:00:45

Continued on next page

Table 3.4 – continued from previous page

instance	type	$\|V\|$	root gap	best	lb gap	bk gap	RS [s]	GS nodes	dd:hh:mm:ss
E-n23-k3	CVRP	25	50.48	569	21.44	0.00	4.9	9321	00:00:00:09
E-n30-k3	CVRP	32	52.28	534	40.97	0.56	70.3		
E-n31-k7	CVRP	37	66.93	379	19.26	0.00	11.3	155737	00:00:02:24
E-n33-k4	CVRP	36	34.61	835	28.50	0.00	119.8		
E-n51-k5	CVRP	55	62.00	521	21.75	3.16	55.0		
E-n76-k10	CVRP	85	63.16	830	29.86	0.48	1899.9		
E-n76-k14	CVRP	89	47.43	1021	35.36	1.35	3552.1		
E-n76-k7	CVRP	82	69.68	682	23.75	2.43	201.5		
E-n76-k8	CVRP	83	61.98	735	26.28	0.94	290.0		
F-n135-k7	CVRP	141	71.80	1162	52.65	0.51	6891.6		
F-n45-k4	CVRP	48	65.61	724	42.99	0.55	32.9		
F-n72-k4	CVRP	75	74.10	237	31.22	0.00	151.2		
fri-n26-k3	CVRP	28	23.56	1353	17.75	0.37	6.1	842175	00:00:06:11
gr-n17-k3	CVRP	19	29.88	2685	28.31	0.00	5.6	13977	00:00:00:09
gr-n21-k3	CVRP	23	36.02	3704	27.54	0.00	6.3	29293	00:00:00:17
gr-n24-k4	CVRP	27	46.04	2053	28.30	0.00	11.7	5919153	00:00:47:50
gr-n48-k3	CVRP	50	66.55	5985	25.71	0.22	28.3		
hk-n48-k4	CVRP	51	56.96	14749	25.74	0.09	361.6		
M-n101-k10	CVRP	110	66.26	820	33.98	0.49	657.2		
M-n121-k7	CVRP	127	67.19	1034	64.71	8.09	37860.9		
M-n151-k12	CVRP	162	67.60	1053	34.00	0.28	12133.0		
M-n200-k17	CVRP	215	66.28	1373	-	-	-		
P-n101-k4	CVRP	104	71.61	681	15.04	2.44	219.1		
P-n16-k8	CVRP	23	1.75	450	14.67	0.00	4.1	3033	00:00:00:07
P-n19-k2	CVRP	20	37.09	212	21.70	0.00	4.2	16959	00:00:00:12
P-n20-k2	CVRP	21	43.31	216	19.46	2.26	3.1	10593	00:00:00:08
P-n21-k2	CVRP	22	42.03	211	18.48	0.00	3.7	4787	00:00:00:05
P-n22-k2	CVRP	23	45.04	216	17.13	0.00	5.6	6765	00:00:00:07
P-n22-k8	CVRP	29	20.66	603	39.97	0.00	19.6	818203	00:00:09:20
P-n23-k8	CVRP	30	19.73	529	37.62	0.00	26.7	9609861	00:02:09:07
P-n40-k5	CVRP	44	57.08	458	18.12	0.00	12.8		
P-n45-k5	CVRP	49	61.07	510	19.22	0.00	17.1		
P-n50-k10	CVRP	59	41.46	696	28.43	0.57	407.6		
P-n50-k7	CVRP	56	52.08	554	22.10	1.25	69.3		
P-n50-k8	CVRP	57	50.43	631	31.54	5.68	353.7		
P-n51-k10	CVRP	60	44.62	741	31.44	2.11	372.3		
P-n55-k10	CVRP	64	48.74	694	25.71	0.86	388.4		
P-n55-k15	CVRP	69	28.28	989	38.10	4.35	4983.0		
P-n55-k7	CVRP	61	61.18	568	21.38	2.07	146.9		
P-n55-k8	CVRP	62	62.93	588	19.83	1.18	105.0		
P-n60-k10	CVRP	69	52.55	744	29.61	2.11	656.2		
P-n60-k15	CVRP	74	42.59	968	31.49	0.72	1568.4		
P-n65-k10	CVRP	74	57.67	792	26.28	1.37	339.0		
P-n70-k10	CVRP	79	62.34	827	29.73	1.66	704.2		
P-n76-k4	CVRP	79	74.01	593	16.97	1.33	64.0		
P-n76-k5	CVRP	80	68.19	627	20.59	2.18	90.6		
swiss-n42-k5	CVRP	46	46.79	1668	31.85	1.24	30.9		
ulysses-n16-k3	CVRP	19	100.00	7965	18.75	2.60	6.1	5871	00:00:00:07
ulysses-n22-k4	CVRP	25	32.88	9179	34.51	1.21	21.4	60874403	00:07:31:41

Continued on next page

Table 3.4 – continued from previous page

instance	type	$\|V\|$	root gap	best	lb gap	bk gap	RS [s]	GS nodes	dd:hh:mm:ss
a280	TSP	280	8.16	2579	8.94	3.08	1063.7		
att48	TSP	48	78.68	10628	22.02	1.67	9.4	777323	00:00:21:35
bayg29	TSP	29	65.19	1610	10.56	0.00	6.1	2661	00:00:00:09
bays29	TSP	29	64.88	2020	12.93	0.30	4.3	2441	00:00:00:07
berlin52	TSP	52	66.03	7542	21.54	5.88	14.7	22145	00:00:01:50
bier127	TSP	127	69.98	118282	20.36	1.68	940.1		
brazil58	TSP	58	80.35	25395	35.50	1.12	23.2	741555	00:00:41:02
brg180	TSP	180	98.36	1950	100.00	2.99	256.9		
burma14	TSP	14	27.16	3323	17.33	0.00	3.3	191	00:00:00:05
ch130	TSP	130	87.22	6110	29.68	1.93	600.4		
ch150	TSP	150	87.64	6528	16.09	1.45	323.0		
d198	TSP	198	29.86	15780	33.40	0.92	1752.7		
d493	TSP	493	69.17	35002	15.97	2.89	11463.8		
dantzig42	TSP	42	0.00	699	23.89	0.00	8.6	224263	00:00:03:58
eil101	TSP	101	69.50	629	11.75	2.78	133.3		
eil51	TSP	51	67.43	426	13.16	1.62	16.3	765927	00:00:21:30
eil76	TSP	76	72.68	538	13.26	3.58	63.6	1929865	00:02:55:49
fl417	TSP	417	78.61	11861	37.68	0.40	24856.1		
fri26	TSP	26	17.81	937	11.10	0.00	4.6	553	00:00:00:06
gil262	TSP	262	90.96	2378	21.33	2.66	1593.5		
gr120	TSP	120	86.12	6942	18.18	3.14	107.4		
gr137	TSP	137	28.07	69853	19.10	0.96	211.8		
gr17	TSP	17	55.84	2085	20.77	0.00	5.6	843	00:00:00:03
gr202	TSP	202	30.94	40160	16.45	2.92	442.3		
gr21	TSP	21	59.11	2707	10.60	0.00	3.5	43	00:00:00:05
gr229	TSP	229	25.15	134602	19.48	1.54	3895.2		
gr24	TSP	24	62.98	1272	17.30	0.00	5.9	215	00:00:00:06
gr431	TSP	431	26.45	171414	21.27	5.88	31311.0		
gr48	TSP	48	74.56	5046	18.28	0.30	12.5	3900747	00:01:24:20
gr96	TSP	96	31.85	55209	16.99	0.15	290.8		
hk48	TSP	48	76.21	11461	16.13	2.61	9.6	141947	00:00:04:43
kroA100	TSP	100	88.88	21282	19.71	0.00	222.8		
kroA150	TSP	150	90.79	26524	23.00	5.07	822.6		
kroA200	TSP	200	92.15	29368	24.31	3.76	606.8		
kroB100	TSP	100	85.91	22141	25.83	2.20	237.3		
kroB150	TSP	150	90.44	26130	24.08	3.14	565.2		
kroB200	TSP	200	91.01	29437	23.19	3.41	1018.1		
kroC100	TSP	100	88.69	20749	23.10	4.68	82.9		
kroD100	TSP	100	87.55	21294	27.39	6.52	371.0		
kroE100	TSP	100	88.28	22068	25.71	1.74	120.2		
lin105	TSP	105	60.58	14379	39.56	2.96	181.1		
lin318	TSP	318	64.94	42029	38.36	5.06	3329.4		
pcb442	TSP	442	77.07	50778	11.32	3.84	7646.0		
pr107	TSP	107	29.40	44303	46.48	2.05	1204.9		
pr124	TSP	124	40.34	59030	34.54	0.73	494.6		
pr136	TSP	136	66.28	96772	15.11	3.98	488.2		
pr144	TSP	144	37.41	58537	66.33	1.49	847.3		
pr152	TSP	152	54.23	73682	42.51	1.58	1713.9		
pr226	TSP	226	27.21	80369	39.11	2.01	3059.9		

Continued on next page

Table 3.4 – continued from previous page

instance	type	$\|V\|$	root gap	best	lb gap	bk gap	RS [s]	GS nodes	dd:hh:mm:ss
pr264	TSP	264	36.99	49135	35.98	4.75	8072.2		
pr299	TSP	299	42.29	48191	19.47	2.69	4455.0		
pr439	TSP	439	60.38	107217	31.70	4.75	20186.8		
pr76	TSP	76	28.27	108159	30.33	2.28	147.5		
rat195	TSP	195	42.36	2323	14.17	4.83	551.8		
rat99	TSP	99	42.98	1211	11.46	1.54	67.4		
rd100	TSP	100	84.36	7910	21.89	5.80	229.0		
rd400	TSP	400	92.91	15281	20.93	2.24	3940.3		
si175	TSP	175	18.79	21407	6.00	0.59	382.1		
st70	TSP	70	80.21	675	25.22	2.74	65.5		
swiss42	TSP	42	55.08	1273	22.44	2.15	9.9	19241	00:00:00:42
ts225	TSP	225	54.20	126643	11.63	3.20	1431.0		
tsp225	TSP	225	62.16	3916	12.98	0.31	494.2		
u159	TSP	159	3.00	42080	17.66	0.00	139.4		
ulysses16	TSP	16	29.03	6859	18.38	0.00	3.5	549	00:00:00:05
ulysses22	TSP	22	42.51	7013	24.58	0.00	4.3	10923	00:00:00:11

Part II.

RotOR

Chapter 4.

RotOR's Hypergraph Model

In Chapter 1 we defined the rolling stock rotation problem (RSRP) in terms of a hypergraph $(V \cup S, A, H)$ which we claimed to be very useful for the industrial application of the RSRP, see Table 1.1. In fact, the current chapter presents the proper arguments for this claim, i.e., we describe the requirements of DB Fernverkehr AG (DBF) for the optimization of their rolling stock rotations that we exclusively model by $(V \cup S, A, H)$ within RotOR. The hypergraph serves as an input structure for RotOR's overall mixed-integer programming (MIP) model, which we present by the next chapter.

We start here by addressing versatile aspects of the vehicle composition requirement for railway vehicles in Sections 4.1 to 4.9. There, the difference between vehicle composition and vehicle configuration on that RotOR's coarse-to-fine (C2F) implementation is based is of particular importance, see Chapter 2. Note that the application of the vehicle composition aspects of this chapter is neither limited to the dedicated application at DBF, nor to the cyclic planning horizon (i.e., we assume the standard week in the RSRP). However, we also describe assumptions of RotOR's implementation that turn out to be necessary and reasonable for the particular railway application at DBF.

In Section 4.10 we come to the regularity requirement for rolling stock rotations. To the best of our knowledge, regularity for rolling stock has neither been investigated, nor mentioned in the scientific railway literature so far. This is surprising because it turns out to be of essential interest in industrial scenarios with a planning horizon of more than one day of operation, e.g., for standard week RSRPs.

Industrial RSRPs originate from two major use cases that we call *greenfield optimization* and *re-optimization*. By greenfield optimization we just mean the situation where a railway timetable has to be covered by rolling stock rotations such that they minimize operational cost. Apart from the timetable concept and rotation's cost the rolling stock rotations can be freely constructed in greenfield optimization.

In re-optimization instances we additionally try to construct the rolling stock rotations in such a way that they are as similar as possible to so called *reference (rolling stock) rotations*. The reference rotations are considered as part of RotOR's input data and reflect a planning situation that has already been completely or partially applied to the production process of a railway operator, e.g., DBF. Re-optimization is the more relevant the more it comes to the proper railway operation.

In Section 4.11 we explain RotOR's re-optimization capability, which is developed on top of the greenfield optimization model. In fact, from a modeling point of view, we only make a few modifications to RotOR's hypergraph $(V \cup S, A, H)$ and its objective

function $c : H \mapsto \mathbb{Q}_+$ when dealing additionally with reference rotations.

Finally, we close the chapter by Section 4.12 with the hypergraph's objective function $c : H \mapsto \mathbb{Q}_+$, which can be configured for both greenfield and re-optimization.

Parts of this chapter are published under the following titles:

- ▷ "A Hypergraph Model for Railway Vehicle Rotation Planning" [4],
- ▷ "Integrated Optimization of Rolling Stock Rotations for Intercity Railways" [6],
- ▷ "Re-optimization of Rolling Stock Rotations" [9], and
- ▷ "Template Based Re-Optimization of Rolling Stock Rotations" [10].

4.1. Fleets and Vehicle Configuration

As is well known, rolling stock is operated on railway tracks. At a dedicated point in time and location, there can only be one vehicle on a railway track. Thus, railway vehicles have to run one after the other through the railway infrastructure. For security reasons, this is constrained by a *minimum headway time*, see Schlechte (2012, [94]). Therefore, it is advantageous to couple vehicles together in railway systems because, then, minimum headway times are saved, which increases the capacity of the railway infrastructure.

Furthermore, the energy consumption of a set of vehicles that are coupled together is lower w.r.t. the case where the vehicles are operated individually. Finally, an estimated passenger demand needs to be served on timetabled trips. This may require more than one railway vehicle. In this situation it is advantageous if the vehicles arrive and depart in a coupled state at passenger platforms in order to optimize the space usage of the platform and the confusion of the passengers.

In this section we distinguish coupled railway vehicles by their *fleet* and their *vehicle configuration* (and later in Sections 4.6 and 4.7 by some more details).

A *fleet* is a basic type of railway vehicles. For example, the slightly more than 220 intercity express (ICE) vehicles of DBF are partitioned into several structurally identical sets of vehicles, namely fleets. We consider fleets as the basic types of vehicle units that need to be distinguished in rolling stock rotation optimization. A railway vehicle that operates according to a rolling stock rotation may traverse the standard week $n \in \mathbb{N}$ $(n \geq 1)$ times before it reruns the rotation. Such a rotation requires n railway vehicles, which must belong to the same fleet.

Several further constraints in connection with fleets exist. First of all, the number of railway vehicles of a dedicated fleet is limited. In addition, there exist constraints that require dedicated timetabled passenger trips to be exclusively operated by vehicles of dedicated fleets. E.g., an electric supply is not available for all railway tracks. For timetabled trips that use those tracks, fleets with vehicles that have a diesel engine must be chosen. Furthermore, there are other technical constraints that depend, e.g., on the country where a railway vehicle is operated. Different countries have different security systems to be supported by the equipment of the rolling stock.

Figure 4.1.: Possible vehicle configurations composed of two fleets, i.e., a red fleet and a blue fleet.

Railway vehicles of (possibly different) fleets can be coupled together. To this end, we define a *vehicle configuration* as a multiset of fleets. A vehicle configuration models that vehicles of the involved fleets appear to be coupled all together. It is defined as a multiset such that vehicles of a dedicated fleet can appear multiple times. Note that railway vehicles that have the same fleet can not be distinguished in a vehicle configuration–it is only known how many of them are coupled. To this end, we refine the handling of individual vehicles in vehicle configurations in Section 4.7.

Let F be the set of fleets. If we consider the set $F = \{\text{Red}, \text{Blue}\}$ of fleets, one could create the vehicle configurations of size two at most, i.e., $\{\text{Red}\}$, $\{\text{Blue}\}$, $\{\text{Red}, \text{Red}\}$, $\{\text{Red}, \text{Blue}\}$, and $\{\text{Blue}, \text{Blue}\}$. This is illustrated in Figure 4.1.

The relation of fleets and vehicle configurations plays one of the most important roles in rolling stock rotation optimization. In particular, in the application at DBF, ICE vehicles can be coupled together *on the fly*, i.e., no technical equipment or crew is needed for coupling or decoupling activities. There exist fleets at DBF for which the coupling time does not exceed ten minutes. Coupling activities can happen very frequently and, therefore, the handling of different vehicle configurations is essential in rolling stock rotation optimization. Thus, to determine which vehicle configuration to use for which timetabled trip is a basic question for a railway operator as DBF.

Figure 4.2 provides an example for the choice of different vehicle configurations for timetabled passenger trips. Suppose that the timetable consists of the passenger trips $T = \{t_1, \ldots, t_6\}$ to be covered by rolling stock rotations. These trips have to be operated in the driving direction as indicated by the six small blue trees. Assume, that the locations where the trips depart and arrive are arranged as the gray boxes are arranged (w.r.t. the pages' two-dimensional plane). Therefore a connection of t_1 to t_5 has a higher deadhead distance (i.e., a distance traversed without passengers, see Section 4.4) to be passed than a connection of t_1 to t_3.

Each of the trips t_1 and t_2 in Figure 4.2 can be operated by a single red railway vehicle and trip t_6 by a single blue vehicle. For trips t_3, t_4, and t_5 there are two options for the choice of vehicle configurations in each case. We assume that all connections between the timetabled trips are feasible except for those that connect by using different fleets (e.g., a connection between t_1 and t_6).

Suppose that one red and one blue vehicle is available and that we want to manually optimize the rolling stock rotations for this example. It seems natural that there must be a connection of t_2 and t_1. Now we have to determine what happens to the vehicle that arrives after operating t_1. A possibility is to close the rotation with a connection to t_2 again. But this might be expensive since we have to pass the whole distance traversed through t_2 and t_1 without passengers again. Therefore, we try to send the red vehicle through the three trips in the middle. This implies that we have to choose the vehicle

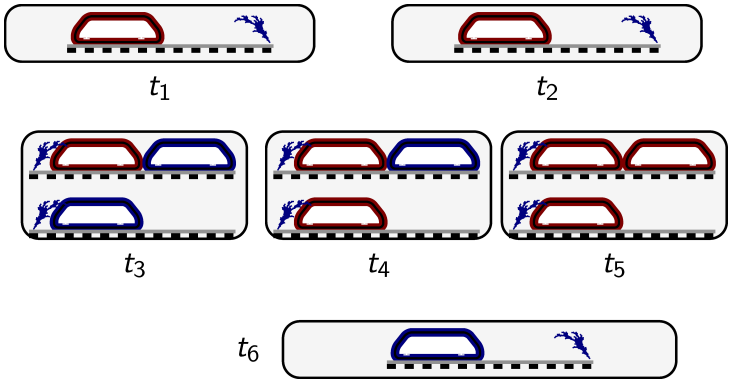


Figure 4.2.: Example for the choice of vehicle configurations.

configuration $\{\text{Red}, \text{Blue}\}$ for t_3.

Further, we choose the same vehicle configuration for t_4 because the red and blue vehicle from t_3 can not vanish. At trip t_5 we note that by choosing a single red vehicle and by connecting t_4 to t_5 and t_5 to t_2 the rolling stock rotation for the red vehicle has become closed. In addition, it seems reasonable to send the blue vehicle after the arrival at t_4 back to t_3 via operating t_6.

Finally, we decided to use two vehicles both for t_3 and t_4 but only a single vehicle for t_5. But what would have happened if we started our manual construction at t_3 for the timetable of Figure 4.2? A too shortsighted decision would have been to choose only one blue vehicle for the operation of t_3.

The example demonstrates that it is not obvious which vehicle configuration to choose for which timetabled trip; in particular for large and complex timetables. In addition, we have seen that it can be beneficial to choose vehicle configurations providing a passenger capacity that de facto exceeds the estimated passenger demand for timetabled trips.

The railway tracks for the timetabled passenger trips are assumed to be already allocated (i.e., reserved for operation) in rolling stock rotation optimization. But the tracks for additional deadhead trips (see Section 4.4) have to be still allocated (after rolling stock rotation optimization). This motivates to say that railway vehicles get *hauled* if a too large (w.r.t. the estimated passenger demand) vehicle configuration is used. *Hauling* is an often seen way to be efficient in the railway industry and a particular result of rolling stock rotation optimization. E.g., in the manually constructed rotations for the timetable of Figure 4.2 the blue vehicle at t_3 can be seen to be hauled by the other red vehicle[1].

We proceed with the explanation how we model vehicle configurations by a hypergraph. The nodes and hyperarcs of RotOR's hypergraph have the following interpretation. A node of the RSRP hypergraph is associated with the departure or arrival of a vehicle of a dedicated fleet of a timetabled trip. Each hyperarc of the RSRP hypergraph represents the connection of the involved nodes with a dedicated vehicle configuration. The vehicle configuration that is modeled is implied by the fleets of the involved tail and head nodes

[1] Although, it is not clear which vehicle hauls which vehicle in all cases.

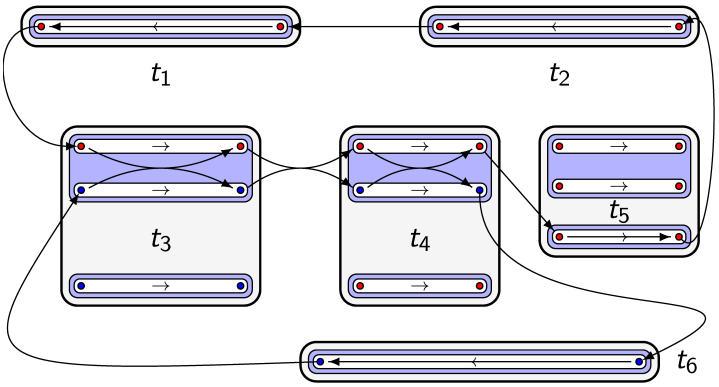


Figure 4.3.: Hypergraph model for vehicle configuration.

of a hyperarc.

Figure 4.3 shows a small example to provide the relation between hyperarcs and vehicle configurations. The hypergraphs arising in industrial instances are very dense because almost all connections between timetabled passenger trips exist.

The trips $\{t_1, \ldots, t_6\}$ in Figure 4.3 are the same as in Figure 4.2. All red and blue circles are nodes of the node set V of RotOR's hypergraph, i.e., departures or arrivals of railway vehicles operating the timetabled trips. The colors of the circles indicate the two fleets, namely the red and the blue one. The drawn hyperarcs form the solution that we constructed manually above.

The hyperarcs that connect the departure with the arrival of the timetabled trip $t \in T$ are elements of the set $H(t)$, i.e., the set of hyperarcs that can be used to cover the timetabled trip. For $H(t_3)$, $H(t_4)$, and $H(t_5)$ only the hyperarcs that were chosen in the solution are shown. Imagine that the others also exist, e.g., the hyperarc that covers t_5 by a vehicle configuration composed of two red vehicles.

To haul or not to haul an additional vehicle of the red fleet while operating t_5 corresponds to the chosen hyperarc for the operation of t_5. Clearly, only those hyperarcs for the operation of timetabled trips exist that are feasible in operation, e.g., w.r.t. the length of the passenger platform, the estimated passenger demand, and the required security equipment.

The hyperarcs that connect arrivals and departures of timetabled trips model how the trips can be connected to build a set of feasible rotations. Again, only those hyperarcs are drawn in Figure 4.3 that appear in the solution. Imagine that all hyperarcs exist that connect an arrival with a departure through nodes such that the colors of the nodes (i.e., the fleets) match. E.g., there is no hyperarc between t_1 and t_6 and also no hyperarc that connects two vehicles operating t_5 with two vehicles operating t_3.

The (hyper-) arc that connects trips t_1 and t_3 in Figure 4.3 implements a coupling activity before the departure of t_3, while the (hyper-) arc that connects t_4 and t_6 implements a decoupling activity after the arrival of t_4. The hyperarcs connecting t_2 and t_1 as well as t_3 and t_4 model connections between trips without coupling activities. On those connections the vehicle configuration is not changed.

Suppose that $\{\hat{h}_i\}_{i=1,\dots,6}$ are the hyperarcs that cover the timetabled trips in Figure 4.3 (i.e., trip t_i is covered by $\hat{h}_i$) and that $\{\tilde{h}_i\}_{i=1,\dots,6}$ are the hyperarcs that connect the arrival (i.e., $\hat{h}_i$ connects the arrival of trip t_i) of the timetabled trips with their succeeding departure in our solution. The two rolling stock rotations appear to be defined by $\{\hat{h}_1, \tilde{h}_1, \hat{h}_3, \tilde{h}_3, \hat{h}_4, \tilde{h}_4, \hat{h}_5, \tilde{h}_5, \hat{h}_2, \tilde{h}_2\}$ for the red fleet and $\{\hat{h}_3, \tilde{h}_3, \hat{h}_4, \tilde{h}_4, \hat{h}_6, \tilde{h}_6\}$ for the blue fleet.

A hyperarc is defined as set of standard arcs which ensures that the individual cycles (namely rotations) for the railway vehicles are well defined. More precisely, let $\hat{a}_{i,1} \subseteq \hat{h}_i$ and $\tilde{a}_{i,1} \subseteq \tilde{h}_i$ ($\hat{a}_{i,2} \subseteq \hat{h}_i$ and $\tilde{a}_{i,2} \subseteq \tilde{h}_i$) for $i = 1, \dots, 6$ be the standard arcs that are associated with the red (blue) fleet and of which the hyperarcs are composed. Under this notation, the two rotations that are illustrated in Figure 4.3 appear as $\{\hat{a}_{1,1}, \tilde{a}_{1,1}, \hat{a}_{3,1}, \tilde{a}_{3,1}, \hat{a}_{4,1}, \tilde{a}_{4,1}, \hat{a}_{5,1}, \tilde{a}_{5,1}, \hat{a}_{2,1}, \tilde{a}_{2,1}\}$ for the red fleet and $\{\hat{a}_{3,2}, \tilde{a}_{3,2}, \hat{a}_{4,2}, \tilde{a}_{4,2}, \hat{a}_{6,2}, \tilde{a}_{6,2}\}$ for the blue fleet[2].

In this section we illustrated how to model vehicle configurations by hyperarcs. Note that the illustrated model is a simplified version of RotOR's proper hypergraph for vehicle composition. But before we come to that in Section 4.7, we refine our considerations w.r.t. the rules that have to be considered when connecting arrivals and departures of timetabled trips by hyperarcs in Section 4.2. Indeed, these rules "only" apply to the involved vehicle configurations.

4.2. Turn Duration Rules

The hyperarcs of a solution of the RSRP particularly determine the connections between timetabled trips. But, given an arriving trip $t_1 \in T$ and a departing trip $t_2 \in T$ RotOR has to decide under which circumstances a connection of t_1 and t_2 is feasible. This is described in this section with a focus on rules that apply to the duration of so called *turns*, namely *turn duration rules*.

Connections are composed of states of operation. The states of operation that we distinguish are:

- ▷ operating a timetabled trip,
- ▷ operating a deadhead trip (see Section 4.4),
- ▷ maintaining (see Section 5.2),
- ▷ parking (see Section 5.3.2),
- ▷ and *turns*.

A turn appoints a dedicated period of time between succeeding states of operation. These times are used to, e.g., change staff, transfer staff, clean something, shunt, modify technical equipment (e.g., for a certain country), update data for seat reservations, and

[2]Note that our notation is not sufficient to distinguish vehicle configurations with vehicles of the same fleet (e.g., for t_5). Fortunately, this is not necessary for the example.

supply catering as well as newspaper material. Note that a turn is not directly related to the orientation (see Section 4.6) of a railway vehicle even if the name suggests this.

The important thing about turns is that there is a detailed set of rules that affect the feasibility and cost of connections, i.e., the shape of RotOR's hypergraph-based model. We remark that those rules are very sophisticated at DBF and may differ among other railway operators[3]. Therefore, it is not the intention of this section to present this set of rules in full detail, but the major aspects that have to be taken into account in rolling stock rotation optimization should not be suppressed here.

In the railway industry, a common phrase is to say "trip t_1 *turns* onto trip t_2" which means that a vehicle that arrives after operating t_1 continues with the operation of t_2. This turn type is called "platform turn" since it is performed directly on the passenger platform. Also other types like "maintenance turn" and "parking turn" that describe a connection of two timetabled trips with a maintenance service or a parking activity in between are common.

In general, all concatenations of states of operation and appropriate turns could be considered as potential connections that translate to hyperarcs of RotOR's hypergraph. This would lead to very huge hypergraphs. Instead, we model a proper subset of connections.

Assumption 1. *Given the timetabled passenger trips $t_1 \in T$ and $t_2 \in T$ we consider all connections of t_1 to t_2 that cover at most one additional location that is different from the arrival location of t_1 as well as different from the departure location of t_2.*

By Assumption 1 we consider all platform turns as well as parking and maintenance turns that may visit a third *service location*, see Section 4.5. The assumption excludes for example a connection of t_1 and t_2 by a path composed of a maintenance service in Munich and a parking activity in Berlin. In addition, the set of locations to be considered as parking locations for a dedicated arrival-departure pair of timetabled trips is restricted in RotOR's implementation–to a list that consists of, say, up to five locations in most of our instances. This list is a rather slight assumption since it is adjustable and turns out to be sufficient in our industrial application at DBF.

All rules for turns that are implemented in RotOR depend on the following properties:

- ▷ location of a turn,
- ▷ time of a turn[4],
- ▷ type of a turn, and
- ▷ vehicle configuration of a turn.

The reasons for this quite complex set of dependencies are highly specialized situations w.r.t. staff, frequency of traffic, occupied infrastructure capacity, and special shunting

[3] See the publicly available diploma thesis of Ullah (2005, [104]) that also deals with turn duration rules in Section 3.2.1 (also in the context of the ICE fleet of DBF).

[4] By "time" the time in terms of day and night is meant here. It is determined if a turn covers a predefined point in time, e.g., 2:30 a.m., in order to distinguish "night turns" and "day turns".

requirements caused by dedicated infrastructure topologies. Thus, the input data provided by DBF for RotORs hypergraph is a kind of essential expert knowledge for ICE rotation optimization.

As the name suggests, all turn duration rules apply to the *duration* (not to be confused with "time") of a turn. Given two dates $\theta_1, \theta_2 \in [0, \Theta]$ w.r.t. a continuous time horizon of length Θ, the duration $\mathrm{dur}(\theta_1, \theta_2)$ is defined as

$$\mathrm{dur}(\theta_1, \theta_2) := \begin{cases} \theta_2 - \theta_1 & \text{if } \theta_1 < \theta_2, \\ \Theta - \theta_1 + \theta_2 & \text{otherwise.} \end{cases} \tag{4.1}$$

In principle there is a minimum duration $d_{\min}$, a planned duration d_{plan}, and a maximum duration $d_{\max}$ in minutes for each dedicated situation that can arise by combining the properties of a turn. The minimum and maximum durations express hard operational constraints, i.e., a turn with a duration of d minutes is considered to be infeasible if $d < d_{\min}$ or $d > d_{\max}$.

For example the minimum duration to clean a railway vehicle or to move the train driver from one end to the other end in a terminus station is hard constrained. The maximum durations are mainly to restrict the maximal time that vehicles can spend on a passenger platform (other departing and arriving vehicles can be expected soon).

For a turn with a duration of d minutes the *turn duration deviation* is equal to $\max\{0, d_{\mathrm{plan}} - d\}$. We penalize deviations of the turn duration in the objective function by a constant linear multiplier, see Section 4.12.1. This is very important for the operation of the German ICE network and the data for those planned durations is very detailed and realistic, i.e., DBF knows bottlenecks in their operations that particularly influence the propagation of delays.

A typical situation that points out important aspects of turn duration rules is as follows. If the time between the arrival of trip t_1 and trip t_2 is short, it is not possible to perform any shunting or coupling activities such that the connection can only be made by all vehicles coupled together while arriving at t_1. We can model this by creating a single hyperarc–and no others–between the nodes that correspond to the arrival of t_1 and the departure of t_2. This simple construction formulates the "all or nothing constraint", i.e., those hyperarcs that would decouple the vehicle configuration after t_1 and recompose it before t_2 simply do not exist in RotOR's hypergraph (if it is required by the concrete situation).

An alternative approach to model connections between timetabled trips is the use of so called *timelines* [74] (also called "Kanal" in [103]). Roughly speaking, a timeline is a path or cycle that represents events at a dedicated location. By this construction one only has to decide the connections between timetabled trips and timelines and the connections between timelines. The connections between timetabled trips are anonymous, i.e., it is not possible to distinguish the rotations along timelines. On the one hand, timelines may save a huge number of connections if the number of different locations is large w.r.t. the number of timetabled trips. On the other hand, it is neither possible to handle maintenance constraints nor to take rules for maximal turn durations into account.

The most important consequences from this section are:

Figure 4.4.: Uncompressed (left) and compressed (right) hypergraph models for vehicle configuration.

1. The detailed industrial data given to evaluate the feasibility and cost of connections strongly depends on the current vehicle configuration. This is considered explicitly in RotOR's hypergraphs.

2. Planned turn durations need to be considered in RotOR's objective function in order to model bottleneck connections.

3. Timelines seem to be not appropriate for our purposes.

In particular the first consequence is crucial, because it leads to the circumstance, that a dedicated connection can only be made by two coupled vehicles, whereas the same connection is not possible to be made by one or two individual vehicles. This is the reason why RotOR's hypergraph has individual nodes for the operation of trip t_3 with one or two vehicles in Figure 4.3 on page 117. An alternative way to compress the hypergraph model w.r.t. the number of nodes and, finally, potential hyperarcs would be to only use two pairs of red and blue nodes for t_3 in this situation. This is indicated on the right of Figure 4.4. The proposed way to model vehicle configurations for the operation of timetabled trips by hyperarcs works as well as indicated by hyperarcs on both sides of Figure 4.4. But, by using the right, namely compressed hypergraph model, we a priori do not know which vehicle configuration is used for t_3 and can, therefore, not handle turn duration rules that depend on it when constructing hyperarcs connecting t_3 to others.

This gives also the natural but important insight that it can not be decided for a single rolling stock rotation on its own if it is feasible or not because the feasibility always depends on vehicles in vehicle configurations that might appear in other rolling stock rotations as well.

Suppose that turn duration rules do not depend on the vehicle configuration. An assumption giving this would be, e.g., that turn durations for cleaning only depend on the maximal duration over all cleaning times for the individual fleets of a vehicle configuration (a consequence of this assumption is that the vehicles of a vehicle configuration are cleaned parallel instead of sequentially). Then, the compressed hypergraph model could be used and the feasibility of single rolling stock rotations can be decided independently. This independence and the reduced size of the hypergraph model could possibly be exploited in algorithms (e.g., algorithms based on decision variables for paths or whole rotations). Unfortunately, the gain of such assumptions has not yet been investigated.

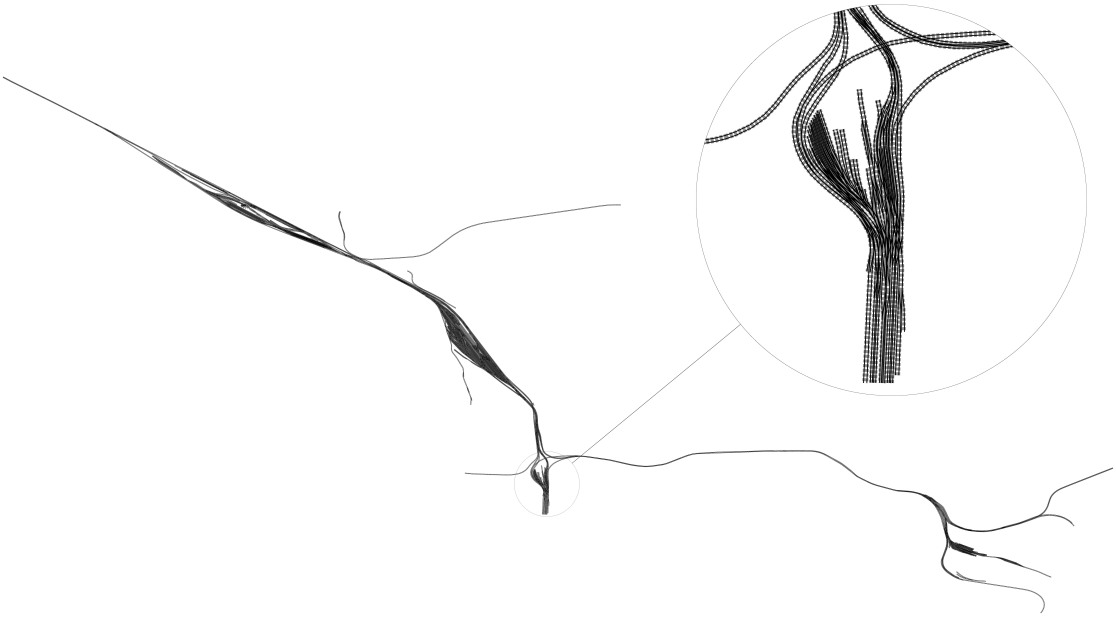


Figure 4.5.: ICE railway tracks of Hamburg. The station Hamburg-Altona is magnified.

4.3. Railway Topology

For the requirements vehicle orientation and vehicle composition that we explain later in Sections 4.6 and 4.7, we need to consider the railway network that we assume to be traversed by the rolling stock.

Figure 4.5 shows some tracks of the ICE network of Hamburg. Usually, ICE passenger trips coming from other parts of Germany enter Hamburg by the tracks in the south-east of the figure and have their end-station in Hamburg-Altona, which is magnified. After the passengers have left the railway vehicle, it is often parked at one of the parking tracks at Hamburg-Eidelstedt in the north-east of the figure or at Hamburg-Langenfelde, which is between Hamburg-Eidelstedt and Hamburg-Altona. As can easily be seen, a railway vehicle will turn around in Hamburg-Altona since it is a terminus station. Those turn around events have to be taken into account in the application of the RSRP.

To this end, we consider a railway network as a directed graph $(\mathcal{S}, \mathcal{T})$ in that the nodes $\mathcal{S}$ represent stations that are connected by a set of railways $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{S}$. E.g., we consider Hamburg-Altona as well as Hamburg-Eidelstedt and Hamburg-Langenfelde as stations in the railway network $(\mathcal{S}, \mathcal{T})$ connected through railways of $\mathcal{T}$ where we consider a railway to be a path traversing railway tracks.

In order to define a topology of the railway network $(\mathcal{S}, \mathcal{T})$ w.r.t. turn around events we consider each station of $\mathcal{S}$ to have exactly two *sides* $\{1, 2\}$. Through each of these two sides railway vehicles can enter and leave the station.

Figure 4.6 provides an example for the concept of sides. It shows the main station of

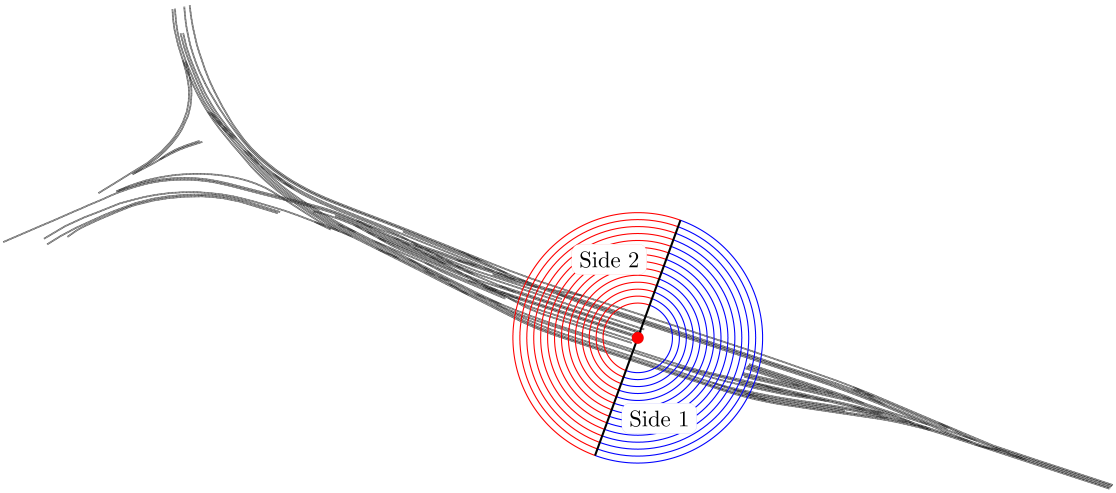


Figure 4.6.: Concept of sides: The example shows railway tracks of the station Dresden Hauptbahnhof.

Dresden. This station has the special characteristic that it has terminus tracks as well as tracks that run through the station. All tracks that come from the south east are considered to enter the station through side 1, while all tracks that come from the north west enter the station through side 2. Note that the concept of sides that we consider for the turn-around topology of a railway network derives from "microscopic" models for the train timetabling problem (TTP) in terms of double vertex graphs, see Schlechte (2012, [94]). But in the TTP context a station can have more than two sides (which is enough for our considerations).

Each railway $r = (u, v) \in \mathcal{T}$ with departure station $u \in \mathcal{S}$ and arrival station $v \in \mathcal{S}$ is now mapped by $\text{SIDE}_{\text{dep}} : \mathcal{T} \mapsto \{1, 2\}$ and $\text{SIDE}_{\text{arr}} : \mathcal{T} \mapsto \{1, 2\}$ to its departure side $\text{SIDE}_{\text{dep}}(r)$ and its arrival side $\text{SIDE}_{\text{arr}}(r)$. For example, $\text{SIDE}_{\text{dep}}(r) = 1$ and $\text{SIDE}_{\text{arr}}(r) = 2$ means for the railway $r = (u, v) \in \mathcal{T}$ that the railway vehicle departs through side 1 at station u and enters the arrival station v through side 2.

These functions, namely SIDE_{dep} and SIDE_{arr}, completely characterize the topology of a railway network $(\mathcal{S}, \mathcal{T})$ w.r.t. turn around events without considering detailed track data and without losing any necessary information. For a path $P \subseteq \mathcal{T}$ of the form

$$P = \{(u_0, u_1), (u_1, u_2), (u_2, u_3), \ldots, (u_{n-1}, u_n), (u_n, u_{n+1})\}$$

we can easily find out if a railway vehicle will turn around while traversing P. In fact, it will turn around if and only if

$$|\{u_i \in \mathcal{S} \,|\, i = 1, \ldots, n \,:\, \text{SIDE}_{\text{arr}}((u_{i-1}, u_i)) = \text{SIDE}_{\text{dep}}((u_i, u_{i+1}))\}| \mod 2 = 1.$$

In easy words, if and only if the number of stations traversed by P through equal arrival and departure sides is odd, the vehicle will turn around when traversing P.

With the concept of sides even the special structure of the main station of Dresden can be easily handled. Note that it is always possible to split an even more complicated

station into several individual stations. For example, the current main station of Berlin separates by one station with tracks connecting the south with the north and another station with tracks between the west and the east.

In operation it is sometimes desired to turn around and sometimes to not turn around. To this end, more than only two railways could be considered for the connection of any two stations $u, v \in \mathcal{S}$, i.e., each combination of a departure side of u with an arrival side of v results in a potential railway that could be taken. In order to keep the data manageable within RotOR we make the following assumption.

Assumption 2. *Let $(\mathcal{S}, \mathcal{T})$ be a railway network. We assume that for each two stations $u, v \in \mathcal{S}$ at most two railway paths $P_1, P_2 \subseteq \mathcal{T}$ that both start at u and end at v are given in RotOR's input data. In addition,*

- ▷ *P_1 and P_2 are assumed to traverse the same departure (arrival) side at u (v), and*
- ▷ *the number of turn around events of P_1 is odd while it is even for P_2.*

Assumption 2 cuts off the degree of freedom which sides to use by a railway connecting stations. In fact, up to eight possible connections between $u \in \mathcal{S}$ and $v \in \mathcal{S}$ could be considered: Leave u at side 1 (2), turn (not) around underway, and enter v at side 1 (2). We expect a low gain and a huge increase of complexity if we give up Assumption 2. In addition, we explicitly consider additional turn around trips in Section 4.4, which can be interpreted as a connection between different sides of a single station.

4.4. Deadhead Trips

A deadhead (trip) is a movement of a composition of rolling stock without passengers (or load in a railway freight context). Different to timetabled trips they are a result of rolling stock rotation optimization. RotOR's input timetable T, which has already been accepted by an infrastructure manager, does normally not contain any deadhead trips. For deadhead trips arising in RotOR's output, the infrastructure manager has to be requested again for the allocation of an appropriate railway path.

Since deadhead trips are usually performed without any passengers on board, a railway operator as DBF can not expect any direct profit from deadhead trips (except if a deadhead trip is miraculously redefined to be a passenger trip). Therefore, deadhead trips, or more precisely the distance they are covering, cause operational cost that have to be minimized by RotOR.

Deadhead trips are essentially required to move rolling stock between *different* stations of a railway network $(\mathcal{S}, \mathcal{T})$. Indeed, the movements induced by the timetabled trips T in $(\mathcal{S}, \mathcal{T})$ can not be expected to be constructed in a "deadhead avoiding" way. An obvious situation for the unexpected need of deadhead trips arises during disruptions. To this end, it is a common approach in industry to solve RSRPs in order to just see how much deadhead trip distance would be needed for a particular timetable concept.

Deadhead trips that run between different stations are required to reach service locations in order to perform maintenance or parking activities, see Section 5.2. Under

Assumption 1 from Section 4.2, a dedicated connection of two timetabled passenger trips of RotOR's hypergraph can implement up to two deadhead trips: At most two to reach and leave the service location and up to one if there is no extra service location, as well as no deadhead (in case of a platform turn). For these deadhead trips RotOR's input data contains a complete matrix that states the distance and time needed to perform a deadhead trip between any two locations (that appear as arrival, departure, or service location). In this data, the time required for a deadhead trip depends on the vehicle configuration as the turn duration rules from Section 4.2 do.

Indeed, also deadhead trips that move rolling stock from a station $v \in \mathcal{S}$ to the same station $v \in \mathcal{S}$ exist in RotOR's model. These deadhead trips are distinguished if they turn around the rolling stock. A deadhead trip that does not turn around but still moves vehicles within the same station is called *shunting operation*. Detailed shunting operations are approximately considered in RotOR's implementation by minimum turn durations, see Section 4.2.

If a deadhead trip moves rolling stock within the station and turns around the rolling stock, it is called *(plain) turn around trip*. Certainly, also a deadhead trip that connects different stations may turn around. But, during our industrial cooperation with DBF, plain turn around trips turn out to be very unpopular within rolling stock rotations. Nevertheless, turn around trips are sometimes necessary to compensate the shape of the railway infrastructure. They are also sometimes necessary to provide dedicated orientations (see Section 4.6) of railway vehicles that are required for technical aspects or for reasons of convenience.

The duration for a turn around trip depends on the vehicle configuration in order to be able to model different driving speeds of different vehicle configurations. This can be translated to hyperarcs of RotOR's hypergraph since each hyperarc $h \in H$ is associated with a distinguished vehicle configuration as we have seen in Section 4.1. In this way, by checking if the duration that is available for a deadhead trip is less or equal to the duration necessary to perform it, it is easy to verify if a hyperarc exists within RotOR.

Also the cost for a deadhead trip depend on the vehicle configuration. The reason is that the track allocation cost mainly depend on the duration for that a certain railway track is occupied. This duration is not significantly higher if more than one railway vehicle is used. Therefore, on the one hand, it is desirable to couple as many vehicles as possible for the operation of deadhead trips.

On the other hand, RotOR's input data contains an almost complete list of deadhead connections (that follow Assumption 2). Thus, all possible hyperarcs that model deadhead trips with arbitrary coupling and decoupling activities while traversing the railway network $(\mathcal{S}, \mathcal{T})$ could be considered by RotOR's model. We expect that this flood of hyperarcs is hard to be controlled computationally. Therefore, RotOR follows Assumption 3, which we suggest as a reasonable compromise for those situations.

Assumption 3. *Let $t_1, t_2 \in T$ be two timetabled trips. Let $h \in H$ be a hyperarc that connects t_1 to t_2. We assume that h models one of the following situations:*

- ▷ *The vehicle configuration during the arrival of t_1 is decoupled (if it is larger than one) after the arrival, the hyperarc h implements a vehicle configuration of size one,*

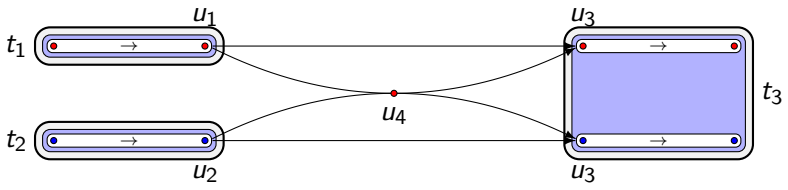


Figure 4.7.: Sophisticated situation for deadhead trips: The connection of $t_1 \in T$ (with arrival at $u_1 \in \mathcal{S}$) to $t_3 \in T$ (with departure at $u_3 \in \mathcal{S}$) and $t_2 \in T$ (with arrival at $u_2 \in \mathcal{S}$) also to t_3 could be realized by a vehicle configuration of size two that is created underway at $u_4 \in \mathcal{S}$.

and the vehicle configuration for the departure of t_2 is coupled (if it is larger than one) by vehicle configurations of size one.

▷ *The vehicle configuration during the arrival of t_1, the vehicle configuration that h implements, and the vehicle configuration during the departure of t_2 are all equal, i.e., no coupling or decoupling activity appears.*

Assumption 3 applies for example for the following sophisticated situation illustrated in Figure 4.7. A timetabled trip $t_1 \in T$ arrives in station $u_1 \in \mathcal{S}$ with a vehicle configuration of size one and another timetabled trip $t_2 \in T$ arrives in station $u_2 \in \mathcal{S}$ with $u_1 \neq u_2$ with a vehicle configuration also of size one. A third timetabled trip $t_3 \in T$ departs at station $u_3 \in \mathcal{S}$ with $u_3 \neq u_2 \neq u_1$ with a vehicle configuration of size two. This situation requires the movement of a vehicle from u_1 to u_3 and another movement from u_2 to u_3. These movements could be realized by coupling together at an appropriate meeting point $u_4 \in \mathcal{S}$, i.e., the railway vehicles that come from t_1 and t_2 move to u_4, then they are coupled to a vehicle configuration of size two at u_4, and proceed their movement together to reach the departure of t_3 at station u_3. This would possibly save track allocation cost for the movement from u_4 to u_3.

Nevertheless, we expect that the computational price for including hyperarcs that exactly represent those situations to be very high (many of those hyperarcs exist). Instead, we apply Assumption 3 to the construction of RotOR's hypergraph and handle situations similar to Figure 4.7 by two hyperarcs that individually connect t_1 to t_3 and t_2 to t_3. Note that this approach does not exclude principally underway couplings if they are, e.g., handled by a post-processing routine.

4.5. Service Paths for Maintenance Services

A major requirement of industrial RSRP instances are the maintenance constraints. A formal description of maintenance constraints is given in Section 1.4 of Chapter 1 of the thesis. Informally, a maintenance constraint restricts the maximal driven distance (or, alternatively, the operation time) of a railway vehicle between two consecutive maintenance services. A classical example is a maintenance constraint that ensures appropriate refueling

of diesel-driven railway vehicles. But also other preventive maintenance constraints and services have to be taken into account in the application of the RSRP at DBF.

Note that we distinguish between *maintenance services* and *maintenance constraints* in our terminology. Maintenance constraints restrict the solution space of an RSRP instance. We handle them by linear equalities, inequalities, and additional continuous variables in RotOR's MIP model, see Section 5.2.

Maintenance services are activities that increase the solution space of a RSRP instance. A maintenance service can be responsible for *several* maintenance constraints. E.g., we consider a refueling activity with a subsequent inspection as one single maintenance service that is associated with two maintenance constraints (i.e., refueling and periodic inspection).

Maintenance services are exclusively modeled by RotOR's hypergraph. To this end, we consider the data that we are given for a dedicated maintenance service:

- ▷ fleet to which the maintenance service is dedicated,
- ▷ location where the maintenance service can be performed,
- ▷ planned and minimum duration of the maintenance service,
- ▷ cost for performing the maintenance service, and
- ▷ the set of maintenance constraints associated with the maintenance service.

As denoted, maintenance services are associated with railway vehicles of a dedicated fleet. In fact, this is a result of the modeling, i.e., RotOR implements the following assumption.

Assumption 4. *We assume that railway vehicles become maintained individually, i.e., only a vehicle configuration of size one is considered for a maintenance service. Larger vehicle configurations are assumed to be decoupled before they run through a maintenance service.*

Assumption 4 excludes the consideration of maintenance services for coupled railway vehicles. In combination with Assumption 1 from Section 4.2 it implies additionally that we do not consider multiple maintenance services at different locations between succeeding timetabled trips. It turns out that both assumptions are rather weak in the application at DBF, i.e., all existing rolling stock rotations that we have seen during cooperating with DBF follow them.

The day and time when a maintenance service is to be performed belongs to RotOR's output while at the time of RotOR's hypergraph construction it is only known that maintenance services have to appear between timetabled trips (and not exactly when). This gives rise to model maintenance services as service nodes, which are *not* associated with a day or time when they are constructed, in RotOR's hypergraph model.

The set of service nodes is denoted by S. Obviously, service nodes have to be covered between arrival and departure nodes of timetabled trips in a solution of a RSRP instance.

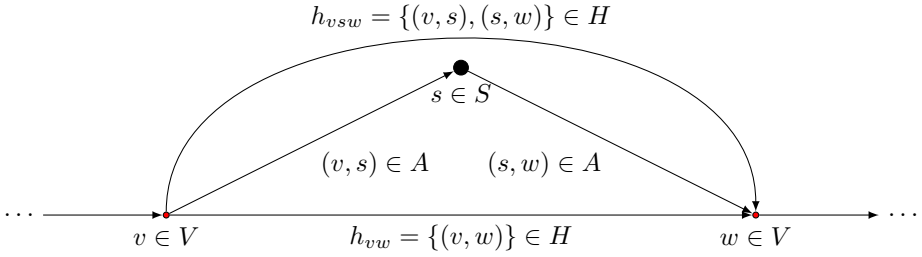


Figure 4.8.: Service paths are represented as replenishment hyperarcs.

To this end, we consider *service paths*. A service path is a path of length two of the following form:

$$\{(v,s),(s,w)\} \subseteq A \quad \text{with} \quad s \in S \text{ and } v,w \in V. \tag{4.2}$$

We assume that each service path of the form (4.2) starts at a node v that represents the arrival of a timetabled trip, traverses a service node s and ends at a node w that represents the departure of a timetabled trip.

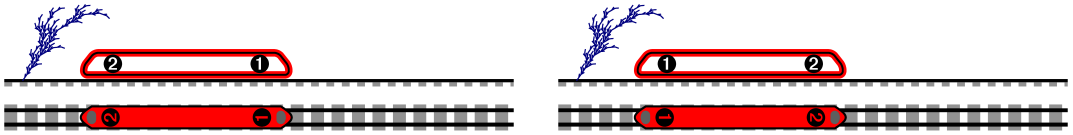
In RotOR's hypergraph each service node is dedicated to a particular connection of arrival and departure nodes. Consequently, a service node has exactly one incoming and one outgoing standard arc according to its service path. This allows to denote service paths simply also as "hyper"-arcs, i.e., the service path $\{(v,s),(s,w)\} \subseteq A$ is considered as the *replenishment hyperarc*

$$h = \{(v,s),(s,w)\} \in H$$

in RotOR's hypergraph. If h is part of a solution, it implies the traversal of its service path in a rolling stock rotation. Therefore, also the cost for the maintenance service can be easily associated with h. In addition, the proposed construction of h allows to handle the planned and minimum durations of maintenance services in the same way as explained in Section 4.2 for planned and minimum turn durations.

Note that we do not consider a maximum duration for maintenance services. This implies that railway vehicles may have to be parked after a maintenance service is completed. Those parking activities are not considered explicitly in RotOR's model. Instead, we assume that a parking area is nearby each location where maintenance services are performed.

Figure 4.8 illustrates the treatment of service paths in RotOR's hypergraph. The service path $\{(v,s),(s,w)\} \subseteq A$ is represented by the hyperarc $h_{vsw} \in H$. It models that a vehicle configuration arrives at $v \in V$, traverses the service $s \in S$, and finally departs on $w \in V$. In addition, the hyperarc h_{vw} models a direct connection of the arrival node $v \in V$ and the departure node $w \in V$ without performing a service.



(a) Tick: First class carriage in front (b) Tack: Second class carriage in front

Figure 4.9.: Vehicle orientations w.r.t. the driving direction indicated by the tree.

4.6. Vehicle Orientation

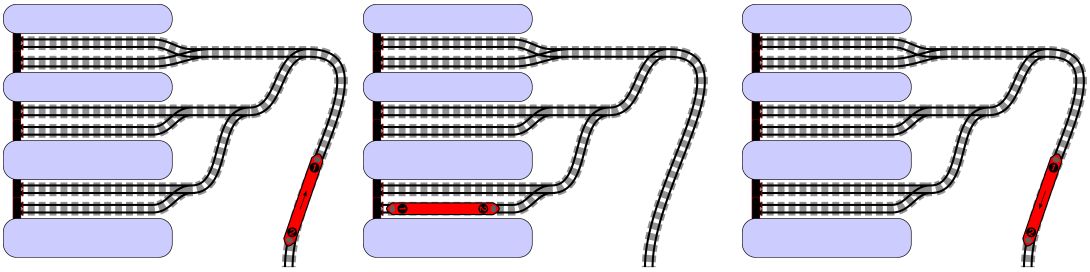
The *orientation* of a railway vehicle describes one of the two options of how vehicles can be placed on a railway track. Rolling stock rotations specify the orientations of each used vehicle in detail, i.e., the vehicle orientation is a result of rolling stock rotation optimization. To the best of our knowledge, our approach is the first that takes the vehicle orientation into account.

An orientation is an element of the set $O = \{\text{Tick}, \text{Tack}\}$. These terms originate from DBF where the vehicle orientation is distinguished by the position of the first class carriage of the vehicle w.r.t. the driving direction. Tick (Tack) means that the first class carriage is located at the head (tail) of the vehicle w.r.t. its driving direction, see Figures 4.9a and 4.9b.

The ICE vehicles of DBF are constructed such that the orientation does not influence or restrict their use, i.e., the ability to move and limitations on the driving speed are independent of the vehicle orientation. The vehicle orientation is of interest for the passengers, in particular in Germany. The reservations that passengers make for timetabled trips are associated with seats that have a dedicated location in the carriage. Among other things, this provides the ability to offer seats with special characteristics like a table or a seat next to the window as well as the level of quality (first or second class). In particular, the fact that the first and second class carriages are placed next to each other makes it necessary to distinguish the vehicle's orientation. Alternatives are to place the first or second class in the middle of a carriage or to offer reservations without a dedicated seat (as it is the case in the airline industry). At the passenger platforms, there are car position indicators that provide the orientation of the departing vehicles to the passengers. This makes it particularly necessary to consider the vehicle orientation in rolling stock rotation optimization.

Nevertheless, changes of the vehicle orientation occur naturally when traversing a railway network. At terminus stations the most common situation w.r.t. the change of the vehicle orientation appears as indicated in Figure 4.10. A railway vehicle that arrives at a terminus station (see Figure 4.10a) with orientation Tick will depart with the opposite orientation Tack (see Figure 4.10b) after the passenger platform is visited, see Figure 4.10c.

Since we consider a cyclic planning horizon we have to create rolling stock rotations that are *consistent* w.r.t. the vehicle orientation. This means, that the vehicle orientation needs to be constant at every event along a rotation (also when it is traversed multiple times). A rolling stock rotation that is not consistent w.r.t. the vehicle orientation is not



(a) Arrival at terminus station (b) At terminus platform (c) Departure at terminus station

Figure 4.10.: Change of orientation at terminus stations.

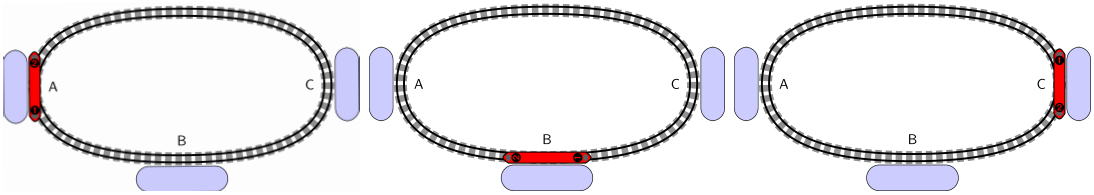


Figure 4.11.: Rolling stock rotation visiting the locations in the order A, B, C, A is always consistent w.r.t. the vehicle orientation.

applicable for further planning steps (at DBF). The assumption that everything repeats while traversing a rotation multiple times is necessary for the implementation in the production process. E.g., a car position indicator at passenger platforms that changes from week to week is clearly not desired.

In Figures 4.11, 4.12, 4.13, and 4.14 several examples of railway networks that are traversed according to rolling stock rotations are given. The orientation of the railway vehicle is indicated by the location of the first and second class carriage. It is assumed that the railway vehicle always traverses the shortest distance through the illustrated infrastructure. Imagine that timetabled trips are operated when traversing through locations A, B, and C in the order that is described by the captions.

On the one hand, we obviously observe that the vehicle orientation is always consistent if we traverse the railway infrastructure shown in Figures 4.11 and 4.12. This is even independent of the number of traversals.

On the other hand, we are not consistent for the examples in Figures 4.13 and 4.14 if we traverse even a single time through the infrastructure. By departing at the platform of location A with orientation Tack in Figure 4.13 and traversing the railway infrastructure

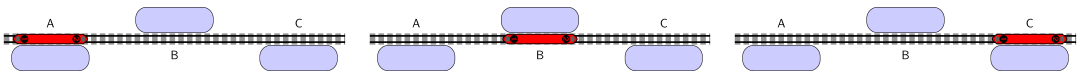


Figure 4.12.: Rolling stock rotation visiting the locations in the order A, B, C, B, A is always consistent w.r.t. the vehicle orientation since there are two changes of orientation at A and C.

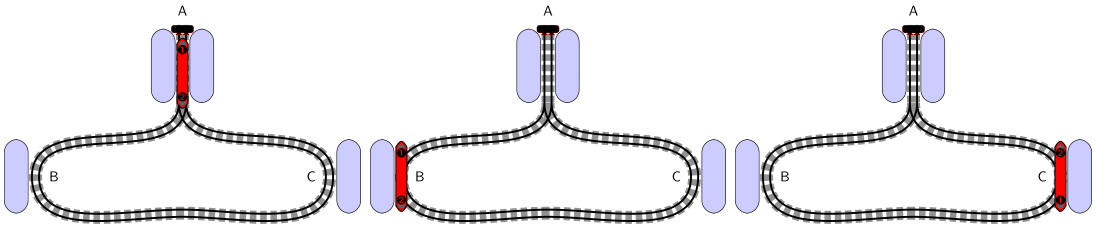


Figure 4.13.: Rolling stock rotation visiting the locations in the order A, B, C, A is only consistent w.r.t. the vehicle orientation if the rotation is traversed an even number of times since there is one change of orientation at location A.

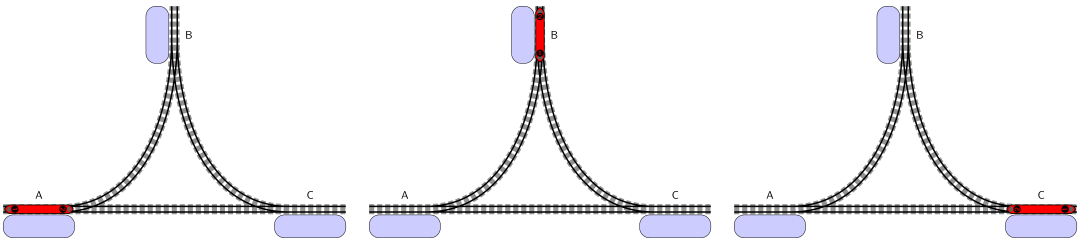


Figure 4.14.: Rolling stock rotation visiting the locations in the order A, B, C, A is only consistent w.r.t. the vehicle orientation if the rotation is traversed an even number of times. Rolling stock rotation visiting the locations in the order A, B, C, B, A is always consistent w.r.t. the vehicle orientation.

exactly for one time we arrive with orientation Tack at location A again. But if we traverse for the second time we depart at location A with the opposite orientation Tick, which is not consistent. If we traverse another time we arrive in orientation Tick at location A again. Then, we end up with a consistent rolling stock rotation since we would depart with vehicle orientation Tack at location A again.

From this examples we observe that we have to run an even number of times through this infrastructure to have a constant vehicle orientation at any event w.r.t. number of traversals, i.e., to be consistent.

The difference between the examples in Figures 4.11, 4.12 and Figures 4.13, 4.14 is the number of changes of the vehicle orientation imposed by the railway infrastructure. For Figure 4.11 it is zero, for Figure 4.12 it is one, for Figure 4.13 it is two, and for Figure 4.14 it is three. An odd number of changes of the vehicle orientation always leads to a rolling stock rotation that is not consistent, while an even number of changes of the vehicle orientation always leads to a consistent rolling stock rotation.

Consistency is the most important but not the only constraint to be satisfied in terms of vehicle orientation. There are a few hard constrained situations arising during the operations of DBF where only one of the two vehicle orientations is allowed due to infrastructure requirements, e.g., at maintenance or parking facilities. In addition, there are some stations where a dedicated vehicle orientation is preferred, e.g., for the first

class carriage.

Railway networks are typically rather complex. An important question is how to provide the data for orientation changes in a manageable way. In fact, this question is answered by the concept of sides, which we describe in Section 4.3.

All constraints w.r.t. the vehicle orientation can be handled by an appropriate construction of RotOR's hypergraph with applicable objective function values. This is described in detail in the next Section 4.7.

4.7. Vehicle Composition

As already mentioned, railway vehicles can be coupled together in order to form vehicle configurations, see Section 4.1. In our terminology, a vehicle configuration is a multiset of fleets. Indeed, the real-world situation is more complex in terms of coupling railway vehicles. In fact, it is not sufficient to only distinguish the fleets of the individual vehicles. Some industrial requirements apply to the individual positions and orientations of the railway vehicles that are coupled together. To this end, we define *vehicle compositions* that are based on the set of fleets F (see Section 4.1) and the set of orientations $O = \{\text{Tick}, \text{Tack}\}$, see Section 4.6. A *vehicle composition* of size $n \in \mathbb{N}_+$ is a n-tuple of the form

$$((f_1, o_1), (f_2, o_2), \ldots, (f_n, o_n)) \in (F \times O)^n.$$

The subscripting index $p \in \{1, \ldots, n\}$ is called the *position* of an individual vehicle in a vehicle composition. Each position is associated with an individual fleet $f_1, \ldots, f_n \in F$ and an individual orientation $o_1, \ldots, o_n \in O$. Which vehicle composition is used for the operation of timetabled trips is a result of rolling stock rotation optimization, i.e., it is computed by RotOR.

If we consider the set of fleets $F = \{\text{Red}, \text{Blue}\}$ we get the following potential vehicle compositions of size one: (Red, Tick), (Red, Tack), (Blue, Tick), and (Blue, Tack). Figure 4.15 illustrates the 16 possibilities for such vehicle compositions of size two. The fleet Red is represented by the red vehicle, while the blue vehicles represent the fleet Blue. Each gray vehicle has orientation Tack and each white vehicle has orientation Tick w.r.t. the driving direction as indicated by the small blue individual trees.

A vehicle composition is always a specialization of a vehicle configuration. We say that the configuration k is *realized* by the vehicle composition $c = ((f_1, o_1), (f_2, o_2), \ldots, (f_n, o_n))$ if $k = \{f_1, \ldots, f_n\}$, i.e., if the multiset of fleets used in the composition c is equal to the vehicle configuration k. In the example above the vehicle configurations $\{\text{Red}, \text{Red}\}$, $\{\text{Red}, \text{Blue}\}$, and $\{\text{Blue}, \text{Blue}\}$ are realized by the 16 vehicle compositions that are illustrated in Figure 4.15.

Despite the desired property that ICE vehicles operated by DBF can equivalently move in both driving directions there are exceptions if they are coupled together. We know about one such exception that establishes the special vehicle composition $((f, \text{Tack}), (f, \text{Tick}))$ (i.e., the second vehicle composition in the second row of Figure 4.15) for a dedicated ICE fleet $f \in F$ of DBF to be infeasible for operation (the technical reason is sophisticated).

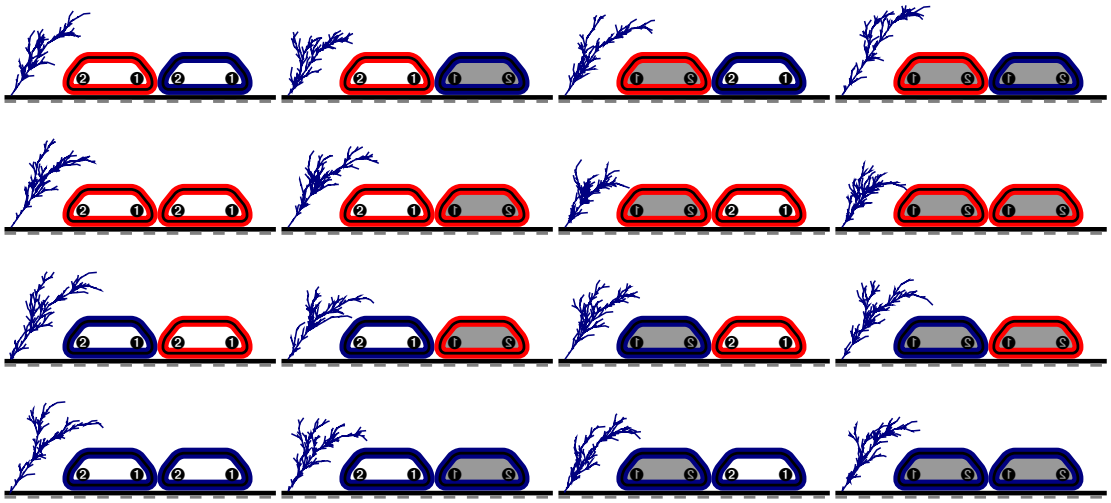


Figure 4.15.: All possible vehicle compositions of size two for fleets $F = \{\text{Red}, \text{Blue}\}$ and orientations $O = \{\text{Tick}, \text{Tack}\}$. The manifold blue trees indicate the driving direction of the vehicle composition. White (gray) vehicles are assumed to have orientation Tick (Tack).

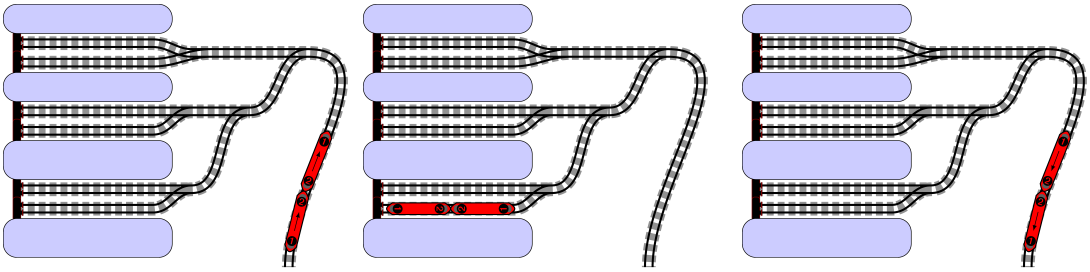
When the driving direction of a railway vehicle that is operated in a vehicle configuration changes, i.e., the whole vehicle configuration turns around, the vehicle's orientation changes too. If the vehicle orientation $o \in O$ is taken before turning around, we denote the opposite orientation after the turn around by $\bar{o} \in O \setminus \{o\}$.

In addition to the change of orientation, it is also possible (but not mandatory) that the whole vehicle composition of a vehicle configuration changes during a turn around. In general, a vehicle composition $c = ((f_1, o_1), (f_2, o_2), \ldots, (f_n, o_n)) \in (F \times O)^n$ will change to $c' = ((f_n, \bar{o}_n), (f_{n-1}, \bar{o}_{n-1}), \ldots, (f_1, \bar{o}_1)) \in (F \times O)^n$ when turning around. Note that the vehicle compositions c and c' can also be equal even if a turn around occurs.

Figure 4.16 illustrates this issue for a terminus station. The vehicle configuration of two red vehicles arrival in vehicle composition $((\text{Red}, \text{Tick}), (\text{Red}, \text{Tack}))$ at the terminus station (see Figure 4.16a) and halts at the passenger platform, see Figure 4.16b. During halting the vehicle composition is undefined since there is no movement (i.e., driving direction), see Figure 4.16a. Afterwards, the driving direction w.r.t. the arrival is changed when departing. But the vehicle composition $((\text{Red}, \text{Tick}), (\text{Red}, \text{Tack}))$ is still the same, even though the physical railway vehicles in the vehicle configuration have changed their positions. This is of course not always the case, e.g., a vehicle configuration that arrives in vehicle composition $((\text{Red}, \text{Tick}), (\text{Blue}, \text{Tick}))$ would depart with the vehicle composition $((\text{Blue}, \text{Tack}), (\text{Red}, \text{Tack}))$ in the same situation as in Figure 4.16.

We proceed with the explanation of how we model vehicle compositions by RoTOR's hypergraph. In fact, we simply extend the hypergraph model for vehicle configurations of Section 4.1 in order to handle vehicle compositions too.

Figure 4.17 illustrates the vehicle composition model in terms of RoTOR's hypergraph. It is a further refinement of Figure 4.3 of Section 4.1 where the timetabled trips $\{t_1, \ldots, t_6\}$



(a) Arrival at terminus (b) Platform halt (c) Departure at terminus

Figure 4.16.: Vehicle composition at terminus stations.

in Figure 4.17 are the same as in Figures 4.2 and 4.17. We kindly invite the reader to examine detailed aspects revealed by Figure 4.17 in the following.

The red and blue circles represent nodes of V of the hypergraph where the colors of the circles indicate the two fleets $F = \{\text{Red}, \text{Blue}\}$ again. Also the possible vehicle configurations that can be used to operate the timetabled trips are arranged in the same way as before. They are indicated by the blue boxes that surround the circles.

In order to deal with vehicle compositions in $(V \cup S, A, H)$ each node of V is now additionally associated with an individual position and an individual orientation. The position and orientation of a departure (an arrival) node of V correspond to the situation of an individual vehicle arranged in a vehicle composition while departing (arriving). The position taken w.r.t. the departure of a vehicle configuration is denoted within the gray or white boxes surrounding pairs of departure and arrival nodes in Figure 4.17. The colors of the white (gray) boxes declare the vehicle orientation Tick (Tack) also w.r.t. the situation while departing. Note that the position and orientation w.r.t. the departure is not necessarily equal to the situation when the vehicle configuration arrives because turn around events can occur along timetabled trips.

By the arrangement of the nodes the handling of vehicle compositions by RotOR's hypergraph is now obvious, i.e., each hyperarc represents the connection of the involved nodes performed with a dedicated vehicle composition. We use this concept for the operation of timetabled trips and for the connection of timetabled trips via hyperarcs. All the hyperarcs illustrated in Figure 4.17 are refinements of the hyperarcs illustrated in Figure 4.3. They form a set of rotations that define additionally the orientation and position of the railway vehicles.

As already introduced, for each timetabled trip $t \in T$ there is a subset $H(t) \subseteq H$ of hyperarcs that connect nodes associated with the departure and arrival of t. Each $h \in H(t)$ models the operation of $t \in T$ with a distinguished vehicle composition at departure. For the timetabled trips t_1, t_2, and t_6 in Figure 4.17 RotOR is allowed to choose a single vehicle configuration of size one. Therefore, all hyperarcs for their operation are de facto standard directed arcs. But, still, $|H(t_i)| = 2$ for $i \in \{1, 2, 6\}$ since one of the two orientations of $\{\text{Tick}, \text{Tack}\}$ has to be decided by RotOR. Note, that the number of turn around events along timetabled trips is precisely defined because

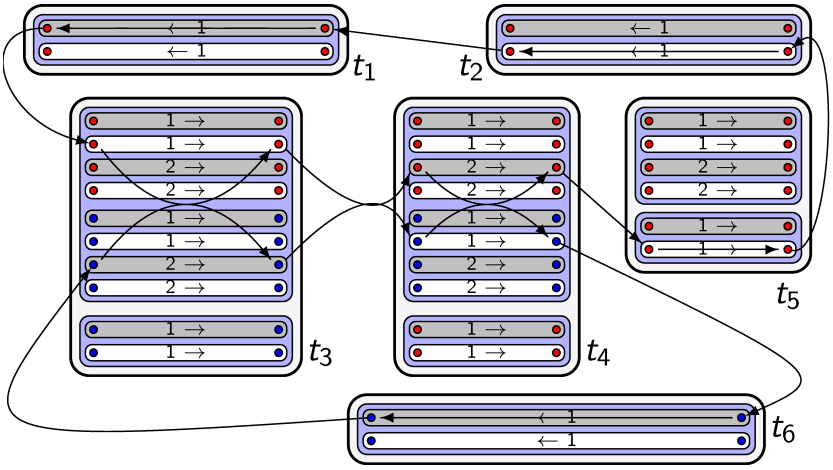


Figure 4.17.: Hypergraph model for vehicle composition.

the railway tracks traversed are already allocated in the planning step that we consider in rolling stock rotation optimization. Therefore, the orientation at the arrival of a timetabled trip is already defined by the orientation at departure and vice versa. Since the number of turn around events along a timetabled trip can be odd (and even as well), the orientation (and position) of railway vehicles at departure may be different from the situation at arrival (this is not depicted in Figure 4.17).

In comparison to trips t_1, t_2, and t_6 we have more possibilities for the operation of trips t_3, t_4, and t_5. For them RotOR has to decide if a vehicle configuration of size one or of size two is used in operation. For t_3 and t_4 the vehicle configuration of size two is composed of two different fleets, i.e., $F = \{\text{Red}, \text{Blue}\}$ while the vehicle configuration for t_5 of size two is configured from two vehicles of the red fleet. The exemplarily shown hyperarcs for t_3 and t_4 are composed of two standard arcs of A, which connect nodes associated with railway vehicles of the red fleet and the blue fleet. Indeed, each of them is only one out of eight possibilities to realize the vehicle configurations of size two for t_3 and t_4. These eight possibilities are illustrated in the first and third row of Figure 4.15. For the operation of t_5 a vehicle configuration of size one is used in the exemplary solution in Figure 4.17. However, all four vehicle compositions shown in the second row of Figure 4.15 that realize the vehicle configuration of size two could be alternatively used in order to operate t_5. By taking together the eight and four possible vehicle compositions of size two with the two vehicle compositions of size one we have $|H(t_i)| = 10$ for $i \in \{3, 4\}$ and $|H(t_5)| = 6$, respectively.

Beside hyperarcs for the operation of timetabled trips, RotOR's hypergraph $(V \cup S, A, H)$ contains a much larger set of hyperarcs dedicated to the connection of timetabled trips. They are created by RotOR under the Assumptions 1, 2, and 3 explained in the previous sections.

Let $h(t_i, t_j) \in H$ denote the hyperarc in Figure 4.17 that connects arrival nodes of trip t_i with departure nodes of trip t_j (with indices $i, j = 1, \ldots, 6$), e.g., $h(t_2, t_1) \in H$ denotes the hyperarc of size one that connects t_2 to t_1 in the top middle of Figure 4.17.

In order to sharpen understanding, we proceed with a discussion of the situations that these hyperarcs model.

The connection of t_2 and t_1 is performed by a railway vehicle from the Red fleet. The different orientations at the departure of t_2 and t_1 imply that at least one turn around event has occurred between the departures. We assume that t_2 does not run through any turn around events. Thus, the turn around occurs during the hyperarc $h(t_2, t_1)$. There can be two reasons for this situation. Either the railway topology (see Section 4.3) forces automatically a turn around or $h(t_2, t_1)$ models an additional turn around trip, see Section 4.4. The latter is only possible if there is enough time for the additional turn around trip. Such additional turn around trips are penalized in RotOR's objective function $c : H \mapsto \mathbb{Q}_+$, see Section 4.12.1. In general, there could be up to four different hyperarcs between t_2 and t_1. The same discussion can be applied to $h(t_5, t_2)$, i.e., either the railway topology forces the illustrated orientations at departure or an additional turn around trip is included in $h(t_5, t_2)$ in order to establish equal orientations.

The situation directly before the departure of t_3 is modeled by $h(t_1, t_3)$ and $h(t_6, t_3)$, i.e., one vehicle of the fleet Blue coming from t_1 is coupled to another vehicle of the fleet Red that comes from the operation of t_6. Also here, $h(t_1, t_3)$ or $h(t_6, t_3)$ may implement an additional turn around trip in order to establish the vehicle composition $((\text{Red}, \text{Tick}), (\text{Blue}, \text{Tack}))$ at the departure of trip t_3.

The trips t_3 and t_4 are both operated with the same vehicle configuration of size two in the exemplary rotations. Also during their connection by $h(t_3, t_4)$ the vehicle configuration is unchanged. But the realized vehicle compositions differ. Assumption 3 reveals that no coupling or decoupling activities appear through the connection modeled by $h(t_3, t_4)$. The only possibility in this situation in terms of the vehicle composition is that it turns around somewhere after the departure of t_3 and before the departure of t_4, i.e., the vehicle composition $((\text{Red}, \text{Tick}), (\text{Blue}, \text{Tack}))$ turns into $((\text{Blue}, \text{Tick}), (\text{Red}, \text{Tack}))$, see Figure 4.17.

Even under Assumption 3 RotOR still considers 16 different hyperarcs for the connection of t_3 and t_4 with a vehicle configuration of size two. There are eight possible vehicle compositions of size two for the arrival of t_3. All of them can or can not be turned around before they proceed with the departure of t_4. This gives 16 possibilities.

After the arrival of trip t_4 the vehicle configuration is decoupled into two individual vehicle configurations of size one that proceed with the operation of trips t_5 and t_6.

Beside the constraints mentioned in this section, there are further requirements that constrain the individual positions and orientations that railway vehicles can take within vehicle compositions. Those constraints particularly arise during coupling and decoupling activities, see Sections 4.9 and 5.4.

4.8. Trip Sequences

RotOR is allowed to change the vehicle configuration before or after but not during the operation of timetabled trips along the rolling stock rotations. It is also allowed to change the vehicle configuration even when passengers are on board if the situation

allows this. In order to refine technically the timetable for those configuration changes *trip sequences* are provided in RotOR's input data. A *trip sequence* of length $k \in \mathbb{Z}_+$ is a series $\{t_i\}_{i=1}^k$ of k timetabled trips $t_i \in T$ for $i = 1, \ldots, k$ that should *successively* appear (one immediately after the other in the order that the trip sequence defines) in RotOR's output, i.e., in the rolling stock rotations. In this way, potential changes of the vehicle configuration are handled by allowing or disallowing dedicated vehicle configurations for the individual trips of a trip sequence.

Trip sequences are further distinguished into *timetabled trip sequences*[5] and *enforced trip sequences*[6].

The individual trips of a timetabled trip sequence are announced in the timetable under a single train number. Two successive trips of a timetabled trip sequence have a scheduled (i.e., timetabled) intermediate stop in between. Therefore, a connection between two successive trips of a timetabled trip sequence does not have to follow the turn duration rules described in Section 4.2.

An enforced trip sequence is always of length two. The trips of an enforced trip sequence $\{t_i\}_{i=1}^2$ are announced as "train t_1 continues as train t_2" to the passengers. Thereby, the train numbers of t_1 and t_2 are different. Enforced trip sequences have to follow turn duration rules described in Section 4.2 but are sometimes specified exceptionally in order to bypass turn duration rules.

We handle the vehicle configurations as well as vehicle compositions for trip sequences in the same way as explained in Sections 4.1 and 4.7, i.e., trip sequences need almost no special treatment in RotOR's hypergraph model. Although, trip sequences are important because they constitute strongly the shape of the hypergraph. In addition, some particular constraints are associated with trip sequences, see Section 5.4.

The concept of timetabled trip sequences provides the possibility to disassemble any timetabled trip $t \in T$ into a finer trip sequence $\{t_i\}_{i=1}^k$ such that t and $\{t_i\}_{i=1}^k$ represent the same vehicle movements for passenger transport. The shorter the trips of the timetabled trip sequence, the more changes of vehicle configurations are possible. In this way, the degree of freedom for coupling and decoupling activities can be adjusted to a desired level by specifying the granularity of the timetabled trips.

For an example of trip sequences we consider the procedures illustrated in Figure 4.18. At the beginning, two railway vehicles become coupled together in Hamburg while another one waits in Frankfurt. The two coupled vehicles run to Berlin. Then, the blue vehicle on the back is decoupled and runs to Hanover. At the same time, the red vehicle proceeds to Leipzig. In the meantime, the waiting vehicle from Frankfurt has arrived and is coupled to the back of the red one at Leipzig. Finally, the new vehicle configuration of size two operates a timetabled trip to Munich. This example shows a typical application of trip sequence and contains three trip sequences:

- $\{t_i^1\}_{i=1}^4$, i.e., timetabled trip sequence from Hamburg to Munich with:
  - t_1^1: short trip of the red vehicle before coupling in Hamburg

[5] At DBF the trips of a timetabled trip sequence are called "Fahrlagenabschnitte".
[6] At DBF an enforced trip sequence are called "Zwangsbindung".

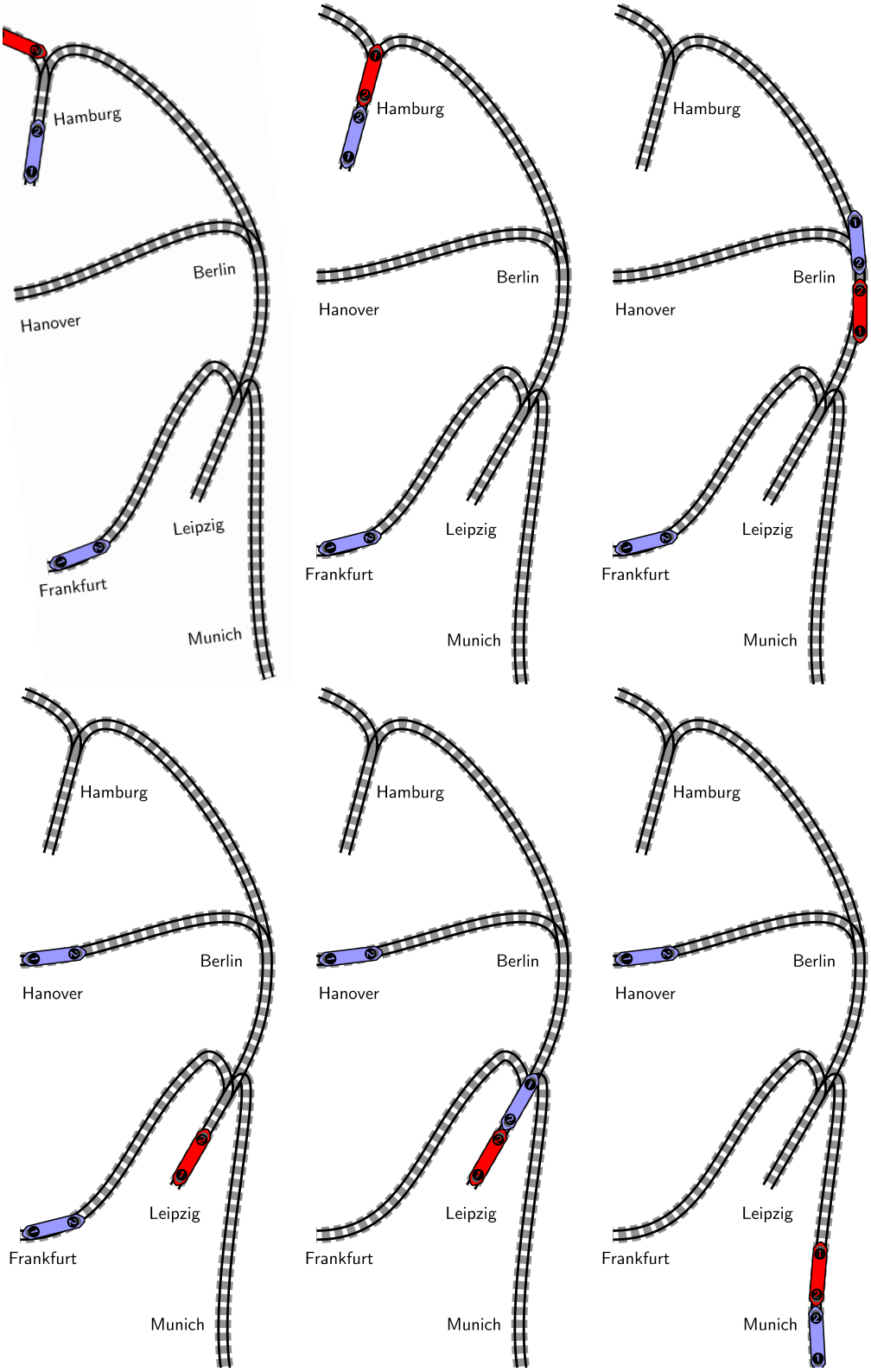


Figure 4.18.: Trip sequences.

- ▷ t_2^1: trip from Hamburg to Berlin
- ▷ t_3^1: trip from Berlin to Leipzig
- ▷ t_4^1: trip from Leipzig to Munich

▷ $\{t_i^2\}_{i=1}^2$, i.e., enforced trip sequence from Hamburg to Hanover with:

- ▷ t_1^2: trip from Hamburg to Berlin ($t_1^2 = t_2^1$)
- ▷ t_2^2: trip from Berlin to Hanover

▷ $\{t_i^3\}_{i=1}^2$, i.e., enforced trip sequence from Frankfurt to Munich with:

- ▷ t_1^3: trip from Frankfurt to Leipzig
- ▷ t_2^3: trip from Leipzig to Munich ($t_2^3 = t_4^1$)

In a proper industrial application at DBF it may be beneficial to refine the trip from Leipzig to Munich in order to allow one of the vehicles to move to Stuttgart (located in Germany's south-west) for example. This idea can be continued until each timetabled trip represents an atomic rolling stock movement. As a result, RotOR's hypergraph can become (too) large. Naturally, not every location is reasonable for a change of the vehicle configuration. The granularity of trip sequences in our industrial RSRP instances is adjusted by planners of DBF. A little more than 50 stations distributed over Germany (and neighboring countries partially) are considered as candidates for configuration changes. This results in manageable trip sequences $\{t_i\}_{i=1}^k$, with, say, $k \leq 6$.

Basic constraints for a trip sequence $\{t_i\}_{i=1}^k$ are that additional turn around trips, parking as well as maintenance activities, and shunting operations are not allowed when two successive trips of $\{t_i\}_{i=1}^k$ are connected. The reason for those constraints is that passengers may pass through the trips of a trip sequence. These constraints can be easily modeled by excluding corresponding hyperarcs.

For example, assume that trips t_3 and t_4 in Figure 4.17 are both operated with a vehicle configuration of size two and that t_3 and t_4 are successive trips of a trip sequence. Then, all hyperarcs that connect t_3 to t_4 have to be of size two as well in order to avoid any coupling activities in between. In addition, the number of hyperarcs that connect t_3 to t_4 is exactly eight because every vehicle composition at the arrival of t_3 ($|H(t_3)| = 8$ if we assume that only the vehicle configuration of size two is allowed for t_3) will–depending on the topology–either turn around or not turn around before the departure of t_3 (but there is no option for turning around).

Another constraint for a trip sequence $\{t_i\}_{i=1}^k$ is that one railway vehicle has to perform the whole sequence at once, i.e., a vehicle has to depart at t_1 and is not allowed to leave the trip sequence before the arrival of t_k. This can not be modeled exclusively by RotOR's hypergraph model in general. Section 5.4 addresses those constraints.

Note that trip sequences are not necessarily disjoint in their timetable, e.g., $\{t_i^1\}_{i=1}^4$ and $\{t_i^2\}_{i=1}^2$ share the trip from Hamburg to Berlin. This is a particular subject of Section 4.9.

4.9. Coupling, Decoupling, Combining, Splitting, Extending, and Reducing

In rolling stock rotation planning there are differences between coupling/decoupling, combining/splitting, and extending/reducing[7]. Combining is a special case of coupling and splitting is a special case of decoupling. In addition, extending is a special case of combining and reducing is a special case of splitting. What makes the differences is not completely sharp in the terminology of the railway industry. Here, we distinguish combining from coupling (as splitting from decoupling) as follows. If coupling or decoupling activities require (e.g., if the activities are to be performed fast or are complicated) to consider the individual positions of the railway vehicles within vehicle compositions we call the technical situations combining and splitting, respectively. The most specialized situations, namely extending and reducing, arise if an intermediate stop of a timetabled trip sequence (see Section 4.8) is involved.

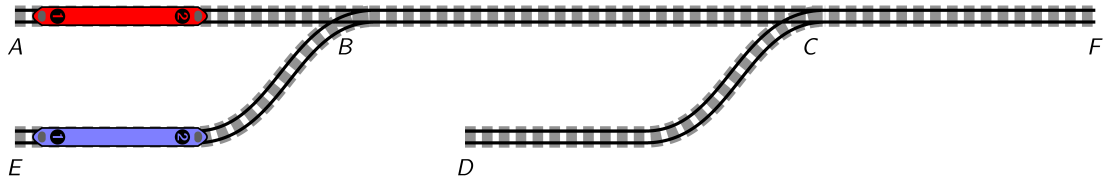
The specialization to extending and reducing is "only" a technical distinction made in the railway industry, i.e., DBF. But the difference between coupling/decoupling and combining/splitting affects RotOR's RSRP model. Interestingly, almost exactly the same distinction between coupling/decoupling and combining/splitting is made by Fioole et al. (2006, [47]) where a different model for vehicle composition is described and applied to instances of the Dutch railway operator Nederlandse Spoorwegen.

Typically, the consideration of the individual positions becomes necessary if there is only little time available for coupling and decoupling. Therefore, the distinction of coupling and decoupling from combining and splitting is essential to be considered in rolling stock rotation optimization.

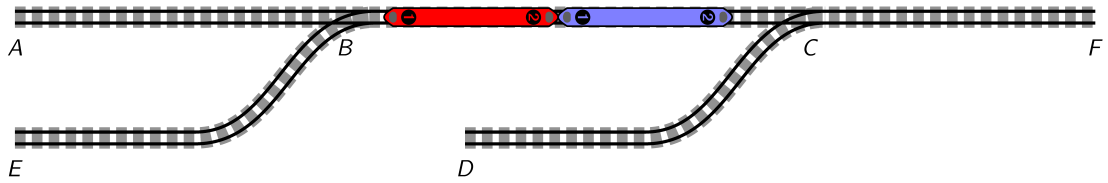
We explain the different situations for coupling and decoupling by referring to Figure 4.19. To this end, we assume that we are given five timetabled trips that we denote by $t_{AB} \in T$, $t_{BC} \in T$, $t_{CD} \in T$, $t_{EB} \in T$, and $t_{CF} \in T$. The notation of each of the five trips indicates at which stations the trips depart and arrive in Figure 4.19. The railway vehicles that operate t_{AB} and t_{EB} in Figure 4.19a meet at B and are coupled together, see Figure 4.19b. Then, they move coupled to C in order to operate t_{BC} and become decoupled again. Afterwards, both vehicles proceed individually with the operation of t_{CD} and t_{CF}, see Figure 4.19c.

Reducing and extending of timetabled trip sequences. At first we explain the most specialized case, namely extending and reducing. To this end, we assume $\{t_{AB}, t_{BC}, t_{CD}\}$ to be a timetabled trip sequence, i.e., the trip sequence provides a continuous passenger connections from A to D. In the railway industry it is said that the passenger capacity of the timetabled trip sequence is extended by the blue vehicle at B and reduced again at C. Typically, the durations of the intermediate stops at B and C are little, say not more than five minutes. In this situation the coupling and decoupling activities specialize to combining and splitting because the duration of the stop at B is short. Thus, it is

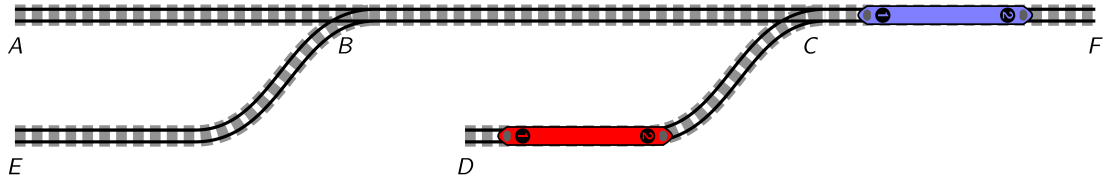
[7] At DBF coupling/decoupling, combining/splitting, and extending/reducing are referred as "Kuppeln und Entkuppeln", "Vereinigen und Flügeln", and "Stärken und Schwächen", respectively.



(a) Before coupling/combining



(b) Coupled/combined



(c) After uncoupling/splitting

Figure 4.19.: Splitting and combining

required that the red vehicle is coupled to the back of the blue vehicle. In other words, the blue vehicle needs to arrive at the passenger platform at B before the red vehicle arrives at B. Otherwise, the red vehicle would arrive at B. Then, the blue vehicle would move to the back of the red vehicle and then they would become coupled together. This procedure would require too much time, i.e., is often not possible in operation. The analog situation arises when decoupling at C. The blue vehicle needs to be at the front, i.e., position one, of the vehicle composition. Otherwise, the red vehicle would have to wait until the departure of the blue vehicle at C, which is also often not possible in operation because the red vehicle performs a short intermediate stop, while the blue vehicle gets, e.g., cleaned.

In this situation the timetabled trip sequence $\{t_{AB}, t_{BC}, t_{CD}\}$ is given as an input requirement RotOR's RSRP model. The connections of t_{EB} to t_{BC} as well as the connection of t_{BC} to t_{CF} are results of RotOR's computations. There are also situations where additional enforced trip sequences (e.g., $\{t_{EB}, t_{BC}\}$) are given as input requirement in order to force dedicated extensions or reductions.

Extending and reducing are highly typical procedures that arise very often in the RSRP instances provided by DBF. The corresponding coupling and decoupling activities distinguish themselves by the characteristic that they are performed within a timetabled trip sequence, i.e., during short intermediate stops of a continuous passenger connection. In these cases the individual positions of the railway vehicles are completely defined by the timetable concept, i.e., extensions and reductions always apply to the back or the

front of a vehicle composition. Therefore, extending and reducing can be directly handled by the design of RotOR's hypergraph. E.g., hyperarcs that model the connection of t_{EB} to t_{BC} such that the red vehicle is coupled to the front of the blue vehicle do simply not exist.

Decoupling and coupling outside timetabled trip sequences. Coupling and decoupling can also involve departures and arrivals of timetabled trips that do not belong to intermediate stops of a timetabled trip sequence. These cases are not referred by reducing or extending in the railway industry. Suppose that all timetabled trips associated with Figure 4.19 are not contained in any timetabled trip sequence, i.e., none of the four incoming and outgoing connections of trip t_{BC} are associated with an intermediate stop of a continuous passenger connection. In this case, a minimum turn duration (see Section 4.2) for the connections of t_{AB} to t_{BC}, t_{EB} to t_{BC}, t_{BC} to t_{CD}, and t_{BC} to t_{CF} are given. These minimum turn durations are not allowed to be violated and are usually significantly higher than the duration of intermediate stops. Therefore, most of those cases do not specialize to combining and splitting. This means that we do not have to constrain the position of the individual railway vehicles because we (can) assume that there is enough time for establishing the vehicle composition that is taken on the succeeding trips (possibly by additional shunting operations).

In some cases the positions of the individual vehicles, indeed, need to be considered explicitly. Fortunately, almost all of these situations are provided explicitly as enforced trip sequences in RotOR's input data. For example, suppose that the situation in Figure 4.19 violates minimum turn duration rules but is still desired for some reason. In this case, enforced trip sequences are specified. For the situation in Figure 4.19 the enforced trip sequences $\{t_{AB}, t_{BC}\}$,$\{t_{EB}, t_{BC}\}$, $\{t_{BC}, t_{CD}\}$, and $\{t_{BC}, t_{CF}\}$ could be created. Sometimes only one of the trip sequences $\{t_{AB}, t_{BC}\}$ and $\{t_{EB}, t_{BC}\}$ are enforced in order to leave the other trip that serves in t_{BC} as an option for RotOR.

In this way, we are also able to handle combining and splitting activities outside timetabled trip sequences exclusively by RotOR's hypergraph model.

Note that this approach is not completely aware of all situations that might require to take the individual positions of vehicles into account. For example, suppose that the minimum turn duration after the arrival of t_{BC} is short, say ten minutes and assume that the blue vehicle continues to B (again) while the red vehicle continues to D. In this situation it is theoretically possible that the departure of the blue vehicle is blocked by the red vehicle if it waits too long at the passenger platform at C. Nevertheless, those cases have not occurred during our cooperation with DBF (so far). Instead, the assumption that the minimum turn duration rules leave enough time for possibly necessary shunting operations turns out to be sufficient in our application.

4.10. Regularity Patterns

As motivated in the introduction of Chapter 1, we consider a cyclic planning horizon with seven individual days of operation in RotOR's model. We call this horizon standard

week in which the timetable is allowed to be different on each of the seven single days of operation, while it is assumed that the trips repeat from week to week.

A closely related concept is called *standard (operational) day* where the timetable is required to be equal on each day of operation. In the RSRP the standard day concept is not sufficient because it is much more restrictive in comparison to the standard week. But still, it has a significant importance; in particular for the regularity stipulations of the RSRP. Moreover, the standard day is often directly used by other transportation companies for several reasons.

The first occasion for the standard day is a transportation business that does not require the consideration of seven individual days. E.g., a local bus operator or shipping company can assume that all buses or trucks are located at a certain depot during the night. Those assumptions do not carry over to large railway companies.

Another reason for the usage of the standard day is that it is sometimes necessary to make long term concepts in which the input data does not differ over the days of operation or is only for an ideal day of operation at hand. These concepts are made in order to estimate, e.g., the required number of vehicles vehicles if this or that timetable is applied. Of course, this also applies to railway companies from time to time.

The major reason for the widespread distribution of the standard day (also in the railway industry) is its definition itself, i.e., the characteristic that everything repeats from day to day. This is strongly desired in operation (also at DBF) because the opposite, i.e., that everything differs from day to day is much more complex and therefore costlier to perform.

Indeed, most timetables for a standard week were derived from a standard day timetable at some point of the production process. During this derivation, many timetabled trips or at least parts of them still repeat on several days of operation. It is desired to take over these repetitions into the planning of the rolling stock rotations even when we consider a standard week. This requirement is what we call regularity. Regularity of rolling stock rotations simply means that operational routines repeat w.r.t. the days of operation.

Viewed from the outside, the consideration of regularity in rolling stock rotations might appear as a cosmetic improvement. Surprisingly, this is definitely not the case. Rolling stock rotations that do not follow regularity requirements are not accepted to be implemented in operation. This has been discovered during cooperating with DBF.

From an applied point of view, the purpose of the regularity requirement is to create rolling stock rotations that are compactly representable, easy to communicate, and easy to operate. This is important because rolling stock rotations are the basis for many further planning steps, e.g., duty scheduling and dispatching. In all further planning steps it is preferred to work with simple structures.

Repeating routines are not only desired from the perspective of a railway operator. Indeed, it is also an important convenience property for the users (i.e., passengers) of an ICE network. This is the reason for the popular concept of a periodic timetable. Periodicity in timetabling, see Liebchen and Möhring (2007, [71]), means that the timetabled trips follow dedicated periods, e.g., a trip that runs within a period of every two hours from Berlin to Munich. In addition, passengers can also expect that a certain trip from Berlin to Munich operated on Monday will also be available on Tuesday, i.e, the

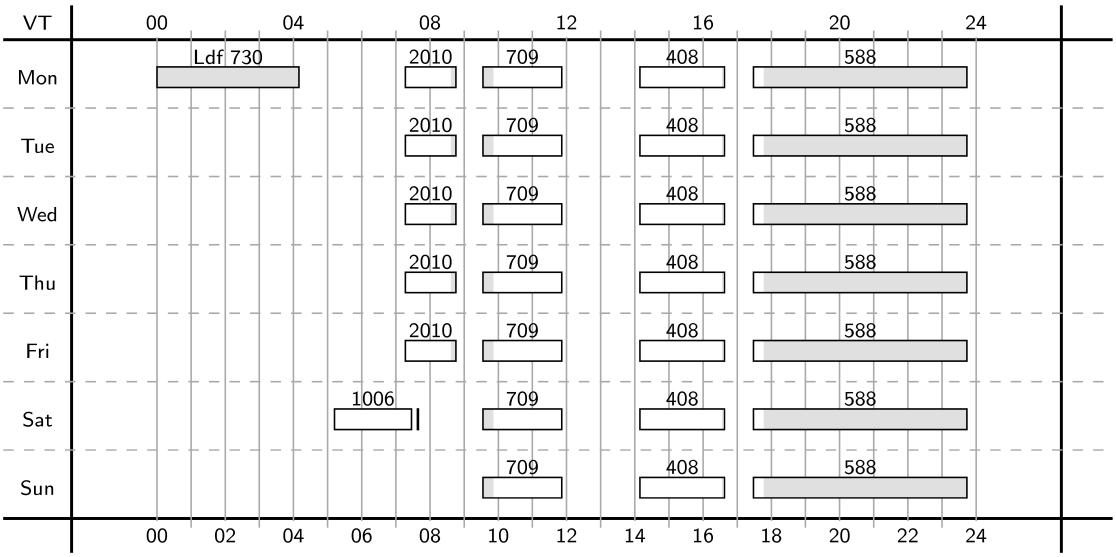


Figure 4.20.: Visualization of a rolling stock rotation.

trips follow a period of one day. In this way, we see regularity in rolling stock rotation optimization as the counterpart of periodicity in timetabling.

The maximal amount of regularity one can ever achieve can be obtained by creating rolling stock rotations for a standard day and carrying them over to the standard week. This clearly shows that the amount of regularity that can be achieved for rolling stock rotations always depends on the amount of periodicity that is included in the timetable.

In the application at DBF the timetable varies over the days of operation, e.g., in ICE networks the passenger demand on the weekend is different from the demand on workdays. Nevertheless, the structure of the given timetable is periodic to a large extent. Only a few trips of the timetable differ over the individual days of operation, i.e., the departure and arrival locations and corresponding times are often equal.

In order to identify repeating trips we introduce the following definition. We define a set of *equal* trips as a *train* if the trips have equal times of the day and locations w.r.t. departure and arrival. Let $\mathfrak{T}$ be the set of trains.

Figure 4.20 illustrates this definition. It shows a part of a real rotation plan of DBF, which is visualized anonymously by one of RotOR's drawing routines. It particularly shows five trains that can be distinguished by their train numbers, i.e., 1006, 2010, 709, 408, and 588. Each row is a part of a rolling stock rotation. The boxes stand for timetabled trips (except for "Ldf 730", which is a deadhead trip with a turn around event), which are colored by the current orientation taken along the trips. Gray means Tack, while white indicates the orientation Tick. Train 2010 is not operated on the weekend and train 1006 is exclusively operated on Saturday. All other timetabled trips, i.e., trains completely repeat over the days of operation. The figure shows a relatively suitable example for regular rolling stock rotations, i.e., only at Monday and Saturday morning there are deviations from the regular "schedule".

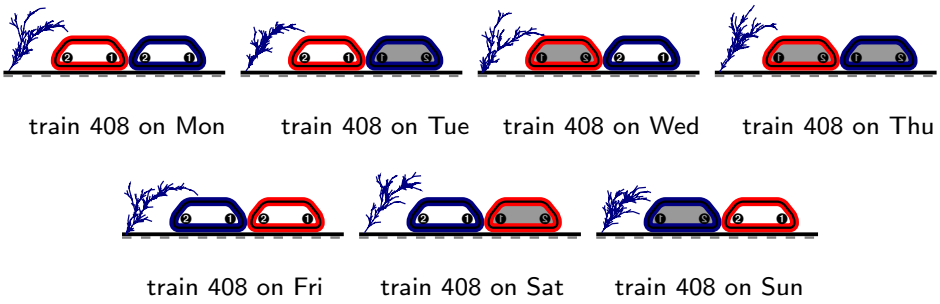


Figure 4.21.: Most irregular operation of train 408.

Our regularity approach is divided into three parts that tackle individual regularity *patterns*, which can be recognized in the rolling stock rotations and address different aspects:

- ▷ Section 4.10.1: Regular trips
  E.g., train 408 departs always with orientation Tick in Figure 4.20.
- ▷ Section 4.10.2: Regular turns
  E.g., train 408 is always followed by train 588 in Figure 4.20.
- ▷ Section 4.10.3 Regular handouts
  The visualization of rolling stock rotations, e.g., the seven rows illustrated in Figure 4.20.

We handle the first two regularity patterns, namely regular trips and turns, by defining hyperarcs with appropriate objective function coefficients. Then, these hyperarcs are optimized within RotOR's overall RSRP model. In this way, these two patterns are considered in an integrated manner.

As can be seen in Figure 4.20, the last pattern depends on the first two patterns, i.e., the arrangement of the seven rows in Figure 4.20 would not have been possible if there are less regular trips and less regular turns. Hence, among other things, the first two patterns prepare for the visualization, i.e., handout of a rolling stock rotation.

Within RotOR, the last pattern is optimized after the rolling stock rotations are created by a post-processing routine. In this way, the rolling stock rotations are not further modified in order to improve the visualization.

We discuss RotOR's approach for the three regularity patterns in the following.

4.10.1. Regular Trips: Regular Vehicle Compositions for Trains

The first regularity pattern is directly recognized by the passengers. The pattern is to maximize the number of equal vehicle compositions that are used for the operation of the trips of a train. RotOR's model can assign an individual vehicle composition to each trip of a train as illustrated in Figure 4.21.

In this example train 408 is operated by seven different vehicle compositions on each day of operation. In fact, it is desired to operate train 408 with the same vehicle composition at each day of operation. In this way, Figure 4.21 shows a worst case regularity pattern for the operation of train 408.

Let $\mathfrak{t} \in \mathfrak{T}$ be a train with $m \leq 7$ individual trips $t_1, \ldots, t_m \in T$, i.e., $\mathfrak{t} = \{t_1, \ldots, t_m\}$. Each trip $t \in \mathfrak{t}$ can be operated with an individual vehicle composition, which is modeled by a hyperarc of $H(t) \subseteq H$. Our idea is to introduce a set of hyperarcs $H(\mathfrak{t}) \subset H$ that model the operation of *several* trips of the train $\mathfrak{t}$ *at once*, i.e.,

$$H(\mathfrak{t}) \subseteq \bigcup_{(h_1,\ldots,h_m) \in H(t_1)\times\ldots\times H(t_m)} 2^{\{h_1,\ldots,h_m\}}.$$

A hyperarc $h \in H(\mathfrak{t})$ is called *regularity hyperarc*. As denoted, a regularity hyperarc is an union over hyperarcs operating timetabled trips of a train. Figure 4.22 illustrates seven different regularity hyperarcs that could be used for the operation of train 408 with a red and a blue vehicle. Each of the seven regularity hyperarcs would operate train 408 with an equal vehicle composition on every day of operation.

The notation above reveals the constraint that at most one hyperarc of $H(t)$ is included in a regularity hyperarc h for each $t \in \mathfrak{t}$. This construction ensures that h does not need any special attention, i.e., regularity hyperarcs can be handled as all other hyperarcs are handled in RotOR's hypergraph-based approach. But, a regularity hyperarc $h \in H(\mathfrak{t})$ gives access to the more or less regular pattern of the vehicle compositions that it applies to the operation of the trips of a train.

The overall number of possible regularity hyperarcs for a dedicated train $\mathfrak{t} \in \mathfrak{T}$ can be very large, i.e., $|H(\mathfrak{t})| \leq \prod_{t\in\mathfrak{t}} (|H(t)| + 1)$. Not all of them are worth to be considered, e.g., to only change the vehicle composition on Tuesday to the one that is taken on Friday in Figure 4.21 is not really an improvement that matters in operation. In addition, the detailed choice of the cost coefficients for regularity hyperarcs that implement this or that pattern can be discussed endlessly.

In our industrial application at DBF we made assumptions in order to keep the size of the sets $H(\mathfrak{t})$ manageable and to provide a transparent objective function.

The first assumption depends on the vehicle configurations (see Section 4.1, not to be confused with vehicle composition) that are allowed for the operation of the trips of a train. For the train $\mathfrak{t} \in \mathfrak{T}$ we consider an inclusion maximal subset of timetabled trips $\mathfrak{t}' \subseteq \mathfrak{t}$ such that all trips $t \in \mathfrak{t}'$ are allowed to be operated with an equal vehicle configuration. We denote all these subsets for the train $\mathfrak{t}$ by $R(\mathfrak{t}) \subseteq 2^{\mathfrak{t}}$. In addition, we also consider the set $\bar{R}(\mathfrak{t})$ that consists of all sets that are complementary to a set in $R(\mathfrak{t})$, i.e., $\bar{R}(\mathfrak{t}) := \{\bar{\mathfrak{t}} \in 2^{\mathfrak{t}} \mid \bar{\mathfrak{t}} = \mathfrak{t} \setminus \mathfrak{t}' \,:\, \mathfrak{t}' \in R(\mathfrak{t})\}$.

E.g., if we assume that all six trips of the Figure 4.3 (or equivalently of Figure 4.17) form a single train $\mathfrak{t} = \{t_1, \ldots, t_6\}$ we have

$$\begin{aligned} R(\mathfrak{t}) &= \{\{t_1, t_2, t_4, t_5\}, \{t_3, t_6\}, \{t_3, t_4\}, \{t_5\}\}, \text{ and} \\ \bar{R}(\mathfrak{t}) &= \{\{t_3, t_6\}, \{t_1, t_2, t_4, t_5\}, \{t_1, t_2, t_5, t_6\}, \{t_1, t_2, t_3, t_4, t_6\}\}. \end{aligned}$$

Assumption 5 addresses the size of the sets $H(\mathfrak{t})$ for the train $\mathfrak{t} \in \mathfrak{T}$.

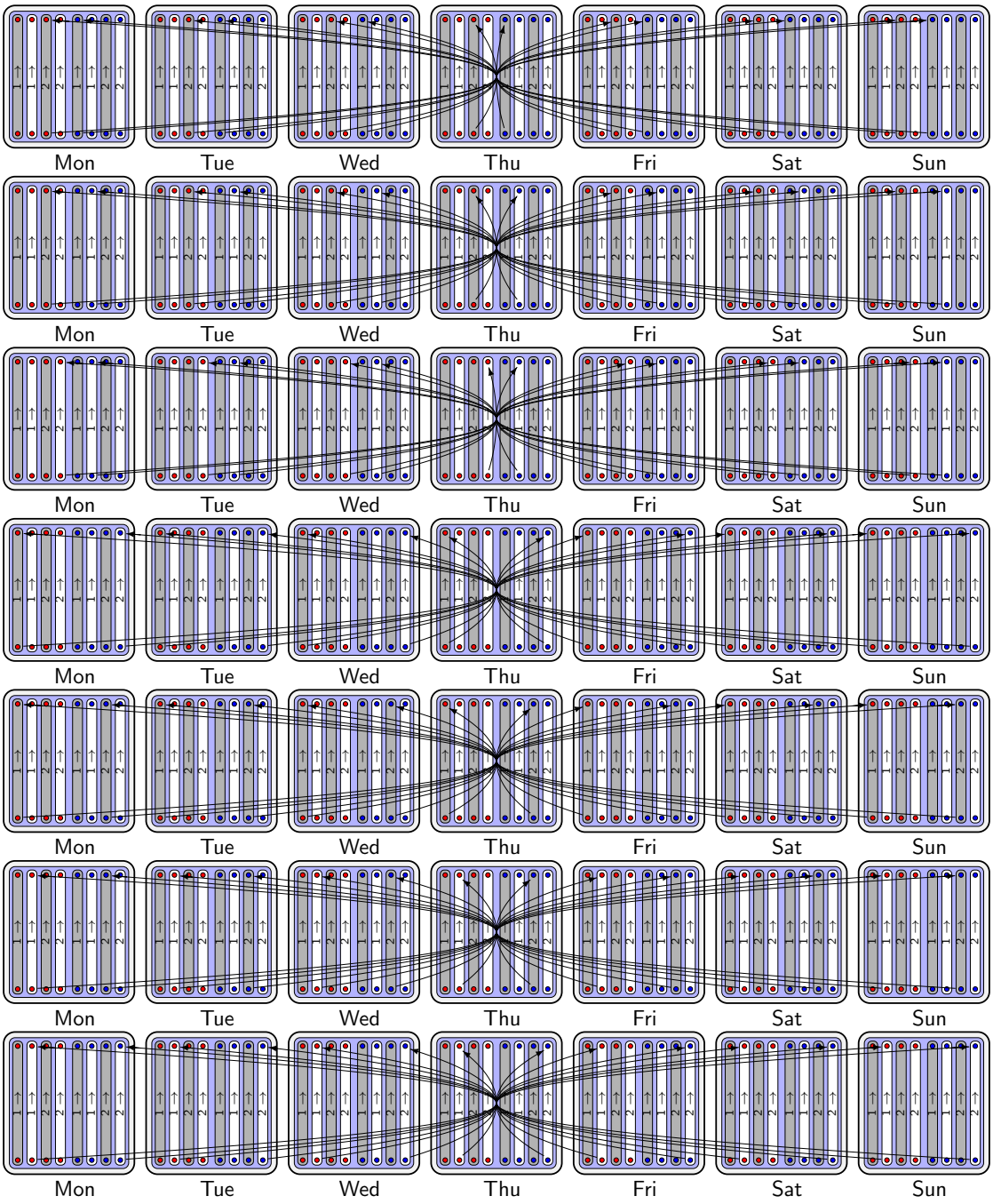


Figure 4.22.: Hyperarcs modeling regularity for timetabled trips, i.e., the operation of the trips of a train with equal vehicle compositions.

Assumption 5. *Let* $\mathfrak{t} \in \mathfrak{T}$ *be a train,* $h \in H(\mathfrak{t})$ *be a regularity hyperarc of* $\mathfrak{t}$*, and* $T(h) \subseteq T$ *be the set of trips that* h *operates. We assume:*

- ▷ *All trips of* $T(h)$ *are operated with exactly the same vehicle composition, and*
- ▷ $T(h) \in R(\mathfrak{t}) \cup \bar{R}(\mathfrak{t})$.

Assumption 5 excludes some situations to be recognized as rewarding regularity patterns. For example, we assume that all trips in Figure 4.17 form a train that is operated exclusively with vehicle configurations of size one. In this situation, it could be desired to reward the regularity pattern in that all trips are operated by the same vehicle orientation at all six departures of the train. For this pattern both characteristics of Assumption 5 do not apply, i.e., the vehicle configuration of the trips of $\{t_3, t_6\}$ are different from the ones taken for $\{t_1, t_2, t_4, t_5\}$ and the set $\{t_1, \ldots, t_6\}$ is not an element of the family $R(\mathfrak{t}) \cup \bar{R}(\mathfrak{t})$ as explained above.

Note that this pattern is not excluded by Assumption 5 in general. We still have regularity hyperarcs that model equal orientations at the departures of trips $\{t_3, t_6\}$ and $\{t_1, t_2, t_4, t_5\}$.

A situation that often arises in industrial instances is that a vehicle configuration of size two is allowed for the operation of the trips of a train. On weekdays size two is often mandatory, while on the weekend it is optional, i.e., a vehicle configuration of size one or of size two can be chosen. Then, $R(\mathfrak{t}) \cup \bar{R}(\mathfrak{t})$ has the form

$$R(\mathfrak{t}) \cup \bar{R}(\mathfrak{t}) = \{\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}, \{t_1, t_2, t_3, t_4, t_5\}, \{t_6, t_7\}\}$$

and equal vehicle compositions during weekdays and during the weekend are rewarded by the objective function. This is a major motivation for Assumption 5.

The remaining question is now, how to translate the quality of the regularity patterns modeled by regularity hyperarcs into the minimization of RotOR's linear objective function $c : H \mapsto \mathbb{Q}_+$ of the RSRP. In fact, a simple approach turns out to be transparent and sufficient enough in RotOR's industrial application at DBF.

To this end, we introduce a parameter $R_{\mathfrak{T}} \in \mathbb{Q}_+$ where its subscript tags the correspondence to regularity for trains. Then, the usage of each hyperarc $h \in H$ that operates a single timetabled trip but is not a regularity hyperarc, i.e.,

$$h \in \bigcup_{\mathfrak{t} \in \mathfrak{T}} \bigcup_{t \in \mathfrak{t}} H(t)$$

is penalized. To this end, $R_{\mathfrak{T}}$ is included in the objective coefficient $c(h)$, i.e., $c(h) - R_{\mathfrak{T}}$ exactly states the remaining proportion of the objective coefficient of h that is not influenced by regularity for trains.

Finally, let $h_{\mathfrak{t}} \in H(\mathfrak{t})$ be a regularity hyperarc for the train $\mathfrak{t} \in \mathfrak{T}$. We define the objective coefficient $c(h_{\mathfrak{t}})$ of $h_{\mathfrak{t}}$ by the objective coefficients of the individual hyperarcs that $h_{\mathfrak{t}}$ is composed of:

$$c(h_{\mathfrak{t}}) := \sum_{t \in \mathfrak{t},\, h \in H(\mathfrak{t}) \cap H(t)} (\, c(\, h\,) - R_{\mathfrak{T}}\,)\,.$$

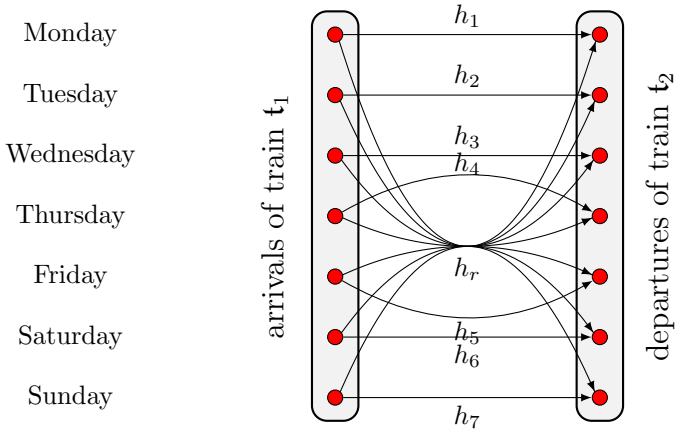


Figure 4.23.: Hyperarc model for regular turns.

By this cost structure it is more expensive to operate a timetabled trip with a hyperarc modeling an individual vehicle composition than to use a regularity hyperarc that follows a regularity pattern for the vehicle compositions for trips of a train.

During the industrial cooperation with DBF we discovered that it is more difficult to justify deviating vehicle compositions for larger subsets of trips of a train than for smaller ones. E.g., for a train with seven trips it is somehow better to operate three trips with an equal vehicle composition and four trips with another one than to operate two trips with different vehicle compositions and the remaining trips with a third vehicle composition. We conjecture that the reason behind this is related to the amount of information that is needed to communicate the vehicle compositions used for the trips of a train. This also motivates Assumption 5 since we consider inclusive maximal subsets of trips.

Moreover, it is interesting that the concrete value of the parameter $R_{\mathfrak{T}}$ is set to a relatively large value in industrial RSRP instances. It is in a range where it is paying for the use of additional turn around trips. This means, that additional turn around trips are accepted in operation in order to provide equal vehicle compositions for the trips of a train, which underlines the importance of the regularity pattern.

4.10.2. Regular Turns: Regularity for Connections

RotOR's second regularity pattern is not visible for the passengers. The pattern is to minimize the number of different connections that precede and succeed the trips of a train. For example, the connection of all trips of train 408 to trips of train 588 in Figure 4.20 is a desired pattern. We call these patterns *regular turns* because they cause repeating turns.

The modeling approach for this requirement is almost equal to the approach for regular trips, i.e., we handle regular turns by an appropriate construction of RotOR's hypergraph.

Figure 4.23 illustrates RotOR's regularity approach for connections. The circles on

the left for train $\mathfrak{t}_1 \in \mathfrak{T}$ are supposed to be arrivals of train $\mathfrak{t}_1 \subseteq T$ that are equal w.r.t. the location and time of day. The set of circles for train $\mathfrak{t}_2 \in T$ represent equal departures of trips of train $\mathfrak{t}_2$. The hyperarcs $\{h_1, \ldots, h_7\}$ are individual connections of the trips of the trains that are arranged as a desired regularity pattern: On each day of operation a trip of $\mathfrak{t}_1$ is connected to a trip of $\mathfrak{t}_2$. In order to have control about this pattern we additionally introduce the regularity hyperarc

$$h_r := \bigcup_{i=1}^{7} h_i,$$

which exactly models the desired situation in Figure 4.23.

Naturally, RotOR's hypergraph contains much more hyperarcs that model turns than those that model the operation of timetabled trips. In addition, regular turns are of less importance in comparison to regular trips. Therefore, the construction and the treatment in RotOR's objective function of regular turns is slightly different from the approach that we make for regular trips in the previous Section 4.10.1. But, analogically to the previous section, the set of regularity hyperarcs that RotOR's implements is a compromise between the size of the hypergraph and the amount of regularity patterns to be rewarded by the objective function.

In order to identify regularity patterns for turns, we introduce an equivalence relation for hyperarcs that connect timetabled trips by turns. Let $h_1, h_2 \in H$ be two hyperarcs that connect timetabled trips (and are not already regularity hyperarcs themselves). We define that h_1 is *turn equivalent* to h_2 if there exist two trains $\mathfrak{t}_1, \mathfrak{t}_2 \in \mathfrak{T}$ such that all of the following conditions hold:

- ▷ Both, h_1 and h_2, connect the arrival of a trip of train $\mathfrak{t}_1$ to the departure of a trip of train $\mathfrak{t}_2$,
- ▷ both, h_1 and h_2, model exactly the same vehicle compositions, and
- ▷ the duration between the arrival and the departure that h_1 connects is equal to the duration between the arrival and the departure that h_2 connects.

This equivalence relation for hyperarcs naturally induces equivalence classes of hyperarcs, which we translate directly into RotOR's regularity hyperarcs:

Assumption 6. *Let $h \in H$ be a regularity hyperarc that is composed of $2 \leq m \leq 7$ individual hyperarcs $h_1, \ldots, h_m \in H$, i.e., $h = \bigcup_{i=1}^{m} h_i$. We assume that $\{h_1, \ldots, h_m\}$ is an equivalence class w.r.t. turn equivalence, i.e., all hyperarcs of $\{h_1, \ldots, h_m\}$ are pairwise turn equivalent and m is maximal.*

By Assumption 6 our model is restricted to hyperarcs that model a maximal set of individual connections, e.g., we do not create a hyperarc that is the union of $h_2, \ldots, h_6$ w.r.t. the situation in Figure 4.23. Note that this does not mean that RotOR's solution can not contain the pattern modeled by this subset.

The objective function for regularity hyperarcs for regular turns is handled in the same way as in Section 4.10.1 for regular trips. We introduce a parameter $R_{\mathfrak{T}\times\mathfrak{T}} \in \mathbb{Q}_+$ (again,

the subscript indicates the correspondence to regular turns) that is responsible for the amount of regular turns in a solution. In view of Figure 4.23 we penalized each of the individual hyperarcs $h_1, h_2, \ldots, h_7$ by the parameter $R_{\mathfrak{T}\times\mathfrak{T}}$ and compute the cost for h_r as

$$c(h_r) := \sum_{i=1}^{7} (c(h_i) - R_{\mathfrak{T}\times\mathfrak{T}}).$$

Depending on the value of the parameter $R_{\mathfrak{T}\times\mathfrak{T}}$ a solution contains more or less regularity hyperarcs for regular turns.

In the industrial instances of DBF the concrete value of $R_{\mathfrak{T}\times\mathfrak{T}}$ is typically set to be minimally affective. Indeed, its value is much smaller than the corresponding penalty for regular trips from the previous section, i.e.,

$$R_{\mathfrak{T}\times\mathfrak{T}} \ll R_{\mathfrak{T}}.$$

In addition, irregular turns are typically least penalized in the industrial RSRP instances (in comparison to the other terms of RotOR's objective function). Nevertheless, we observed that this setting provides results which are accepted by the planners of DBF.

We want to point out explicitly that by using a model completely without regularity, we find connections between whole trains only very rarely and already a low value of $R_{\mathfrak{T}\times\mathfrak{T}}$ strongly increases the amount of regularity. The low value of $R_{\mathfrak{T}\times\mathfrak{T}}$ might appear as a contradiction to the high importance of regularity for turns that we claimed. We suppose that the reason for this is that there are many solutions for a dedicated RSRP instance with almost equal objective values when $R_{\mathfrak{T}\times\mathfrak{T}} = 0$.

4.10.3. Regular Handouts: Regularity for Visualization

As already mentioned, Figure 4.20 is an example for the visualization of rolling stock rotations. Those visualizations are made in order to communicate conveniently rolling stock rotations in further planning steps. Interestingly, visualizations for rolling stock rotations made for the standard week are (more or less) standardized across many railway companies from different (European) countries. For example, Nederlandse Spoorwegen from the Netherlands, the Österreichische Bundesbahn from Austria, Trenitalia from Italy, and, of course, DBF from Germany is using the visualization concept that we call *handout concept* here. Although, the terminology among these companies is not standardized, the methodology is exactly the same.

Despite the standardization, there exist different visualizations for a dedicated rolling stock rotation in the handout concept. The quality of a visualization is measured by the regularity patterns that are visible. Indeed, the regularity patterns for trips and turns, which we discussed in the past two sections, influence the visualization quality. But these patterns are not completely sufficient for a high-quality visualization that is required by the railway industry.

We assume here that each rolling stock rotation is visualized separately. This assumption does not completely hold in industry, i.e., sometimes a whole set of cycles is meant by a rotation (i.e., "Umlauf") and also the whole set of cycles is visualized together. If

several cycles are identified as a "rotation" in industry, a single cycle is called *sub-rotation* (i.e., "Teilumlauf"). The explicit consideration of sub-rotations complicates notation and does not contribute here. Therefore, we assume that we are given a single rolling stock rotation to be handed out in this section.

By construction, each rolling stock rotation runs an integral number of times through the standard week. We denote this number by $\mathfrak{v} \in \mathbb{Z}_+$ for the rolling stock rotation that we want to visualize. The number of railway vehicles needed to operate the rotation is also equal to $\mathfrak{v}$, i.e., $\mathfrak{v}$ railway vehicles run through the rotation one by one.

For the visualization, we imagine a rolling stock rotation as a cycle that runs through timetabled trips as illustrated in Figure 4.24. In order to find a convenient visualization the rotation is split into $7 \cdot \mathfrak{v}$ *segments*. A segment represents the operation of a rolling stock rotation on a single day of operation. In this way, the rolling stock rotation with $\mathfrak{v} = 2$ in Figure 4.24 decomposes into $7 \cdot \mathfrak{v} = 14$ segments. A segment may not contain any timetabled trip, i.e., can also be empty.

Let[8] $\mathcal{S}$ be the set of segments of the rolling stock rotation to be visualized and let $\mathbb{D}(s) \in \{\text{Mon}, \ldots, \text{Sun}\} =: \mathbb{D}$ denote the day of operation of the segment $s \in \mathcal{S}$ and the set of days of operation, respectively. In addition, we denote by $[\mathfrak{v}] := \{k \in \mathbb{N} \mid k \leq \mathfrak{v}\}$ the set of the first $\mathfrak{v}$ natural numbers. A *handout* is a map $\Omega : \mathcal{S} \mapsto [\mathfrak{v}]$ such that

$$\mathbb{D}(s) = \mathbb{D}(t) \quad \Rightarrow \quad \Omega(s) \neq \Omega(t) \quad \forall\, s, t \in \mathcal{S},$$

i.e., Ω assigns different values of $[\mathfrak{v}]$ to each pair of segments that are both associated with the same day of operation.

Always exactly $\mathfrak{v}$ segments of a rolling stock rotation are associated with the same day of operation. Thus, if a handout $\Omega : \mathcal{S} \mapsto [\mathfrak{v}]$ is at hand each segment $s \in \mathcal{S}$ can be precisely identified by its $\Omega(s)$ and its $\mathbb{D}(s)$. This is an evident motivation for the concept of handouts. For most of the rolling stock rotations in industry $\mathfrak{v}$ is much greater than two. Indeed, rolling stock rotations with $\mathfrak{v} > 50$ are not an exception at DBF. For those rotations a handout provides obviously a gain, i.e., segments can now be distinguished in all further planning steps.

The major objective of a handout $\Omega : \mathcal{S} \mapsto [\mathfrak{v}]$ is to create the standardized visualization for a rolling stock rotation. Indeed, $\Omega : \mathcal{S} \mapsto [\mathfrak{v}]$ completely defines this visualization. The visualization appears by printing all segments one below the other such that they are lexicographically sorted according to Ω and the day of operation.

Figure 4.25 illustrates two different handouts, namely their prints, for the rolling stock rotation of Figure 4.24. In both prints the first three columns state the value $\Omega(s)$ for a segment $s \in \mathcal{S}$ associated with $\mathbb{D}(s)$. The underlying rolling stock rotation is *not* modified by those visualizations, i.e., the successor relations of the segments that are defined by the rolling stock rotation remain. These successor relations are stated in the last columns Ω_{next} of the two prints. The function $\Omega_{\text{next}} : \mathcal{S} \mapsto [\mathfrak{v}]$ directly arises from Ω and the successor relations, i.e., if the segment $t \in \mathcal{S}$ directly follows after the segment $s \in \mathcal{S}$ along the rolling stock rotation we have $\Omega_{\text{next}}(s) := \Omega(t)$.

[8] We apologize for using the symbol $\mathcal{S}$ again, which is also used to denote a set of stations, e.g., in Section 4.3.

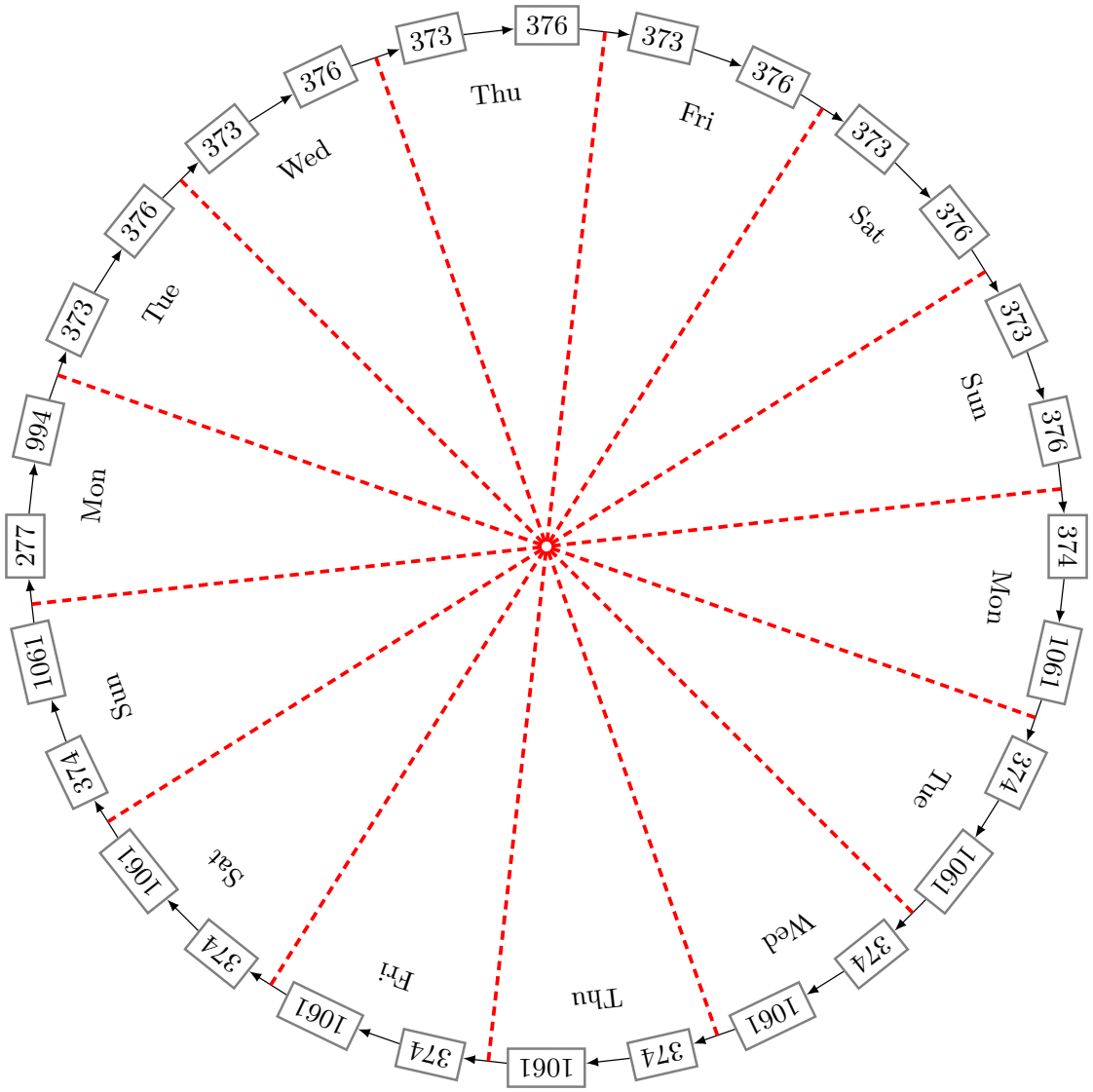


Figure 4.24.: Splitting of a rolling stock rotation with $\mathfrak{v} = 2$ into 14 segments.

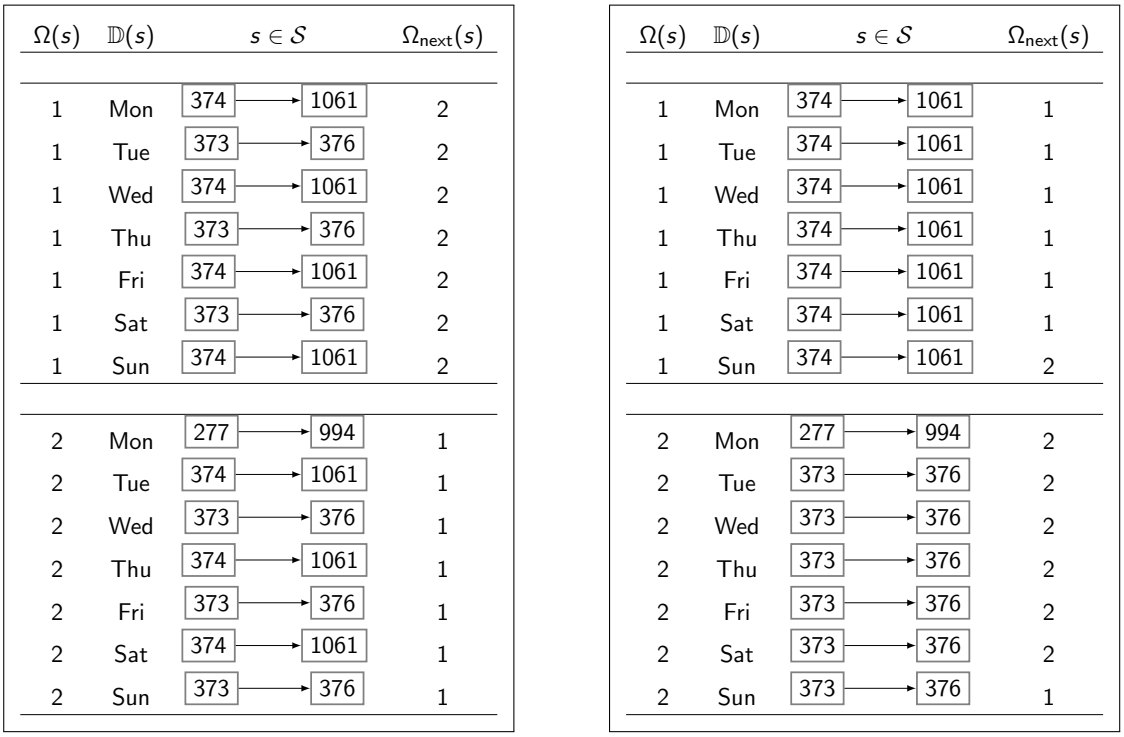


Figure 4.25.: Two different handouts for the rolling stock rotation of Figure 4.24.

The rolling stock rotation that is visualized by a handout Ω can now be followed along a print. For example, for the successor segment $t \in \mathcal{S}$ of the segment $s \in \mathcal{S}$ with $\Omega(s) = 2$ and $\mathbb{D}(s) = \text{Sun}$ on the left of Figure 4.25 we know that $\Omega_{\text{next}}(s) = \Omega(t) = 1$ and that $\mathbb{D}(t) = \text{Mon}$ from the visualization. In this way the successor segment t can be easily found and we are able to double-check that both handouts in Figure 4.25 visualize the rolling stock rotation of Figure 4.24.

The function Ω is extensively distributed as a planning tool in the railway industry (but it is not called Ω there). In particular, the manual planning of rolling stock rotations at DBF is based on it. There, timetabled trips of T are equipped with values of $[\mathfrak{v}]$. Thus, it is not surprising that the function Ω has some expectations in the railway industry. For example, for all segments $s \in \mathcal{S}$ arranged in the left of Figure 4.25 holds

$$\Omega_{\text{next}}(s) = (\,\Omega(s) \mod \mathfrak{v}\,) + 1. \tag{4.3}$$

If a successor relation follows (4.3) we call it *logical turn*. It is desired to have as much logical turns as possible in a handout in order to obtain a visualization in that the rolling stock rotations can be followed easily. It is notable that it is not always possible to find a handout such that all successor relations are logical turns. If we consider a handout with $\mathfrak{v} = 7 \cdot k$ for some $k \in \mathbb{Z}_+$ such that it maximizes the number of logical turns (i.e., $7 \cdot \mathfrak{v}$), we recognize (by following the logical turns) that the handout visualizes seven individual sub-rotations. In this way, a single rotation (without sub-rotations) with $\mathfrak{v} = 7 \cdot k$ for

$k \in \mathbb{Z}_+$ can not have the maximal number of logical turns.

By definition, Ω induces a natural partition of the segments of the rotation into *blocks*. All segments $s \in B$ of a block $B \subseteq \mathcal{S}$ have the same $\Omega(s)$ but a different day of operation. Therefore, each block is of cardinality seven and there are always $\mathfrak{v}$ blocks induced by a handout. In the left of Figure 4.25 the first seven segments $B \subset \mathcal{S}$ with $\Omega(s) = 1$ for all $s \in B$ form a block[9].

Another desired property (if not the most important) is related to the blocks of a handout. For example, in the right of Figure 4.25 we recognize that the first seven segments, (i.e., the segments of the first block) are all equal. We say that two segments $u, v \in \mathcal{S}$ with $\Omega(s) = \Omega(t)$ have a *similarity* if they cover trips of the same train. In reality, some more details are taken into account for the quantification of similarities but those details do not affect the presentation here. It is desired to have similarities in one block as much as possible. The two patterns discussed in the past two sections are somehow the preparation for the appearance of many similarities in blocks, e.g., the connection of trips of train 374 to trips of train 1061 is reflected in the handout on the right of Figure 4.25.

The desired properties logical turns and similarities compete. A handout that maximizes similarities may not maximize the number of logic turns as well and vice versa, see Figure 4.25. For a better understanding we present a program for the optimization of similarities as well as logical turns of handouts. To this end, we introduce a binary decision variable $z_{s\omega} \in \{0, 1\}$ that takes value one if $\Omega(s) = \omega$ for the segment $s \in \mathcal{S}$ and is zero otherwise.

In order to qualify handouts we denote the number of similarities of $s_1 \in \mathcal{S}$ and $s_2 \in \mathcal{S}$ by $\mathrm{SI}(s_1, s_2) \in \mathbb{Z}_+$. In addition, we introduce the parameter $\mathrm{LO}(s_1, s_2) \in \{0, 1\}$ where $\mathrm{LO}(s_1, s_2) = 1$ if and only of $s_2 \in \mathcal{S}$ is the direct successor of $s_1 \in \mathcal{S}$ within the rolling stock rotation.

Under this notation, a program for the optimization of handouts reads:

$$\max \sum_{\substack{s_1 \in \mathcal{S} \\ s_2 \in \mathcal{S} \\ s_1 \neq s_2}} \left(\sum_{\omega \in [\mathfrak{v}]} \mathrm{SI}(s_1, s_2)\, z_{s_1\omega}\, z_{s_2\omega} + \sum_{\substack{\omega_1 \in [\mathfrak{v}],\, \omega_2 = \\ (\,\omega_1 \bmod \mathfrak{v}\,)+1}} \mathrm{LO}(s_1, s_2)\, z_{s_1\omega_1}\, z_{s_2\omega_2} \right) \tag{SILO}$$

$$\sum_{\omega \in [\mathfrak{v}]} z_{s\omega} = 1 \qquad \forall\, s \in \mathcal{S}, \tag{4.4}$$

$$\sum_{\substack{s \in \mathcal{S} \\ \mathbb{D}(s) = d}} z_{s\omega} = 1 \qquad \forall\, d \in \mathbb{D},\, \omega \in [\mathfrak{v}], \tag{4.5}$$

$$z_{s,\omega} \in \{0, 1\} \quad \forall\, s \in \mathcal{S},\, \omega \in [\mathfrak{v}].$$

Equalities (4.4) and (4.5) of program (SILO) constrain the binary z-variables to perfectly match all segments of $\mathcal{S}$ to pairs of $\mathbb{D} \times [\mathfrak{v}]$, i.e., to form an assignment in a bipartite

[9] At DBF a block is called *Umlauftag*. This translates to the English term "rotation day" or "day of rotation", which we do not use here. The arrangement in Figure 4.20 on page 144 is what is called Umlauftag.

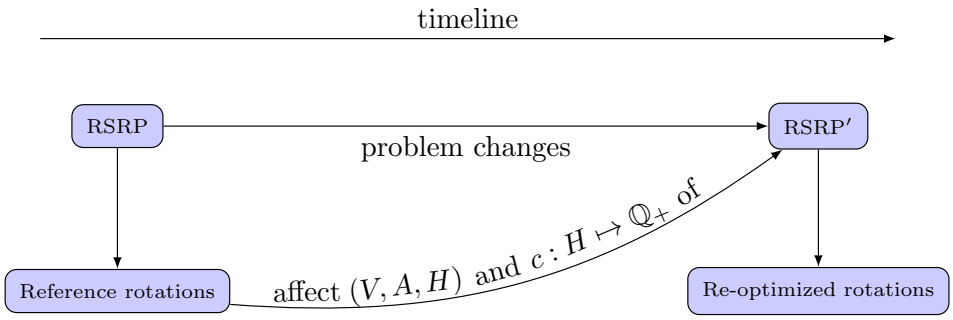


Figure 4.26.: Re-optimization setting.

graph that is composed of the two disjoint node parts $\mathcal{S}$ and $\mathbb{D} \times [\mathfrak{v}]$. Thus, any feasible solution to program (SILO) defines a handout Ω with $\Omega(s) = \omega$ if and only if $z_{s\omega} = 1$.

The objective function of program (SILO) qualifies each feasible solution according to similarities and logical turns. This is made by multiplying the corresponding parameters SI and LO to quadratic terms. Hence, program (SILO) is a non-linear formulation, more precisely the quadratic assignment problem (QAP) is an extension of program (SILO). The QAP is $\mathcal{NP}$-hard and problem instances of it can be hard to solve to proven optimality, see Burkard, Dell'Amico, and Martello (2009, [33, Chapter 7]). Note, that this is not a proof that program (SILO) is $\mathcal{NP}$-hard as well but we notify a close relation to the QAP.

We finally remark here that we supervised the Bachelor's thesis of Dod (2010, [41]) in the context of handout optimization. This thesis deals with handout optimization by a different terminology and presents particularly our construction heuristic for handout optimization, see Section 6.7.

4.11. Re-optimization by Rotation Templates

In this section we focus on RotOR's *re-optimization* capability that is frequently required in rolling stock rotation optimization. RotOR's re-optimization approach is published in the papers [9] and [10], where [9] $\subset$ [10] holds.

The re-optimization setting for the RSRP arises in the following situation. At some point in time a railway undertaking has to tackle an instance of the RSRP and constructs rolling stock rotations. We call this planning step *greenfield planning* or *greenfield optimization*. At a later point in time conditions of and assumptions about the original instance change. Then, it is inefficient and often even not feasible to operate the originally constructed rolling stock rotations that we call *reference rotations*, see Figure 4.26.

The circumstances that lead to re-optimization scenarios are manifold. Examples are:

- ▷ construction sites (see Section 7.5),
- ▷ technical failures,
- ▷ accidents, and

- ▷ strikes[10].

In such situations, a new RSRP′ has to be solved. In the RSRP′ all requirements of the original RSRP have to be taken into account. The main difference to greenfield planning is that the reference rotations were already completely or partially implemented in operation. Crew was scheduled for vehicle operations and maintenance services, capacity consumption of parking areas was reserved, and most important railway tracks were already allocated for the deadhead trips of the reference rolling stock rotations. Hence, a major goal in constructing a solution to the RSRP′ is to change as little as possible in comparison to the reference rotations. This is called *re-optimization*.

Re-optimization problems have received a huge amount of attention in the literature, e.g., see Haahr et al. (2014, [59]) and Budai et al. (2010, [32]) for rolling stock applications, Secomandi and Margot (2009, [96]) for vehicle routing with stochastic demands, and Huisman (2007, [65]) as well as Borndörfer et al. (2013, [28]) for crew scheduling applications.

When it comes to re-optimization almost everything about the original scenario can change. Rather typical examples are:

- ▷ the number of turn around events during timetabled trips can change,
- ▷ the set of allowed vehicle compositions of a timetabled trip can change,
- ▷ timetabled trips can be shortened, enlarged, or canceled,
- ▷ new timetabled trips can appear,
- ▷ fleet capacities can change, or
- ▷ new fleets may have to be introduced.

This large variety of possible changes makes it hard to insist on maintaining any particular properties of the reference rotations. A major question in re-optimization applications is therefore to identify appropriate structures in the reference solutions that can be recognized after re-optimization. An obvious candidate in rolling stock rotation optimization is a dedicated connection between two timetabled trips in a reference rotation. To this end, we introduce *connection templates*, which we use in order to configure the objective function for re-optimization towards the connections included in the reference rotations.

In our application, however, it turns out that it is not enough to re-optimize on the basis of such "local" templates. To this end, we introduce additional *rotation templates*, which we announce as the main contribution of RotOR's re-optimization approach.

RotOR's re-optimization capability is designed to tackle the RSRP and the RSRP′ in one go, i.e., we consider the RSRP′ just as a special RSRP. The specialized parts are the shape of the hypergraph $(V \cup S, A, H)$ and the objective function $c : H \mapsto \mathbb{Q}_+$, see Figure 4.26. Connection templates are described in Section 4.12.2 (just after the presentation of the objective function for greenfield optimization) while the rotation templates are the subject of the current section.

[10] See Ahmadi et al. (2015, [14]) for an application of RotOR in days of strike.

Rotation templates. A rotation template is a set of timetabled trips that are all contained in a reference rotation. In order to motivate the idea of rotation templates we consider a single reference rolling stock rotation as a cycle that covers a subset of timetabled trips $T_{\text{ref}} \subseteq T$ (and possibly more in its associated original RSRP). We assume that the set T_{ref} is contained in (the timetable of) the RSRP′, i.e., re-optimization scenario. A seemingly "small" change to this reference rotation is to interchange two connections of timetabled trips, i.e., if $t_1 \in T_{\text{ref}}$ is connected to $t_2 \in T_{\text{ref}}$ and $t_3 \in T_{\text{ref}}$ is connected to $t_4 \in T_{\text{ref}}$ in the reference rotation we might consider to connect t_1 to t_4 and t_2 to t_3 in a solution for the re-optimization scenario. Even if all other connections between timetabled trips remain unchanged, this exchange results in a split of the reference rotation into two completely independent rotations. This is not desired in industrial practice. In fact, an important "non-local" requirement refers to the whole set T_{ref} of timetabled trips of a dedicated reference rotation. Namely, it is generally desired that most of the timetabled trips of T_{ref} remain together in a rolling stock rotation *after* re-optimization. The purpose of RotOR's *rotation templates* is to provide control about the distribution of the timetabled trips among the rotations produced by re-optimization.

RotOR's implementation of this purpose works as follows. As already introduced, we denote by F the set of fleets that is available for the operation of the timetable of the re-optimization scenario. Let R be the set of rolling stock rotations that appear in the reference rotations. The idea is to "refine" the set of fleets F into a larger set of fleets such that its elements can be distinguished by the reference rotations R. To this end, we are given a non-empty set of *refined fleets* $F_R(f)$ for each (original) fleet $f \in F$ in the input data for the re-optimization scenario. The set $F_R(f)$ is created in an appropriate way, e.g., if three reference rotations are operated by railway vehicles of the fleet $f \in F$ we would create at least three refined fleets in $F_R(f)$ that correspond to the reference rotations operated by f. Note that it might also be of practical interest to create refined fleets for $f \in F$ that correspond to reference rotations that were not operated by f before re-optimization. This is an exceptional case, which is made, e.g., to allow to only change the fleet of a reference rotation. In this way, it is possible to consider all elements of the Cartesian product $F \times R$ as refined fleets. But the typical case is to refine $f \in F$ to as many refined fleets as reference rotations are operated by f. Therefore, we assume $\left|\bigcup_{f \in F}(F_R(f))\right| = |R|$ in the following.

The (refined) re-optimization hypergraph G_R of RotOR is built in a (more or less) straightforward way on the basis of the refined fleets. All rules and requirements for the original case directly carry over to the refined hypergraph that can be denoted as

$$G_R = \left(\bigcup_{r \in R} (V_r \cup S_r), \bigcup_{r \in R} A_r, H_R \right).$$

The nodes and *standard arcs* of G_R decompose into independent *standard graphs* $(V_r \cup S_r, A_r)$ for each reference rotation $r \in R$. By construction, each such graph $(V_r \cup S_r, A_r)$ is an exact one-to-one copy of nodes and arcs that we consider in the case without reference rotations.

The nodes as well as the standard arcs of a hyperarc can now be distinguished in terms

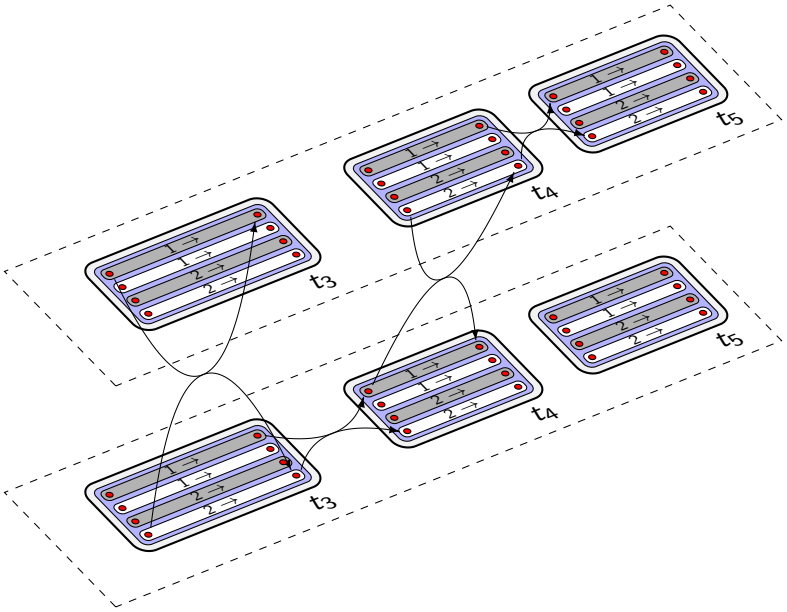


Figure 4.27.: Refined hyperarcs of the re-optimization hypergraph.

of reference rotations. E.g., traversing $v \in V_r$ has the meaning of traversing a node in a rolling stock rotation that corresponds to the reference rotation $r \in R$. By choosing appropriate objective function coefficients we gain full control over the distribution of the timetabled trips among the reference rotations in a solution to the re-optimization scenario, see Section 4.12.2.

The crux of G_R is that its hyperarcs H_R do not refine as easy as the nodes and standard arcs do. The hyperarcs of RotOR's original hypergraph particularly model movements of vehicle configurations, which are multisets of original fleets. Now, if we refine the original fleets, we, consequently, have to also refine the set of original vehicle configurations, which movements are modeled by hyperarcs H_R of G_R. This is much more complicated.

For example, consider a hyperarc $h = \{a_1, a_2\} \in H$ of the original hypergraph and assume that $F(a_1) \in F$ and $F(a_2) \in F$ denote the original fleets that the standard arcs $a_1, a_2 \in h$ model, i.e., $\{F(a_1), F(a_2)\}$ is the vehicle configuration of h. Now, in the re-optimization scenario we have $|F_R(F(a_1))|$ alternative refined fleets for a_1 as well as $|F_R(F(a_2))|$ for a_2. The crucial point is that it is necessary to consider all multisets of size two of elements (i.e., fleets) of $F_R(F(a_1)) \cup F_R(F(a_2))$ because an optimal distribution of the railway vehicles that are coupled together at h among the refined fleets is not known a priori. In fact, in a solution to the re-optimization scenario it could be efficient to perform the movement of h by two coupled railway vehicles with *equal* refined fleets. This already gives $|F_R(F(a_1)) \cup F_R(F(a_2))|$ alternatives. Moreover, it could also be even optimal to (arbitrarily) take two *different* refined fleets from $F_R(F(a_1)) \cup F_R(F(a_2))$ in order to refine h.

By Figure 4.27, which is based on Figure 4.17, we illustrate the situation of the refined

hyperarcs in the re-optimization hypergraph G_R. Suppose that the original hypergraph $(V \cup S, A, H)$ is illustrated by one of the two layers. Further, assume that two reference rotations for the red fleet are given. As explained, the nodes of $(V \cup S, A, H)$ become refined, i.e., just copied. Obviously, these copies are represented by the two layers that lie on top of each other in Figure 4.27. We also assume that we want to operate the timetabled trips t_3, t_4, and t_5 by two coupled railway vehicles of the red fleet. Now, in the re-optimization scenario, we have to decide from which refined fleet (which corresponds to a reference rotation for the original red fleet) the two individual railway vehicles originate. This translates to hyperarcs as is exemplarily shown in Figure 4.27. It is possible to operate the individual railway vehicles by vehicles of the same refined fleets. This situation is implied if one chooses the hyperarc that connects t_3 to t_4 (see the layer below) or t_4 to t_5 (see the layer above) in a solution. But, as can be seen by the hypearcs that operate t_3 and t_4 in Figure 4.27, it is naturally also possible to take vehicles from different refined fleets.

Thus, an original hyperarc may be refined to many hyperarcs of the refined hypergraph. The refinement of the original hyperarcs H to H_R of the re-optimization hypergraph G_R directly derives from the refinement of the vehicle configurations. But, it turns out that the number of vehicle configurations increases–dramatically.

In order to illustrate this blow-up, we consider a vehicle configuration that has not been refined yet, i.e., a multiset of elements of F, by using the following notation:

$$\{f_1^1, f_1^2, \ldots, f_1^{m_1}, f_2^1, f_2^2, \ldots, f_2^{m_2}, \ldots, f_n^1, f_n^2, \ldots, f_n^{m_n}\}. \tag{4.6}$$

The multiset (4.6) denotes a single vehicle configuration of size $\sum_{i=1}^{n} m_i$ that is allowed for the operation of a dedicated timetabled trip of T. In this notation we assume $n \in \mathbb{Z}_+$ and that two fleets $f_i^{\cdot}, f_j^{\cdot} \in F$ have different subscripts $i, j \in \{1, \ldots, n\}$ if and only if they identify different fleets of F. Therefore, the vehicle configuration (4.6) is composed of n different fleets where multiple appearances of a dedicated fleet are distinguished by the superscripts. In this way, fleet $f_i^{\cdot}$ appears exactly $m_i \in \mathbb{Z}_+$ times in the vehicle configuration, $i \in \{1, \ldots, n\}$.

In the re-optimization model we consider $F_R(f_i^{\cdot})$ as the new (i.e., refined) fleets for the original fleet $f_i^{\cdot} \in F$ that appeared m_i times in the original vehicle configuration (4.6). Already for this m_i appearances we have

$$\binom{|F_R(f_i^{\cdot})| + m_i - 1}{m_i} = \frac{(|F_R(f_i^{\cdot})| + m_i - 1)!}{m_i!(|F_R(f_i^{\cdot})| - 1)!} \tag{4.7}$$

possibilities to form a (refined) multiset from the elements of the set $F_R(f_i^{\cdot})$. The Formula (4.7) denotes the number of possibilities to choose m_i elements from a base set of size $|F_R(f_i^{\cdot})| + m_i - 1$. The base set can be interpreted as the refined fleets $F_R(f_i^{\cdot})$ plus $m_i - 1$ artificial elements that indicate the multiple choice of an already taken element. As already mentioned, all of this (refined) multisets have to be considered because an optimal distribution of the railway vehicles that are coupled together while operating a timetabled trip among the refined fleets is not known a priori.

Formula (4.7) only denotes the blow-up for a single original fleet (contained in an original vehicle configuration). The final number of refined vehicle configurations for the original vehicle configuration (4.6) is

$$\prod_{i=1}^{n} \binom{|F_R(f_i)| + m_i - 1}{m_i}. \tag{4.8}$$

In the RSRP applications at DBF, a typical re-optimization case involves $|F_R(f)| = 7$ refined fleets for a dedicated original fleet $f \in F$. Let $f = \text{Red}$ be the original fleet and let $F_R(\text{Red}) = \{\text{Red}_1, \text{Red}_2, \text{Red}_3, \text{Red}_4, \text{Red}_5, \text{Red}_6, \text{Red}_7\}$. Then, the number of possible (refined) vehicle configurations for the original vehicle configuration $\{\text{Red}, \text{Red}\}$ increases to not less than 28. In this situation, the refined vehicle configurations for $\{\text{Red}, \text{Red}\}$ read:

$$\begin{array}{llll}
\{\text{Red}_1, \text{Red}_1\}, & \{\text{Red}_2, \text{Red}_2\}, & \{\text{Red}_3, \text{Red}_4\}, & \{\text{Red}_4, \text{Red}_7\}, \\
\{\text{Red}_1, \text{Red}_2\}, & \{\text{Red}_2, \text{Red}_3\}, & \{\text{Red}_3, \text{Red}_5\}, & \{\text{Red}_5, \text{Red}_5\}, \\
\{\text{Red}_1, \text{Red}_3\}, & \{\text{Red}_2, \text{Red}_4\}, & \{\text{Red}_3, \text{Red}_6\}, & \{\text{Red}_5, \text{Red}_6\}, \\
\{\text{Red}_1, \text{Red}_4\}, & \{\text{Red}_2, \text{Red}_5\}, & \{\text{Red}_3, \text{Red}_7\}, & \{\text{Red}_5, \text{Red}_7\}, \\
\{\text{Red}_1, \text{Red}_5\}, & \{\text{Red}_2, \text{Red}_6\}, & \{\text{Red}_4, \text{Red}_4\}, & \{\text{Red}_6, \text{Red}_6\}, \\
\{\text{Red}_1, \text{Red}_6\}, & \{\text{Red}_2, \text{Red}_7\}, & \{\text{Red}_4, \text{Red}_5\}, & \{\text{Red}_6, \text{Red}_7\}, \\
\{\text{Red}_1, \text{Red}_7\}, & \{\text{Red}_3, \text{Red}_3\}, & \{\text{Red}_4, \text{Red}_6\}, & \{\text{Red}_7, \text{Red}_7\}.
\end{array}$$

All of these refined vehicle configurations are, indeed, considered explicitly when RotOR computes. Therefore, our modeling trick has a dramatic impact on the size of the arising hypergraphs as well as on the fractionality of corresponding solutions of linear programming (LP) relaxations.

However, it turns out that these disadvantage are often mitigated substantially by the information gain that the reference solution provides, in particular when large parts do not have to be changed, which is the usual case in industry. Indeed, the resulting increase in integrality is completely paying for the increase in size, see the computational study in Borndörfer et al. (2015, [10])[11]. Of course, one must be able to deal with such very large scenarios in the first place. The algorithmic key technology that allows this is the C2F hyperarc generation method of Chapter 2.

Relation to duty scheduling templates. A similar template concept for re-optimizing duty schedules is proposed in Borndörfer et al. (2013, [28]). The main idea is to define *duty scheduling templates*, which are able to model "non-local" requirements such as the (more or less detailed) distribution of breaks in duties. To this end, a pricing problem with a dedicated graph is solved for each duty type template individually. The individual graphs allow to model re-scheduling requirements and can be seen as copies of some original graph. Then, these copies are modified in order to provide a template for some "non-local" structures. In this sense, rotation templates and duty scheduling templates are related.

[11] RotOR's template approach including its implementation is a contribution of this thesis. The computational study called "The Benefit of Templates" [10] was made by Boris Grimm.

4.12. Objective Function

This section deals with RotOR's objective function $c : H \mapsto \mathbb{Q}_+$ that is defined to map hyperarcs to rational numbers. The RSRP that RotOR is dedicated to solve is to minimize the function c. The objective value $c(H^\star)$ of a solution $H^\star \subseteq H$ to the RSRP is calculated as $c(H^\star) := \sum_{h \in H^\star} c(h)$. Indeed, we only need hyperarcs with their objective function and nothing more to completely express the objective function for the RSRP. This is notable in the respect of the versatile requirements presented in the preceding part of this chapter, whereas a linear sum objective function is pretty much standard for many combinatorial optimization problems.

We are able to configure RotOR's objective function c in such a way that it is able to model almost all desires both for greenfield and for re-optimization too. The individual value that c assigns to the hyperarc $h \in H$ is of the following shape:

$$c(h) := \left\langle \begin{pmatrix} c_1(h) \\ c_2(h) \\ c_3(h) \\ c_4(h) \\ c_5(h) \\ c_6(h) \\ c_7(h) \\ c_8(h) \\ c_9(h) \\ c_{10}(h) \\ c_{11}(h) \\ c_{12}(h) \\ c_{13}(h) \\ c_{14}(h) \\ c_{15}(h) \end{pmatrix}, \begin{pmatrix} p_1(h) \\ p_2(h) \\ p_3(h) \\ p_4(h) \\ p_5(h) \\ p_6(h) \\ p_7(h) \\ p_8(h) \\ p_9(h) \\ p_{10}(h) \\ p_{11}(h) \\ p_{12}(h) \\ p_{13}(h) \\ p_{14}(h) \\ p_{15}(h) \end{pmatrix} \right\rangle \begin{matrix} \dots \text{ vehicles} \\ \dots \text{ trip distance} \\ \dots \text{ deadhead distance} \\ \dots \text{ turn duration} \\ \dots \text{ regular trips} \\ \dots \text{ regular turns} \\ \dots \text{ maintenance services} \\ \dots \text{ coupling activities} \\ \dots \text{ deviating configurations} \\ \dots \text{ deviating fleets} \\ \dots \text{ deviating rotations} \\ \dots \text{ deviating orientations} \\ \dots \text{ deviating positions} \\ \dots \text{ deviating connections} \\ \dots \text{ deviating maintenance services} \end{matrix} \begin{matrix} \} \text{ greenfield optimization} \\ \} \text{ re-optimization} \end{matrix} \tag{4.9}$$

As denoted, the objective function for $h \in H$ is composed of a vector of *cost coefficients* $c_i(h) \in \mathbb{Q}_+$ and a vector of *property values* $p_i(h) \in \mathbb{Q}_+$ for $i = 1, \dots, 15$ (for that we calculate the inner product) that correspond to 15 different objective features. Note that all objective features for greenfield optimization are also present in re-optimization scenarios. In the next two sections we discuss individual aspects of these objective features separated by greenfield and re-optimization.

4.12.1. Objective Function for Greenfield Optimization

The most important and, thus, typically largest term states the vehicle costs of a hyperarc. To this end, monetary vehicle cost are given per fleet. We handle these costs by computing the duration (see explanation around formula (4.1) on page 120) of a hyperarc and calculating the amount of vehicles w.r.t. the duration of the standard week. For example, if the duration of a standard arc $a \in A$ is $1/2$ days we have $p_1(\{a\}) = 1/14$

and the cost coefficient $c_1(\{a\})$ takes the value of the monetary cost of the fleet that a models. In case of a hyperarc $h \in H$ the values of $p_1(h)$ and $c_1(h)$ are computed as sums over the standards arcs that h includes. If $H^\star \subseteq H$ is a solution to the RSRP the number of vehicles that $H^\star$ takes is equal to $\sum_{h \in H^\star} p_1(h)$. Therefore the monetary vehicle costs are distributed over all hyperarcs of a solution.

The cost coefficients (c_2) for the operation of a timetabled trip are given per kilometer and per vehicle configuration–different vehicle configurations have different power consumptions, wear marks, and servicing cost. This cost coefficient is multiplied by the kilometrage (p_2) of a timetabled trip and, of course, only affects hyperarcs that cover timetabled trips.

The third objective feature for penalizing kilometrage of deadhead trips is tackled analogically but particularly worthy of emphasis is the structure of the cost coefficients. Suppose that we consider a hyperarc that connects the timetabled trip $t_1 \in T$ to $t_2 \in T$ by a large deadhead distance, e.g., 100 km, and much time, e.g., three hours, between the arrival of t_1 and the departure of t_2. Assume that the cost for allocating a deadhead trip per kilometer is 100 Euros for a vehicle configuration of size one and 120 Euros for a vehicle configuration of size two. We can express directly this cost structure by the cost coefficients c_3 for individual hyperarcs of our hypergraph because they distinguish different vehicle configurations. Two individual deadhead trips would increase the objective function by 20.000 Euros, while one coupled deadhead trip of both vehicles coupled together costs only 12.000 Euros. This reflects the track allocation cost for railway tracks, i.e., deadhead trips are cheaper for coupled vehicles than individual deadhead trips. The cost structure is the same for proper deadhead trips (that connect different locations) as well as additional turn around trips (that "connect" equal locations).

As explained in Section 4.2, we are given a minimum turn duration $d_{\min}$ and a planned turn duration d_{plan} for each turn that a dedicated hyperarc implements. As the cost coefficients for trip and deadhead distances also the turn durations depend on the vehicle configurations that are involved in turns. As already explained in Section 4.2, the turn duration deviation of a single turn is equal to $\max\{0, d_{\text{plan}} - d\}$. The property value $p_4(h)$ takes the sum over all turn duration deviations that the hyperarc $h \in H$ implements. All turn duration violations of a hyperarc $h \in H$ are then penalized by a constant cost coefficient ($c_4(h)$).

The fifth and sixth objective features are responsible for the amount of regular trips and regular turns that a solution contains. Our regularity model including the regularity parameters $R_{\mathfrak{T}}$ and $R_{\mathfrak{T}\times\mathfrak{T}}$ are explained in Sections 4.10.1 and 4.10.2. For the property value $p_5(h)$ we have $p_5(h) = 1$ if $h \in H$ is not a regularity hyperarc and h covers a timetabled trip. Similarly, $p_6(h) = 1$ if $h \in H$ is not a regularity hyperarc and h connects timetabled trips. For a regularity hyperarc $h \in H$ we set $p_5(h) = p_6(h) = 0$. The cost coefficients $c_5(h)$ and $c_6(h)$ take the values of the regularity parameters $R_{\mathfrak{T}}$ for regular trips and $R_{\mathfrak{T}\times\mathfrak{T}}$ for regular turns, respectively.

If and only if a hyperarc $h \in H$ implements a service path, see Section 4.5, that models a maintenance activity we simply have $p_7(h) = 1$. The costs $c_7(h)$ for maintenance activities are distinguished by the individual maintenance tasks that the maintenance service implements. These cost coefficients are also distinguished by the fleet of the

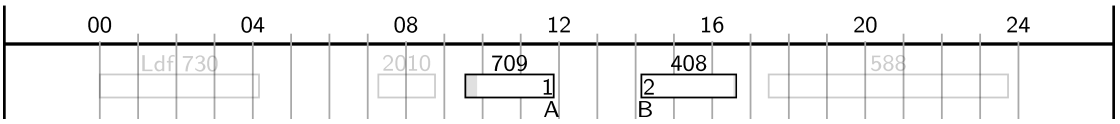


Figure 4.28.: Connection of a reference rotation performed by a railway vehicle that is contained in a vehicle configuration. The arrival is performed with position one in orientation Tick at location A. The vehicle departs with position two and orientation Tick at location B.

vehicle that is under maintenance.

In the railway industry it is desired to avoid unnecessary coupling and decoupling activities, see Section 4.9. We tackle this by the last objective feature for greenfield optimization. In our hypergraph we are able to recognize the number of coupling activities that a particular hyperarc $h \in H$ implements. For example, the hyperarc that connects trips t_3 and t_4 in Figure 4.17 does not implement any coupling activity, while the hyperarc that connects t_4 to t_5 implements a decoupling activity of the vehicle at position two from the vehicle at position one during the arrival of t_4. Finally, all coupling and decoupling activities of a hyperarc $h \in H$ are penalized by a constant cost coefficient ($c_8(h)$).

4.12.2. Objective Function for Re-optimization

As explained in Section 4.11, RotOR's mathematical model does not differ for greenfield and re-optimization of rolling stock rotations. We consider re-optimization for the RSRP simply as extended problem version in that we are additionally given a reference rotation plan. The major re-optimization requirement for the RSRP is to create a solution to the re-optimization scenario that differs from the reference rotation plan as little as possible. We handle this requirement by designing an appropriate hypergraph (see Section 4.11) with an appropriate objective function.

The configuration of the objective function for re-optimization is complex and the subject of this section. It is built on top of the objective function for greenfield optimization, i.e., the objective features 9 to 15 in Equation (4.9) are responsible for re-optimization. All of these seven objective features are derived from the connections of the trips in the reference rotations.

Connection templates. During re-optimization, these connections are considered as connection templates. The connection templates are prepared in a configuration routine that is called for a re-optimization scenario.

This configuration routine iterates all connections of trips of the reference rotations. Much data is associated with a connection between timetabled trips in rolling stock rotation planning. Figure 4.28 illustrates such a connection for a trip with train number 709 that is connected to a trip with train number 408. The preparation associates the following data with each connection template:

- ▷ arrival and departure times,

- ▷ arrival and departure locations,
- ▷ train identifiers at arrival and departure,
- ▷ fleet of the vehicle performing the connection,
- ▷ reference rotation in that the connection is performed,
- ▷ configurations at arrival, taken for the connection, and at departure,
- ▷ orientations at arrival and departure,
- ▷ positions at arrival and departure, and
- ▷ maintenance services performed during the connection.

Now, a connection template refers to all the details that constitute a connection of the reference rotations.

For a single connection template of the reference rotations it might not be beneficial to carry over all details in the solution to the re-optimization scenario. In addition, this can be impossible because some timetabled trips might be modified or even canceled in the re-optimization scenario. Also, new trips may have to be covered and requirements may have changed (e.g., an increased minimum turn duration) that forbid dedicated details.

Our idea to reasonably carry over the data of the connection templates to the re-optimization RSRP is as follows. We penalize deviations for each detail and for each hyperarc individually by the objective features 9 to 15 (see Equation (4.9)) in the following way.

Let $h \in H$ be a hyperarc of the hypergraph of the re-optimization scenario. In a first step we reinterpret h in the reference rotations, i.e., we search through all connection templates and check if h matches to some details of a template. A property value $p_i(h)$ for $i = 9, \ldots, 15$ is then set to an appropriate value if h models a deviation w.r.t. the reference rotation plan. This is possible because the hyperarcs of RotOR's hypergraph are distinguished by the same details. In particular, the hyperarcs are elaborately distinguished by the reference rotation as described in Section 4.11. If a deviation is unavoidable, e.g., if a succeeding timetabled trip is canceled w.r.t. p_{14}, the property values are clearly not affected.

We also consider departures and arrivals of the re-optimization scenario to match connection templates of the reference rotation plan if the corresponding times and locations slightly vary. Thereby, small timetable changes, e.g., a shift of an arrival time by five minutes or a shortened timetabled trip (by, e.g., one stop located ten kilometers before the stop location in the reference rotations) do not prevent the configuration routine to recognize desired (details of) connections of the reference rotations for re-optimization.

Examples for the adjustment of the property values of deviations are:

- ▷ If $h \in H$ implies that one vehicle operates $t \in T$ in a different rotation (with a different fleet) we set $p_{11}(h) = 1$ ($p_{10}(h) = 1$), otherwise $p_{11}(h) = 0$ ($p_{10}(h) = 0$).

- ▷ If $h \in H$ implies that trip $t \in T$ is operated by a different vehicle composition than in the reference rotations, we adjust $p_{12}(h)$ and $p_{13}(h)$ to the number of orientation and position deviations.
- ▷ Let $h \in H$ a hyperarc connecting the timetabled trips $t_1 \in T$ and $t_2 \in T$. If the arrival of t_1 and the departure of t_2 exist in the reference rotations and are not connected there, we set $p_{14}(h) = |h|$.
- ▷ If $h \in H$ implies a different maintenance service before or after a timetabled trip $p_{15}(h) = 1$, otherwise $p_{15}(h) = 0$.

All of these deviations prepared in the form of distinguished property values are now multiplied by individual cost coefficients c_i for $i = 9, \ldots, 15$ in order to penalize their occurrences in the solution to the re-optimization scenario. In this way RotOR is able to handle a lot of technical re-optimization details simply by changing objective function coefficients and by the "simple" extension of the hypergraph described in Section 4.11.

Chapter 5.

RotOR's Mixed-Integer Programming Model

RotOR's algorithms presented in Chapter 6, which are called in order to solve industrial scenarios for DB Fernverkehr AG (DBF), are based on a mixed-integer programming (MIP) formulation for the rolling stock rotation problem (RSRP). This formulation is the subject of this chapter.

As we have seen in Chapter 4 many industrial requirements are already captured by RotOR's hypergraph. In fact, the hypergraph defines the major structure of RotOR's MIP model through the *flow part* of the MIP model. We present the MIP model in Section 5.1 in a moment. It is composed as follows.

The **flow part** is formed by constraints (covering) and (vehicle-flow) that we already presented in terms of the basic model in Section 1.5 of Chapter 1. The flow part does not cover all industrial requirements to be considered in the RSRP application at DBF, (unfortunately). To this end, it is supplemented by the following further model parts.

The **maintenance part** of the MIP model presented in Section 5.2 is used in order to handle the maintenance constraints of the RSRP by additional continuous (i.e., rational) variables and linear constraints.

The **capacity part** denoted in Section 5.3 models specialized capacity constraints of the RSRP arising in its application at DBF. The available number of vehicles of a fleet (i.e., its fleet capacity) is treated in Section 5.3.1. In addition, we deal with the capacity of the railway infrastructure in terms of maintenance services and parking facilities in Section 5.3.2.

The **trunk part** of RotOR's MIP model explained in Section 5.4 is responsible for industrial constraints of DBF, namely *trunk constraints*. Trunk constraints appear in the context of trip sequences (see Section 4.8) and force continuous passenger connections along the rolling stock rotations.

The flow, maintenance, and capacity part of RotOR's model is published in our paper [6] which is published in the journal Transportation Science and which is a further development of the paper [5] whose contribution includes the paper [4]. Thus,

$$[4] \subset [5] \subset [6] \subset \text{Chapter 5}.$$

5.1. The Model

Before we denote RotOR's complete MIP model, we list all needed data:

- ▷ The sets of timetabled trips, trip sequences, service nodes, maintenance constraints, capacity constraints, and fleets are denoted by T, $\mathbb{T}$, S, L, B, and F.
- ▷ We assume that a hypergraph $(V \cup S, A, H)$ modeling all requirements presented in Chapter 5 is given. The hypergraph particularly contains
  - ▷ hyperarcs $H(t) \subset H$ that model the operation of each timetabled trip $t \in T$ by vehicle compositions (see Section 4.7),
  - ▷ hyperarcs modeling the connection of trips by vehicle compositions through arcs of A that link the arrival and departure nodes of V,
  - ▷ services paths (also modeled as hyperarcs) that cover service nodes of S in order to assign a maintenance service to railway vehicles (see Section 4.5), as well as
  - ▷ regularity hyperarcs, see Section 4.10.

  The hypergraph's cost function $c : H \mapsto \mathbb{Q}_+$ is composed of 15 objective features with property values $p_i(h)$ and cost coefficients $c_i(h)$ for $i = 1, \ldots, 15$ and $h \in H$, see Section 4.12.
- ▷ Each maintenance constraint $l \in L$ is associated with a resource function $r_l : S \cup A \mapsto \mathbb{Q}_+$ and a resource upper bound $U_l \in \mathbb{Q}_+$, see Section 5.2.
- ▷ The maximum number of railway vehicles of each fleet $f \in F$ is denoted by the parameter $U_f \in \mathbb{Z}_+$, see Section 5.3.1.
- ▷ Each capacity constraint $b \in B$ is associated with a resource function $r_b : H \mapsto \mathbb{Q}_+$ and a capacity bound $U_b \in \mathbb{Q}_+$. Capacity constraints model infrastructure capacity, which is illustrated in Section 5.3.2.
- ▷ We consider a set of *trunk constraints* that are represented by values $\mathfrak{v}\left(\{t_i\}_{i=1}^k\right) \in \mathbb{Z}_+$ for each trip sequence $\{t_i\}_{i=1}^k \in \mathbb{T}$, see Section 4.8. In Section 5.4 we construct a *trunk network* $(V_\mathbb{T}, A_\mathbb{T})$, in which the nodes $V_{\langle\mathbb{T}\rangle} \subset V_\mathbb{T}$ carry on so called *trunk vehicles*. Trunk constraints are important for the intercity express (ICE) network of DBF, but appear less crucial in terms of RotOR's model and algorithm (therefore, they have not been mentioned in the general description of the RSRP in Chapter 1).

We introduce a binary decision variable x_h for each hyperarc $h \in H$. In addition, we define a non-negative continuous variable $w_a^l \in [0, U_l]$ for each standard arc $a \in A$ and each maintenance constraint $l \in L$. The variable w_a^l models the current resource consumption since the last maintenance service after the arc a has been traversed. Further, we denote by y_a a continuous flow variable for each arc $a \in A_\mathbb{T}$ of the trunk network. Now, RotOR's complete MIP model for the RSRP, which we apply at DBF, reads:

$$\min \sum_{h\in H} \sum_{i=1}^{15} c_i(h)\, p_i(h)\, x_h, \tag{RMIP}$$

$$\sum_{h\in H(t)} x_h = 1 \qquad \forall\, t \in T, \tag{covering}$$

$$\sum_{h\in H(v)^{\text{in}}} x_h = \sum_{h\in H(v)^{\text{out}}} x_h \qquad \forall\, v \in V, \tag{vehicle-flow}$$

$$w_a^l \le \sum_{h\in H(a)} U_l\, x_h \qquad \forall\, a \in A,\, l \in L, \tag{res-coupling}$$

$$\sum_{a\in A(v)^{\text{out}}} w_a^l - \sum_{h\in H(v)^{\text{out}}} r_l^v(h)\, x_h = \sum_{a\in A(v)^{\text{in}}} w_a^l \qquad \forall\, v \in V,\, l \in L, \tag{res-flow}$$

$$\sum_{a\in A(v)^{\text{out}}} w_a^l - \sum_{h\in H(v)^{\text{out}}} r_l^v(h)\, x_h = 0 \qquad \forall\, v \in S,\, l \in L, \tag{res-reset}$$

$$\sum_{h\in H} p_1^f(h)\, x_h \le U_f \qquad \forall\, f \in F, \tag{fleet capacity}$$

$$\sum_{h\in H} r_b(h)\, x_h \le U_b \qquad \forall\, b \in B, \tag{rail capacity}$$

$$y_a \le \sum_{h\in H(a)} x_h \qquad \forall\, a \in A_{\mathbb{T}}, \tag{trunk-couple}$$

$$\sum_{a\in A_{\mathbb{T}}(v)^{\text{out}}} y_a = \sum_{a\in A_{\mathbb{T}}(v)^{\text{in}}} y_a \qquad \forall\, v \in V_{\langle\mathbb{T}\rangle}, \tag{trunk-flow}$$

$$\sum_{v\in V_{\mathbb{T}}(t_1)} \sum_{a\in A_{\mathbb{T}}(v)^{\text{out}}} y_a \ge \mathfrak{v}\left(\{t_i\}_{i=1}^k\right) \qquad \forall\, \{t_i\}_{i=1}^k \in \mathbb{T}, \tag{trunk start}$$

$$x_h \in \{0,1\} \qquad \forall\, h \in H, \tag{x-domain}$$

$$w_a^l \in [0, U_l] \subset \mathbb{Q}_+ \qquad \forall\, a \in A,\, l \in L, \tag{w-domain}$$

$$y_a \ge 0 \qquad \forall\, a \in A_{\mathbb{T}}. \tag{y-domain}$$

The linear objective function minimizes the cost of operating the timetable. For each trip $t \in T$ the covering constraints (covering) assign exactly one hyperarc of $H(t)$ to t. The equalities (vehicle-flow) are flow conservation constraints for each node $v \in V$ that imply a set of cycles, namely rotations. The constraints (covering), (vehicle-flow), and (x-domain) state the flow part of RotOR's model[1].

[1] For the definitions of $H(v)^{\text{out}}$, $H(v)^{\text{in}}$, $A(v)^{\text{out}}$, and $A(v)^{\text{in}}$ for $v \in V$ see definition (1.4) on page 22.

We call (res-coupling), (res-flow), and (res-reset) *resource flow constraints* that form the maintenance part of the model. The resource flow constraints ensure that the set of cycles is feasible w.r.t. all maintenance constraints $l \in L$. The maintenance constraints and their resource flow constraints are the subject of Section 5.2.

The capacity part is composed of the linear inequalities (fleet capacity)[2] and (rail capacity), which we introduced as capacity constraints in terms for the basic model for the RSRP in Section 1.4. Both kinds of capacity constraints are illustrated in Section 5.3.

The last part of RotOR's industrial model is the trunk part formed by inequalities (trunk-couple) as well as equalities (trunk-flow) and (trunk start). We explain it in Section 5.4.

5.2. Maintenance Constraints

The topology of the German railway system w.r.t. the maintenance service facilities and the shape of the timetables in ICE passenger traffic does not allow to ignore maintenance constraints. For some major maintenance constraints there exists a limited set of dedicated locations where appropriate service activities can be performed. Thus, by using the non-maintenance relaxation of the RSRP, the resulting rolling stock rotations will not (or not with the needed frequency) cover those locations. In fact, the number of vehicles required to operate a timetable may be underestimated by a solution to the non-maintenance relaxation. Therefore, maintenance constraints have to be considered in the RSRP.

During our cooperation with DBF we investigated four maintenance constraints[3] for which we give a high-level description in the following.

Regular inspections have to appear frequently along the rolling stock rotations. In the industrial instances approximately five to twelve timetabled trips can be operated one after another without undergoing regular inspection[4]. This is a result of the distance dependent resource function for regular inspections. The duration of the corresponding maintenance services varies from two to four hours and the number of locations, where the services can be performed, is less than six[5].

Major inspections can be performed almost at the same locations were the regular inspections take place. But the duration of a major inspection is much longer, e.g., one full day is a rather typical case. The resource function for major inspections is time dependent and is constrained such that appropriate service activities appear once a week on average.

Refueling is needed for diesel-driven engines. DBF operates only a few (i.e., not more than 16) diesel-driven traction units at the moment. The distance dependent

[2] The (fleet capacity)-specific coefficients $p_1^f(h)$ for $h \in H$ and $f \in H$ are introduced in Section 5.3.1.

[3] The proposed naming of the maintenance constraints is fictional (different technical German terms are used at DBF). Our translations follow the main technical purposes of the maintenance constraints.

[4] Note that, five to twelve is a rather vague observation and not a constraint.

[5] The resource constraint that we consider in the rolling stock instances for the RCAP in Chapter 3 as well as in the computational study of the C2F Chapter 2 is a regular inspection.

resource function and its resource upper bound for refueling are similar to the ones for regular inspections. A refueling activity takes approximately two hours.

Home bases are locations that are dedicated to fleets. Each vehicle of a fleet has to visit its home base, which is a dedicated location, one time per rolling stock rotation at least. This is constrained, e.g., in order to create opportunities for vehicle exchanges.

Cleaning services need to be performed at least every 24 hours. The usual case is to clean the vehicles during the night within a time window of not more than, say, four hours.

All of these four constraints are of practical interest but of different importance for RotOR's model. Regular inspections and refueling are essential for feasible rolling stock rotations. Therefore, these constraints are frequently considered explicitly in industrial RSRP scenarios of DBF.

A little less attention is given to major inspections, which are sometimes not considered explicitly. In these cases it is assumed that they can be scheduled in a subsequent planning step, which is not part of our considerations.

For many timetables the coverage of home bases is fulfilled implicitly. Often many trips depart and arrive near home bases. In these cases it is unlikely to obtain a rolling stock rotation that is separated from the corresponding home base when the corresponding constraint is not considered explicitly. Therefore, home base constraints appear only very rarely in an explicit fashion in industrial RSRP instances of DBF.

Obviously, cleaning has the structure of a maintenance constraint. In addition, cleaning is important to offer a high-quality service to the passengers. Nevertheless, RotOR does not handle this constraint explicitly as a maintenance constraint. Instead, the needed time windows for cleaning services are included in the turn duration rules presented in Section 4.2. More precisely, if a hyperarc $h \in H$ that connects timetabled trips covers a dedicated point during the night (e.g., 2:30 a.m.), h has to provide enough time for a cleaning service. This approach turns out to be sufficient for cleaning services w.r.t. the current application at DBF.

We proceed with the model for the maintenance constraints. To this end, we revise the notation for the maintenance constraints from Section 1.4 and present the MIP part afterwards.

The set of all maintenance constraints that we take into account explicitly is denoted by L. A maintenance constraint $l \in L$ consists of a resource function $r_l : S \cup A \mapsto \mathbb{Q}_+$, a resource upper bound $U_l \in \mathbb{Q}_+$, and a set of maintenance services $S_l \subseteq S$ that are dedicated to l. A path $P \subseteq A$ that starts and ends at nodes of S_l and does not cover any other nodes of S_l is called resource path w.r.t. $l \in L$. A resource path $P \subseteq A$ is feasible w.r.t. the maintenance constraint $l \in L$ if $\sum_{a \in P} r_l(a) + \sum_{s \in S(P) \setminus S_l} r_l(s) \leq U_l$ (1.2) holds. Here, $S(P) \subseteq S$ denotes the set of service nodes that the resource path $P \subseteq A$ covers. In the RSRP each node of $V \cup S$ that is covered by a rolling stock rotation needs to be contained in a feasible resource path for all maintenance constraints of L.

The MIP formulation for the maintenance constraints works as follows. We define a non-negative continuous variable $w_a^l \in [0, U_l]$ for each standard arc $a \in A$ and each maintenance constraint $l \in L$. The variable w_a^l models the current resource consumption since the last maintenance service $l \in S_l$ after the arc $a \in A$ has been traversed.

For the simplicity of notation we define a resource consumption $r_l(v) := 0$ for all nodes $v \in V$ and set $r_l^v(h) := r_l(\,(v,w)\,) + r_l(v)$ to the sum of the resource consumption of the standard arc $(v,w) \in H$ that goes out of the node $v \in V \cup S$ plus the resource consumption of v.

The formulation of the maintenance constraints, namely the maintenance part of RotOR's model, for the RSRP can now be stated as follows:

$$w_a^l \leq \sum_{h \in H(a)} U_l \, x_h \qquad \forall a \in A,\, l \in L, \qquad \text{(res-coupling)}$$

$$\sum_{a \in A(v)^{\text{out}}} w_a^l - \sum_{h \in H(v)^{\text{out}}} r_l^v(h)\, x_h = \sum_{a \in A(v)^{\text{in}}} w_a^l \qquad \forall v \in V,\, l \in L, \qquad \text{(res-flow)}$$

$$\sum_{a \in A(v)^{\text{out}}} w_a^l - \sum_{h \in H(v)^{\text{out}}} r_l^v(h)\, x_h = 0 \qquad \forall v \in S,\, l \in L. \qquad \text{(res-reset)}$$

As already mentioned, the linear constraints (res-coupling), (res-flow), and (res-reset) are called resource flow constraints. The resource flow constraints ensure that the set of cycles is feasible w.r.t. all maintenance constraints $l \in L$.

Inequalities (res-coupling) formulate that a variable w_a^l can only be non-zero if a x-variable for a corresponding hyperarc of $H(a)$ is non-zero. Suppose that the x-variables are fixed such that constraints (covering), (vehicle-flow), and (x-domain) are fulfilled. Hence, the corresponding hyperarcs form a set of cycles and the flow implied by the w-variables traverses the same set of cycles (i.e., rotations).

The constraints (res-flow) state that for each node $v \in V$ the sum of the outgoing (or succeeding) resource consumption must be equal to the sum of the incoming (or preceding) resource consumption plus the current resource consumption caused by the hyperarc that goes out of v. The same idea applies to service nodes by constraints (res-reset), but we do not include the preceding flow such that the resource consumption is reset at service nodes.

Let $v \in V$ be a node and let $l \in L$ be a maintenance constraint. Suppose that $a^{\texttt{out}} \in A(v)^{\text{out}}$, $a^{\texttt{in}} \in A(v)^{\text{in}}$, and $h \in H(v)^{\text{out}}$ are the (hyper-) arcs that go into and out of v in a certain solution. In this case we have $x_h = 1.0$ and equation (res-flow) for $v \in V$ and $l \in L$ reduces to $w_{a^{\texttt{out}}}^l = w_{a^{\texttt{in}}}^l + r_l^v(h)$, which states that the flow value of an arc, namely $w_{a^{\texttt{out}}}^l$, is always the sum of the flow value of the predecessor arc $w_{a^{\texttt{in}}}^l$ and the actual resource consumption $r_l^v(h)$.

In order to illustrate our model for the maintenance constraints we give a brief example in the following.

Example for resource flow constraints. We give an example for the resource flow constraints as follows. We consider only a single maintenance constraint $l \in L$, which is not explicitly subscripted in this example. We further assume that we are given a part of a hypergraph as shown in Figure 5.1 on page 174 with resource consumptions $r^1(h_1) = r^2(h_2) = r^3(h_3) = r^4(h_4) = r^5(h_5) = r^s(h_6) = r^1(h_6) = 1$ and an upper bound $U = 6$. For this example, the constraints (res-coupling), (res-flow), and (res-reset) read as follows.

$$
\begin{array}{lcll}
w_1 & \leq & Ux_{h_1} & \text{((res-coupling) for arc 1),} \\
w_2 & \leq & Ux_{h_2} & \text{((res-coupling) for arc 2),} \\
w_3 & \leq & Ux_{h_3} & \text{((res-coupling) for arc 3),} \\
w_4 & \leq & Ux_{h_4} & \text{((res-coupling) for arc 4),} \\
w_5 & \leq & Ux_{h_5} & \text{((res-coupling) for arc 5),} \\
w_{\text{in}} & \leq & Ux_{h_6} & \text{((res-coupling) for in } s \text{ incoming arc),} \\
w_{\text{out}} & \leq & Ux_{h_6} & \text{((res-coupling) for from } s \text{ outgoing arc),} \\
w_2 & = & w_1 + \quad r^1(h_2) \cdot x_{h_2} + w_{\text{out}} & \text{((res-flow) for node 1),} \\
w_3 & = & w_2 + \quad r^2(h_3) \cdot x_{h_3} & \text{((res-flow) for node 2),} \\
w_4 & = & w_3 + \quad r^3(h_4) \cdot x_{h_4} & \text{((res-flow) for node 3),} \\
w_5 & = & w_4 + \quad r^4(h_5) \cdot x_{h_5} & \text{((res-flow) for node 4),} \\
w_1 + w_{\text{in}} & = & w_5 + \quad r^5(h_1) \cdot x_{h_1} + r^5(h_6) \cdot x_{h_6} & \text{((res-flow) for node 5),} \\
w_{\text{out}} & = & \quad r^s(h_6) \cdot x_{h_6} & \text{((res-reset) for node } s\text{).}
\end{array}
$$

If we set $x_{h_1} = 1.0$ it follows by constraints (vehicle-flow) that $x_{h_2} = \ldots = x_{h_5} = 1.0$ and $x_{h_6} = 0.0$. $x_{h_6} = 0.0$ implies $w_{\text{out}} = w_{in} = 0.0$ by constraints (res-coupling). After inserting the values of the resource consumption function we get: $w_2 = w_1 + 1.0$, $w_3 = w_2 + 1.0$, $w_4 = w_3 + 1.0$, $w_5 = w_4 + 1.0$, $w_1 = w_5 + 1.0$. By successively inserting the equations starting with the last into the first we get $w_2 = w_2 + 4.0$, which shows that the resulting model is infeasible. If we set $x_{h_6} = 1.0$ we get the only feasible solution $x_{h_2} = \ldots = x_{h_6} = 1.0$ that uses h_6 as replenishment hyperarc.

5.3. Capacity Constraints

The operation of a timetable requires to utilize infrastructure and rolling stock. Both utilities have a limited capacity, which is the subject of this section. In our model we consider constraints that limit the capacity of a fleet (i.e., the number of vehicles used in a solution) and constraints that limit the amount of infrastructure capacity that is occupied by the rolling stock rotations of a solution to the RSRP.

5.3.1. Fleet Capacity

The number of physical vehicles of a fleet is limited. We denote the number of available physical vehicles of fleet $f \in F$ by $U_f \in \mathbb{Z}_+$. The total number of vehicles used in a solution $H^\star \subseteq H$ to the RSRP can be recognized by counting how often the cycles (i.e.,

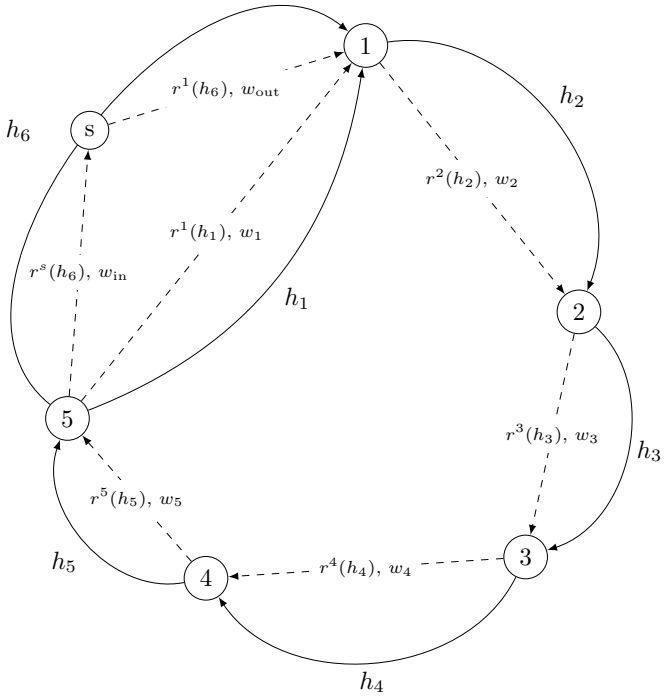


Figure 5.1.: Example of the resource flow model.

rolling stock rotations) that $H^\star$ defines cover a specific date, e.g., Monday 0:00 a.m. in the standard week. We simply carry over this idea to the individual hyperarcs of the solution $H^\star$ but distinguish by fleets.

A hyperarc $h \in H$ defines connections of nodes of V that correspond to arrivals or departures of timetabled trips, see Figure 4.17 for a recapitulation (note that also service paths connect arrivals to departures, see Section 4.5). These nodes have specific dates. Now we are able to denote by[6] $p_1^f(h) \in \mathbb{Z}_+$ how many vehicles of fleet $f \in F$ cover the specific date Monday 0:00 a.m. if $h \in H$ belongs to a solution. Consequently, if $H^\star \subseteq H$ is a feasible solution to the RSRP it has to fulfill $\sum_{h \in H^\star} p_1^f(h) \leq U_f$ for all fleets $f \in F$.

Fleet capacity constraints translate easily to the following linear inequalities, which we use in RotOR's MIP approach:

$$\sum_{h \in H} p_1^f(h)\, x_h \leq U_f \quad \forall\, f \in F. \tag{5.1}$$

In the application at DBF it is not always completely obvious if the capacity of a fleet

[6] The notation $p_1^f(h)$ is inspired by the notation of the property value for the (fractional) number of vehicles that $h \in H$ imposes, see definition (4.9) in Section 4.12 about the objective function. Note that $p_1^f(h) \in \mathbb{Z}_+$ denotes the *integral* number of vehicles that pass Monday 0:00 a.m. whereas $p_1(h) \in \mathbb{Q}_+$ denotes the *fractional number* of vehicles that $h \in H$ models. This modeling difference is made in order to minimize the number of non-zeros in constraints. In the objective function we prefer $p_1(h) \in \mathbb{Q}_+$ in order to distribute the vehicle cost over all hyperarcs of a solution.

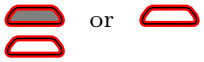
track	length	feasible assignments of vehicle configurations
1	570 m	
2	480 m	
3	430 m	
4	420 m	
5	420 m	
6	420 m	
7	410 m	
8	390 m	
9	240 m	
10	240 m	
11	240 m	
12	210 m	
13	210 m	
14	210 m	

Figure 5.2.: Enumerated vehicle configuration assignments for a parking facility.

is a constraint (to be considered as inequality (5.1)) or is an optimization objective. The questions how many vehicles are required and how many vehicles are available are not completely independent in several industrial use cases. E.g., one would optimize the fleet capacity for a strategical estimate on the number of vehicles for a future conceptual timetable whereas in many re-optimization scenarios the fleet capacities are proper constraints. The number of railway vehicles is always minimized by RotOR's model, even if a constraint for the capacity of a fleet is imposed.

5.3.2. Infrastructure Capacity

Figure 5.2 gives an example of a set of tracks of a parking area for two different fleets, i.e., a black and a white one. We enumerate feasible assignments of vehicle configurations for a single track w.r.t. a specific date in the standard week. This assignment is done by comparing the lengths of the vehicle configurations to the lengths of the tracks. In addition this enumeration is constrained by several other technical requirements, e.g., we can not park a black unit at tracks 9 to 14 because black units need a track with a refueling facility.

Parking capacity constraints. We model the infrastructure capacity of a dedicated parking area by capacity constraints that restrict the maximal number of vehicles per fleet as well as the total number of vehicles (independent from the fleet) w.r.t. a specific date in the standard week. Our notation for these capacity constraints is equal to the notation presented in Section 1.4, i.e., we deal with a set of capacity constraints B and each capacity constraint $b \in B$ is represented by a resource function $r_b : H \mapsto \mathbb{Q}_+$ and an upper bound $U_b \in \mathbb{Q}_+$. The concrete resource functions and upper bounds that model infrastructure capacity are as follows.

By means of the feasible assignments as shown in Figure 5.2 we estimate $14 \cdot 1 = 14$ vehicles for the white fleet, $7 \cdot 2 + 1 = 15$ vehicles for the gray fleet, and $7 \cdot 2 + 7 = 21$ vehicles in general that can be parked at the same time. If $H^\star \subseteq H$ is a solution of the RSRP, the capacity constraint b for the white fleet on Monday midnight can be denoted as $\sum_{h \in H^\star} r_b(h) \leq 14$. The resource function $r_b(h)$ is equal to the number of vehicles of the white fleet used in the vehicle composition of $h \in H$ if h covers Monday midnight and zero otherwise.

It can be observed, that bottlenecks of infrastructure capacity appear at specific dates, e.g., parking areas are almost empty during the day and almost completely full during the night. Therefore, we use the seven midnights in the standard week as specific dates, i.e., we define capacity constraints for the total number of vehicles and for the number of vehicles per fleet are defined for each of these seven dates.

Maintenance capacity constraints. Beside infrastructure capacity constraints for parking facilities, RotOR can also handle capacity constraints for maintenance services. Those constraints are handled similarly by counting the number of services of a solution that cover a specific date. But, since the date of the start of a maintenance service along its service path is not defined by RotOR's hypergraph, the "date approach" is implemented by the following heuristic rule: If a maintenance service can *not* be shifted between its corresponding arrival-departure pair such that it is *not* covered by the specific date it is considered within a respective constraint that limits the capacity of maintenance services of a certain fleet at a specific location and date within the standard week.

We finally point out the Bachelor's thesis by Brückner (2012, [31]), which was written during our cooperation with DBF. In easy words, this thesis investigates how to derive capacity constraints for parking facilities.

5.4. Trunk Constraints for Trip Sequences

The concept of trip sequences is introduced in Section 4.8. A trip sequence of length $k \in \mathbb{Z}_+$ is a series $\{t_i\}_{i=1}^k$ of k timetabled trips $t_i \in T$ for $i = 1, \ldots, k$. Trip sequences come from timetabling, i.e., they are defined in the input data of the RSRP. The industrial intention behind trip sequences is to offer continuous connections (i.e., connections without changes) to the passengers.

As explained in Section 4.8, some timetabled trips are split into smaller parts that form timetabled trip sequences in order to increase the granularity for vehicle configuration requirements. For those "artificial" cases the requirement for a continuous passenger connection through a timetabled trip sequence is obvious.

However, there exists another type of trip sequence: *Enforced trip sequences* are made particularly in order to provide continuous connections. These enforced trip sequences originate from the line plan. A clip of the German ICE network of the year 2015 is plotted in Figure 5.3. In this line plan the purple line runs from Berlin to Cologne (Köln) and to Düsseldorf as well. The line splits in Hamm (Westfalen). Therefore, the purple line promises two continuous connections, i.e., one from Berlin to Cologne and another

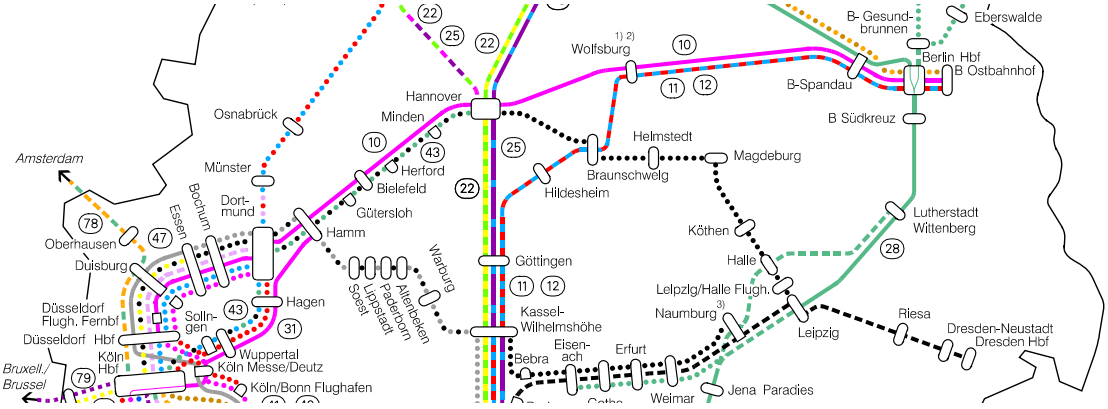


Figure 5.3.: Clip of the German ICE network 2015 published by DBF [40].

one from Berlin to Düsseldorf. The timetabled trips that belong to the purple line are operated by vehicle configurations of size two that split in Hamm.

The trips that operate the purple line by a departure in Berlin are arranged as follows. One of the two continuous connections is announced under a single train number, i.e., it is a timetabled trip sequence with an intermediate stop at Hamm. The other continuous connection is given as an enforced trip sequence that is composed of trips through Hamm with different train numbers[7].

A railway vehicle that successively runs through all trips $t_1, \ldots, t_k \in T$ in the order that is defined by the timetabled trip sequence $\{t_i\}_{i=1}^k$ is called *trunk vehicle*. Trunk vehicles implement the requirement that a trip sequence really constitutes a continuous connection for passengers. In the input data for a RSRP instance a value

$$\mathfrak{v}\left(\{t_i\}_{i=1}^k\right) \in \mathbb{Z}_+$$

is associated with each trip sequence $\{t_i\}_{i=1}^k$. The value $\mathfrak{v}\left(\{t_i\}_{i=1}^k\right)$ states the minimum number of trunk vehicles that have to pass through $\{t_i\}_{i=1}^k$ by starting with the departure of t_1 and ending with the arrival of t_k. The constraint that is associated with $\mathfrak{v}\left(\{t_i\}_{i=1}^k\right)$ for the trip sequence $\{t_i\}_{i=1}^k$ is called *trunk constraint*.

In order to take trunk constraints into account we construct a directed graph that is dedicated to them. We denote the set of trip sequences given in RotOR's input data for the RSRP by $\mathbb{T}$ (the elements of $\mathbb{T}$ are series of timetabled trips, i.e., trip sequences). The symbol $\mathbb{T}$ is also used as subscript for the nodes $V_\mathbb{T}$ and arcs $A_\mathbb{T}$ of the directed graph that is dedicated to the trunk constraints. Note that the nodes and arcs of the directed graph $(V_\mathbb{T}, A_\mathbb{T})$ are distinct from the nodes and arcs of the directed graph (V, A) for the RSRP.

The node set $V_\mathbb{T}$ of $(V_\mathbb{T}, A_\mathbb{T})$ is composed as follows. For each timetabled trip $t \in T$ we have as many nodes in $V_\mathbb{T}$ as vehicles can be maximally coupled together in order to operate t. We denote the nodes that are dedicated to $t \in T$ by $V_\mathbb{T}(t) \subseteq V_\mathbb{T}$. A node

[7] Enforced trip sequences are called "Zwangsbindungen" at DBF.

$v \in V_{\mathbb{T}}(t)$ models the departure of t at a dedicated position of a trunk vehicle in a trip sequence that includes t. The set $V_{\mathbb{T}}$ is simply the union over these node sets, i.e., $V_{\mathbb{T}} := \bigcup_{t \in T} V_{\mathbb{T}}(t)$. The arc set $A_{\mathbb{T}}$ is of the form

$$A_{\mathbb{T}} := \bigcup_{\{t_i\}_{i=1}^{k} \in \mathbb{T}} \{(u, v) \mid u \in V_{\mathbb{T}}(t_i),\, v \in V_{\mathbb{T}}(t_j),\, i \in \{1, \ldots, k-1\},\, j = i+1\}.$$

Hence, $(V_{\mathbb{T}}, A_{\mathbb{T}})$ is composed of $|\mathbb{T}|$ connected components (w.r.t. the underlying undirected graph). Each of these connected components is dedicated to a trip sequence.

The interpretation of the traversal of the arc $(u, v) \in A_{\mathbb{T}}$ is to move a trunk vehicle (within a trip sequence) from the departure position of the timetabled trip that u represents to the departure position of the timetabled trip that v represents. Each path in $(V_{\mathbb{T}}, A_{\mathbb{T}})$ of maximal length represents the traversal of a trip sequence with a trunk vehicle (by construction).

We introduce a binary decision variable $y_a \in \{0, 1\}$ for each arc $a \in A_{\mathbb{T}}$. The variable y_a takes value one if the movement of a trunk vehicle that is modeled by $a \in A_{\mathbb{T}}$ appears in a solution to the RSRP and is equal to zero otherwise. In order to force as many maximal paths for trunk vehicles in the graph $(V_{\mathbb{T}}, A_{\mathbb{T}})$ we introduce the following linear constraints:

$$\sum_{v \in V_{\mathbb{T}}(t_1)} \sum_{a \in A_{\mathbb{T}}(v)^{\text{out}}} y_a \geq \mathfrak{v}\left(\{t_i\}_{i=1}^{k}\right) \quad \forall\, \{t_i\}_{i=1}^{k} \in \mathbb{T} \qquad \text{(trunk start)}$$

and

$$\sum_{a \in A_{\mathbb{T}}(v)^{\text{out}}} y_a = \sum_{a \in A_{\mathbb{T}}(v)^{\text{in}}} y_a \quad \forall\, v \in V_{\langle \mathbb{T} \rangle}. \qquad \text{(trunk-flow)}$$

The set $V_{\langle \mathbb{T} \rangle} \subset V_{\mathbb{T}}$ denotes all nodes that belong to timetabled trips that are part of a trip sequence but are not identical to the first or last trip of a trip sequence, i.e.,

$$V_{\langle \mathbb{T} \rangle} := \{v \in V_{\mathbb{T}} \mid v \in V_{\mathbb{T}}(t) \,:\, t \in \{t \in T \mid t_1 \neq t \neq t_k \,\forall\, \{t_i\}_{i=1}^{k} \in \mathbb{T}\}\}.$$

The inequalities (trunk start) constrain the number of trunk vehicles that depart at t_1 for each trip sequence $\{t_i\}_{i=1}^{k} \in \mathbb{T}$ to be at least $\mathfrak{v}\left(\{t_i\}_{i=1}^{k}\right)$. Equalities (trunk-flow) ensure that the flow induced by inequalities (trunk start) is conserved when it traverses nodes of $V_{\mathbb{T}}$. Nodes that belong to the first (t_1) or last trip (t_k) of a trip sequence $\{t_i\}_{i=1}^{k} \in \mathbb{T}$ do not have to conserve flow. Therefore, equalities (trunk-flow) are exclusively for nodes of $V_{\langle \mathbb{T} \rangle}$. In this way, the linear constraints (trunk start) and (trunk-flow) imply a set of trunk vehicles (i.e., a set of maximal paths) in the graph $(V_{\mathbb{T}}, A_{\mathbb{T}})$ that is required by the trunk constraints.

Up to now, the flow of the y-variables is not related to the x-variables that define the set of hyperarcs in a solution to the RSRP. In order to constrain the y-variables to follow the x-variables we relate the arcs of $A_{\mathbb{T}}$ to the hyperarcs of H in the following way. For each $a \in A_{\mathbb{T}}$ we denote the set of hyperarcs that imply the movement that a models by $H(a) \subseteq H$. Suppose that $a \in A_{\mathbb{T}}$ models the movement from departure position one of t_3 to departure position two of t_4 in Figure 4.17 on page 135. Then, the hyperarc that

connects t_3 to t_4 in Figure 4.17, namely $h(t_3, t_4) \in H$, is an element of the set $H(a) \subseteq H$, i.e., $h(t_3, t_4) \in H(a)$.

The following inequalities imply the movements of the trunk vehicles expressed by the y-variables in the x-variables as well, i.e., if a trunk vehicle performs the movement modeled by $a \in A_{\mathbb{T}}$ (i.e., $y_a = 1$) an appropriate hyperarc variable x_h with $h \in H(a)$ needs to have value one as well. This is formulated by inequalities (trunk-couple):

$$y_a \leq \sum_{h \in H(a)} x_h \quad \forall\, a \in A_{\mathbb{T}}. \tag{trunk-couple}$$

Note that the model for trunk constraints, namely the standard directed graph $(V_{\mathbb{T}}, A_{\mathbb{T}})$ as well as constraints (trunk start), (trunk-flow), and (trunk-couple), is structurally very similar to the configuration model of Schlechte (2012, [94]), which is made for the train timetabling problem (TTP). There, train requests are also modeled by arc-based variables (the y-variables in our case) that are coupled in almost the same way to configuration variables, which are the hyperarc variables in our case. In this way, the requests for train paths in the TTP correspond to the requests for trunk vehicles in the RSRP.

Even though trunk constraints need special modeling attention as we have seen in this section, we do not see them as main problem aspect of the RSRP. The reason for this is that trunk constraints do not appear to be computationally hard in the industrial application at DBF. In many cases it is sufficient to just eliminate hyperarcs of the hypergraph that violate obviously trunk constraints in a pre-processing procedure. This has the effect that trunk constraints are often automatically fulfilled and do not need extra variables and constraints. Another reason is that the connected components of the graph $(V_{\mathbb{T}}, A_{\mathbb{T}})$ are small. A typical trip sequence has length four where the trips of the trip sequence can be operated with vehicle configurations of size two. In this case the corresponding connected component of $(V_{\mathbb{T}}, A_{\mathbb{T}})$ consists of eight nodes and twelve arcs.

Chapter 6.

RotOR's Algorithms

In this chapter we describe RotOR's algorithms that we use in order to solve the mixed-integer programming (MIP) formulation (RMIP) denoted on page 169 of the rolling stock rotation problem (RSRP), see Section 5.1. The algorithm combines three main algorithmic components: A coarse-to-fine (C2F) constraint matrix generation approach (which is based on but not limited to the ideas of Chapter 2), RSRP specific heuristics, and an adaptation of the rapid branching search scheme.

The C2F constraint matrix generation approach turns out to be necessary because computing the linear programming (LP) relaxation of RotOR's model is time consuming and even storing the whole model in the memory of an usual computer is not possible for all industrial instances that we investigated during our cooperation with DB Fernverkehr AG (DBF). The rapid branching algorithm is needed since the LP relaxation of the model is too hard to tackle within a standard branch and bound (B&B) search tree, while the problem specific heuristics aim to detect high-quality integer solutions. During the chapter we will see how these algorithmic components work together within RotOR from an algorithmic point of view.

The chapter is organized as follows. In Section 6.1 we describe RotOR's general workflow. This is followed by RotOR's C2F constraint matrix generation algorithm presented in Section 6.2. The rapid branching search technique is (briefly) reviewed in Section 6.3. Further heuristic ideas are presented in Sections 6.4 to 6.6.

As advertised in the introduction of the thesis, RotOR also computes handouts for rolling stock rotations. What handouts are and how they play together with RotOR's hypergraph-based model is explained in Section 4.10 of Chapter 4. How RotOR computes handouts is described in Section 6.7 of this chapter.

Sections 6.3 and 6.4 of this chapter have been published as parts of the following papers:

- "Rapid Branching" [7] and
- "Integrated Optimization of Rolling Stock Rotations for Intercity Railways" [6].

6.1. General Workflow

Before we state RotOR's general workflow, we introduce some notation in order to distinguish variable states during the execution of RotOR's algorithms. Consider a hypergraph $(V \cup S, A, H)$ and let $\overline{H} \subseteq H$ and $\underline{H} \subseteq H$ be disjoint subsets of hyperarcs.

Denote by RMIP $\left(\overline{H}, \underline{H}\right)$ the MIP formulation from Section 5.1 with the additional bound constraints $x_h = 1$ for all $h \in \overline{H}$ and $x_h = 0$ for all $h \in \underline{H}$. Therefore, $\overline{H}$ defines the set of variables fixed to one, while $\underline{H}$ defines the set of variables fixed to zero. We assume that all variables associated with $\underline{H}$ are not contained in the corresponding models. Thus, particularly, we do not have to explicitly consider dual variables for the corresponding fixing constraints.

Similarly, LP $\left(\overline{H}, \underline{H}\right)$ denotes the LP relaxation of RMIP $\left(\overline{H}, \underline{H}\right)$, i.e., the problem that results from relaxing all integrality constraints (x-domain) to

$$x_h \geq 0 \quad \forall\, h \in H.$$

The upper bound constraints, i.e., $x_h \leq 1$ for all $h \in H$, are redundant because the constraints (covering) of model (RMIP) constrain either the incoming or the outgoing flow of a node $v \in V$ to be less or equal to one. This constraint carries over to all other hyperarc variables by the equalities (vehicle-flow). Therefore, values $x_h > 1$ can not occur for any $h \in H$.

Under this notation, RotOR aims to solve RMIP $(\emptyset, \emptyset)$, which is outlined by Algorithm 6.1.

```
1  // Assumed data: { timetabled trips T; trip sequences 𝕋; maintenance services S;
                     maintenance constraints L; capacity constraints B; reference rotations R }
2  {
3    // elements of A and H are computed dynamically when needed
4    (V ∪ S, A, H) := computeHypergraph(T, 𝕋, S, R);
5
6    for( i ∈ {1, ..., 15} )
7    {
8      computeObjectiveFeature(i, (V ∪ S, A, H), R);
9    }
10
11   H̲ := constraintMatrixGeneration(); // see Algorithm 6.2
12
13   H̄ := ∅;
14
15   H* := rapidBranching(H̄, H̲); // see Algorithm 6.3
16
17   𝒮 := createSegments(H*);
18
19   maximizeSuccessors(maximizeSimilarities(𝒮)); // see Algorithms 6.8 and 6.9
20 }
```

Algorithm 6.1: General workflow of RotOR.

Up to line 13 RotOR "only" prepares the hypergraph and its objective function for greenfield and re-optimization. Within RotOR's implementation the preparation is

much more sophisticated than presented here. Especially the procedure that analyses the reference rotations and configures the corresponding objective features towards them is rather detailed and complex. In addition, routines that propagate turn around events through the trips and trip sequences of the input timetable, the refinement of the hypergraph for re-optimization templates, as well as many checks that test contradicting or suspect input rules belong to this early stage.

RotOR's hypergraph is *not* explicitly stored in memory as line 4 of Algorithm 6.1 suggests. Instead, RAM is really allocated by the computer only for the smaller and coarser configuration layer, see Section 2.4.2. In order to access the hyperarc $h \in H$ within RotOR's implementation, one has to take its coarse hyperarc $[h] \in [H]$ and to call an iterator routine that enumerates locally all $h' \in H$ with $[h'] \in [H]$ on the *stack* of the computing process that runs RotOR.

Especially these iteration routines contribute to RotOR's column generation procedure, namely the C2F constraint matrix generation algorithm that is called in line 11 of Algorithm 6.1. The C2F constraint matrix generation algorithm computes an optimal primal and an optimal dual solution to the LP relaxation of model (RMIP).

The solutions obtained from the C2F constraint matrix generation algorithm are the major results of this procedure. But there is even more that we gain there. In fact, we use exactly the subset of the hyperarc variables and linear constraints that have proven to be sufficient for the optimality proof of the solutions to the LP relaxation of RotOR's overall model in order to define a restricted MIP model. Then, the (remaining) integer programming (IP) procedures are based on this restricted MIP model. The restricted MIP model simply derives from the C2F constraint matrix generation algorithm by taking all variables and constraints that have been generated and by declaring all continuous hyperarc variables to be binary decision variables (again). Under our notation the restricted MIP model is completely defined by the subset $\underline{H} \subset H$ that contains those hyperarcs that we do not consider in the restricted MIP model (anymore).

Note that the way how we deal with the restricted MIP model is a substantial algorithmic detail. In fact, it is completely heuristic because we implicitly assume that the restricted MIP model contains hyperarc variables associated with an (near) optimal integral solution to the RSRP. In general, this assumption is not true but turns out to be reasonable for the industrial instances provided by DBF during our cooperation.

After the restricted MIP model is set up (i.e., the set $\underline{H}$ of hyperarc variables that we assume to be unnecessary is computed), we start the rapid branching procedure on top of it. There, we try to compute integer feasible solutions for the RSRP by branching with calls to heuristics within the search tree. This is indicated in line 15 of Algorithm 6.1.

Note that the assumption that all variables defined by $\underline{H}$ are not contained in the denoted models is only precise because we do not have to distinguish between variables that are not generated and variables that are fixed explicitly to zero. The reason for this is exactly that the branching scheme is called on top of the C2F constraint matrix generation algorithm and does not further generate variables within the tree, i.e., RotOR performs "price-and-branch" (instead of branch-and-price).

When a solution, namely feasible rolling stock rotations, is found, RotOR computes a handout for each rotation. To this end, the solution is translated to a more technical

description that does not rely on hyperarcs anymore. Imagine a technical description of a rolling stock rotation as a cycle that covers rectangles. The rectangles correspond to timetabled and deadhead trips as well as maintenance services, see Figure 4.24 on page 153.

The translation procedure is indicated by function createSegments(·) in line 17 of Algorithm 6.1. A technical description of a rolling stock rotation divides into a set of so called *segments* that represent the rotation on each day of operation. Finally, RotOR's remaining task is to assemble the segments in order to form a high-quality handout. The handouts are computed by a two stage construction heuristic, which is indicated by the nested function calls in line 19 of Algorithm 6.1. How this works is described in Section 6.7.

6.2. C2F Constraint Matrix Generation

Since the number of variables and rows of RMIP $(\emptyset, \emptyset)$ can be large in industrial instances, even solving its LP relaxation LP $(\emptyset, \emptyset)$ is often not possible in a reasonable amount of computation time by using standard static LP solvers. We, therefore, developed the C2F constraint matrix generation algorithm in order to solve LP $(\emptyset, \emptyset)$. This algorithm is based on but not limited to the ideas of Chapter 2.

By constraint matrix generation we just mean to dynamically generate the constraint matrix of a MIP model by extending the (dynamic) column generation algorithm (CGA) with a procedure that also generates (some) linear constraints of the model dynamically.

In column generation approaches the overall model is split into a *restricted master problem (RMP)* and a *pricing problem*. The RMP contains a subset of all original variables and the pricing problem is solved to find–depending on the dual solution vector–new variables that might improve the objective function value of the RMP. This procedure is iterated until it is proven that there is no variable left that can improve the value of the objective function. RotOR's column generation approach generally follows this line but distinguishes itself from other approaches by the following facts:

1. All variables of program (RMIP) are given explicitly, e.g., the set H that has associated x-variables can be relatively easily iterated (this does not apply to models with variables that correspond to, e.g., paths to be found in a graph for example).

2. Program (RMIP) includes several types of variables, namely, x-variables, w-variables, and y-variables.

3. Both, the number of columns and the number of rows is at least in the order of $\mathcal{O}(|A|)$ (for columns because of $H \subseteq 2^A$; for rows because of constraints (res-coupling)).

The first aspect is considered for the x-variables in Chapter 2 in terms of a C2F hyperarc generation routine that is specialized to the RSRP. The second aspect has not been tackled yet: A CGA for the LP relaxation of program (RMIP) has to deal with three

types of variables. In addition, a traditional CGA may suffer from the third fact, i.e., there are almost as many rows as columns. The second and third aspects are the subject of this section.

First of all, we do not expect a gain by generating the y-variables dynamically. As mentioned in Section 5.4, the number of trunk arcs $|A_{\mathbb{T}}|$ is rather small in the industrial instances of DBF in comparison to the number of x-variables associated with hyperarcs (and w-variables associated with standard arcs). Therefore, RotOR does not generate any y-variables dynamically. Instead, we assume that all y-variables are present in all models that we consider. Note that this is only an implementation detail. More precisely, if instances have to be considered where the number of trunk arcs is large, the approach that we present in the following for the x- and w-variables also works well. This becomes more valuable in view of models for other applications (e.g., the configuration model for the train timetabling problem (TTP), see Section 5.4).

The remaining variables to be generated are the x-variables and the w-variables. The x-variables correspond to hyperarcs of H and the w-variables correspond to standard arcs of A. By definition we have $H \subseteq 2^A$. As we will see, this makes it possible to reduce the pricing problem of the CGA for model (RMIP) to only one type of variables.

Let $m \in \mathbb{N}$ be the number of rows of LP $(\emptyset, \underline{H})$. For the hyperarc $h \in H$ let $\mathbb{A}(x_h) \in \mathbb{Q}^m$ be the column vector of the hyperarc variable x_h, which can be expressed as $\mathbb{A}(x_h) = \sum_{a \in h} \mathbb{A}(x_{\{a\}})$. Further, let $\pi \in \mathbb{Q}^m$ be an optimal dual solution vector of the restricted master problem LP $(\emptyset, \underline{H})$. The reduced cost of x_h is given by $d_h := c_h - \pi^T \mathbb{A}(x_h)$. Similarly, we denote the reduced cost of the variable w_a^l for the standard arc $a \in A$ and the maintenance constraint $l \in L$ by $d(w_a^l) := -\pi^T \mathbb{A}(w_a^l)$ where $\mathbb{A}(w_a^l) \in \mathbb{Q}^m$ is the column vector of the coefficients of the variable w_a^l in model (RMIP). Under this notation, the pricing problem of RotOR's column generation approach for model (RMIP) reads:

$$\text{Find } h \in H \text{ s.t. } d(x_h) < 0 \quad \text{and find} \quad (a, l) \in A \times L \text{ s.t. } d(w_a^l) < 0. \tag{6.1}$$

Therefore, the algorithm needs to be sure that all reduced cost of all x-variables and all w-variables are not negative anymore at termination. In addition, a strategy for the generation of both types has to be defined. Obvious examples for this strategy are the simultaneous generation of x- and w-variables within one column generation iteration or an alternating procedure. In fact, the following lemma suggests a natural alternative.

Lemma 11. *The pricing problem (6.1) for program (RMIP) is equivalent to*

$$\textit{Find } h \in H \textit{ s.t. } d(x_h) < 0. \tag{6.2}$$

if the following rule is applied:

- ▷ *As soon as any $h \in H(a)$ for $a \in A$ has entered the RMP all variables of $\left\{ w_a^l \,|\, l \in L \right\}$ are added to the RMP as well.*

Proof. Case 1: The variables $\left\{ w_a^l \,|\, l \in L \right\}$ are already in the RMP for an arc $a \in A$. Then, solving the pricing problem (6.2) is sufficient to prove optimality because only x-variables remain to be priced.

Case 2: The variables $\{w_a^l \mid l \in L\}$ are not in the RMP for an $a \in A$. Under the rule suggested by the lemma, we know that also no hyperarc $h \in H(a)$ has been generated yet. Suppose that we have solved LP $(\emptyset, \underline{H})$ and let $a \in A$ be an arc and $l \in L$ be a maintenance constraint such that LP $(\emptyset, \underline{H})$ does not contain the variable w_a^l and does not contain any hyperarc variables of $H(a)$. The inequality (res-coupling) corresponding to a and l reads $0 \leq 0$. By introducing an additional slack variable $s_a^l \geq 0$ for this empty constraint we get its normal form $s_a^l = 0$. Let π_a^l be the free dual variable associated with this empty constraint. The only dual constraint that has a non-zero for π_a^l is $\pi_a^l \leq 0$, which is implied by the corresponding primal variable s_a^l. Since the dual objective coefficient of π_a^l is zero (i.e., the right hand side of constraints (res-coupling)) it can be set to any negative value preserving optimality. The smaller we chose π_a^l the larger are the reduced cost of w_a^l and the smaller are the reduced cost of any hyperarc variable corresponding to $H(a)$. In this way it is always possible to manipulate the dual solution vector (while preserving optimality) in order to force that the reduced cost of the w-variables are not negative. □

If the rule mentioned in Lemma 11 is applied, we only have to price x-variables. More precisely, if we found a hyperarc $h \in \underline{H}$ with negative reduced cost to be removed from $\underline{H}$, we also add all w_a^l for all $a \in h$ and for all $l \in L$ to the RMP.

In order to utilize Lemma 6.2 within RotOR we assume that the LP solver for LP $(\emptyset, \underline{H})$ always sets the value of π_a^l such that w_a^l has reduced cost of zero for the constraint that corresponds to $a \in A$ and $l \in L$. This is precisely the case if we set

$$\pi_a^l := -\pi^T \mathbb{A}'(w_a^l) \tag{6.3}$$

where $\mathbb{A}'(w_a^l) \in \mathbb{Q}^m$ is the column vector of coefficients of the variable w_a^l in model (RMIP) *without* constraints (res-coupling), i.e., constraints (res-coupling) are considered to be relaxed (by zero values as coefficients) in $\mathbb{A}'(w_a^l)$. The configuration of the dual variables π_a^l according to (6.3) ensures that the reduced cost $d(x_a)$ (to which the π_a^l variables contribute) of the x-variables reflect simultaneously the contribution of the dual variables associated with the w-variables.

The C2F constraint matrix generation idea is summarized in Algorithm 6.2. At the beginning of the algorithm it is assumed that the current RMP, i.e., LP $(\emptyset, \emptyset)$ contains all columns and rows of model (RMIP) except for the x-variables, the w-variables, and constraints (res-coupling). In a routine (referred as findInitialColumns() in Algorithm 6.2) we add a small set of initial columns to the model that consist of

- ▷ all hyperarcs that operate trips,
- ▷ all hyperarcs that have a duration of not more than 30 minutes, and
- ▷ binary slack variables for the constraints (covering).

The slack variables are introduced in order to force that the model is always feasible, while the other two criteria for the initial columns hopefully provide (more or less) meaningful

```
1  constraintMatrixGeneration( LP (∅, ∅) )
2  {
3     H := findInitialColumns();
4
5     do
6     {
7        π' := solve( LP (∅, H) );  // π' ∈ ℚ^{m'}  (m' ≤ m)
8
9        π := completeDuals( π' );  // π ∈ ℚ^m ; complete according to (6.3)
10
11       H^Δ := C2FHyperarcGeneration( π );  // see Algorithm 2.2
12
13       H := H \ H^Δ;
14
15    } while( H^Δ ≠ ∅ );
16 }
```

Algorithm 6.2: C2F constraint matrix generation.

dual variables in the first column generation round. The current RMP, namely LP $(\emptyset, \underline{H})$ is always built according to the rule denoted in Lemma 11.

Note that constraints (res-coupling) are added dynamically to the model within RotOR, i.e., constraints that read $0 \leq 0$ are not considered in LP $(\emptyset, \underline{H})$. This is the reason why the dual solution vector π' that is obtained by solving the current RMP in line 7 is of dimension $m' \leq m$, i.e., only a subset of the constraints of program (RMIP) is generated. Afterwards, the "remaining" dual variables are computed according to (6.3), which is indicated by the function completeDuals$(\cdot)$. When the dual variables are completed, a valid dual solution vector $\pi \in \mathbb{Q}^m$ for the RMP is at hand.

Then, the vector π serves as an input for the C2F hyperarc generation Algorithm 2.2, which we explain in Chapter 2. In RotOR's implementation the vector π is not computed explicitly. Instead, the dual values are computed when needed according to equation (6.3). Finally, the C2F constraint matrix generation procedure terminates if the set of new hyperarcs H^Δ is empty, i.e., if it is proven that π is a feasible solution to the dual of the LP relaxation of model (RMIP).

There is an interesting but dangerous paradox that is associated with the simultaneous generation of the x- and w-variables. If we assume that the model always contains all w-variables, we clearly only have to price the x-variables. In this case we will observe rows (res-coupling) of the form $w_a^l \leq 0$ for $a \in A$ and $l \in L$ during the column generation iterations if there are no x-variables yet that allow the variable w_a^l to be non-zero. Now, if this happens we could conclude that w_a^l is fixed to zero and, therefore, can be eliminated from the current RMP before computing its solution. In the succeeding round we would assume analogously that all w-variables are in the current RMP and continue in the same way again.

Following this approach appears as basically the same as pricing x-variables and

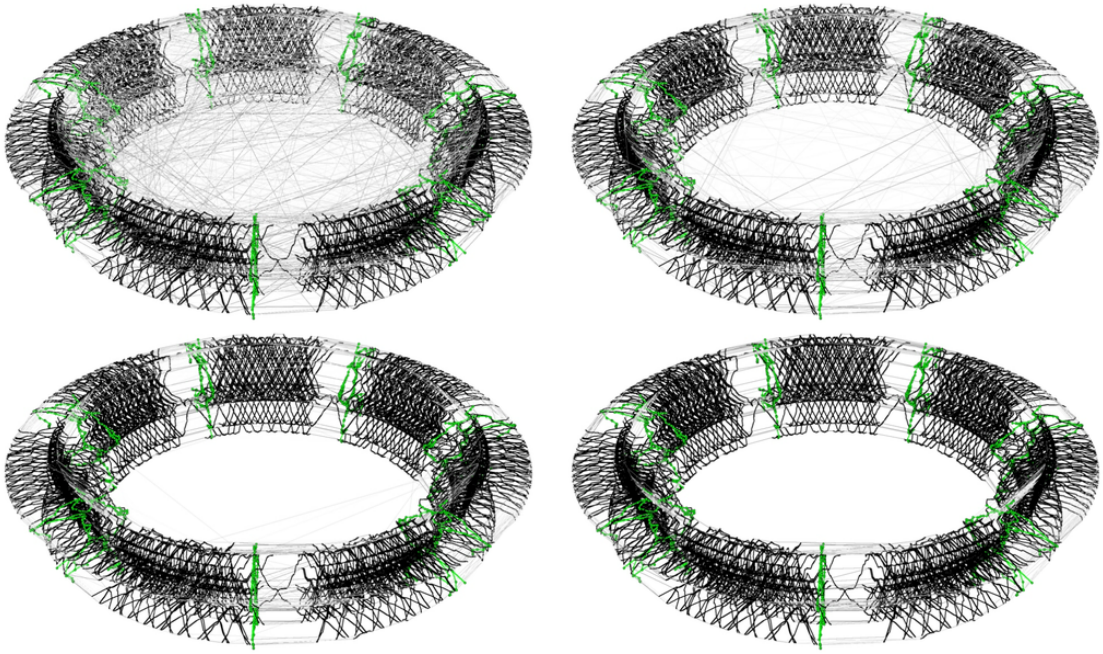


Figure 6.1.: C2F constraint matrix generation iterations.

w-variables simultaneously but without any modification to the dual solution vector as required by equation (6.3). This appears (slightly) simpler. In fact, it is a pitfall because this procedure lacks a proof that all w-variables have positive reduced cost in the end.

A counterexample that this procedure will not converge in general can be constructed as follows. Perform the procedure for any instance and check if all w-variables have positive reduced cost. If not, the counterexample is already found. Otherwise, a proper counterexample can be constructed by simply changing the objective coefficient of a w-variable that has not been generated such that their reduced costs become negative or, in other words, change the objective coefficient of the w-variable such that the corresponding dual constraints becomes violated.

Note that in program (RMIP) we assume that the objective coefficients of all w-variables are zero. We do not expect that this weak assumption results in a correct version of the alternative procedure. Indeed, a modification according to equation (6.3) needs to be performed to the dual solution vector in order to obtain a correct algorithm.

Finally, Figure 6.1 provides an illustration of four C2F constraint matrix generation iterations in terms of the rolling stock rotation torus. In this figure, hyperarcs are visualized by their standard arcs. The level of density of the arcs indicate their factionality, i.e. black arcs can be assumed to have value one, while gray arcs can be assumed to take a value around 0.5. The illustration shows the primal solution that corresponds to the dual solution computed in line 7 of Algorithm 6.2. Hyperarcs that appear inefficient are generated in early iterations: The arcs that cross complete operational days represent idle times of railway vehicles. Those arcs are expensive but replaced in further iterations

when more efficient (but still fractional) solutions come up.

6.3. Rapid Branching

The main idea of the rapid branching heuristic is that fixing a single variable to zero or one has almost no effect on the objective value of the LP relaxation of some MIP model, see our paper Borndörfer et al. (2013, [7]). Rapid branching tries to tackle this problem by a combination of cost perturbation to "make the LP relaxation more integer", a selective branching scheme to fix large sets of variables, and an associated backtracking mechanism to correct wrong decisions. This method has already been successfully applied to integrated vehicle and duty scheduling, see Weider (2007, [107]), and track allocation, see Borndörfer, Schlechte, and Weider (2010, [30]). Rapid branching belongs to the class of objective diving heuristics guided by solutions of the LP relaxation for the construction of high-quality integer solutions for large scale MIP models.

Algorithm 6.3 gives a simplified overview of RotOR's rapid branching implementation. After applying the variable fixings defined by the sets $\overline{H}$ and $\underline{H}$, we solve the current LP relaxation, see line 4. After that we call the local search algorithm of Section 11. In our implementation, we use the objective coefficients $c_h := -x_h$ for the computation of the solution of the non-maintenance relaxation of the RSRP in line 9. This is a heuristic method to round the fractional solution of the current LP relaxation to an integer feasible solution, which is usually infeasible w.r.t. maintenance constraints. It provides a starting solution for the local search method described in Section 6.4 called in line 11 of Algorithm 6.3. If the local search method finds a feasible solution RotOR updates the incumbent solution $H^\star$.

A main feature of the rapid branching search scheme is the perturbation of the objective function in line 15. We iteratively decrease the objective coefficient of each hyperarc by a linear perturbation function $\alpha : \mathbb{Q} \mapsto \mathbb{Q}$. We do not describe the detailed implementation of α since it is very technical, and refer the reader to Borndörfer et al. (2013, [7, Section 2.1]). The most important fact about the perturbation in line 15 is, that the perturbation is quadratic in the value of the corresponding hyperarc variable. The idea behind this quadratic perturbation is that variables with values close to 1 are driven towards 1.

In each perturbation round i we determine the *candidate* set $\overline{H}_i$ of hyperarc variables that have a value close to 1.0 and that are not fixed yet, see line 19. We compute a score for each of the ten candidate sets. This score states how many of the variables have become integer and how much the original objective function was increased by the perturbation. If we have not found any candidate set, we solve the remaining sub-problem by traditional B&B (of a global MIP solver) with the original objective function (line 25). Those subproblems are expected to be small. Otherwise, if we found non-empty candidate sets we sort these sets by the computed score to explore the most promising candidate sets first in line 29.

To increase the diversity of the search we choose one variable that was fixed in the previous set $\overline{H}_{i-1}$ and should not be fixed to zero in the current set $\overline{H}_i$, see line 32 to line 35.

```
1  rapidBranching (H̄, H̲)  // H̄ and H̲ denote current fixations, i.e.,
2  {                        // x_h = 1 ∀h ∈ H̄ ⊆ H and x_h = 0 ∀h ∈ H̲ ⊆ H
3
4     x := solve ( LP (H̄, H̲) );  // x ∈ [0,1]^|H|
5
6     if ( H* = ∅ ) // H* ⊆ H denotes the incumbent solution
7     {
8        // RMIP (H̄, H̲)_NM denotes the non-maintenance relaxation
9        H̃ := solve ( RMIP (H̄, H̲)_NM );  // H̃ ⊆ H
10
11       cutDateHeuristic ( H̃ );
12    }
13    for ( i ∈ {1, 2, ..., 10} ) // perturbation rounds
14    {
15       for ( h ∈ H \ (H̄ ∪ H̲) ) { c_h := c_h − α(x_h^2); }
16
17       x := solve ( LP (H̄, H̲) );  // x ∈ [0,1]^|H|
18
19       H̄_i := { h ∈ H \ H̄ | x_h ≥ 0.9 };
20
21       score ( H̄_i );
22    }
23    if ( ⋃_{i=1}^{10} H̄_i = ∅ )
24    {
25       solve ( RMIP (H̄, H̲) );  // traditional branch and cut
26    }
27    else
28    {
29       sort (H̄_1, ..., H̄_10);  // sort according to score(·);
30       for ( i ∈ {1, 2, ..., 10} )
31       {
32          H̲_i := ∅;
33          if ( i > 1 and H̄_{i−1} \ H̄_i ≠ ∅ )
34          {
35             H̲_i := { choose ( H̄_{i−1} \ H̄_i ) };
36          }
37          H̄ := H̄ ∪ H̄_i;  H̲ := H̲ ∪ H̲_i;
38
39          rapidBranching (H̄, H̲);
40
41          H̄ := H̄ \ H̄_i;  H̲ := H̲ \ H̲_i;
42       }
43    }
44 }
```

Algorithm 6.3: Rapid branching that outputs a feasible solution $H^\star$ of an RSRP instance (if successful).

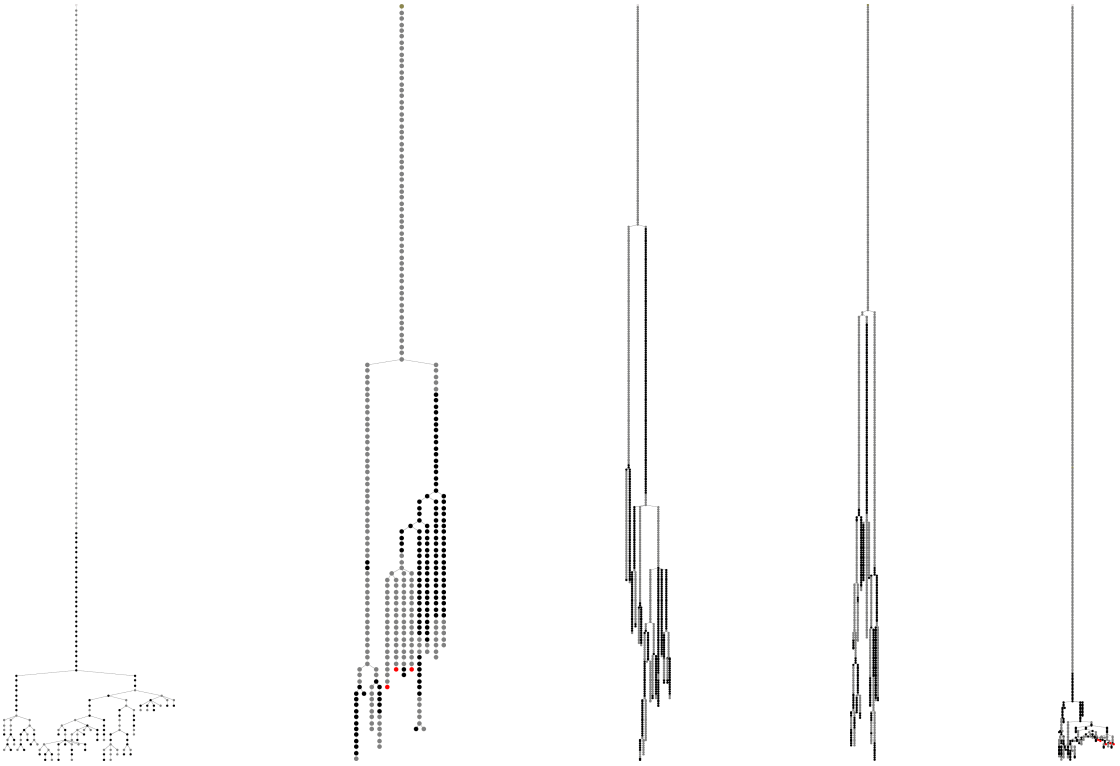


Figure 6.2.: Rapid branching search trees.

We create the search tree by a recursion, i.e., by calling the rapid branching function again. Since the recursion is called inside the loop over the candidate sets, we execute a depth first search over those candidate sets. One can view the sorted candidate sets as backtracking alternatives. In [29] these backtracking sets were constructed by an alternative strategy that is not based on the results of the perturbation rounds. This is the most important difference we made in RotOR's rapid branching adaptation.

Figure 6.2 gives three examples of search trees that arise while executing the rapid branching algorithm for industrial RSRP instances. Each continuous black or gray path represents a set $\overline{H}_i$ that have been fixed during one call of Algorithm 6.3. Note that these search trees can contain branchings with degrees more than two, e.g., the most left branch has degree three in depth four.

6.4. Cut Date Heuristic

The cut date heuristic (CDH) is an effective local search algorithm for industrial RSRP instances. The general procedure of a local search algorithm is to compute an initial solution and to iteratively improve the solution by exploring a neighborhood structure. In the case of an "infeasible solution" at the beginning of a local search procedure one tries to reduce the infeasibility first and in the case of a feasible solution one tries to

decrease the value of the objective function. We call $H_{\texttt{inf}} \subseteq H$ a *maintenance-infeasible solution* to an instance of the RSRP if $H_{\texttt{inf}}$ is a feasible solution for the non-maintenance relaxation of the RSRP but is not a feasible solution for the RSRP itself.

The main idea of the CDH is to iteratively remove the "infeasibilities" of a maintenance-infeasible solution by exploring what we call the *cut date neighborhood*. After obtaining a feasible solution we try to improve the value of the objective function by exploring the cut date neighborhood again.

In order to handle maintenance-infeasible solutions we extend the set of hyperarcs of RotOR's hypergraph as follows. Let $h \in H$ be a hyperarc and $a = (u, v) \in h$ be a standard arc of h. We introduce an artificial service node s_{art} that resets all existing maintenance constraints and extend a to an artificial service path $\{(u, s_{\text{art}}), (s_{\text{art}}, v)\}$. By this construction we are able to construct *artificial hyperarcs* that are sets of artificial service paths. Thus, any maintenance-infeasible solution $H_{\texttt{inf}} \subset H$ of a given RSRP instance, can be "made feasible" by additionally inserting artificial hyperarcs. By introducing artificial hyperarcs that have a sufficiently large objective coefficient we are able to tackle maintenance constraint violations by minimizing the objective function.

The cut date neighborhood, which is explored during the single iterations of the CDH, is constructed as follows. Each timetabled trip $t \in T$ has a departure and arrival date in the standard week. A node $v \in V$ represents the departure or arrival of a vehicle operating a timetabled trip. Therefore, also $v \in V$ is associated with a specific date which we denote by $\theta_v \in [0, \Theta) \subset \mathbb{Q}_+$ (Θ denotes the time horizon which can be assumed to be seven days). We define the function isBetween(u, v, w) for the nodes $u, v, w \in V$ that returns `true` if and only if the date associated with v *is between* the dates that are associated with u and w. More precisely, in terms of definition (4.1) we have

$$\text{isBetween}(u, v, w) \quad :\Leftrightarrow \quad \text{dur}(\theta_u, \theta_v) \leq \text{dur}(\theta_u, \theta_w).$$

We define $\text{tail}\,(h) := \{u \in V \mid \exists\, v \in V \,:\, a = (u, v) \in h\}$ as the *tail nodes* of h and $\text{head}\,(h) := \{v \in V \mid \exists\, u \in V \,:\, a = (u, v) \in h\}$ as the *head nodes* of h. Let $H^\star \subset H \cup H_{\text{art}}$ be a (possibly maintenance-infeasible) solution to a given RSRP instance and let $v \in V$ be a node. We define the *cut date neighborhood* $\text{CDN}(H^\star, v) \subseteq H$ as:

$$\begin{aligned}\text{CDN}(H^\star, v) := H^\star \cup \{\, h \in H \mid \exists\, (u, w) \in \text{tail}\,(h) \times \text{head}\,(h) \,:\\ \text{isBetween}(u, v, w) = \texttt{true}\,\}.\end{aligned}$$

Figure 6.3 illustrates the main idea of the cut date neighborhood $\text{CDN}(H^\star, v)$. It includes all hyperarcs of a given (possibly maintenance-infeasible) solution $H^\star$ as well as all hyperarcs covering a specific date that is defined by a dedicated node v.

Algorithm 6.4 outlines the CDH. We start with a possibly maintenance-infeasible solution. Let the set H_{art} denote all artificial hyperarcs. We assume that M is sufficiently large such that any infeasible solution, i.e., a solution that contains artificial hyperarcs, is much more expensive than any feasible solution. In our overall implementation, the Algorithm 6.4 runs in two phases: First it tries to remove all infeasibilities, i.e., all artificial hyperarcs, and second it tries to improve the value of the objective function of

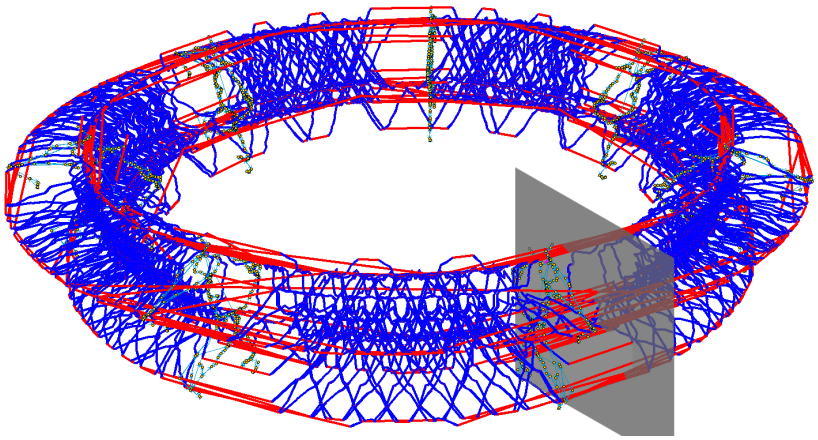


Figure 6.3.: Idea of the cut date heuristic (CDH).

```
1  cutDateHeuristic ( H* )
2  {
3     H := H ∪ H_art ;  // allow artifical hyperarcs
4
5     FindImprovements :
6
7     for ( v ∈ V )
8     {
9        Ĥ := solve ( RMIP (∅, H \ (CDN(H*, v) ∪ H_art)) ) ;
10
11       if ( Σ_{h∈H*} c_h > Σ_{h∈Ĥ} c_h )
12       {
13          H* := Ĥ ;
14          goto FindImprovements ;  // go to line 5
15       }
16    }
17    H := H \ H_art ;
18 }
```

Algorithm 6.4: Cut date heuristic (CDH) that turns a (possibly infeasible) solution $H^\star \subseteq H \cup H_{\text{art}}$ into a solution that is locally optimal w.r.t. the cut date neighborhood.

the incumbent solution. The algorithm terminates with a locally optimal solution $H^\star$, i.e., a solution that can not be improved by exploring any cut date neighborhood.

A disadvantage of the CDH heuristic is that the number of railway vehicles used in the starting solution is likely to be the number of vehicles in the locally optimal solution as well. This is because it is unlikely that a reassembly of the arcs that we cut in the torus (see Figure 6.3) will have a decreased or increased number of standard arcs that cover the cut date.

Currently the variation of the number of vehicles is handled by various starting solutions that are constructed within RotOR's rapid branching search tree. An obvious way to improve the CDH is to use more general cuts (i.e., that do not rely just on a date) of the tours. But how to efficiently find those? In fact, a promising approach is already at hand by the alternating cycle region that we investigate in Chapter 3. But for now, this remains further work.

6.5. Coupling Components

In this section we present the coupling component heuristic (CCH), which is dedicated to a particular characteristic of the LP relaxation of RotOR's hypergraph-based MIP model. This characteristic appears if the allowed vehicle configurations for a certain timetabled trip follow a certain arrangement, which can lead to extremely difficult RSRP instances. Those instances are artificial by no means, i.e., they play an important role in the industrial application at DBF.

We explain the issue by an example of a fractional solution to the LP relaxation of RotOR's model. This example has really occurred in an industrial use case. Based on the example, we identify some violated cutting planes. Even if those cutting planes indicate how to derive a more tighter LP relaxation, we suggest to solve respective instances through a dedicated type of decision variables. These variables are suitable for an exact branching scheme or a less time-consuming local search approach. The latter is currently implemented in RotOR under the name CCH and is described in the last part of this section.

6.5.1. Infeasible Couplings

An *infeasible coupling* is a non-trivial (i.e., of size two at least) vehicle configuration that is not allowed in operation for technical reasons. A particular weakness of the LP relaxation of model (RMIP) makes it possible to "simulate" infeasible couplings by fractional solutions.

Figure 6.4 illustrates a situation of interest. It shows a part of a fractional solution to the LP relaxation of model (RMIP) (where positions and orientations of the nodes have been aggregated according to the configuration layer, see Section 2.4.2). In Figure 6.4, all variables x_i have value 0.5 for $i \in \{1, 2, \ldots, 22\}$. The illustrated arrangement can appear even if the underlying instance has neither capacity, nor maintenance, nor trunk constraints, and consists of hyperarcs of size not greater than two. In other words, the flow part of model (RMIP) is enough to produce these fractionalities. The dots on the left

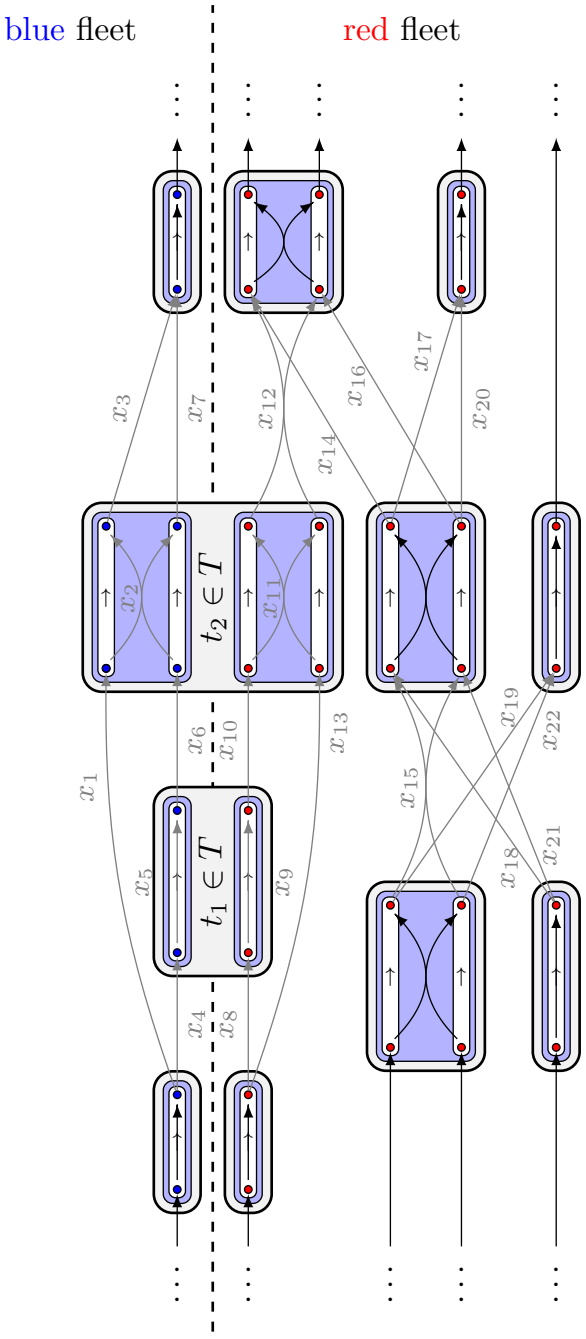


Figure 6.4.: Example of a solution to the LP relaxation with infeasible couplings.

and right of the figure indicate that five railway vehicles come from somewhere, traverse the (fractional) part of the solution that is shown, and leave to somewhere. The dashed line indicates that we deal with two fleets, i.e., a red and a blue one, in this instance. The crucial circumstances are:

- ▷ The railway vehicles of the blue fleet can not be coupled to vehicles of the red fleet, i.e., any vehicle configuration of the form $\{\ldots, \text{Red}, \ldots, \text{Blue}, \ldots\}$ is an infeasible coupling in this instance.

- ▷ A timetabled trip shares alternative *non-trivial* vehicle configurations of the red and blue fleet.

At trip $t_2 \in T$ in Figure 6.4 both circumstances come together and lead to a fractional assignment of non-trivial vehicle configurations to t_2. More precisely, $x_2 = 0.5$ contributes one half of the vehicle configuration $\{\text{Blue}, \text{Blue}\}$, while $x_{11} = 0.5$ contributes another one half of the vehicle configuration $\{\text{Red}, \text{Red}\}$ to the operation of t_2. An obvious interpretation of this situation is that the vehicle configuration $\{\text{Blue}, \text{Red}\}$ is assigned to t_2. In other words, it is possible to couple fleets together in the solution space of the LP relaxation of model (RMIP) that are forbidden to be coupled together in an integer feasible solution, i.e., infeasible couplings can appear.

Note that, it is possible to also find fractionalities where "only" trivial vehicle configurations (i.e., of size one) are involved as we can see at $t_1 \in T$ in Figure 6.4. But we observed that the computational hardness increases *dramatically* when non-trivial vehicle configurations participate.

An illustrative explanation for this increase in hardness compared to instances with only trivial vehicle configurations is as follows. Obviously, in any integer feasible solution we either have $x_2 = 0$ or $x_{11} = 0$. By $x_{11} = 0$ we impose that the blue fleet needs to cover t_2 once more than in the original fractional solution. Typically, this additional cover is performed either by an additional railway vehicle of the blue fleet or by a vehicle of the blue fleet that does not operate any timetabled trip, i.e., that idles, at the time when t_2 takes place. We suspect that it is much more likely that an additional vehicle of the blue fleet will be used under $x_{11} = 0$ when we compare to a similar situation where only trivial vehicle configurations are involved (as at t_1). The reason for this conjecture is that for t_1 we already know how to connect t_1 by railway vehicles of the blue fleet and, therefore, we probably do not need an additional railway vehicle when branching $x_5 = 1$ (or $x_9 = 0$).

If the argumentation above appears too esoteric to the reader, our computational observation remains: If infeasible couplings appear in fractional solutions the number of railway vehicles of fleets is over- or underestimated by the fractional solutions[1]. This causes huge integrality gaps and leads to particularly hard RSRP instances.

[1] For the RSRP instance from that Figure 6.4 is derived, the LP solution uses one vehicle of the blue fleet and nine vehicles of the red fleet, while an optimal solution contains two blue and eight red vehicles.

6.5.2. Cutting Planes

An obvious way to reduce the integrality gap is to apply linear inequalities that forbid dedicated fractional structures. For the instance in Figure 6.4 it is worth to consider its *conflict graph* according to the definition of Atamtürk, Nemhauser, and Savelsbergh (1998, [19]). The conflict graph (N, E) is an undirected graph where each node of N identifies a binary variable $x_h \in \{0, 1\}$ associated with $h \in H$ or its complementary variable $1 - x_h \in \{0, 1\}$. Thus, we have $|N| = 2 \cdot |H|$. The conflict graph (N, E) contains an edge $\{x, y\} \in E$ if and only if at most one of both variables can be one (i.e., $x \cdot y = 0$) in any feasible solution to model (RMIP). Obviously, for any clique $Q \subseteq N$ of the conflict graph (N, E) the *clique inequality* $\sum_{x \in Q} x \leq 1$ holds. In addition, any cycle $C \subseteq N$ with $|C|$ odd implies an *odd cycle inequality* $\sum_{x \in C} x \leq \left\lfloor \frac{|C|-1}{2} \right\rfloor$.

Figure 6.4 shows many violated clique and odd cycle inequalities. Examples are:

$$\begin{aligned} x_{15} + x_{19} + x_{22} &\not\leq 1 && \text{(clique)} \\ x_{15} + x_{18} + x_{21} &\not\leq 1 && \text{(clique)} \\ x_3 + x_7 + x_{12} &\not\leq 1 && \text{(clique)} \\ x_3 + x_7 + (1 - x_2) &\not\leq 1 && \text{(clique)} \\ x_1 + (1 - x_2) + x_6 + (1 - x_5) + x_4 &\not\leq \left\lfloor \tfrac{5-1}{2} \right\rfloor && \text{(odd cycle)} \end{aligned}$$

We use SCIP [13] in order to test the effects of separating these two classes of inequalities for some small instances that can be handled without column generation. The following has been observed.

Many clique inequalities are rapidly found in early separation rounds and slightly increase the lower bound. But clique inequalities do not significantly help in resolving infeasible couplings. E.g., the first two examples of violated clique inequalities do not even involve fractional vehicle configurations. And, for example, the solution is only very slightly changed by separating the third exemplary clique inequality on its own: x_{12} becomes zero but is just split into its standard arcs, i.e., $x_{12} =: x_{\{a_1,a_2\}} = 0.5$ is replaced by $x_{\{a_1\}} = x_{\{a_2\}} = 0.5$. For the hypergraph assignment problem (HAP) (that the RSRP extends, see Section 1.6.2), clique inequalities turn out to be computationally very effective, see Heismann (2014, [62]). Unfortunately, this does not carry over to our setting.

After many clique inequalities have been separated by SCIP some progress is made by separating odd cycle inequalities in further separation rounds. For example, the fractionality around x_{12} in Figure 6.4 (which is was already tried to separate by the third clique inequality as explained above) is separated again by the odd cycle inequality $x_{20} + x_{17} + x_{14} + x_{\{a_1\}} + (1 - x_{11}) + x_{\{a_2\}} + x_{16} \not\leq \left\lfloor \frac{7-1}{2} \right\rfloor$. Finally, we observed that the separation of odd cycle inequalities involves many separation rounds, but, unfortunately, does not significantly resolve any infeasible couplings (w.r.t. the instances that we investigated).

What has been found out during our investigations is that an aggressive separation of so called $\{0, 1/2\}$-*cuts* [36] leads to a very small integrality gap and resolves infeasible couplings. But the separation of those inequalities is $\mathcal{NP}$-hard and turns out to be too

time consuming even for very small instances.

From these observations we decide to not pursue a separation approach in order to resolve infeasible couplings within RotOR.

6.5.3. Coupling Components

For the set of fleets F we are given a set of vehicle configurations that determine which fleets of F can be coupled together, see Section 4.1. These relations can be modeled by an undirected graph (F, E) where the node set is the set of fleets and where an edge $\{f_1, f_2\} \in E$ exists if and only if there exists a vehicle configuration that contains both fleets $f_1 \in F$ and $f_2 \in F$ at the same time. We call (F, E) the *coupling graph* and denote by $\mathbb{F}_1, \ldots, \mathbb{F}_n \subseteq F$ with $n \in \mathbb{N}$ its connected components. Obviously, if (F, E) decomposes into $n > 1$ connected components, infeasible couplings can appear. We call the connected components $\mathbb{F}_1, \ldots, \mathbb{F}_n$ of (F, E) *coupling components*.

An example reads as follows. We consider the set of fleets $F = \{\text{Blue}, \text{Red}, \text{White}\}$ and assume that the vehicle configurations $\{\text{Blue}\}$, $\{\text{Red}\}$, $\{\text{White}\}$, $\{\text{Blue}, \text{Blue}\}$, $\{\text{Red}, \text{Red}\}$, $\{\text{White}, \text{White}\}$, and $\{\text{Blue}, \text{Red}\}$ are given. Then, the coupling graph (F, E) is $(\{\text{Blue}, \text{Red}, \text{White}\}, \{\{\text{Blue}, \text{Red}\}\})$ and decomposes into the two coupling components $\mathbb{F}_1 = \{\text{Blue}, \text{Red}\}$ as well as $\mathbb{F}_2 = \{\text{White}\}$.

Obviously, in an integer feasible solution the railway vehicles that operate a dedicated timetabled trip $t \in T$ need to be taken from fleets of a single coupling component of (F, E). From the previous section we know in which situations particular computational hardness in terms of infeasible couplings can arise. Therefore, we define the set $T' \subseteq T$ as the subset of timetabled trips such that at least two non-trivial vehicle configurations with railway vehicles from different coupling components are allowed for each $t \in T'$. Typically (but not always), we have $|T'| \ll |T|$ in industrial RSRP instances.

Our idea is simply to consider directly the decision from which coupling component the vehicles for $t \in T'$ are taken. To this end, we introduce additional binary variables $q_{t,i} \in \{0, 1\}$ for all $t \in T'$ and for all $i \in \{1, \ldots, n\}$. The variable $q_{t,i}$ takes value zero if and only if $t \in T'$ is not operated by a vehicle from coupling component $\mathbb{F}_i$. We couple these new q-variables to the x-variables of model (RMIP) by

$$\sum_{h \in H(\,t, \mathbb{F}_i\,)} x_h = q_{t,i} \quad \forall\, t \in T',\, i \in \{1, \ldots, n\} \tag{6.4}$$

where $H(\,t, \mathbb{F}_i\,) \subseteq H(t)$ denotes all hyperarcs that cover t with at least one railway vehicle of a fleet of the coupling component $\mathbb{F}_i$.

The intention behind the q-variables is as follows. If they are all fixed to binary values the particular computational hardness caused by infeasible couplings is resolved within the branching tree. Moreover, we expect that imposing $q_{t,i} = 0$ has almost the same computational effect for the resulting sub-problem as $q_{t,i} = 1$ has. Note that this is not the case for the x-variables, i.e., imposing $x_h = 1$ has a much stronger meaning than $x_h = 0$. The way how we deal algorithmically with these variables is the subject of the next section.

6.5.4. Coupling Component Heuristic

An obvious idea is to branch on the q-variables that we introduced in the previous section. Our investigations showed that branching on a particular variable $q_{t,i}$ affects the values of the other q-variables from only very slightly to not at all. This results in a large branching tree even if the number of fractional q-variables is small, say 20. Therefore, a complete branching scheme on top of these variables turns out to be too time consuming (for the instances that we considered).

Even rapid branching on the q-variables more or less fails for respective instances. An illustrative reason for this can be seen by considering Figure 6.4 on page 195 again. All variables that cause infeasible couplings take value 0.5. The perturbation function of rapid branching perturbates all these variables in a single iteration by decreasing their objective coefficients. Our observation is that the appearance of infeasible couplings is enhanced. For example, if it was effective to operate trip t_2 by one half of $\{\text{Blue}, \text{Blue}\}$ and another one half of $\{\text{Red}, \text{Red}\}$ for trip t_2 before perturbation, it is even more effective under objective perturbation.

Finally, our suggestion is to perform a simple local search over the q-variables. To this end, we compute initial binary values $q^\star \in \{0,1\}^{|T'|\cdot n}$ at the basis of the LP relaxation of model (RMIP) as follows. Let $\widetilde{q}_{t,i} \in [0,1]$ be the fractional values of the q-variables if we compute an optimal solution to the LP relaxation of model (RMIP) including the q-variables (as continuous variables) and constraints (6.4). We round to initial binary values for the q-variables as follows. For each timetabled trip $t \in T'$ we (arbitrarily) choose a coupling component $\mathbb{F}_i$ such that $\widetilde{q}_{t,i} = \max_{j=1,\dots,n}\{\widetilde{q}_{t,j}\}$ and define $q^\star_{t,i} := 1$. All other $q^\star$-variables are defined to be zero.

We denote by $c(q^\star)$ the objective value of the LP relaxation of model (RMIP) under constraints $q_{t,i} = q^\star_{t,i}$ for all $t \in T'$ and $i \in \{1,\dots,n\}$. Thus, $c(q^\star)$ is a lower bound (in the sense that the x-variables are still allowed to be fractional) for the distribution of the coupling components of the coupling graph among the trips of T' according to the rounded values $q^\star$. We use the values $q^\star$ as starting point for the CCH that locally minimizes the lower bound $c(q^\star)$ by moving timetabled trips between coupling components. This is denoted by Algorithm 6.5 on page 200, i.e., while(couplingComponentHeuristic($z^\star$)); states the execution of the CCH.

In order to find an improvement for $q^\star$ we iterate all timetabled trips of T' and test whether it is beneficial to move a single trip to another coupling component. This is made by appropriately fixing the q-variables as denoted in lines 7 and 10 of Algorithm 6.5. The bottleneck of the CCH is the solution of the remaining LP relaxation, which is often only a little smaller than the original LP relaxation (especially when $|T'| \ll |T|$, which is often the case in the industrial instances of DBF).

Since only one coupling component move is performed in each iteration, the CCH can be seen as a "1-opt" improvement heuristic. In this respect, the CCH searches through a rather small neighborhood. This neighborhood (or the "look-ahead") can be increased by a backtracking scheme that works as follows.

Assume that $q^\star$ is the current incumbent solution and is improved by Algorithm 6.5 to the new solution $q^{\star\star}$ (i.e., $c(q^{\star\star}) < c(q^\star)$). In this situation, we store (i.e., *backtrack*) the

```
1  couplingComponentHeuristic( q* ) // q* ∈ {0,1}^{|T'|·n}
2  {
3    for( t ∈ T' )
4    {
5      for( i ∈ { j | q*_{t,j} = 0 } )
6      {
7        q_{t,i} := 1; // move t to another coupling component
8
9        // fix other q-variables according to q*
10       for( u ∈ T' \ {t} ) for( j ∈ {1,...,n} \ {i} ) { q_{u,j} := q*_{u,j}; }
11
12       // solve LP relaxation and store q-variables
13       q := solve( LP(∅,∅) ); // q ∈ {0,1}^{|T'|·n}
14
15       if( c(q) < c(q*) )
16       {
17         q* := q;
18         return true;  // report improvement
19       }
20     }
21   }
22   return false; // report local optimality
23 }
```

Algorithm 6.5: Coupling component heuristic (CCH).

old incumbent solution $q^\star$ in a global heap that is sorted by the objective function values (i.e., by $c(q^\star)$). Then, we call couplingComponentHeuristic($q^\star$) again but ensure that the move that leads to $q^{\star\star}$ is performed in the very last iteration (this is accomplished by changing the execution order of the first two for-loops) of Algorithm 6.5[2]. This is made in order to test if there is an improvement of $q^\star$ that is better than $q^{\star\star}$.

Note that this procedure would completely enumerate a tree of solutions where each edge represents a coupling component move that improves the parent node's solutions. On the one hand, this can result in very large search trees. On the other hand, the solution quality might be significantly increased by backtracking. The computational compromise that is implemented in RotOR is to restrict the number of backtracks by a fixed parameter value that is chosen as, say, maximal three backtracks.

Naturally, RotOR only calls the CCH if $T' \neq \emptyset$. In addition, the CCH is called only once in the root of the rapid branching search tree. In this way, the distribution of the coupling components among the crucial timetabled trips T' as computed by the CCH is also the result of the final rolling stock rotations that RotOR returns. Of course, we recognize that RotOR's current approach to deal with infeasible couplings is simple and less sophisticated, i.e., a 1-opt that is called only for one time. But until now nothing else turns out to be superior in the respective situation.

[2]Note that the backtracking scheme is not illustrated by Algorithm 6.5.

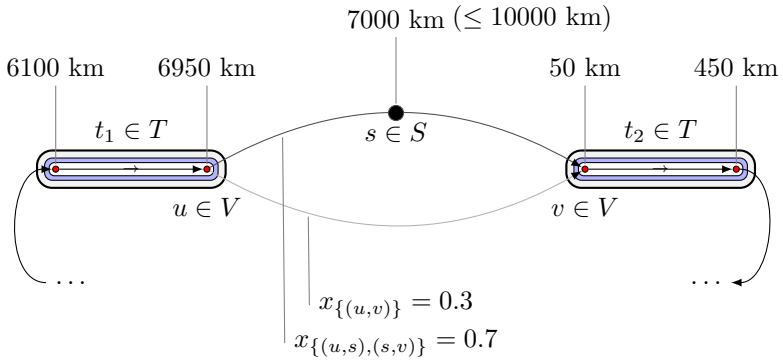


Figure 6.5.: Fractional service path $\{(u, s), (s, v)\} \in H$.

6.6. Fractional Maintenance Relaxation

If rolling stock goes under maintenance in order to reset a maintenance constraint $l \in L$, it can be expected that the actual resource consumption of a feasible resource path $P \subset A$ that enters a service node of S_l is smaller than the maximal resource consumption that is actually allowed for P, see page 20. More precisely, the *resource slack* $\bar{r}_l(P)$ that we denote as

$$\bar{r}_l(P) := U_l - \sum_{a \in P} r_l(a) - \sum_{s \in S(P) \setminus S_l} r_l(s) \tag{6.5}$$

can have an almost arbitrary value, i.e., $\bar{r}_l(P) \in (0, U_l]$.

Resource slacks have been experienced (by us) as challenging. In order to resolve those challenges we develop a special relaxation of model (RMIP) that is used by a heuristic application in RotOR's overall algorithm: We apply this relaxation in order to derive the so called *fractional maintenance bound*, which is described in this section.

Figure 6.5 provides an example for the situation of interest. We consider a feasible solution $H^\star \subseteq H$ to a RSRP instance that consists of a single resource path that starts at $s \in S$ traverses some unspecified timetabled trips between the timetabled trips $t_2 \in T$ as well as $t_1 \in T$, and, finally, comes back to the service node s. The cumulated distance along the resource path is illustrated in the figure and takes the value of 7000 km just before the maintenance service at s is started. We consider this as a feasible solution to this RSRP instance that has a single distance dependent maintenance constraint $l \in L$. The solution $H^\star$ is feasible because we assume the maintenance upper bound $U_l = 10000$ km, which is not exceeded.

We now relate this situation to the LP relaxation of RotOR's model, i.e., we ask how does a solution to the LP relaxation appear? To this end, we try to construct the instance in such a way that it is almost trivial to solve but still has fractionalities. We do this by assuming $H = H^\star \cup \{\{(u, v)\}\}$. Therefore, the only proper variable decisions in the RSRP instance are represented by $x_{\{(u,v)\}}$ and $x_{\{(u,s),(s,v)\}}$. We observe that $x_{\{(u,v)\}} = 0$ and $x_h = 1$ for all $h \in H^\star$ is the optimal solution in this situation.

Unfortunately, this is not the solution to the LP relaxation of model (RMIP) where we

have $x_{\{(u,v)\}} = 0.3$ and $x_{\{(u,s),(s,v)\}} = 0.7$ (and $x_h = 1$ for $h \in H \backslash \{\{(u,v)\}, \{(u,s),(s,v)\}\}$) as soon as $c(\{(u,v)\}) < c(\{(u,s),(s,v)\})$. It is easy to see that this solution is feasible, cheaper than the integer feasible solution $H^\star$, and, unfortunately, fractional. In order to prove that it is also optimal, we consider the constraints (res-coupling) for $(u,s) \in A$ and $l \in L$:

$$7000 = w^l_{(u,s)} \leq U_l \cdot x_{\{(u,s),(s,v)\}} = 10000 \cdot x_{\{(u,s),(s,v)\}}.$$

This inequality is tight for $x_{\{(u,s),(s,v)\}} = 0.7$, which proves optimality of the solution to the very trivial RSRP instance that we consider. This shows that the contribution of the resource slack, which has a value of 3000 km in the integer feasible solution $H^\star$, is not reflected in the fractional solution to the LP relaxation.

The assumption $c(\{(u,v)\}) < c(\{(u,s),(s,v)\})$ is reasonable in our application because the service path $\{(u,s),(s,v)\}$ can be assumed to be more expensive than just $\{(u,v)\}$. Particular reasons for that are the higher deadhead distance to reach the service node s and the cost for the maintenance service that is performed at s, see objective features three and seven in Section 4.12. Especially instances with high maintenance cost have caused algorithmic problems in the past. This is not surprisingly because the integrality gap of an instance can be increased arbitrarily by increasing the cost for a maintenance service as we have seen.

The rolling stock rotations pass through further tests and changes until they are implemented in operation. Those changes especially affect the distribution of maintenance services. Therefore, the minimization of cost for maintenance services is of less importance in our application but can not be completely ignored as we discovered during cooperating with DBF. Therefore, we develop a heuristic approach that is based on the *fractional maintenance relaxation* of model (RMIP) and is dedicated to RSRP instances with high maintenance cost, which appear from time to time.

In order to denote the fractional maintenance relaxation we define the set of *connections* $C \subset V \times V$ that is composed of all pairs $(u,v) \in V \times V$ for that u represents an arrival node and v represents a departure node in RotOR's hypergraph. In addition, we introduce a binary decision variable $z_{(u,v)} \in \{0,1\}$ for each connection $(u,v) \in C$. The variable $z_{(u,v)}$ takes value one if the arrival associated with u is connected to the departure associated with v in a solution. We further define the set $H(\,(u,v)\,) \subset H$ for $(u,v) \in C$ that denotes the set of all hyperarcs $h \in H$ for that $(u,v) \in h$ or $\{(u,s),(s,v)\} \subseteq h$ holds. In other words, we denote $h \in H(\,(u,v)\,)$ for $(u,v) \in C$ if h directly connects u to v or implements a service path through a service $s \in S$. The fractional maintenance relaxation of model (RMIP) is

$$\begin{aligned} &\text{LP relaxation of model (RMIP)} && \text{constrained by} && \\ &z_{(u,v)} - \sum_{h \in H(\,(u,v)\,)} x_h = 0 && \forall\,(u,v) \in C && \text{and} \\ &z_{(u,v)} \in \{0,1\} && \forall\,(u,v) \in C. && \end{aligned} \tag{FMR}$$

Model (FMR) is to compute a solution to the LP relaxation of model (RMIP) with the additional constraint that all connection variables need to be binary, which is the "only"

```
1  fractionalMaintenanceBound( H* )
2  {
3    // fix all connections implied by H* in model (FMR)
4    for(  (u,v) ∈ {(u,v) ∈ C | ∃h ∈ H* : h ∈ H((u,v))}  )
5    {
6      z_(u,v) := 1;
7    }
8
9    (FMR)(H*) := solve((FMR)); // (FMR)(H*) ∈ ℚ+ is the objective value
10
11   return (FMR)(H*);
12 }
```

Algorithm 6.6: Computation of the fractional maintenance bound for a feasible solution $H^\star \subseteq H$ (in terms of model (RMIP)).

purpose of the additional z-variables and equalities. Therefore, the optimal objective value of model (FMR) is not smaller than the optimal objective value of the LP relaxation of model (RMIP) and not greater than the optimal objective value of model (RMIP). This indicates that model (FMR) could be computationally easier compared to model (RMIP), which turns out to be true in experiments that we made. Of course, the underlying problems of models (RMIP) and (FMR) are different. But this does not stop us from using model (FMR) in a heuristic as follows.

Fractional maintenance bound. Obvious exact ways to handle large cost for maintenance services are to use cutting planes or a complete branching scheme. These techniques aim at increasing the lower bound in order to decrease the integrality gap. The idea of the fractional maintenance bound is different to both. In fact, it can be seen as a "heuristic objective value" for the current incumbent solution.

The motivation for the fractional maintenance bound is that computing the optimal solution to model (FMR) for the example above produces the same fractional solution as the LP relaxation yields. Therefore, model (FMR) is completely *invariant* w.r.t. the cost difference $c(\{(u,s),(s,v)\}) - c(\{(u,v)\})$ between performing the direct connection of $u \in V$ to $v \in V$ and the connection of u to v via s. This is a desired property that we *heuristically* transfer into RotOR's overall algorithm as indicated in Algorithm 6.6.

In Algorithm 6.6 we "lift" the current incumbent solution $H^\star$ to a solution to model (FMR) by fixing the z-variables according to the connections implied by $H^\star$, see line 6. The remaining problem is a very (very) small LP model in which fractional values of service paths (as illustrated in Figure 6.5) are possible (and desired).

The purpose of the returned value $(\text{FMR})(H^\star) \in \mathbb{Q}_+$, i.e., the fractional maintenance bound of $H^\star$ is as follows. Let $\underline{z} \in \mathbb{Q}_+$ be the current optimal objective value of the LP relaxation of model (RMIP). Obviously, we have

$$\underline{z} \leq (\text{FMR})(H^\star) \leq c(H^\star).$$

Therefore, the value $c(H^\star) - (\text{FMR})(H^\star)$ is an error w.r.t. the approximation quality of the LP relaxation of model (RMIP) that is definitely caused by resource slacks. This error occurs even if $H^\star$ is an optimal solution to model (RMIP). Clearly, also the value $c(H^\star) - \underline{z}$ (i.e., the traditional absolute integrality gap) is an error in the same sense. But $c(H^\star) - (\text{FMR})(H^\star)$ is *systematic*. Based on this observations we choose a small $\epsilon \in \mathbb{Q}_+$ and denote the following heuristic bounding criterion:

$$(\text{FMR})(H^\star) - \underline{z} \leq \epsilon \quad \Rightarrow \quad \text{Detach the current (rapid) branching node.} \tag{6.6}$$

The bounding criterion (6.6) is invariant w.r.t. the systematic error $c(H^\star) - (\text{FMR})(H^\star)$. An interpretation of criterion (6.6) is as follows. If $(\text{FMR})(H^\star) - \underline{z}$ is small, we assume that $H^\star$ is near optimal. If $(\text{FMR})(H^\star) - \underline{z}$ is large, we rather expect that $c(H^\star)$ can be further improved. This criterion is integrated as a heuristic pruning rule within RotOR's rapid branching scheme that we explain in Section 6.3.

6.7. Handout Optimization

As already mentioned, RotOR does not "only" compute rolling stock rotations. In fact, RotOR also provides an advice, namely a handout, for the visualization of each rotation that is computed. In this section we describe RotOR's approach to compute the handouts. We refer to Section 4.10.3 for the explanation what a handout formally is and how it interplays with RotOR's regularity model.

As already mentioned, a rolling stock rotation is translated into a technical description that can be imagined as cycle covering rectangles. The rectangles indicate timetabled and deadhead trips as well as maintenance services, see Figure 4.24 on page 153.

The rectangles are naturally arranged in so called segments $\mathcal{S}$ that correspond to individual days of operation $\mathbb{D} = \{\text{Mon}, \ldots, \text{Sun}\}$, again see Figure 4.24 on page 153. A rolling stock rotation that traverses the standard week for $\mathfrak{v} \in \mathbb{N}$ times requires exactly $\mathfrak{v}$ railway vehicles to be operated and is composed of exactly $7 \cdot \mathfrak{v}$ segments, i.e.,

$$|\mathcal{S}| = 7 \cdot \mathfrak{v}.$$

```
1  handoutHeuristic( S ) // S is the set of segements
2  {
3    // partition S into blocks B = {B_1,...,B_v}, B_i ⊆ S, |B_i| = 7
4    B := maximizeSimilarities(S);
5
6    // compute bijection Ω_B : B ↦ [v]
7    Ω_B := maximizeSuccessors(B);
8
9    // finally create Ω : S ↦ [v]
10   for( B ∈ B ) { for( s ∈ B ) { Ω(s) := Ω_B(B); } }
11 }
```

Algorithm 6.7: Two-stage handout heuristic.

RotOR's task is now to compute a handout $\Omega : \mathcal{S} \mapsto [\mathfrak{v}] = \{1, 2, \ldots, \mathfrak{v}\} \subset \mathbb{N}$ for a certain rolling stock rotation requiring $\mathfrak{v}$ railway vehicles. To this end, a construction heuristic is applied. This heuristic has two stages that are denoted in Algorithms 6.8 as well as 6.9 and are arranged as outlined by Algorithm 6.7.

As denoted in Algorithm 6.7, the construction heuristic partitions the segments into so called blocks $B_1, \ldots, B_{\mathfrak{v}} \subseteq \mathcal{S}$ where each block is of size seven. This forms the first stage. In the second stage, a value of $[\mathfrak{v}]$ is assigned to each of the blocks. Then, the segments take over the values assigned to their blocks. The algorithm always produces a feasible handout but not necessarily an optimal one (in terms of program (SILO), see page 155).

```
1  maximizeSimilarities(S) // S is the set of segments
2  {
3    B := ∅;
4
5    // initialize blocks
6    for( s ∈ S : D(s) = Mon ) { B := B ∪ {s}; }
7
8    for( d ∈ D \ {Mon} )
9    {
10     S_d := {s ∈ S | D(s) = d}; // |S_d| = v
11
12     // assign segments of S_d to blocks B by solving
13   solve( max{ Σ_{s∈S_d} Σ_{B∈B} Σ_{s_B∈B} SI(s, s_B) z_sB  |  Σ_{s∈S_d} z_sB = 1   ∀B ∈ B,
                                                               Σ_{B∈B} z_sB = 1   ∀s ∈ S_d,
                                                               z_sB ∈ {0,1}   ∀s ∈ S_d, B ∈ B } );
14
15     // update blocks B according to the optimal assignment
16     for( B ∈ B ) { for( s ∈ S : z_sB = 1 ) { B := B ∪ {s}; } }
17   }
18 }
```

Algorithm 6.8: First stage of the handout heuristic.

The first stage of the handout heuristic, i.e., the partition of the segments into blocks, is denoted in Algorithm 6.8. The first steps initialize the set of blocks such that each block contains exactly one segment associated with Monday (the choice of Monday is arbitrary). Afterwards, the blocks are increased iteratively. The iterations are performed for each day of operation independently. In every iteration a standard assignment problem (AP) is set up and solved. The denoted AP is defined through the bipartite graph for which the two node parts are the set $\mathcal{S}_d$ (all segments associated with the day $d \in \mathbb{D}$ of operation of the current iteration) and the set of blocks $\mathcal{B}$.

The binary variable[3] $z_{sB} \in \{0, 1\}$ decides if the segment $s \in \mathcal{S}_d$ w.r.t. $d \in \mathbb{D}$ is assigned

[3] Note that the z-variables used in the first stage of the handout heuristic are different from the z-variables of Section 6.6 (in terms of the fractional maintenance bound), Section 4.10.3 (with program (SILO)), and are also defined differently in the second stage of the handout heuristic.

```
1  maximizeSuccessors(ℬ) // ℬ are 𝔳 blocks of segments
2  {
3    // compute assignment relaxation of
4    // ATSP over graph (ℬ, ℬ × ℬ) by solving
5
6    solve ( max { Σ_{B1∈ℬ, B1∋s1} Σ_{B2∈ℬ, B2∋s2} LO(s1,s2) z_{B1B2} |
                   Σ_{B1∈ℬ} z_{B1B2}  = 1          ∀ B2 ∈ ℬ,
                   Σ_{B2∈ℬ} z_{B1B2}  = 1          ∀ B1 ∈ ℬ,
                   z_{B1B2}  ∈ {0,1}               ∀ B1, B2 ∈ ℬ } );
7
8    // compute Ω_ℬ : ℬ ↦ [𝔳] by patching subtours as follows
9    v := 1;
10
11   for( B1 ∈ ℬ )
12   {
13     if( Ω_ℬ(B1) is not computed yet )
14     {
15       Ω_ℬ(B1) := v;
16       B1 := B2; // s.t. z_{B1B2} = 1
17       v := v + 1;
18       goto 13;
19     }
20   }
21 }
```

Algorithm 6.9: Second stage of the handout heuristic.

to the block $B \in \mathcal{B}$. Note that this is different from program (SILO). The objective function of this AP is designed to maximize the similarities (denoted by $\mathrm{SI}(s_1, s_2) \in \mathbb{N}$ for $s_1, s_2 \in \mathcal{S}$, see Section 4.10.3) of the segments within the blocks.

The second stage of the handout heuristic is denoted in Algorithm 6.9. The blocks $\mathcal{B} = \{B_1, \ldots, B_{\mathfrak{v}}\}$ with $|B| = 7$ for all $B \in \mathcal{B}$ that are created in the first stage serve as input data for this procedure. It remains to assign a value of $[\mathfrak{v}]$ to each block of $\mathcal{B}$.

In other words, a permutation of the blocks $\mathcal{B}$ has to be found. This is an asymmetric traveling salesman problem (ATSP) in that the blocks $\mathcal{B}$ take over the role of the cities and the weight of a connection between blocks corresponds to the number of logical turns (denoted by $\mathrm{LO}(s_1, s_2) \in \mathbb{N}$ for $s_1, s_2 \in \mathcal{S}$, see Section 4.10.3) between them. Not surprisingly, we also solve this ATSP heuristically as denoted in Algorithm 6.9. At first, the assignment relaxation[4] of the ATSP is solved in the second stage of the handout heuristic. There, each binary variable $z_{B_1B_2}$ is equal to one if and only if block $B_2 \in \mathcal{B}$ is the directed successor of block $B_1 \in \mathcal{B}$ in the handout (assuming no subtours). Possibly appearing subtours are patched in order to form a proper Hamiltonian cycle (or path) through the blocks $\mathcal{B}$.

[4] We promise that this is the last AP that appears in this thesis.

As presented, we handle the visualization of the rolling stock rotations in a post-processing routine that calls the explained algorithms. This may seem as a contradiction to our claim that regularity can be handled completely by RotOR's hypergraph approach (especially in an integrated manner). But, still, we adhere to this statement because we feel that the visualization of rolling stock rotations should not cause any operational cost. This is trivially guaranteed by our post-processing approach. In addition, we remark (again, see Section 4.10.3) that the quality of a handout visualization strongly depends one the amount of regularity patterns that have been accomplished w.r.t. turns and trips. These patterns do cause operational costs and are directly integrated in RotOR's model.

Indeed, during our cooperation with DBF it has been repeatedly and, moreover, seriously discussed whether it is necessary to also consider the handouts of the rolling stock rotations in an integrated manner. We claim in this thesis, that a post-processing routine is sufficient if the regularity patterns for turns and trips are integrated in the optimization of the rolling sock rotations. This was not evident at the beginning of our cooperation with DBF (not at all). In this way, we argue that we integrated as much regularity as necessary (but not more than that).

Chapter 7.

RotOR's Computations and Applications

This chapter concludes the thesis and is dedicated to computational results as well as applications of RotOR.

In all our previous publications various computational results of RotOR have already been published, see references [1] to [10]. However, the results presented in this chapter are not already published in former publications, except for Section 7.4, which is [11, Section 3.3]. For the experiments of Section 7.3 RotOR version 2.4 has been used.

The chapter is organized as follows. After describing some implementation details of RotOR in Section 7.1, we introduce the way how we present computational results here in Section 7.2. Then, we report computational results for 116 industrial instances of the rolling stock rotation problem (RSRP) provided by DB Fernverkehr AG (DBF) in Section 7.3.

Finally, we illustrate two (one rather academic and one proper industrial) exemplary RSRP applications for that RotOR has been used. In fact, the list of applications for that RotOR is appropriate is too long for a thesis. Typically, one of the following questions are investigated:

- How to operate a given timetable in detail?
- What are the influences of parts of the objective function?
- What could be the profit of an investment in infrastructure capacity?
- A new intercity express (ICE) fleet should be constructed. Should the new railway vehicles be able to be coupled together or not?
- Miraculously, new railway vehicles are available. How to use them?
- Why do the ICEs often depart or arrive in an orientation different from the timetable announcement and what is the price to resolve this issue?
- Are the current maintenance facilities appropriately located, i.e., what is the gain of alternative locations?
- What is the price of long turn durations?
- What is the price of regularity?
- How to operate a timetable under a drastic line closure?

For the third last question we refer to [6, Figure 9]. The second last question is addressed in Section 7.4. Finally, Section 7.5, which concludes this thesis, gives a real-world example in terms of the last question.

The computational results presented in Sections 7.3 and 7.4 have pure academic purposes and were made at Zuse Institute Berlin (ZIB) with an academic license of Cplex[1]. More precisely, the RSRP instances originate from past industrial scenarios of DBF, but the rolling stock rotations produced by RotOR for those instances have not been used by DBF for productive purposes. The results presented in Section 7.5 have been computed at DBF (where a another commercial license of Cplex is at hand) and have really been used within the production process of DBF.

7.1. Implementation Details

RotOR can be used as standalone console application under Linux systems and Microsoft Windows platforms by passing an input file (containing the timetable, rules, and parameters) to RotOR and receiving an output file, which contains the rolling stock rotations. In addition, RotOR has an API through that it is integrated in the IT system of DBF. This system is based on data bases, has a proper GUI, and is designed to be used by multiple rotation planners of DBF.

Sub-solvers. RotOR's implementation makes use of the commercial mixed-integer programming (MIP) solver Cplex [66]. We use the interior point algorithm (IPA) without crossover of Cplex version 12.6.3[2] in order to solve all linear programming (LP) models that arise during the algorithm and we also use the branch and cut (B&C) of Cplex to solve all MIP subproblems that come up. All pure instances of the standard assignment problem (AP) are solved with an $\mathcal{O}(|V|^3)$ implementation of the Hungarian method [70] (where $|V|$ denotes the number of nodes of the AP's bipartite graph).

RotOR is not limited to be used with Cplex only. In fact, RotOR can switch (at run time) to another sub-solver, e.g., Gurobi [57] or SCIP [13]. Although, Cplex is the current software that is involved when RotOR is used at DBF.

Parallelization. RotOR is parallelized as follows. First, the IPA and B&C (of Cplex or Gurobi) run on multiple threads if this is desired. Second, some of RotOR's procedures are parallelized by using OpenMP [84]:

- ▷ RotOR's hypergraph construction is parallelized. This does not really help to decrease its running time when computing industrial RSRP instances but significantly speeds up debugging.
- ▷ When an AP that appears during the coarse-to-fine (C2F) hyperarc generation algorithm (see Section 2.5) decomposes (which is usually the case), the individual

[1] Thanks to the IBM Academic Initiative.

[2] At the time of writing, Cplex version 12.6.3 is the latest release. However, RotOR 2.3 linked to Cplex version 12.5.1 is currently deployed to DBF and used within their production process.

APs are solved in parallel. In addition, the C2F pricing loop is executed in parallel. This is slightly more effective than the parallelization of the hypergraph construction but its speed-up is also limited by a factor of, say, two (independent of the number of threads used).

- ▷ Also the local search iterations of the cut date heuristic (CDH) introduced in Section 6.4 are performed in parallel. To this end, the individual cut date neighborhoods are stored and sorted in a global list. Then, they are iteratively traversed by different threads. When an improvement has been found it is only considered as new incumbent solution if all previous threads have finished their computations and have not found an improvement too. This rule makes the procedure deterministic and always behaving like running in single thread mode (when considering the same starting solution). It is more efficient than the previous parallelizations, especially when many iterations of the CDH do not lead to an improvement.

When does RotOR terminate? RotOR stops computing rolling stock rotations if

- ▷ RotOR is requested to terminate by an user interrupt,
- ▷ there is no rapid branching node left for backtracking,
- ▷ 100 rapid branching nodes[3] have been investigated, or

$$\text{cgap} := \frac{(\text{FMR})(H^\star) - \underline{z}}{c(H^\star)} \cdot 100 \;\leq\; 1.0\,\%. \qquad \text{(cheat gap)}$$

In equation (cheat gap) the lower bound obtained for the LP relaxation and the incumbent solution $H^\star \subseteq H$ are denoted by $\underline{z} \in \mathbb{Q}_+$ and $c(H^\star) \in \mathbb{Q}_+ \setminus \{0\}$, respectively. The value $(\text{FMR})(H^\star)$ has been introduced in Section 6.6, is called fractional maintenance bound, and always fulfills $\underline{z} \leq (\text{FMR})(H^\star) \leq c(H^\star)$. See Section 6.6 for its motivation.

When an instance of the non-maintenance relaxation of the RSRP is computed we have

$$\text{pgap} := \frac{c(H^\star) - \underline{z}}{c(H^\star)} \cdot 100 \quad = \text{cgap}. \qquad \text{(primal gap)}$$

Equation (primal gap) defines the traditional "primal gap" in percent that is usually considered in MIP approaches. It states the *maximum* percentage of the objective value $c(H^\star)$ for the incumbent solution $H^\star \subseteq H$ by that $c(H^\star)$ can possibly be decreased (by proof). We feel that this is a rather pessimistic point of view (w.r.t. the incumbent solution) because RotOR does (currently) not have the ability to increase the lower bound that is obtained from the LP relaxation.

The value cgap and its associated aborting criterion (cheat gap) are more optimistic for a proper instance of the RSRP that includes maintenance constraints. In fact, it is (heuristically) cleared up by the systematic error that the LP relaxation of RotOR's model makes for fractional service paths, again see Section 6.6.

[3] A rapid branching node is represented by one call of Algorithm 6.3.

Handouts. Whenever a new incumbent solution is found, RotOR computes a handout for all new rolling stock rotations and reports them to the user by a callback functionality. We do not report computation times for the handout optimization because they are insignificant w.r.t. the other algorithms.

Computers. All the computations presented in this chapter (except for Section 7.5) were performed on computers of the optimization department of ZIB. All computers had an Intel(R) Xeon(R) CPU X5672 with 3.20 GHz, 12 MB cache, and 48 GB of RAM and have been used in parallel mode by using four threads.

7.2. How to Compare?

In this thesis, computational studies have already been presented in Chapters 2 and 3. From a scientific point of view, these computational studies are meaningful. Our argument for that is that both studies allow the reader to compare the results accomplished with RotOR to something alternative, i.e., how does the C2F approach behave w.r.t. a standard LP solver and how do the regional search (RS) procedures perform on public available instances of the vehicle routing problem (VRP).

In this chapter we try to illustrate RotOR's capabilities on proper industrial RSRP instances of DBF. From a pure scientific point of view, this is much harder compared to the other computational studies. The reason is that it is not at all obvious to what we should compare to. Moreover, the industrial RSRP instances are confidential data, which is understandable but does not make it easier here. In addition, RotOR is not an open source software at the moment. The operations research community suggest two options in this situation.

Comparison with standard static MIP solvers. The first option for comparison is to run RotOR on some RSRP instances, compare the results to a standard static MIP solver on these instances, and claim a contribution (if there is one). We do not agree that such a comparison really helps *here*, because even the solution of the LP relaxation of RotOR's model is not always possible for large RSRP instances with standard software as we have seen in Chapter 2. Moreover, even if a standard MIP solver beats RotOR on small RSRP instances (those instances can be constructed), we do not have an idea what that contributes here. In reality, for that RotOR is made, it is possible to just start a standard MIP solver with RotOR's complete model on another thread in parallel. This is not implemented in RotOR, but a valid approach in order to compensate RotOR's overhead w.r.t. small RSRP instances.

Comparison to manually constructed rolling stock rotations. The second option for the comparison is to consider manually planned rolling stock rotations, run RotOR on the scenario behind, and claim a contribution (again, if there is one). Those comparisons have been made for many times during our cooperation with DBF in order to debug RotOR and its input data. In those situations, RotOR's re-optimization capability

has proved to be essential. To this end, manually constructed rolling stock rotations are considered as reference rotations for re-optimization. Then, RotOR is started in order to recreate the manually constructed rotations. By this procedure, one can very efficiently detect issues when the solution to the re-optimization scenario deviates from the manually constructed rotations or, even earlier, if RotOR finds inconsistencies in the preparation routines for re-optimization.

Thus, a comparison to manually constructed rotations is useful for debugging purposes. But we find it not helpful and, moreover, not appropriate in this chapter for three reasons:

- ▷ Consider the case when RotOR finds a "better" solution than the planning division of DBF for a past scenario. Then, we could claim a contribution (here). But do we really do things the right way when comparing in this way? We do not agree because the manually constructed rotations have already been proved to be implementable in operation. This is impossible to verify for RotOR's solution in this situation (because we can not change the past, unfortunately).
- ▷ Consider a present scenario that really has to be solved in the near future. Then, we basically provoke a "Man against Machine"-battle by comparing the solutions and publishing the results. In this situation, we find it better to cooperate, i.e., to load the manually constructed solution–if it already exists–into RotOR as a starting solution (this is currently not implemented but possible) or to provide RotOR's solution ("from scratch") to the railway planners, which is RotOR's original intention.
- ▷ The run time behavior of a human being is not comparable to a computing machine.

How to compare RotOR's computations instead? When RotOR is used by planners of DBF and RotOR comes up with rolling stock rotations, typically the experience of the railway planner, who has designed the instance of the RSRP, is sufficient to evaluate the rotations. We, as scientists, do not have this experience. But we have something different, i.e., the proven optimal solution to the LP relaxation of RotOR's model for the respective instance! In fact, this is what we compare to RotOR's incumbent rolling stock rotations in Section 7.3.3.

Note that some conclusions drawn by this comparison are speculative because we *independently* compare in terms of RotOR's 15 objective features, which are linearly combined to an overall objective function. Of course, such a comparison can lack because the objective features are dependent by definition. We know that! Nevertheless, those comparisons have helped a lot in the past when working together as scientist with planners of DBF on industrial RSRP instances. In fact, the LP relaxation perfectly supplements the experience of the railway planners of DBF.

7.3. Industrial Test Set

As already mentioned, we consider 116 industrial instances of the RSRP in this computational study. These instances were created by DBF during our cooperation in order to

evaluate RotOR. In fact, much more than "only" 116 have been investigated during the last years. The subset that we present here has been selected from all of these instances in order to draw a meaningful conclusion of RotOR's optimization capabilities by sustainably using the computer cluster of the optimization department of ZIB (its queuing system is socialistically shared among the scientists of the department).

Greenfield and re-optimization instances. The names of all instances appearing in Tables 7.1, 7.2, 7.3, and 7.4 of this section follow the regular expression

`RSRP_[0-9][0-9][0-9][GR]`

where the digits provide a continuous index and the last capital letter indicate if it is a greenfield (G) or re-optimization (R) scenario. Note that the 116 considered instances in this section are different from the RSRP's of Section 2.6. The reason is just that the study in Section 2.6 was made more than two years ago and industrial test sets typically change over the years.

Maintenance constraints. A particular aspect of the RSRP that has a significant effect on the computational results are the maintenance constraints. Therefore, all rows of all tables of this section (i.e., Tables 7.1, 7.2, 7.3, and 7.4) are colored by light gray, gray, dark gray, and very dark gray. An instance with light gray does not have any maintenance constraints, gray rows mark instances with one maintenance constraint, and dark gray indicates that there are two maintenance constraints. One instance, namely `RSRP_087G`, includes three maintenance constraints, i.e., regular inspection, major inspection and home base. The rows associated with `RSRP_087G` are very dark gray.

At the moment, instances of the RSRP with one or zero maintenance constraints do have a proper industrial purpose at DBF *when using* RotOR. In these instances regular inspections or the refueling constraint are contained, see Section 5.2. These constraints turn out to be currently enough for a sufficient amount of maintenance services along the rolling stock rotations w.r.t. succeeding planning stages of the production process of DBF. In fact, they take over the role of a somehow "prototype constraint" that leads to successful succeeding planning steps with higher probability.

The instances with two maintenance constraints, which also involve the major inspections (again see Section 5.2), are "slightly artificial". In fact, it turns out that major inspections have to be taken into account in the planning step that RotOR currently is dedicated to at DBF. Therefore, various instances also involving this constraint in addition to regular inspections or refueling have been created. Although, it turns out that it is much more applicable (and sufficient as well) to just predefine (w.r.t. time, duration, and location) some regular inspections and pass them to RotOR in order to cover them just as timetabled trips. Properly constraining the cumulative operational time between two succeeding major inspections turns out to be too restrictive for the scenarios that we considered. Note that this seems to be not an issue of RotOR's solution. It appears just too expensive (primary in terms of ICE vehicles) to exactly constrain (in terms of a resource constraint) the rolling stock rotations by a proper maintenance constraint for

the major inspections. Nevertheless, we also provide results for RSRP instances dealing with those constraints because they show how the computational results are affected when a rather hard maintenance constraint is considered.

7.3.1. Problem-oriented View

Table 7.1, which spreads over the pages 217 to 220, provides a problem-oriented view on the 116 industrial RSRP instances, which we consider here[4]. Note that the industrial RSRP instances that we consider here are very different in terms of size and hardness. This, of course, is a desired property of an industrial test set. But in industry it is often even not of interest what can be computed for past scenarios; what can be computed tomorrow is the crucial point. Naturally, this can not be evaluated, but diversification in terms of size and hardness helps in order prepare for tomorrow. The columns of Table 7.1 present the following details of RotOR's input data.

Timetable. Columns two to five describe the input timetable. To this end, the number of timetabled trips, the number of trains (i.e., the number of different train identifiers, see Section 4.10), the number of timetabled and enforced trip sequences (see Section 4.8) as well as the number of intrusive trunk constraints. A trunk constraint is considered to be intrusive if it imposes a proper constraint, i.e., if $\mathfrak{v}\left(\{t_i\}_{i=1}^{k}\right) > 0$ holds for the respective timetabled trip sequence $\{t_i\}_{i=1}^{k} \in \mathbb{T}$, see Section 5.4.

Note that the largest instance (in terms of the timetable) is `RSRP_013G` with 5600 timetabled trips. This instance does not contain a maintenance constraint. Instances of such a large size that also include maintenance constraints are rather hard to be computed by RotOR in a reasonable amount of computation time, i.e., within one weekend.

Vehicle configuration. The next three columns report how many fleets and vehicle configurations (not to be confused with vehicle compositions) are involved, see Section 4.1. Note that typically not all vehicle configurations are allowed for all timetabled trips. The concrete distribution of vehicle configurations is problem specific and confidential. At most three individual ICE vehicles of DBF can be coupled together (not all fleets are allowed to be coupled together for technical reasons) in order to form a vehicle configuration. But three is a rather exceptional case where are no passengers on board (usually). Thus, the largest vehicle configurations that we consider here is of size two.

The experienced reader might wonder why there are up to 16 ICE fleets considered, e.g., in `RSRP_012G`. The main reason that so many fleets are required to be distinguished is that ICE vehicles running through Germany's neighboring countries (e.g., France, Denmark, or Austria) need to have special equipment, e.g., w.r.t. their security system.

[4] In fact, 119 RSRPs have been considered while preparing the computational study. For three of them, RotOR did not produce a solution after computing two days. One of those instances has four maintenance constraints, i.e., regular as well as major inspection, home base, and another sophisticated inspection. For another instance RotOR seems to suffer from a huge amount of infeasible couplings that lead to many iterations of the CCH. The issue of the third instance is unclear to us at the moment (a sub-MIP solve did not return). These three RSRPs have been ruled out.

Rotation templates. Also the number of refined vehicle configurations is denoted in Table 7.1 if it is different from the original number of vehicle configurations. Recall that an original vehicle configuration becomes refined (by one of RotOR's preparation routines) when a fleet becomes refined in order to create a rotation template for re-optimization[5], see Section 4.11. Note that `RSRP_109R` originally has ten vehicle configurations that become refined to not less than 76 vehicle configurations for this re-optimization scenario.

Maintenance. Columns ten and eleven of Table 7.1 report how many maintenance constraints are involved in the respective instance and how many different maintenance services can be performed in order to maintain railway vehicles. Two maintenance services are considered to be different if their locations, fleets, or addressed set of maintenance constraints differ. Thus, for instances with exactly one maintenance constraint and exactly one fleet the number of maintenance services is equal to the number of different locations where railway vehicles can become maintained.

Capacity constraints. The next three columns of Table 7.1 report how many capacity constraints for fleets, maintenance services, and parking activities are contained in the scenarios. From those numbers, we could suspect[6] that capacity constraints do not really play a very important role for RotOR's current purpose at DBF. But this could also be only a specific characteristic of our test set and might change in the future.

Hypergraph. Finally, the last four columns of Table 7.1 state the number of nodes divided by two[7] and hyperarcs of RotOR's composition (configuration) layer, i.e., $|V|/2$ ($|[V]|/2$) and $|H|$ ($|\,[H]\,|$), respectively. Note that there is no obvious correlation between the number of timetabled trips and the number of hyperarcs. The number of hyperarcs is primary determined by the size, refinement (for re-optimization), and distribution of the vehicle configurations distributed among the timetabled trips.

As already mentioned, the maximal size of the vehicle configurations considered here is two. But already for two railway vehicles there are hyperarcs in RotOR's configuration layer that can be realized by up to 16 fine hyperarcs of the composition layer (i.e., up to eight possibilities to operate the tailing timetabled trip of a hyperarc times two options for turning around before operating the heading timetabled trip). Therefore, the factor between the number of hyperarcs of the configuration layer and the number of hyperarcs of the composition layer is limited (but sometimes close to) a factor of 16.

In terms of hyperarcs, `RSRP_109R` is currently the largest RSRP instance of our test set. It is a re-optimization scenario that includes a maintenance constraint and has almost 80 million hyperarcs in its composition layer.

[5] `RSRP_023G` also contains rotation templates. Although, it is not a re-optimization scenario. This is a bug of the input data, which does not prevent RotOR from computing.
[6] Naturally, we do not know all scenarios for that DBF is using RotOR.
[7] This is the number of departure-arrival pairs (of nodes) in RotOR's hypergraph.

Table 7.1.: Problem-oriented view on industrial RSRP instances. Some instances appear to be equal by only considering the characteristics denoted in the table. In fact, all of them are different but it is not possible to report all industrial details here. For instance, `RSRP_016G`, `RSRP_017G`, `RSRP_018G`, and `RSRP_020G` distinguish by different objective functions and different coefficients for the capacity constraints.

instance	timetabled trips ($\lvert T \rvert$)	trains ($\lvert \Sigma \rvert$)	timetabled trip sequences	enforced trip sequences	trunk constraints	fleets ($\lvert F \rvert$)	configurations	refined configurations	maintenance constraints ($\lvert L \rvert$)	maintenance services	fleet cap. constraints	maintenance cap. constraints	parking cap. constraints	$\frac{\lvert V \rvert}{2}$	$\frac{\lvert [V] \rvert}{2}$	$\lvert H \rvert$	$\lvert [H] \rvert$
`RSRP_001G`	174	10	70		8	2	4							898	303	155078	15420
`RSRP_002G`	277	18	124		8	2	4							1464	490	434312	44317
`RSRP_003G`	278	18	124		8	2	4							1472	492	441810	44793
`RSRP_004G`	185	37	185	4		1	1							370	185	151392	38466
`RSRP_005G`	484	49	264	23	145	3	7				2			6668	2766	16342128	1242877
`RSRP_006G`	590	94	447	156	109	1	2				1			1420	710	1523540	236889
`RSRP_007G`	590	94	447	162	109	1	2							1420	710	1530032	236554
`RSRP_008G`	1111	155	830	60	190	4	8							11298	4849	47207978	4522283
`RSRP_009G`	1111	155	830	60	190	4	8				3			11298	4849	47207978	4522283
`RSRP_010G`	1118	156	830	57	197	4	8				3			11374	4873	47890472	4572513
`RSRP_011G`	1118	156	830	5	197	4	9				3			14494	5653	78103582	6867441
`RSRP_012G`	5593	700	4900	476	469	16	25							14250	6701	21711590	3453359
`RSRP_013G`	5600	700	4900	476	476	16	25							14278	6715	21698166	3449823
`RSRP_014G`	147	16	91	20	41	1	2		1	6				410	205	247386	29123
`RSRP_015G`	160	18	97		43	1	2		1	3				470	235	343542	35164
`RSRP_016G`	278	18	124	7	8	2	4		1	16		154		1472	492	812878	81805
`RSRP_017G`	278	18	124	7	8	2	4		1	16		154		1472	492	812878	81805
`RSRP_018G`	278	18	124	7	8	2	4		1	16		154		1472	492	812858	81800
`RSRP_019G`	278	18	124		8	2	4		1	16				1472	492	812942	81805
`RSRP_020G`	278	18	124	7	8	2	4		1	16		154		1472	492	812858	81800
`RSRP_021G`	183	32	173	7	10	2	5		1	12				586	279	320074	68412
`RSRP_022G`	953	130	622	94	58	4	12		1	32				3086	1219	4758994	572353
`RSRP_023G`	960	128	628	96	69	4	12	18	1	32				8058	3126	24265384	2801667
`RSRP_024G`	964	132	632	100	69	4	12		1	32				3769	2252	7212777	1974272
`RSRP_025G`	964	132	632	100	69	4	12		1	32				4612	2252	10549312	1966738
`RSRP_026G`	964	132	632	100	69	4	12		1	32				5334	2252	14030835	1966738
`RSRP_027G`	964	132	632	100	69	4	12		1	32				5856	2252	17243902	1966738

Continued on next page

Table 7.1 – continued from previous page

instance	timetabled trips ($\lvert T\rvert$)	trains ($\lvert\mathfrak{T}\rvert$)	timetabled trip sequences	enforced trip sequences	trunk constraints	fleets ($\lvert F\rvert$)	configurations	refined configurations	maintenance constraints ($\lvert L\rvert$)	maintenance services	fleet cap. constraints	maintenance cap. constraints	parking cap. constraints	$\frac{\lvert V\rvert}{2}$	$\frac{\lvert[V]\rvert}{2}$	$\lvert H\rvert$	$\lvert[H]\rvert$
RSRP_028G	981	126	663	277	286	1	2		1	4				2550	1275	9134530	1175536
RSRP_029G	1014	154	734	92	61	4	12		1	32				5243	2411	14542286	2560671
RSRP_030G	1069	154	734	92	71	4	12		1	32				5471	2591	15545060	2776925
RSRP_031G	1069	154	734	92	71	4	12		1	32				5471	2591	15601487	2787306
RSRP_032G	788	147	788			2	2		1	8	1			2732	1366	7998508	2000617
RSRP_033G	788	147	788	23		2	2		1	8	1			2732	1366	7743590	1936809
RSRP_034G	987	148	789	40	134	4	10		1	26	1			2926	1427	9067926	1099377
RSRP_035G	1116	174	801	298	289	1	2		1	1				2850	1425	11907150	1635209
RSRP_036G	1116	174	801	298	289	1	2		1	1				2850	1425	11907150	1635209
RSRP_037G	1156	174	801	297	319	1	2		1	1				3488	1744	18306113	2228437
RSRP_038G	1156	174	801	297	319	1	2		1	1				3488	1744	18585259	2263360
RSRP_039G	1156	174	801	297	319	1	2		1	1				3488	1744	18316310	2229537
RSRP_040G	805	150	805			2	2		1	8				2928	1464	9081282	2271432
RSRP_041G	884	162	884			2	2		1	8				1830	915	4085180	1022177
RSRP_042G	884	162	884			2	2		1	62				1830	915	4097382	1025180
RSRP_043G	146	7	49		1	2	3		2	40		154		828	268	339820	29265
RSRP_044G	174	10	70	7	8	2	4		2	40		154		898	303	413844	40590
RSRP_045G	77	15	77			1	1		2	6				154	77	69594	18161
RSRP_046G	159	17	96		43	1	2		2	9				470	235	448146	43828
RSRP_047G	277	18	124		71	1	2		2	20				980	490	1660408	179979
RSRP_048G	277	18	124	7	71	1	2		2	20		112		980	490	1659894	179911
RSRP_049G	277	18	124	6	71	1	2		2	20				980	490	1634202	175692
RSRP_050G	277	18	124	7	8	2	4		2	40		154		1443	490	1115598	116512
RSRP_051G	277	18	124		8	2	4		2	40				1464	490	1154548	116547
RSRP_052G	277	18	124	7	8	2	4		2	40		168	28	1464	490	1390954	132924
RSRP_053G	277	18	124	7	8	2	4		2	40		168	28	1464	490	1391618	133007
RSRP_054G	277	18	124	7	8	2	4		2	40		168	28	1464	490	1391618	133007
RSRP_055G	277	18	124	7	8	2	4		2	40		154		1464	490	1154426	116512
RSRP_056G	277	18	124	7	8	2	4		2	40		168	28	1464	490	1391618	133007
RSRP_057G	277	18	124	7	8	2	4		2	40		168	28	1464	490	1391618	133007
RSRP_058G	278	18	124		8	2	4		2	40				1472	492	1169212	117465
RSRP_059G	310	61	310			1	1		2	10		56		620	310	1163370	290969
RSRP_060G	310	61	310			1	1		2	10				620	310	1163370	290969

Continued on next page

Table 7.1 – continued from previous page

instance	timetabled trips ($\|T\|$)	trains ($\|\mathfrak{T}\|$)	timetabled trip sequences	enforced trip sequences	trunk constraints	fleets ($\|F\|$)	configurations	refined configurations	maintenance constraints ($\|L\|$)	maintenance services	fleet cap. constraints	maintenance cap. constraints	parking cap. constraints	$\frac{\|V\|}{2}$	$\frac{\|[V]\|}{2}$	$\|H\|$	$\|[H]\|$
RSRP_061G	336	48	336			1	1		2	31				672	336	1363482	342709
RSRP_062G	927	124	633	300	289	1	2		2	8				2452	1226	18484103	2415071
RSRP_063G	927	124	633	293	289	1	2		2	8				2452	1226	18488637	2415827
RSRP_064G	958	123	663	296	284	1	2		2	8				2532	1268	17218035	2161711
RSRP_065G	1126	136	753	26	112	3	7		2	60		147		4593	1864	20726853	2697088
RSRP_066G	1126	136	753		112	3	7		2	60		147		4696	1864	22064352	2713236
RSRP_067G	1126	136	753		112	3	7		2	60		147		4696	1864	22064352	2713236
RSRP_068G	1126	136	753	26	112	3	7		2	60		147		4696	1864	21886554	2697088
RSRP_069G	1126	136	753		112	3	7		2	60				4696	1864	22064352	2713236
RSRP_070G	1126	136	753		112	3	7		2	60		147		4696	1864	22064352	2713236
RSRP_071G	1126	136	753	26	112	3	7		2	60		147	147	4696	1864	26743422	3263972
RSRP_072G	1126	136	753	26	112	3	7		2	60		147	147	4696	1864	26743384	3263961
RSRP_073G	1126	136	753		112	3	7		2	60		147		4696	1864	22064352	2713236
RSRP_074G	1060	132	783	20	185	3	6		2	45		49		3934	1879	28001538	2802603
RSRP_075G	1033	153	830	27	117	3	5		2	44				3106	1379	12746204	1407269
RSRP_076G	854	166	854			2	2		2	16				1708	854	5155568	1289546
RSRP_077G	884	162	884			1	1		2	8				1768	884	9389816	2348474
RSRP_078G	884	162	884			2	2		2	16				1830	915	5868584	1468088
RSRP_079G	1806	298	1443		55	3	5		2	60				4610	2015	20719848	4066914
RSRP_080G	1488	298	1443			5	5		2	150				2976	1488	11670192	2921880
RSRP_081G	1488	298	1443			5	5		2	150				2976	1488	11670716	2922056
RSRP_082G	2186	268	1536	46	297	6	13		2	105		161		8630	3743	49888092	5499691
RSRP_083G	2030	284	1640		352	4	6		2	38		70		4910	2455	24196022	3795539
RSRP_084G	2030	284	1640		352	4	6		2	38		70		4910	2455	24196022	3795539
RSRP_085G	4216	552	3176	340	649	10	19		2	143		322		13354	6198	70703476	8916032
RSRP_086G	4216	552	3176	340	649	10	19		2	143		322	21	13354	6198	71575584	9005701
RSRP_087G	958	123	663	296	284	1	2		3	19				2532	1268	32662201	4093622
RSRP_088R	156	18	100	14	41	1	2							444	222	166566	18814
RSRP_089R	933	120	631	271	289	1	2	9						7410	3705	15849006	1947093
RSRP_090R	977	125	670	296	289	1	2	9						7674	3837	16719396	2101521
RSRP_091R	127	18	106			2	3		1	13				368	146	120650	23112
RSRP_092R	259	46	259	1		1	1	2	1	4				1036	518	1116128	279534
RSRP_093R	259	46	259	1		1	1	2	1	4				1036	518	1112628	278650

Continued on next page

Table 7.1 – continued from previous page

instance	timetabled trips ($\|T\|$)	trains ($\|\mathfrak{T}\|$)	timetabled trip sequences	enforced trip sequences	trunk constraints	fleets ($\|F\|$)	configurations	refined configurations	maintenance constraints ($\|L\|$)	maintenance services	fleet cap. constraints	maintenance cap. constraints	parking cap. constraints	$\frac{\|V\|}{2}$	$\frac{\|\|V\|\|}{2}$	$\|H\|$	$\|\|H\|\|$
RSRP_094R	953	130	622	94	58	4	12	18	1	32				4770	1911	8818666	1051403
RSRP_095R	964	132	632	100	69	4	12	18	1	32				5130	3134	10174888	2822780
RSRP_096R	964	132	632	100	69	4	12	18	1	32				6302	3134	15057299	2812146
RSRP_097R	927	124	633	300	289	1	2	9	1	5				7356	3678	25046589	3115038
RSRP_098R	927	124	633	300	289	1	2	9	1	5				7356	3678	26527317	3239712
RSRP_099R	1014	154	734	92	61	4	12	18	1	32				7193	3379	20713480	3690310
RSRP_100R	1014	154	734	92	61	4	12	18	1	32				7193	3379	20719500	3691604
RSRP_101R	1014	154	734	92	61	4	12	18	1	32				7193	3379	20712824	3690143
RSRP_102R	1014	154	734	92	61	4	12	18	1	32				7193	3379	20415748	3630979
RSRP_103R	1014	154	734	92	61	4	12	18	1	32				7193	3379	20712292	3689921
RSRP_104R	1069	154	734	92	71	4	12	18	1	32				7526	3644	22176079	4006463
RSRP_105R	1069	154	734	92	71	4	12	18	1	32				7526	3644	22255985	4021315
RSRP_106R	788	147	788	8		2	2	5	1	8				4182	2091	7821712	1956870
RSRP_107R	788	147	788	17		2	2	5	1	8				4182	2091	7725152	1932694
RSRP_108R	788	147	788	23		2	2	6	1	8	1			7776	3888	20885176	5223738
RSRP_109R	987	148	789	41	134	4	10	76	1	26	1			16790	8215	78071394	8337915
RSRP_110R	805	151	805			2	2	7	1	8				9810	4905	29049302	7265959
RSRP_111R	1025	169	838	91	115	7	15	30	1	45	5			11688	4656	69751046	5885729
RSRP_112R	156	18	100	14	41	1	2		2	14		3		444	222	555306	59940
RSRP_113R	167	19	104	14	43	1	2		2	9				486	243	467218	46854
RSRP_114R	927	124	633	300	289	1	2	9	2	8				7356	3678	56398227	7324146
RSRP_115R	927	124	633	293	289	1	2	9	2	8				7356	3678	56439297	7328703
RSRP_116R	927	124	633	300	289	1	2	9	2	8				7356	3678	56398227	7324146

7.3.2. Algorithmic View

We computed solutions, namely rolling stock rotations, for all 116 industrial instances of the RSRP that we introduced in the previous section by using RotOR version 2.4. RotOR has stopped computing (by "himself") according to the explanation in Section 7.1, except for `RSRP_079G`, `RSRP_082G`, `RSRP_085G`, and `RSRP_086G`. For those instances a manual user interrupt has been made because these computations have expired our (self-chosen) "industrial computation time limit" of one weekend[8].

Many of RotOR's algorithmic characteristics w.r.t. the 116 industrial RSRP instances, which we investigate in the following, are denoted in Table 7.2.

C2F column generation algorithm (CGA). The percentage of the overall computation time (as given by the last column of Table 7.2) spend for computing the LP relaxation of RotOR's model by its C2F CGA (or C2F constraint matrix generation algorithm) is denoted in column two of Table 7.2. As we can see, this percentage varies strongly. This depends on how easy it is for RotOR to construct high-quality integer programming (IP) solutions by its rapid branching (RB) implementation and heuristics. Also the number of iterations (i.e., calls of Algorithm 6.2), which is denoted in column three, varies a lot.

Rapid branching (RB). The number of nodes of RotOR's RB search tree is given by column four of Table 7.2. These nodes are counted without considering nodes that have been created by the standard B&C of Cplex. In addition, the number of perturbated LP models that have appeared during RB are denoted in column five of Table 7.2.

In Chapter 2 we concluded that RotOR's C2F approach reduces the computation for re-optimizing LP models by a factor of 20 (recall that all LP models are computed by the IPA of Cplex). As we can see in Table 7.2 often many perturbation rounds are performed. In fact, the proper contribution of the C2F approach w.r.t. computation time is accomplished in these rounds.

Coupling component heuristic (CCH). The number of local search iterations of the coupling component heuristic (CCH) is denoted in column six of Table 7.2. As explained in Section 6.5, the CCH is only activated when infeasible couplings can be expected to appear in the solution to the LP relaxation. As we can see, only a few RSRP instances of our test set are possibly affected by the issues in terms of infeasible couplings (fortunately). Except for `RSRP_005G`, we observe a highly positive contribution of the CCH.

For `RSRP_005G` the CCH has failed in the sense that vehicle configurations have been assigned to some timetabled trips that conflict with some capacity constraints for fleets that are imposed in this scenario. Then, the solution to `RSRP_005G` contains many active slack variables (for constraints (covering) of RotOR's model (RMIP) denoted on page 169) because some timetabled trips have not been covered. The slack variables have very large objective coefficients (i.e., in the range of the cost for 10 railway vehicles of

[8] We observed that an industrial RSRP instance is likely to change after two days during our cooperation with DBF.

the most expensive fleet appearing in the instance). Therefore, the primal gap between lower and upper bound for this instance becomes "astronomical".

Cut date heuristic (CDH). The number of calls to the CDH is described by column seven of Table 7.2. As explained in Sections 6.3 and 6.4, the CDH is called within the RB search tree but only if no feasible solution has been found yet. Thus, the CDH identifies here as the primal heuristic of RotOR that often finds a feasible solution, which is usually also of high-quality.

As highlighted in blue in this column, sometimes the first call of the CDH does not produce a feasible solution. In this situation often the number of railway vehicles of the starting solution is not sufficient and can, unfortunately, also not be easily varied by the CDH. Then, the CDH is called again but deeper in the RB search tree. In those calls its starting solution is likely to use more railway vehicles than in the previous runs because the starting solution is derived from the LP solution of the current RB node and those solutions become more expensive when branching deeper.

Thus, this is an interplay between RB and the CDH, which we find "very heuristic" (today). This may be improved in the future by using the approach presented in Chapter 3 in order to derive "regions" that allow to automatically vary the number of railway vehicles within the CDH.

Gap between lower and upper bound. The "cheat gap" and primal gap are computed according to equations (cheat gap) as well as (primal gap) on page 211 and are denoted in Table 7.2 as well.

An entry in column cheat gap is highlighted in blue if the difference to the primal gap is smaller or equal to one percent. Thus, the fractional maintenance relaxation, which RotOR uses as an aborting criterion, contributes especially for instances involving the hard maintenance constraint. This effect may appear as slight, but it is not. E.g., for `RSRP_076G` we have a strong indication that half of the primal gap originates from the systematic error of the LP relaxation w.r.t. fractional service paths and does not originate from other errors (with high probability), see Section 6.6.

The results denoted in column primal gap of Table 7.2 indicate that RotOR produces rolling stock rotations of high-quality and, moreover, that the quality can also be proved in many cases by the lower bound computed by the C2F CGA. An entry in this column is highlighted in green if RotOR could prove a worst gap between the lower and upper bound below one percent, see column primal gap of Table 7.2. This is often the case and has, especially for almost all re-optimization scenarios.

Also some results are reported by Table 7.2 that are supposed to be negative. From our experience, we find this natural for such a large test set involving really complicated scenarios. Nevertheless, we like to briefly defend also those results in the following.

Sometimes RotOR can "only" (better to say "already") prove a rather high primal gap. E.g., for `RSRP_046G` it is 12.8 %. The root of the issue is that 8 railway vehicles are enough in the LP solution, while RotOR's IP solution requires 9 (we will see these numbers later in Table 7.3). In view of the relatively small set of timetabled trips to be

covered in RSRP_044G and the appearance of the hard maintenance constraint we expect that RotOR is more right there as the LP solution is.

The algorithmic issues that appear for RSRP_005G have already been mentioned above. We suspect that they originate from infeasible couplings, see Section 6.5.2. This will definitely be a subject of further research. The result obtained for RSRP_005G appears as completely useless by only considering Table 7.2 at the moment. *But*: From Table 7.3 we get a completely different impression of the solution for RSRP_005G. In fact, the LP solution covers 768 nodes, while the IP solution covers *already* 761 nodes. While analyzing the detailed situation, we observe that RotOR fails to cover "only" four out of 484 timetabled trips for this instance. Note that the remaining objective features are similar when comparing the LP to the IP solution as we do in the next section by Table 7.3. Therefore, we claim that this solution is not at all useless! In fact, we strongly expect that an experienced railway planner can immediately complete the rolling stock rotations by the six uncovered timetabled trips and will obtain a solution of high-quality too.

Table 7.2.: Algorithmic view on RotOR's algorithms.

instance	C2F CGA [%]	C2F CGA iter.	RB nodes	RB perturbations	CCH iterations	CDH runs	cheat gap	primal gap	time (dd:hh:mm:ss)
RSRP_001G	28.1	8	2	6		1	0.02	0.02	00:00:01:18
RSRP_002G	36.5	10	2	6		1	0.00	0.00	00:00:01:49
RSRP_003G	29.3	10	2	6		1	0.05	0.05	00:00:02:30
RSRP_004G	30.5	7	3	12		1	0.03	0.03	00:00:02:20
RSRP_005G	9.2	11	6	36	223	1	90.69	90.69	00:03:53:13
RSRP_006G	35.4	11	3	12		1	0.05	0.05	00:00:05:28
RSRP_007G	18.8	9	4	18		1	1.22	1.22	00:00:08:08
RSRP_008G	16.4	11	6	35	63	1	0.33	0.33	00:05:19:53
RSRP_009G	16.7	11	5	27	78	1	0.47	0.47	00:07:29:42
RSRP_010G	11.4	9	4	21	92	1	0.36	0.36	00:12:03:23
RSRP_011G	5.8	9	7	45		1	0.42	0.42	01:22:13:07
RSRP_012G	1.4	9	3	12		1	0.08	0.08	00:17:24:39
RSRP_013G	8.2	12	4	18		1	0.30	0.30	00:03:36:31
RSRP_014G	36.6	17	2	6		1	0.79	0.86	00:00:03:06
RSRP_015G	22.9	16	3	22		1	2.12	2.58	00:00:04:29
RSRP_016G	50.6	16	4	22		1	0.17	0.17	00:00:05:21
RSRP_017G	59.7	22	2	2		1	0.64	0.73	00:00:06:06
RSRP_018G	43.3	18	3	14		1	0.48	0.56	00:00:08:00
RSRP_019G	55.8	23	2	6		1	0.53	0.67	00:00:06:03
RSRP_020G	45.4	19	3	14		1	0.11	0.11	00:00:08:03
RSRP_021G	34.5	19	2	6		1	1.19	1.53	00:00:03:23

Continued on next page

Table 7.2 – continued from previous page

instance	C2F CGA [%]	C2F CGA iter.	RB nodes	RB perturbations	CCH iterations	CDH runs	cheat gap	primal gap	time (dd:hh:mm:ss)
RSRP_022G	21.2	28	6	39		1	0.21	0.50	00:01:15:00
RSRP_023G	58.8	22	13	84		1	1.45	1.45	00:06:19:38
RSRP_024G	53.9	32	11	69		1	0.06	0.29	00:01:50:17
RSRP_025G	64.6	32	11	66		1	0.14	0.27	00:02:47:06
RSRP_026G	64.3	29	8	45		1	0.12	0.30	00:02:39:16
RSRP_027G	70.0	35	6	33		1	0.17	0.32	00:04:21:13
RSRP_028G	43.7	35	6	40		1	0.21	0.32	00:01:37:07
RSRP_029G	27.2	30	100	614		1	1.73	1.87	00:05:29:03
RSRP_030G	52.7	35	10	62		1	0.23	0.36	00:04:49:39
RSRP_031G	38.1	32	83	503		1	0.26	0.48	00:05:11:05
RSRP_032G	16.5	24	5	34		1	0.14	0.61	00:01:46:04
RSRP_033G	28.7	28	2	6		1	1.13	1.32	00:01:04:06
RSRP_034G	6.0	23	100	610	9	1	1.87	1.97	00:05:00:04
RSRP_035G	19.5	40	100	607		1	6.92	6.92	00:05:30:40
RSRP_036G	15.4	38	100	626		1	4.61	4.61	00:06:30:24
RSRP_037G	21.7	40	100	613		1	5.11	5.11	00:06:51:08
RSRP_038G	22.6	33	100	612		1	3.84	3.84	00:05:35:01
RSRP_039G	22.8	34	100	621		1	3.69	3.70	00:05:27:53
RSRP_040G	22.3	25	4	36		1	1.52	1.77	00:01:20:32
RSRP_041G	9.5	19	6	44		1	0.65	1.02	00:00:55:12
RSRP_042G	14.7	24	2	6		1	1.48	1.57	00:00:42:31
RSRP_043G	33.8	15	7	44		1	1.10	3.21	00:00:05:38
RSRP_044G	15.5	17	65	387		4	9.61	12.82	00:00:14:26
RSRP_045G	27.3	16	2	6		1	0.05	0.16	00:00:02:33
RSRP_046G	25.5	16	7	42		1	4.81	8.57	00:00:08:23
RSRP_047G	29.9	27	25	150		1	0.75	0.75	00:00:22:01
RSRP_048G	53.6	25	3	14		1	2.29	3.02	00:00:13:19
RSRP_049G	54.7	26	2	6		1	0.80	0.80	00:00:11:32
RSRP_050G	42.1	18	5	28		1	0.31	1.45	00:00:12:50
RSRP_051G	9.8	21	100	613		1	5.85	5.90	00:00:45:45
RSRP_052G	19.7	19	100	606		1	6.35	8.03	00:00:31:36
RSRP_053G	32.1	29	60	360		1	0.28	4.58	00:00:28:02
RSRP_054G	45.3	20	8	48		1	0.29	3.03	00:00:13:45
RSRP_055G	50.4	21	9	52		1	0.25	3.08	00:00:10:19
RSRP_056G	40.9	20	10	60		1	0.30	1.47	00:00:15:21
RSRP_057G	46.6	18	3	10		1	0.37	1.57	00:00:10:57
RSRP_058G	45.8	22	2	6		1	0.70	0.86	00:00:11:35
RSRP_059G	31.5	21	4	31		1	1.26	4.31	00:00:07:47
RSRP_060G	34.8	22	2	6		1	1.01	3.04	00:00:06:34

Continued on next page

Table 7.2 – continued from previous page

instance	C2F CGA [%]	C2F CGA iter.	RB nodes	RB perturbations	CCH iterations	CDH runs	cheat gap	primal gap	time (dd:hh:mm:ss)
RSRP_061G	19.2	14	3	26		1	1.78	3.20	00:00:10:11
RSRP_062G	6.7	30	100	623		1	2.04	2.09	00:13:38:35
RSRP_063G	3.2	30	100	620		1	2.05	2.05	01:09:40:33
RSRP_064G	9.8	36	100	631		1	1.94	1.96	00:09:11:58
RSRP_065G	20.8	26	100	672		1	5.65	7.16	00:12:06:20
RSRP_066G	15.5	31	100	596		1	2.87	2.97	00:12:50:18
RSRP_067G	16.9	33	100	598		1	1.95	2.00	00:13:36:49
RSRP_068G	14.1	25	6	41		1	5.17	6.26	00:14:10:03
RSRP_069G	9.4	28	96	578		1	4.73	6.46	00:16:50:26
RSRP_070G	20.9	27	100	622		1	5.66	7.21	00:13:01:41
RSRP_071G	14.6	24	20	131		1	5.10	6.65	00:16:57:34
RSRP_072G	14.9	29	10	62		1	5.85	7.50	00:17:54:03
RSRP_073G	9.6	29	100	608		1	3.91	5.73	00:17:55:45
RSRP_074G	4.6	24	100	614		1	2.87	4.10	01:03:00:00
RSRP_075G	1.5	22	100	606		1	2.44	3.62	01:15:52:12
RSRP_076G	6.8	23	7	58		1	3.09	6.19	00:02:23:13
RSRP_077G	2.9	25	100	609		1	4.87	7.43	00:07:46:03
RSRP_078G	6.4	21	15	101		1	3.63	6.58	00:02:16:07
RSRP_079G	1.9	22	100	606		1	4.66	6.59	02:01:01:12
RSRP_080G	3.1	21	4	35		1	3.59	5.49	00:09:16:21
RSRP_081G	2.5	20	4	36		1	3.78	5.64	00:13:03:30
RSRP_082G	10.3	32	19	120		1	3.66	5.11	02:09:44:36
RSRP_083G	10.5	33	3	20		1	4.79	7.15	00:10:41:28
RSRP_084G	7.6	34	6	38		1	4.14	6.44	00:16:34:41
RSRP_085G	19.1	35	5	27		1	3.71	5.52	02:10:04:19
RSRP_086G	25.6	38	4	21		1	4.02	5.89	02:09:26:29
RSRP_087G	8.6	36	100	632		1	3.01	3.01	00:20:13:01
RSRP_088R	30.3	7	1	0		1	0.00	0.00	00:00:02:05
RSRP_089R	72.4	11	3	14		1	0.02	0.02	00:00:29:00
RSRP_090R	67.8	12	11	65		1	1.81	1.82	00:00:34:48
RSRP_091R	43.8	12	3	14			0.02	0.23	00:00:01:30
RSRP_092R	46.0	13	1	0			0.04	0.71	00:00:04:19
RSRP_093R	46.7	12	1	0			0.03	0.63	00:00:04:31
RSRP_094R	88.4	22	2	2			0.01	0.05	00:00:42:36
RSRP_095R	77.3	33	8	47		1	0.03	0.07	00:02:33:13
RSRP_096R	79.0	34	2	2		1	0.15	0.15	00:02:56:40
RSRP_097R	92.5	26	2	2			0.00	0.02	00:02:41:35
RSRP_098R	74.5	29	8	53		1	2.11	2.12	00:03:57:49
RSRP_099R	82.0	30	4	20			0.23	0.34	00:04:01:49

Continued on next page

Table 7.2 – continued from previous page

instance	C2F CGA [%]	C2F CGA iter.	RB nodes	RB perturbations	CCH iterations	CDH runs	cheat gap	primal gap	time (dd:hh:mm:ss)
RSRP_100R	68.9	33	26	158		1	0.14	0.23	00:05:08:04
RSRP_101R	68.7	23	5	32		1	0.10	0.20	00:05:24:02
RSRP_102R	76.9	38	11	68		1	0.07	0.17	00:07:20:30
RSRP_103R	82.7	32	3	14			0.04	0.12	00:05:37:19
RSRP_104R	70.7	32	4	30		1	1.74	1.79	00:06:01:29
RSRP_105R	66.1	30	14	86		1	0.05	0.16	00:05:17:24
RSRP_106R	87.3	12	1	0			0.01	0.27	00:00:14:08
RSRP_107R	77.0	13	2	2			0.05	0.53	00:00:17:48
RSRP_108R	98.1	20	1	0			0.04	0.11	00:02:17:29
RSRP_109R	50.6	19	3	14	8		0.67	0.67	01:14:46:10
RSRP_110R	98.9	16	1	0			0.01	0.01	00:02:10:32
RSRP_111R	70.1	46	4	21	2	1	0.76	0.84	01:04:44:28
RSRP_112R	70.1	18	1	0			0.00	0.00	00:00:02:08
RSRP_113R	44.8	20	4	25		1	0.81	0.84	00:00:04:30
RSRP_114R	67.5	23	2	2		1	0.07	0.07	00:05:25:44
RSRP_115R	82.2	22	4	21		3	0.14	0.16	00:04:13:02
RSRP_116R	82.8	23	4	24		3	0.05	0.05	00:04:27:19

7.3.3. Solution-oriented View

Tables 7.3 and 7.4 report the results in terms of the solution itself, i.e., they provide a solution-oriented view on RotOR's computations.

In order to understand both tables, we recommend to recall equation (4.9) on page 162. There, we explained that RotOR's objective function is composed of 15 objective features that correspond to different aspects of the RSRP to be optimized. Each objective feature $i \in \{1, 2, \ldots, 15\}$ is arranged as a property value $p_i(h)$, which is multiplied by an individual cost coefficient, for each hyperarc $h \in H$ of the hypergraph.

As already mentioned, we present the results always in comparison to the solution of the LP relaxation of RotOR's IP model that has been (globally) solved by RotOR's C2F approach, i.e., we compare the IP solutions to the LP solutions. To this end, each entry in Tables 7.3 and 7.4 is denoted by two values. The sum of both values (zeros are not denoted) gives the value w.r.t. the LP solution, while the first summand is the value w.r.t. the IP solution.

For example, the value $160 - 18$ in column four and row five of Table 7.3 means that the rolling stock rotations (i.e., the IP solution) contain 160 deadhead trips, while the LP solution contains $160 - 18 = 142$ deadhead trips. Note that the 142 is the result

of summing over all (possibly fractional) variables of the LP solution that implement deadhead trips. Clearly, each value $p \in \mathbb{Q}$ that corresponds to a LP solution (i.e., is a second summand) can be fractional but we round to their next integer value (i.e., $\lfloor p + 0.5 \rfloor$) *after* creating the sum over the LP solution. This is just made in order to save space within the tables.

Carefully note that we only report about the property values of the solutions here. The cost values on that RotOR internally computes are in Euro and are confidential. In addition, for the objective features two, three, and four we "only" report the number of covered nodes (of V) and the number of deadhead as well as additional turn around trips, respectively. RotOR's proper model is, of course, based on distance values in kilometer for each of those three objective features instead. We do not report those values here in order to obfuscate possible confidentialities. We expect that the proper values do not increase the scientific value of information here. Also note that a hyperarc can implement up to two deadhead trips that can be almost arbitrarily short, e.g., only a few kilometers from the platform at Berlin Südkreuz to a maintenance facility at Berlin Rummelsburg, see Section 4.4.

7.3.3.1. Greenfield Objectives

Table 7.3 on pages 229 and 231 reports all property values for all 116 RSRP instances for the objective features $1, 2, \ldots, 8$, i.e., without re-optimization objective features. The remaining objective features $9, 10, \ldots, 15$ are denoted by Table 7.4 on page 233 for the re-optimization instances and will be investigated in Section 7.3.3.2. All property values for the re-optimization features are naturally zero for instances dealing with pure greenfield optimization.

Number of railway vehicles. From the second column of Table 7.3 we observe that the number of railway vehicles in the LP and IP solutions very often agree. From that, we speculate that the IP solutions are, at least often, optimal w.r.t. the number of required railway vehicles (this number is usually the primary objective in rolling stock rotation optimization). In some cases the LP solution uses less vehicles. Especially for instances that include the hard maintenance constraint (the dark gray lines), we conjecture that the LP underestimates the required number of railway vehicles (but have, of course, no proof at the moment).

Operation of timetabled trips. The third column of Table 7.3 reports the number of nodes that are covered in the LP and IP solutions. It is less spectacular than the overall number of railway vehicles. But it shows that only sometimes the number of railway vehicles for a certain timetabled trip deviates (in both directions) between the LP and IP solutions. The deviations are very exceptional. By that we *motivate* that it is indeed efficient to first branch on variables associated with hyperarcs that cover timetabled trips and branch on hyperarcs connecting timetabled trips later (i.e., deeper w.r.t. the rapid branching search tree). Our argument is that the LP solution is often right w.r.t. the operation of timetabled trips.

Deadhead and additional turn around trips. RotOR's third objective feature is illustrated by two columns of Table 7.3. They distinguish pure deadhead trips (connecting different locations) from additional turn around trips, see Section 4.4. Note that the number of pure deadhead trips *includes* the number of additional turn around trips. E.g., the LP solution of `RSRP_024G` has 199 (and not 227) proper deadhead trips and 28 additional turn around trips. It is "nice" to see that often the LP solution does not include a single additional turn around trip, when the IP solution has a significant amount of additional turn around trips. We feel that the IP is more right here. The reason is that we investigated (in terms of the cycle embedding approach, see Section 2.2) "how easy" it is for the LP solution to safe additional turn around trips "by fractionally traversing a rotation twice", see Borndörfer et al. (2014, [8, Figure 2]).

Turn duration violation. The sixth column of Table 7.3 states the sum over all turn duration violations in minutes, see Section 4.2. In many cases the LP solution has less "short turns" (i.e., a turn that violates the planned turn duration) than the IP solution has. Turn duration violation is typically penalized significantly, but not in a range where its minimization is paying for, e.g., additional deadhead trips. From our experience, we find the deviations w.r.t. the duration violation reasonable. Note that our paper Borndörfer et al. (2015, [6, Figure 9]) presents a computational study in that the "Impact of turn duration violations" is investigated by a multi-criteria approach. This is accomplished in the same way as we describe it in Section 7.4 (i.e., a weighted-sum method on top of RotOR). By this study it turns out that highly penalizing turn duration violations leads to a significant amount of ICE vehicles that are required in addition (because short turns become very expensive).

Regularity. The next two columns of Table 7.3 state the number of hyperarcs of the LP (for that it could be fractional but is rounded as explained above) and IP solutions that are not contained in a regularity hyperarc. The columns distinguish regular trips and regular turns, see Sections 4.10.1 and 4.10.2. These columns show that the number of regularity hyperarcs for regular trips often agree in the LP and IP solutions but differ sometimes more or less slightly. For regular turns those deviations are often much larger.

Especially in view of the choice of the regularity parameters, i.e., the parameter $R_{\mathfrak{T}}$ for penalizing irregular trips is chosen to be much larger than the corresponding parameter $R_{\mathfrak{T}\times\mathfrak{T}}$ for irregular turns (i.e., $R_{\mathfrak{T}} \gg R_{\mathfrak{T}\times\mathfrak{T}}$, see Sections 4.10.1 and 4.10.2), we find the property values of the IP solutions reasonable. Recall that $R_{\mathfrak{T}\times\mathfrak{T}}$ for regular turns is set to a very small value as explained in Section 4.10.2. For the impact of the choice of $R_{\mathfrak{T}}$ on IP solutions we refer to Section 7.4. We remark (again) that the circumstance that a connection is not performed within a regularity hyperarc does not mean that the connection is not considered to be regular in the rolling stock rotations, i.e., it could be (and often is the case) that a connection is regular for several days of operation but not for all as it is required by RotOR's regularity model for regular turns.

Maintenance. The second last column of Table 7.3 denotes the number of maintenance services that have been created by RotOR along the rolling stock rotations. According to the findings in Section 6.6 it is clear that the LP solution is likely to underestimate those numbers. Especially for instances that include the hard major inspection constraint (the dark gray rows) this effect is significant but reasonable (as we find it).

Coupling activities. Finally, the last column of Table 7.3 illustrates the number of coupling activities (which is naturally equal to the number decoupling activities) in terms of the LP and IP solutions. The penalization of the number of coupling activities is "somewhere between" the one for additional turn around trips and regular trips. This makes it hard to state whether the number of coupling activities is reasonable or not. Note that, for some instances (e.g., `RSRP_043G` and `RSRP_109R`) the LP solution has more coupling activities in comparison to the IP solution. In this situation it is not obvious for what price the coupling activities have been created. This can only be evaluated by a very sophisticated investigation. At this point, we clearly observe the limits of how we compare IP and LP solutions here.

Table 7.3.: Solution-oriented view on industrial RSRP instances without re-optimization objective features $9, \ldots, 15$.

instance	p_1: vehicles		p_2: covered nodes		p_3: deadhead trips		p_3: turn around trips		p_4: turn duration		p_5: trips w/o reg. h.arc		p_6: conn. w/o reg. h.arc		p_7: maint. services		p_8: couplings	
RSRP_001G	8		209		32		1	-1	141		8		81	-12			104	-4
RSRP_002G	13		324		34				140		2		83				120	
RSRP_003G	13		326		34				140		2		86	-2			120	
RSRP_004G	12		185		43				198		17		40					
RSRP_005G	29	-2	761	+7	160	-18	16	-16	353	+3	96	-51	346	-209			380	-84
RSRP_006G	23		708		60	-2	10	-10	49		48	-7	176	-83			336	-54
RSRP_007G	23		710		60		11	-11	20	-12	37		140	-45			300	-30
RSRP_008G	62		1515		299	-28	12	-12	833	-221	99	-35	555	-268			776	-289
RSRP_009G	62		1515		284	-10	27	-27	754	-78	103	-20	446	-140			656	-162
RSRP_010G	62		1529		288	-1	2	-2	759	-167	94	-25	447	-141			640	-132
RSRP_011G	64		1529		342	-27	11	-11	952	-86	100	-33	537	-185			676	-184
RSRP_012G	330		6701		2338	-52	21	-21	2377	-76	1		104	-43			2124	-120
RSRP_013G	330	-1	6715		2387	-104	26	-26	2605	-304	1		116	-55			2648	-616
RSRP_014G	9		205		18	-2					4		109	-66	34	-4	148	-34
RSRP_015G	11	-1	235		76	-8	4	-4	140		4		110	-79	16	-2	184	-22
RSRP_016G	13		326		34						75		138	-35	19	-4	120	
RSRP_017G	13		326		36	-2					75		184	-81	17	-2	120	
RSRP_018G	14		326		35	-1					75		185	-82	17	-2	120	
RSRP_019G	13		326		35	-1			140		2		308	-212	18	-3	120	
RSRP_020G	14		326		34						75		308	-205	16	-1	120	
RSRP_021G	9		193		85	-15			185	+11	3	+10	99	-56	16	-3	40	
RSRP_022G	57		1192		215	-10	7	-7	788	-8	95	-1	430	-105	79	-18	536	-36
RSRP_023G	57		1199		260	-45			717	-147	307	-17	656	-302	78	-17	588	-83
RSRP_024G	57		1203		230	-3	26	+2	783	-37	414		484	-106	84	-23	484	
RSRP_025G	57		1203		230	-7	22	-8	792	-26	270		512	-157	73	-12	532	-20
RSRP_026G	57		1203		232	-6	17	-5	825	-63	196	-9	473	-123	75	-14	540	-15
RSRP_027G	57		1203		224	-11	23	-15	811	-44	301	-6	526	-157	74	-13	524	-21
RSRP_028G	41		1275		146	-15	15	-14	12	-4	91	-6	435	-107	65	-14	708	-47
RSRP_029G	61	-1	1263		248	-16	16	-10	799	-55	293	-22	619	-286	73	-12	580	-94
RSRP_030G	62		1338		302	-7	23	-17	1040	-72	278	-6	538	-173	78	-17	612	-53
RSRP_031G	62		1338		267	-22	21	-13	754	-4	278	-6	525	-161	82	-21	600	-42

Continued on next page

Table 7.3 – continued from previous page

instance	p_1: vehicles		p_2: covered nodes		p_3: deadhead trips		p_3: turn around trips		p_4: turn duration		p_5: trips w/o reg. h.arc		p_6: conn. w/o reg. h.arc		p_7: maint. services		p_8: couplings	
RSRP_032G	54		788		172	-12	2	-2	324	-41	53	-4	412	-206	91	-19		
RSRP_033G	53		788		169	-30	18	-18	500	-137	49	-1	543	-328	81	-10		
RSRP_034G	60	-1	1289	+3	312	-33	17	-17	580	-76	75	-19	639	-329	119	-18	588	-150
RSRP_035G	41	-1	1425		214	-62	50	-50	387	-58	87	-15	706	-380	106	-22	856	-36
RSRP_036G	41	-1	1425		173	-22	41	-41	366	-50	77	-5	625	-345	81	-16	828	-38
RSRP_037G	41	-1	1471	+5	210	-62	45	-45	376	-93	103	-29	769	-414	82	-17	828	-34
RSRP_038G	38	-1	1471		146	-26	40	-39	4473	-62	75		621	-273	63	-14	780	-35
RSRP_039G	39	-1	1472		161	-20	28	-28	310	-44	77	-2	641	-324	68	-17	844	-97
RSRP_040G	54		805		72	-18	31	-31	266	-59	41		519	-322	82	-9		
RSRP_041G	55		884		216	-4	15	-15	759	-248	58		474	-263	94	-17		
RSRP_042G	55		884		118	-6	30	-30	880	-481	58		685	-483	87	-11		
RSRP_043G	7		173	+1	42	+3			1		44	-1	91	-4	10	-2	68	+4
RSRP_044G	9	-1	208	+1	31	+1			1		56		105	-17	11	-2	96	+4
RSRP_045G	7		77		33	+2			15		6	-1	68	-39	7			
RSRP_046G	12	-2	235		61	-5			60		3		58	-29	19	-6	172	-6
RSRP_047G	13		324		36	-3			140		2		159	-66	19	-4	124	-6
RSRP_048G	13		324		41	-5					75		184	-81	18	-3	124	+4
RSRP_049G	13		324		40	-7			140		2		307	-214	18	-3	120	-2
RSRP_050G	13		324		34						49		123	-9	17	-2	124	
RSRP_051G	14	-1	324		34				140		2		165	-70	16	-1	124	
RSRP_052G	14	-1	324		40	-1					75		171	-56	21	-6	124	+1
RSRP_053G	13		324		34						75		136	-20	22	-7	120	+4
RSRP_054G	13		324		34						75		163	-47	19	-4	124	
RSRP_055G	13		324		34				1	-1	75		150	-36	21	-6	124	
RSRP_056G	13		324		34						75		136	-20	18	-3	124	
RSRP_057G	13		325	-1	34				1	-1	75		308	-193	18	-3	124	
RSRP_058G	13		326		38	-4			140		2		309	-213	19	-4	124	
RSRP_059G	17		310		136	-3	4	-4	285		25		141	-70	30	-9		
RSRP_060G	17		310		135	-2	8	-8	285		25		310	-236	27	-6		
RSRP_061G	22		336		96	-10	7	-7					161	-140	32	-11		
RSRP_062G	41	-1	1226		142	-19	12	-12	8	-8	54		1110	-921	87	-21	732	-68
RSRP_063G	41	-1	1226		134	-12	18	-18			56	-2	480	-297	92	-26	752	-88
RSRP_064G	40	-1	1254		115	-10	18	-15	18	-6	54		334	-163	86	-20	728	-73
RSRP_065G	68	-2	1350	-5	331	-68	24	-16	1048	+171	498	-14	895	-347	85	-20	520	-19
RSRP_066G	68	-2	1350		281	-11	10	-10	807	+382	79	-12	807	-343	87	-22	476	-26
RSRP_067G	67	-1	1349	+1	292	-20	18	-18	943	+246	66		848	-384	81	-16	468	-22
RSRP_068G	68	-2	1349	-4	325	-62	20	-12	930	+289	512		890	-329	87	-22	528	-27
RSRP_069G	68	-2	1347	-2	314	-48	28	-28	1211	+310	66		933	-418	84	-19	544	-54
RSRP_070G	68	-2	1345		339	-71	26	-19	1083	+437	520	-8	979	-406	84	-19	536	-35
RSRP_071G	68	-2	1353	-8	312	-51	25	-18	959	+260	511		903	-347	84	-19	540	-39
RSRP_072G	69	-2	1348	-4	339	-76	25	-17	915	+304	510	+1	953	-383	88	-23	556	-54
RSRP_073G	67	-1	1351	-6	324	-58	33	-33	1150	+371	71	-5	927	-412	85	-20	560	-70
RSRP_074G	61	-1	1464	-5	294	-41	26	-20	783	-455	442		938	-486	125	-24	752	-234
RSRP_075G	66	-1	1379		289	-54	41	-41	628	-111	78	-11	815	-472	135	-25	820	-211
RSRP_076G	55		854		240	-39	40	-40	685	-60	75	-13	656	-387	90	-17		
RSRP_077G	53		884		160	-66	46	-46	710	-15	56	-7	584	-368	104	-28		
RSRP_078G	55		884		238	-44	36	-36	947	-20	71	-13	617	-373	95	-18		
RSRP_079G	144	-1	2015		954	-160	40	-40	210	+56	190	-38	1500	-798	143	-36	652	-105
RSRP_080G	122		1488		687	-99	39	-39	232	-78	136	-4	1092	-633	126	-32		
RSRP_081G	120		1488		704	-115	45	-45	219	+125	144	-12	1262	-784	124	-30		
RSRP_082G	129	-3	2808	-5	573	-58	61	-48	1643	-95	942	+11	1877	-864	208	-42	1356	-338
RSRP_083G	112	-8	2424		464	-103	62	-62	2374	-631	104	-14	2277	-1716	184	-41	988	-173
RSRP_084G	110	-6	2424		421	-60	67	-67	2208	-464	90		2277	-1716	180	-37	988	-173
RSRP_085G	237	-7	5233	-6	1025	-165	90	-69	3443	-382	2316	+11	4782	-3306	385	-77	2264	-425
RSRP_086G	236	-6	5234	-14	1022	-187	97	-76	3420	-395	2305	+19	4799	-3285	413	-100	2332	-520
RSRP_087G	40	-1	1254		127	-22	20	-17	111	-99	55	-1	466	-286	111	-40	832	-176
RSRP_088R	9		222		44				183		5		26				112	-2
RSRP_089R	39		1235		200		18		59	-10	139		389	-6			792	-39
RSRP_090R	39		1279		210	-14	17	+1	333	-178	189	-25	483	-73			700	-34
RSRP_091R	7		146		87	-4			203		2		35	-14	13	-3	48	
RSRP_092R	18		259		84	-2	13		66		12		76	-16	35	-10		
RSRP_093R	18		259		82		23		66		12		75	-15	34	-9		
RSRP_094R	57		1192		216	-2	45		889		95		435	-9	97	-4	544	
RSRP_095R	57		1203		236	-2	70		1025		419		466	-3	107	-5	548	-2
RSRP_096R	57		1203		233		71		1096	-28	288		637	-167	106	-7	548	-2
RSRP_097R	40		1226		153		10		14		106		351	+1	68	-2	848	
RSRP_098R	41	-1	1226		130	-3	3		8		106		357	-8	67	-1	852	-6
RSRP_099R	61		1263		344	-6	57		1698	-72	276	+2	516	-24	93	-16	576	-1
RSRP_100R	60		1263		253	-4	89	-3	1166	-11	302	-8	581	-32	90	-14	580	-10

Continued on next page

Table 7.3 – continued from previous page

instance	p_1: vehicles		p_2: covered nodes		p_3: deadhead trips		p_3: turn around trips		p_4: turn duration		p_5: trips w/o reg. h.arc		p_6: conn. w/o reg. h.arc		p_7: maint. services		p_8: couplings	
RSRP_101R	61		1263		346	-7	57	+1	1626	-7	291	-11	520	-24	92	-15	592	-16
RSRP_102R	62		1263		394	-11	52		1150	+22	290		535	-13	93	-16	592	-24
RSRP_103R	61		1263		383	-7	58		1429	-10	287		569	-21	90	-12	604	-22
RSRP_104R	63	-1	1338		319	-4	72	-3	1165	+39	303		783	-207	89	-12	640	-14
RSRP_105R	62		1338		294	-9	78	+2	1199	-55	286	+5	569	+3	93	-17	636	-24
RSRP_106R	54		788		147		25		671	-10	85		300	-15	105	-11		
RSRP_107R	53		788		148	-8	46		516	+7	89		310	-36	101	-22		
RSRP_108R	54		788		148	-1	14		653	+60	97		324	-11	108	-4		
RSRP_109R	59		1299		349	-10	47	-5	945	+19	318	+1	659	-28	151	-10	696	+4
RSRP_110R	55		805		153		10		723		77		320		113			
RSRP_111R	69		1412		411	-15	25	-5	1494	-72	109	-8	743	-231	156	-23	760	-128
RSRP_112R	9		222		23				71		5		94		77		196	
RSRP_113R	9		243		35				193		4		59	-14	15		184	-4
RSRP_114R	40		1226		127		10		14		106		608	-243	99	-19	848	
RSRP_115R	40		1226		127		12	-2	14		106		370	-12	100	-20	848	-7
RSRP_116R	40		1226		127		10		14		107	-1	374	-9	88	-7	848	

7.3.3.2. Re-optimization Objectives

Table 7.4 on page 233 finalizes the comparison of the LP and IP solutions in terms of the objective features for the re-optimization of the re-optimization RSRPs of our test set.

The columns of Table 7.4 denote the deviations between the reference rotations and the LP and IP solutions. To this end, the table denotes the (overall) number of *deviations* of the vehicle configuration, the fleet, the (reference) rotation, the vehicle orientation, and the position (within the vehicle composition) that RotOR has created w.r.t. railway vehicles *operating* timetabled trips within the reference rotations. Recall that RotOR does neither count, nor penalize deviations when departures or arrivals of the "new" timetable have significantly changed, see Section 4.12.2. In addition, the last two columns of Table 7.4 report the number of deviations w.r.t. railway vehicles *connecting* timetabled trips within the reference rotations in terms of the connection itself and w.r.t. maintenance services that appear before or after a timetabled trip in the reference rotations.

Carefully note that a zero entry in Table 7.4 *does not* mean that the reference rotations immediately provide the optimal solution to the re-optimization scenario. Even for the solution of `RSRP_092R`[9] RotOR had to find 43 out of 257 connections between timetabled trips "on its own" because not all connections can be directly derived from the reference rotations due to timetable changes (for `RSRP_093R` it is similar).

We clearly observe from Table 7.4:

- ▷ Table 7.4 is rather sparse. This is as nice as reasonable because it suggests that the re-optimization of the reference rotations is almost always possible without changing a lot and, moreover, this has been also accomplished by RotOR.

[9] `RSRP_092R` has 259 timetabled trips, which are all to be operated with a vehicle configuration of size one, see Table 7.1.

- ▷ Often, the (number of) deviations that are made in the IP solution are also already made in the LP solution. From that we suspect that RotOR is often right when deviations are unavoidable (or very inefficient) when operating the new timetable of the re-optimization scenario.
- ▷ By the last column of Table 7.4 we see that even maintenance services can often be "easily" derived from the reference rotations. This is the main reason why we claim that re-optimization is often easier and faster than pure greenfield optimization when using RotOR.
- ▷ Even the hard major inspection constraint could be relatively easily handled by arranging appropriate major inspections into the rolling stock rotations. Note that the reference rotations do not provide such major inspections for the last five re-optimization instances of the RSRP denoted in Table 7.4.

Table 7.4.: Solution-oriented view on industrial re-optimization RSRP instances without greenfield objective features $1, \ldots, 8$.

instance	p_9: configurations		p_{10}: fleets		p_{11}: rotations		p_{12}: orientations		p_{13}: positions		p_{14}: connections		p_{15}: maintenances	
RSRP_088R									1		2			
RSRP_089R											10			
RSRP_090R	44	-2			106	-11		+2	194	+10	91	-15		
RSRP_091R											4			
RSRP_092R														
RSRP_093R														
RSRP_094R							1		13		15		11	
RSRP_095R	3		6		6		528		12		11		13	
RSRP_096R	3		6		6		417		12		25	-1	13	+1
RSRP_097R											21			
RSRP_098R									1		23	-1	2	
RSRP_099R	5						173		3		24	+3	10	
RSRP_100R	5						173		3		22	-1	10	
RSRP_101R	5						173		3		25	+2	10	
RSRP_102R							173		3	+1	27		10	
RSRP_103R	5						174		3		27	+1	10	
RSRP_104R	5		1		1		189		11	-5	35	-12	11	
RSRP_105R	5		1		1		189		5		24	-1	11	
RSRP_106R							3				12			
RSRP_107R							1				1			
RSRP_108R											1			
RSRP_109R	25	-17	65	-20	69	-22	11	-1	15	-3	62	-25	6	
RSRP_110R											1			
RSRP_111R	1				1				13		40	-2	14	
RSRP_112R											14		14	
RSRP_113R									6	-4				
RSRP_114R										+2	4			
RSRP_115R									3	-1	27	-3	9	+1
RSRP_116R									2		25	-1		

7.4. The Price of Regularity

As already mentioned, we use RotOR's hypergraph to model vehicle compositions, but also for regularity requirements, see Section 4.10. In this section we take a closer look at regularity and provide a case study to underline RotOR's benefit for the railway industry, e.g., DBF.

In our context a train is a set of at most seven timetabled trips, which are associated with the seven days of operation of the standard week. In RSRP instances provided by DBF, in most of all cases the trips of a single train only differ w.r.t. the day of operation. This means that the departure and arrival locations and corresponding times are equal. This is made to provide a timetable that is periodic (or regular) to the passengers. It also yields a quality measure for a timetable, i.e., to operate trains that do not differ over the operational days and operate on all seven days is a desired result in timetabling. This also transfers to rolling stock rotation optimization.

In Figure 4.21 on page 145 we consider a train with train number 408. In this example train 408 is operated by seven different vehicle compositions on each day of operation and shows the most irregular case. In rolling stock rotation optimization it is desired to operate train 408 with the same vehicle composition on each day of operation. We handle this by introducing appropriate hyperarcs. Let π be one of the eight possible vehicle compositions to operate a trip of train 408 with a composition of a red and a blue vehicle. Thus, π defines the position and orientation of the red as well as the blue vehicle. Suppose that $h^\pi_{\text{Mon}}, \cdots, h^\pi_{\text{Sun}}$ are those hyperarcs that model the operation of the seven timetabled trips w.r.t. π. We penalize the choice of $h^\pi_{\text{Mon}}, \cdots, h^\pi_{\text{Sun}}$ by a constant factor $R_{\mathfrak{T}} \in \mathbb{Q}_+$ in RotOR's objective function and introduce:

$$h^\pi_{408} := \bigcup_{\text{day=Mon}}^{\text{Sun}} h^\pi_{\text{day}} \quad \text{with} \quad c(h^\pi_{408}) := \sum_{\text{day=Mon}}^{\text{Sun}} (c(h^\pi_{\text{day}}) - R_{\mathfrak{T}}).$$

By this definition it is cheaper to choose the regularity hyperarc h^π_{408} for the operation of train 408 compared to the other seven individual hyperarcs that may differ in their vehicle composition in a solution. These regular hyperarcs are introduced for many possible vehicle compositions of a train, see Section 4.10.1.

In this case study we investigate the price of regularity by a multi-criteria optimization, see the book Ehrgott (2005, [43]). This means, that we compute all Pareto-optimal solutions of the following two objective functions:

1. Minimize RotOR's objective function without maximizing regular trips, i.e., function (4.9) on page 162 except for the fifth objective feature (i.e., $p_5(h) = 0$ is assumed for all $h \in H$) that includes, e.g.,:
   - ▷ cost for railway vehicles
   - ▷ cost for deadhead trips
   - ▷ cost for additional turn around trips
   - ▷ cost for violating planned turn durations

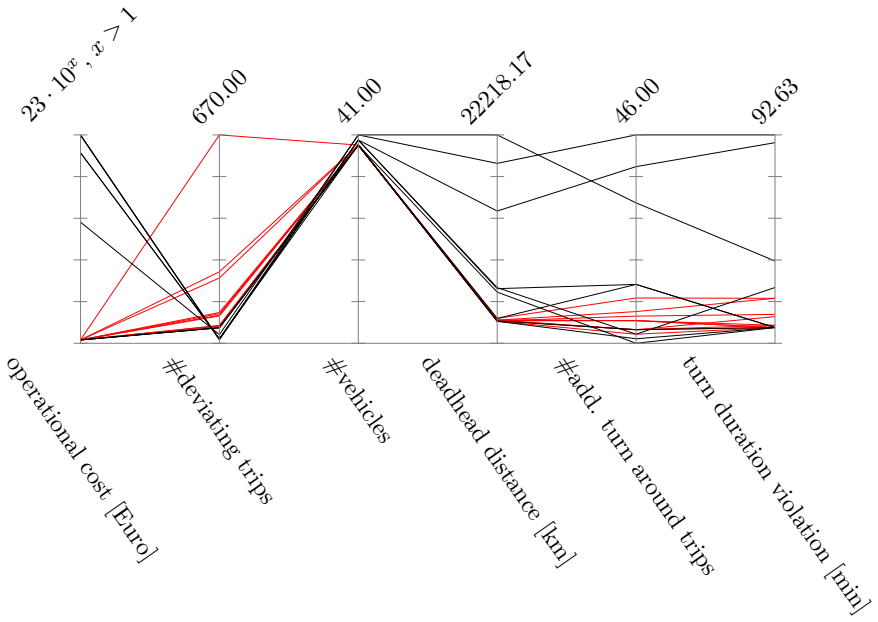


Figure 7.1.: The price of regularity.

2. Maximize regular trips, i.e., RotOR's objective function (4.9) on page 162 restricted to the fifth objective feature (i.e., $p_i(h) = 0$ is assumed for all $h \in H$ and for all $i \in \{1, \ldots, 15\} \setminus \{5\}$)

We use the weighted-sum algorithm [43] for this multi-criteria optimization. In a nutshell, this algorithm starts with two solutions that only minimize one of the two objective functions and refines the Pareto-front in a recursion tree. In each computation step, the value of $R_{\mathfrak{T}}$ is chosen by a gradient argument in the weighted-sum method. A solution of the RSRP w.r.t. the dedicated choice of $R_{\mathfrak{T}}$ is then computed by RotOR. Therefore, this multi-criteria approach can (and does) run completely on top of RotOR.

We consider a dedicated instance provided by our industrial cooperation partner DBF. The instance consists of 670 timetabled trips that are widely spread over the German ICE network. At 52 locations passengers can get on a train or disembark from a train. The instance contains 127 trains overall. Possible vehicle compositions are composed of at most two vehicles of the same fleet. The number of hyperarcs of the resulting hypergraph is 4,946,356.

Figure 7.1 shows the results of the multi-criteria optimization. We obtained eight Pareto-optimal solutions, which are indicated by the black paths in Figure 7.1. Each Pareto-optimal solution has either less operational cost, which are confidential and therefore obfuscated, or less deviating trips compared to all other solutions, i.e., it is non-dominated. The solutions indicated by the red paths are dominated in one objective function by another solution. Those solutions appeared during the computation of the set of Pareto-optimal solutions. We precisely observe:

- ▷ The number of vehicles needed to operate the timetable is almost constant if we vary the amount of regularity in the rolling stock rotations.
- ▷ The number of additional turn around trips varies – from zero to 46. This is directly related to the deadhead distance. Using 46 turn around trips for this instance is practically not implementable, it is much too much.
- ▷ Also the turn duration violation (see Section 4.2) varies, i.e., from 10 minutes overall to almost 93 minutes. From an industrial point of view we claim that a turn duration violation of 93 minutes is not much for the considered instance.

What is the price of regularity? The higher the amount of regularity in the rolling stock rotations, the higher the number of additional turn around trips to appropriate equal vehicle compositions for trains. Additional turn around trips are expensive and very unattractive. Irregularities are also to be avoided as much as possible in operation. Therefore, finding a good compromise between the two objectives is a very challenging task for railway operators. Furthermore, this offers the important insight, that an optimization software can be very beneficial to handle the complexity in the railway competition in order to simulate "what if" real-world scenarios.

7.5. The Köln-Rhein-Main Construction Site

We introduced the thesis by good news and, consequently, conclude it by

Bad news: *Bauarbeiten: ICE-Strecke von Frankfurt nach Köln zeitweise gesperrt* [46].
This news appeared in the "FAZ" (a prominent German newspaper). It announced that the railway tracks for the high-speed ICE lines between Frankfurt (Main) and Cologne would be completely closed on four succeeding weekends in May 2015. The reason was that Deutsche Bahn had to rebuild 17 kilometer long rails. In this section, we call the corresponding occasion the Köln-Rhein-Main construction site (KRM).

In general, the KRM was bad news: It caused an additional driving time of approximately 60 minutes between Frankfurt and Cologne because the railway vehicles (i.e., ICEs) were redirected through rails via the Rhine. The rails of the redirection are not appropriate for high-speed operations. Therefore, the KRM directly affected the timetable of eight ICE lines that usually run over the involved rails via Montabaur with high speed and a high frequency, see Figure 7.2.

However, the KRM is good news for this thesis in the following sense: Before the news were published, RotOR was used by DBF in order to compute appropriate rolling stock rotations for the KRM. This application is the subject of this section.

Redirection via the Rhine. The KRM had drastic consequences for the operation of the ICE vehicles of DBF compared to other construction sites. The main reason is that an increased driving time for eight (out of overall 27) ICE lines causes unusually many "standard" rolling stock rotations to become infeasible. In fact, it is the increase by 60 minutes that matters here because a driving time extension of, e.g., half an hour can sometimes be handled more easily: Many ICE lines operate with a period of one hour and, therefore, an often successful approach is to shift connections between timetabled trips (i.e., turns) by one period. For the KRM it was not obvious how to implement this approach, i.e., an easy solution was not at hand.

Scenario isolation. During the planning of a construction site by DBF, rotation planners try to make minimally invasive changes to the existing timetable and rolling stock rotations. To this end, it is desirable to isolate the planning scenario as much as possible from the remaining part of the ICE network. This is done in order to minimize negative side effects (e.g., the propagation of delays and a decreased volume of passenger of a construction site) on other parts of the ICE network.

For the KRM the isolation procedure was obvious: The timetable changes[10] as illustrated in Figure 7.2 suggest that the operation (in particular the rolling stock rotations) of all ICE vehicles of the category ICE-W[11] of DBF are affected by the KRM. The corresponding ICE lines are the lines 41, 42, 43, 45, 47, 49, 78, 79, and 82. This might

[10] We do not provide the detailed changes of departures and arrivals here. The important aspect is that the eight lines illustrated in Figure 7.2 were directly affected by the 60 minutes increased driving time.
[11] The "W" indicates that the ICE-W vehicles are equipped with "Wirbelstrombremsen", i.e., eddy current brakes.

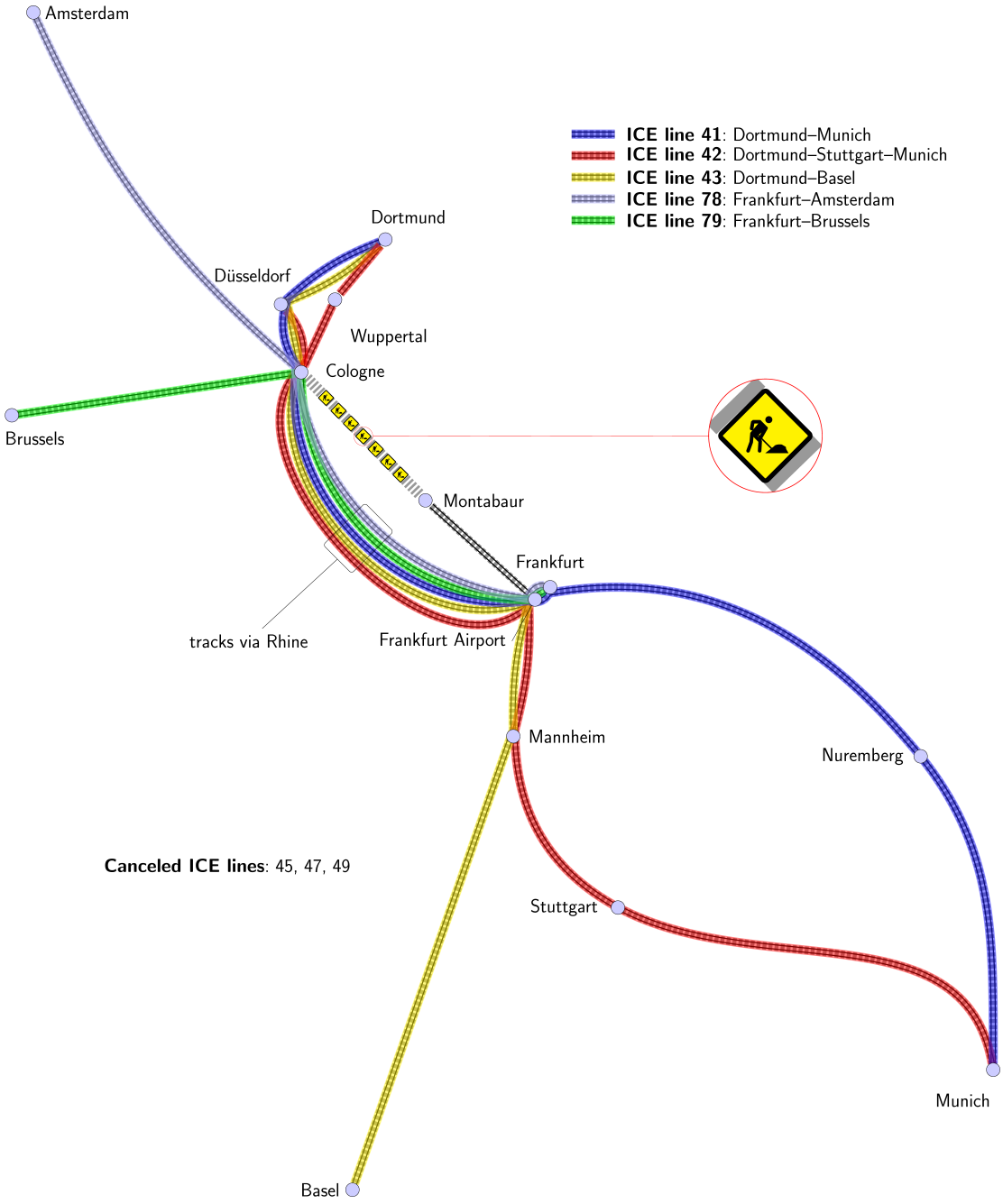


Figure 7.2.: Redirection for the Köln-Rhein-Main construction site (KRM): All ICE lines between Montabaur and Cologne were closed in order to rebuild rails (as indicated by the sleepers without rails) on four succeeding weekends in May 2015. Note that the figure only illustrates the situation at the weekend when the KRM is performed, i.e., the ICE lines 41, 42, 43, 45, 47, 49, 78, 79, and 82 operate as usual on the high-speed tracks between Frankfurt and Cologne from Monday to Friday. Note also that the placement of the locations is true to scale.

not really appear as a minimally invasive isolation at first glance. In fact, in case of the KRM it was the minimal set of ICE lines to consider!

It turned out that line 82, which connects Frankfurt to Paris, could be treated separately. The ICE vehicles for line 82 have special equipment for France and are usually not mixed with the other ICE vehicles on other ICE lines.

RSRP setup. For the remaining eight ICE lines an instance of the RSRP dedicated to the KRM was set up by rotation planners of DBF as follows. The original timetable for the seven remaining ICE lines for Monday to Friday was taken without modifications. The timetabled trips of the ICE lines 41, 42, 43, 78, and 79 were modified on the weekend in order to reflect the increased driving time caused by the redirection via the Rhine. The trips of the ICE lines 45, 47, and 49 that operate on the weekend were canceled. Then, allowed vehicle configuration requirements were specified for the "KRM timetable". Not surprisingly, rotation planners of DBF reported that the setup phase took a long time because a lot of data had to be manually modified for the KRM.

The standard rolling stock rotations were declared as reference rotations. Thus, the KRM RSRP was a proper re-optimization scenario. It contained all industrial requirements described in Chapters 4 and 5 except for infrastructure capacity constraints.

Optimization rounds. It is far too much to expect that RotOR immediately computed an outstanding result for such a complex industrial instance. Beside RotOR's pure computation time it is almost always certain that the first computations will reveal data issues. Naturally, such a complex RSRP instance, which was manually set up "under fire", will not immediately model what is desired in the first shot. Therefore, we are not surprised that approximately four rounds were necessary to obtain the first usable rolling stock rotations. And even if RotOR had produced those rotations in the first place, further rounds were needed to tune the timetable and rotations.

Keep in mind, that rolling stock rotations often give rise to further timetable changes which involve further optimization rounds. What we learn from this is that optimization (equally whether automatic or manual) can easily run into a loop that will never stop. This is a serious issue that all optimization approaches share and which can be minimized by, e.g., software tools dedicated to the preparation of RotOR's input data and also for further processing RotOR's output data. Fortunately, also those software tools are at hand at DBF. Thus, the optimization rounds, indeed, came to an end. Who would have thought it?

The KRM RSRP. Most of the final results were obtained from RotOR by computing a solution to a certain RSRP instance at DBF. We call this instance KRM RSRP in this section. In fact, it was was contributed by DBF to our test set and it is called `RSRP_111R` in Section 7.3. The timetable of the KRM RSRP consists of 169 trains, 1025 timetabled trips, and 838 timetabled trip sequences. Seven reference rotations are to be re-optimized and also seven fleets are given. The rotation templates for these reference rotations lead to a refinement of 15 original vehicle configurations to 30 refined configurations.

RotOR's hypergraph for this instance has 69,751,046 hyperarcs and one maintenance constraint (i.e., the regular inspection) is contained.

At the end of the year 2014, RotOR version 2.3 computed a solution to the KRM RSRP within approximately one day at DBF. This is almost equal to the behavior of RotOR version 2.4 which found a solution to `RSRP_111R` with a primal gap of 0.84% during the computational study of Section 7.3 at ZIB.

KRM rotations. The eight rolling stock rotations that RotOR produced for the KRM RSRP, namely the KRM rotations, are illustrated in Figure 7.3. The rows of the figure alternate between the reference rotations and the KRM rotations. The red blocks indicate those parts of the reference rotations (which would have been operated if the KRM did not exist) that became infeasible on the occasion of the timetable changes for the KRM.

Even if the whole page with anonymous rollings stock rotations appears as slightly over-ambitious to the reader, the figure clearly shows that the KRM rotations are very (very) similar to the reference rotations between Monday noontime and Friday noontime. This provides the opportunity to easily bridge between the reference rotations and the KRM rotations on all the eight boundaries of the four KRM weekends. These rotations were made "camera-ready" by rotation planners of DBF in order to operate them in May 2015.

The following statement is from Sebastian Mänz, who was responsible for the planning of the KRM at DBF.

> "Die Umlaufplanung der Baustelle Köln-Rhein/Main war mit Hilfe von RotOR effizient und auf hohem Qualitätsniveau möglich: Nur eine Person konnte in etwa der Hälfte der üblichen Zeit eine Planung mit höherer Stabilität und Verlässlichkeit in der Kundenkommunikation erstellen."

Finally, I would like to thank Matthias Breuer, Sebastian Mänz, and Patrick Siebeneicher for the support in order to prepare this section.

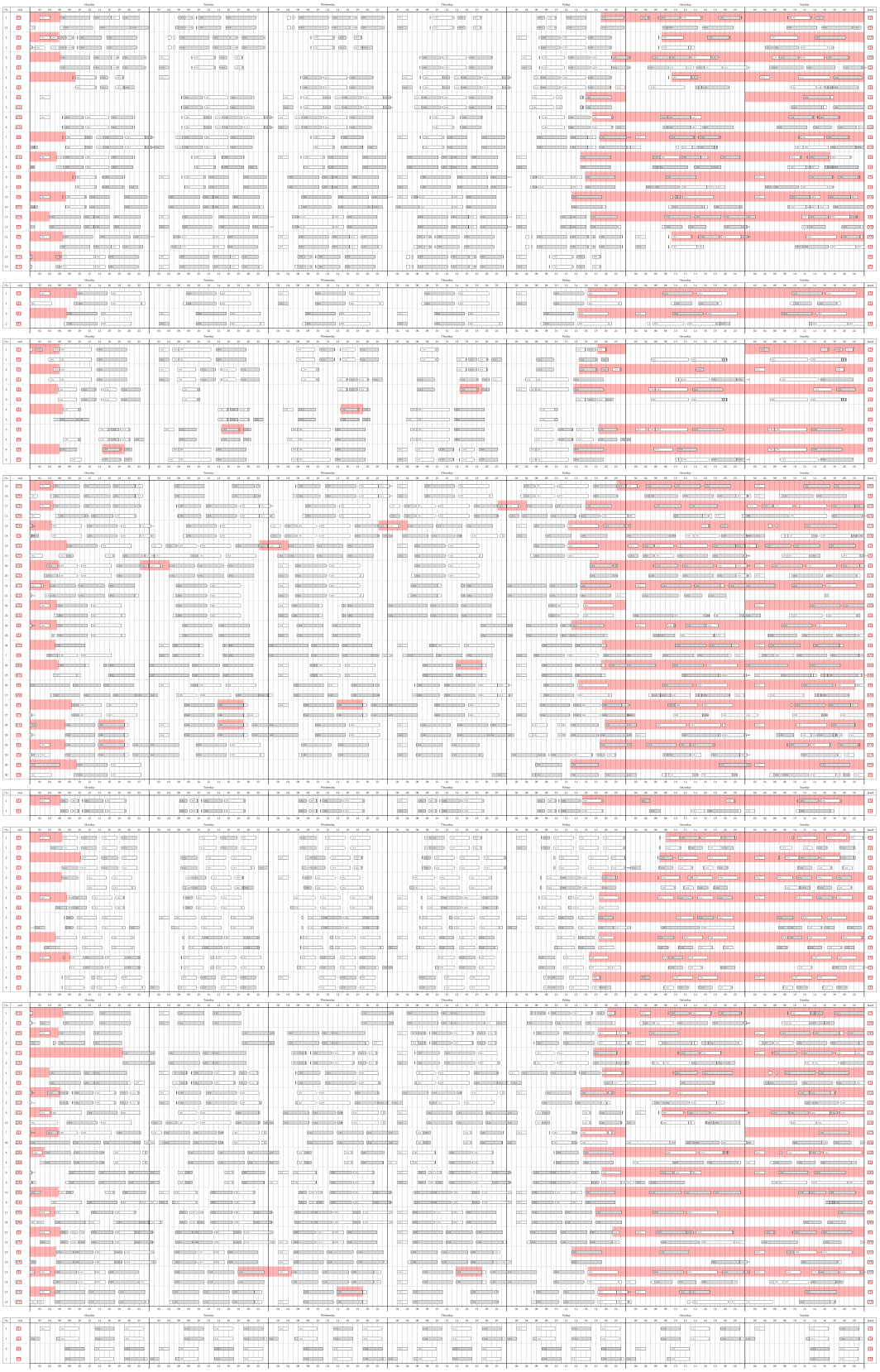


Figure 7.3.: KRM rotations in comparison to the reference rotations.

Abbreviations

RotOR	Rotation Optimizer for Railways
AaD	aggregation and disaggregation
ACVRP	asymmetric capacitated vehicle routing problem
AP	standard assignment problem
ATSP	asymmetric traveling salesman problem
B&B	branch and bound
B&C	branch and cut
BFS	breadth-first search
BP	binary program
C2F	coarse-to-fine
CCH	coupling component heuristic
CDH	cut date heuristic
CEP	cycle embedding problem
CGA	column generation algorithm
CIP	constraint integer program
CPU	central processing unit
CVRP	symmetric capacitated vehicle routing problem
DBF	DB Fernverkehr AG
DPP	disjoint path problem
GB	giga bytes
HAP	hypergraph assignment problem
ICE	intercity express
IP	integer programming
IPA	interior point algorithm
ISUD	integral simplex using decomposition

KRM	Köln-Rhein-Main construction site
LP	linear programming
MB	mega bytes
MDVRP	multi-depot vehicle routing problem
MDVSP	multiple depot vehicle scheduling problem
MIP	mixed-integer programming
MP	master problem
PESP	periodic event scheduling problem
QAP	quadratic assignment problem
RAM	random-access memory
RB	rapid branching
RCAP	resource-constrained assignment problem
RCSPP	resource-constrained shortest path problem
RENS	relaxation enforced neighborhood search
RINS	relaxation induced neighborhood search
RMP	restricted master problem
RS	regional search
RSRP	rolling stock rotation problem
SCIP	SCIP (solving constraint integer programs)
TSP	symmetric traveling salesman problem
TTP	train timetabling problem
VRP	vehicle routing problem
ZIB	Zuse Institute Berlin

List of Figures

1.1. Timetable torus 14
1.2. Problem extensions 27

2.1. Cycle embedding problem 41
2.2. Layers for the RSRP 50
2.3. Coarsening of flow conservation constraints 54

3.1. Extension of the RCAP to the RSRP 67
3.2. RCAP solution 73
3.3. Alternating cycle in bipartite graph 76
3.4. Alternating cycle region 86
3.5. k-opt move 87
3.6. Possible flip operations 88
3.7. Primal-dual gap for move-based RS 103
3.8. Primal-dual gap for branching-based RS 103

4.1. Vehicle configurations 115
4.2. Choice of vehicle configurations 116
4.3. Hypergraph model for vehicle configuration 117
4.4. Uncompressed and compressed hypergraph models 121
4.5. ICE railway tracks of Hamburg 122
4.6. Concept of sides 123
4.7. Sophisticated deadhead trips 126
4.8. Service path 128
4.9. Vehicle orientations 129
4.10. Change of orientation 130
4.11. Orientation consistency I 130
4.12. Orientation consistency II 130
4.13. Orientation consistency III 131
4.14. Orientation consistency IV 131
4.15. Vehicle compositions 133
4.16. Change of vehicle compositions 134
4.17. Hypergraph model for vehicle composition 135
4.18. Trip sequences 138
4.19. Splitting and combining 141
4.20. Rotation visualization 144
4.21. Irregularities 145
4.22. Regular trip hyperarcs 147

4.23. Regular turn hyperarcs . . . 149
4.24. Segments of a rotation . . . 153
4.25. Rotation handouts . . . 154
4.26. Re-optimization . . . 156
4.27. Re-optimization hypergraph . . . 159
4.28. Rotation reference . . . 164

5.1. Resource flow model . . . 174
5.2. Parking facility . . . 175
5.3. German ICE network . . . 177

6.1. C2F constraint matrix generation . . . 188
6.2. Rapid branching trees . . . 191
6.3. Cut date heuristic . . . 193
6.4. Infeasible couplings . . . 195
6.5. Fractional service path . . . 201

7.1. The price of regularity . . . 235
7.2. Redirection for the KRM . . . 238
7.3. KRM rotations . . . 241

List of Tables

1.1. Industrial requirements of ICE rotations 36

2.1. Comparison of C2F ideas 39
2.2. C2F for large instances 61
2.3. C2F for medium instances 62
2.4. C2F for small instances 64

3.1. Twelve out of 384 branching rules 95
3.2. Global and regional search for RSRP instances 101
3.3. Regional search for VRP instances 102
3.4. Global and regional search for VRP instances 106

7.1. Problem-oriented view 217
7.2. Algorithmic view 223
7.3. Solution-oriented view I 229
7.4. Solution-oriented view II 233

Algorithms

2.1. C2F column generation 46
2.2. C2F hyperarc generation 58

3.1. Primal Hungarian method 76
3.2. Dual variable initialization 77
3.4. Alternating cycle check 78
3.3. Alternating cycle search 79
3.5. Regional search loop 82
3.6. Proof of regional optimality 83
3.7. Gram-Schmidt diversification 90

6.1. RotOR's workflow 182
6.2. C2F constraint matrix generation 187
6.3. Rapid branching 190
6.4. Cut date heuristic 193
6.5. Coupling component heuristic 200
6.6. Fractional maintenance bound 203
6.7. Two-stage handout heuristic 204
6.8. First stage of the handout heuristic 205
6.9. Second stage of the handout heuristic 206

Bibliography

[12] Tobias Achterberg. “Constraint Integer Programming”. PhD thesis. Technische Universität Berlin, 2009. URN: urn:nbn:de:0297-zib-11129 (cited on pages 84, 86, 93, 95, 102).

[13] Tobias Achterberg. “SCIP: Solving constraint integer programs”. In: *Mathematical Programming Computation* 1.1 (July 2009), pp. 1–41. DOI: 10.1007/s12532-008-0001-1 (cited on pages 197, 210).

[14] Sepideh Ahmadi, Sascha F. Gritzbach, Kathryn Lund-Nguyen, and Devita McCullough-Amal. *Rolling Stock Rotation Optimization in Days of Strike: An Automated Approach for Creating an Alternative Timetable*. English. Tech. rep. 15-52. Takustr.7, 14195 Berlin: ZIB, 2015. URN: urn:nbn:de:0297-zib-56425 (cited on page 157).

[15] Ravindra K. Ahuja, Jian Liu, James B. Orlin, Dushyant Sharma, and Larry A. Shughart. “Solving Real-Life Locomotive-Scheduling Problems”. In: *Transportation Science* 39 (4 Nov. 2005), pp. 503–517. ISSN: 1526-5447. DOI: 10.1287/trsc.1050.0115 (cited on page 16).

[16] Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis, eds. *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*. Vol. 5868. Lecture Notes in Computer Science. Springer, 2009. ISBN: 978-3-642-05464-8. DOI: 10.1007/978-3-642-05465-5 (cited on page 17).

[17] Luzi Anderegg, Stephan Eidenbenz, Martin Gantenbein, Christoph Stamm, and et al. “Train Routing Algorithms: Concepts, Design Choices, and Practical Considerations”. In: *Proceedings of the 5th Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 2003, pp. 106–118 (cited on page 16).

[18] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press, 2007. ISBN: 9780691129938 (cited on page 70).

[19] Alper Atamtürk, George L. Nemhauser, and Martin W. P. Savelsbergh. “Conflict Graphs in Integer Programming”. In: *European Journal of Operations Research* 121 (1998), pp. 200–. DOI: doi:10.1016/S0377-2217(99)00015-6 (cited on page 197).

[20] M. L. Balinski and R. E. Gomory. “A Primal Method for the Assignment and Transportation Problems”. In: *Management Science* 10.3 (1964), pp. 578–593. DOI: 10.1287/mnsc.10.3.578 (cited on pages 69, 71, 74, 75, 80).

[21] M. L. Balinski and Andrew Russakoff. “On the Assignment Polytope”. English. In: *SIAM Review* 16.4 (1974), pp. 516–525. ISSN: 00361445. URL: http://www.jstor.org/stable/2028692 (cited on page 85).

[22] Andreas Bärmann, Frauke Liers, Alexander Martin, Maximilian Merkert, Christoph Thurner, and Dieter Weninger. “Solving network design problems via iterative aggregation”. English. In: *Mathematical Programming Computation* 7.2 (2015), pp. 189–217. ISSN: 1867-2949. DOI: 10.1007/s12532-015-0079-1 (cited on pages 39, 43).

[23] Sebastian Behrendt. “Dienstreihenfolgeplanung mit ganzzahliger Optimierung”. German. Diplomarbeit. Technische Universität Berlin, July 2008 (cited on page 16).

[24] Timo Berthold. “Heuristic algorithms in global MINLP solvers”. PhD thesis. Technische Universität Berlin, 2014. ISBN: 978-3843919319 (cited on pages 82, 94).

[25] Timo Berthold. “RENS”. In: *Mathematical Programming Computation* 6.1 (2013), pp. 33–54. ISSN: 1867-2957. DOI: 10.1007/s12532-013-0060-9 (cited on pages 70, 71).

[26] Marco Blanco and Thomas Schlechte. “Analysis of Micro–Macro Transformations of Railway Networks”. In: *Operations Research Proceedings 2013*. (URN links to ZIB report). 2014, pp. 37–42. DOI: 10.1007/978-3-319-07001-8_6. URN: urn:nbn:de:0297-zib-42710 (cited on page 40).

[27] Ralf Borndörfer. “Aspects of Set Packing, Partitioning, and Covering”. PhD thesis. Technische Universität Berlin, 1998. URN: urn:nbn:de:0297-zib-10126 (cited on page 71).

[28] Ralf Borndörfer, Andreas Langenhan, Andreas Löbel, Christof Schulz, and Steffen Weider. “Duty scheduling templates”. English. In: *Public Transport* 5.1-2 (2013), pp. 41–51. ISSN: 1866-749X. DOI: 10.1007/s12469-013-0064-x (cited on pages 157, 161).

[29] Ralf Borndörfer, Andreas Löbel, and Steffen Weider. “A Bundle Method for Integrated Multi-Depot Vehicle and Duty Scheduling in Public Transit”. In: *Computer-aided Systems in Public Transport*. Ed. by Mark Hickman, Pitu Mirchandani, and Stefan Voß. Vol. 600. Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, 2008, pp. 3–24. DOI: 10.1007/978-3-540-73312-6_1 (cited on page 191).

[30] Ralf Borndörfer, Thomas Schlechte, and Steffen Weider. “Railway Track Allocation by Rapid Branching”. In: *Proceedings of the 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. Ed. by Thomas Erlebach and Marco Lübbecke. Vol. 14. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, pp. 13–23. ISBN: 978-3-939897-20-0. DOI: 10.4230/OASIcs.ATMOS.2010.13 (cited on page 189).

[31] Michael Brückner. “Polyedrische Approximation von Punktwolken und ihre Anwendung in der ICE-Abstellungsplanung”. Bachelor’s Thesis. Freie Universität Berlin, 2012 (cited on page 176).

[32] Gabriella Budai, Gábor Maróti, Rommert Dekker, Dennis Huisman, and Leo Kroon. “Rescheduling in passenger railways: the rolling stock rebalancing problem”. English. In: *Journal of Scheduling* 13.3 (2010), pp. 281–297. ISSN: 1094-6136. DOI: 10.1007/s10951-009-0133-9 (cited on page 157).

[33] Rainer Burkard, Mauro Dell’Amico, and Silvano Martello. *Assignment Problems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009. ISBN: 9780898716634 (cited on pages 71, 156).

[34] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. “A Fast Heuristic Algorithm for the Train Unit Assignment Problem”. In: *12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS2012*. Ed. by Daniel Delling and Leo Liberti. Vol. 25. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 1–9. ISBN: 978-3-939897-45-3. DOI: 10.4230/OASIcs.ATMOS.2012.1 (cited on page 16).

[35] R. Cambini, G. Gallo, and M.G. Scutellà. “Flows on Hypergraphs”. In: *Mathematical Programming, Series B* 78.2 (1997), pp. 195–217. DOI: 10.1007/BF02614371 (cited on page 19).

[36] Alberto Caprara and Matteo Fischetti. “0, 1/2-Chvátal-Gomory cuts”. English. In: *Mathematical Programming* 74.3 (1996), pp. 221–235. ISSN: 0025-5610. DOI: 10.1007/BF02592196 (cited on page 197).

[37] G. Clarke and J. W. Wright. “Scheduling of Vehicles from a Central Depot to a Number of Delivery Points”. In: *Operations Research* 12.4 (1964), pp. 568–581. DOI: 10.1287/opre.12.4.568 (cited on pages 70, 73, 98, 100).

[38] Jean-François Cordeau, François Soumis, and Jacques Desrosiers. “Simultaneous Assignment of Locomotives and Cars to Passenger Trains”. In: *Operations Research* 49 (4 July 2001), pp. 531–548. ISSN: 0030-364X. DOI: 10.1287/opre.49.4.531.11226 (cited on page 16).

[39] Emilie Danna, Edward Rothberg, and Claude Le Pape. “Exploring relaxation induced neighborhoods to improve MIP solutions”. English. In: *Mathematical Programming* 102.1 (2005), pp. 71–90. ISSN: 0025-5610. DOI: 10.1007/s10107-004-0518-7 (cited on pages 70, 71).

[40] DB Fernverkehr AG. *German Intercity-Express (ICE) lines*. 2015. URL: http://www.bahn.de/p/view/buchung/karten/streckennetz.shtml (visited on 09/01/2015) (cited on page 177).

[41] Markus Dod. “Darstellungsoptimierung von Fahrzeugumläufen”. Bachelor’s Thesis. Hochschule Mittweida (FH), 2010. URN: urn:nbn:de:bsz:mit1-opus-11731 (cited on page 156).

[42] I. Dumitrescu. "Constrained Path and Cycle Problems". PhD thesis. University of Melbourne, 2002. uri: http://hdl.handle.net/11343/36999 (cited on page 70).

[43] M. Ehrgott. *Multicriteria Optimization*. Lecture notes in economics and mathematical systems. Springer, 2005. ISBN: 9783540213987 (cited on pages 234, 235).

[44] Issmail Elhallaoui, Daniel Villeneuve, François Soumis, and Guy Desaulniers. "Dynamic Aggregation of Set-Partitioning Constraints in Column Generation". In: *Operations Research* 53.4 (July 2005), pp. 632–645. ISSN: 0030-364X. DOI: 10.1287/opre.1050.0222 (cited on pages 39, 42).

[45] Ricardo Euler and Gerwin Gamrath. *Hydraw – Hy(pergraph)Draw(ing)*. Version 1.0. Feb. 26, 2016. URL: http://hydraw.zib.de (cited on page 14).

[46] Frankfurter Allgemeine Zeitung (FAZ). *Bauarbeiten: ICE-Strecke von Frankfurt nach Köln zeitweise gesperrt*. German. URL: http://www.faz.net/-gzg-7zfb7 (visited on 02/04/2015) (cited on page 237).

[47] Pieter-Jan Fioole, Leo Kroon, Gábor Maróti, and Alexander Schrijver. "A rolling stock circulation model for combining and splitting of passenger trains". In: *European Journal of Operational Research* 174.2 (2006), pp. 1281–1297. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2005.03.032 (cited on pages 16, 140).

[48] Matteo Fischetti and Andrea Lodi. "Local branching". English. In: *Mathematical Programming* 98.1-3 (2003), pp. 23–47. ISSN: 0025-5610. DOI: 10.1007/s10107-003-0395-5 (cited on pages 70, 94).

[49] Matteo Fischetti, Paolo Toth, and Daniele Vigo. "A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs". In: *Operations Research* 42.5 (1994), pp. 846–859. DOI: 10.1287/opre.42.5.846 (cited on pages 31, 92, 104).

[50] Steven Fortune, John Hopcroft, and James Wyllie. "The directed subgraph homeomorphism problem". In: *Theoretical Computer Science* 10.2 (1980), pp. 111–121. ISSN: 0304-3975. DOI: 10.1016/0304-3975(80)90009-2 (cited on page 96).

[51] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. "Directed Hypergraphs and Applications". In: *Discrete Applied Mathematics* 42.2-3 (Apr. 1993), pp. 177–201. ISSN: 0166-218X. DOI: 10.1016/0166-218X(93)90045-P (cited on page 19).

[52] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979. ISBN: 0716710447 (cited on page 26).

[53] Giovanni Luca Giacco. "Rolling stock rostering and maintenance scheduling optimization". PhD thesis. Roma Tre University, 2014. URL: http://hdl.handle.net/2307/4429 (cited on page 16).

[54] Giovanni Luca Giacco, Donato Carillo, Andrea D'Ariano, Dario Pacciarelli, and Angel Marín. "Short-term Rail Rolling Stock Rostering and Maintenance Scheduling". In: *Transportation Research Procedia* 3 (2014). 17th EURO Working Group on Transportation, EWGT2014, Sevilla, Spain, pp. 651–659. ISSN: 2352-1465. DOI: 10.1016/j.trpro.2014.10.044 (cited on page 16).

[55] Giovanni Luca Giacco, Andrea D'Ariano, and Dario Pacciarelli. "Rolling stock rostering optimization under maintenance constraints". In: *Proceedings of the 2nd International Conference on Models and Technology for Intelligent Transportation Systems*. Leuven, Belgium: MT-ITS 2011, 2011, pp. 1–5. URL: http://www.mech.kuleuven.be/MT-ITS2011/download/proceedings.html (cited on page 16).

[56] Chris Groër, Bruce Golden, and Edward Wasil. "A library of local search heuristics for the vehicle routing problem". English. In: *Mathematical Programming Computation* 2.2 (2010), pp. 79–101. ISSN: 1867-2949. DOI: 10.1007/s12532-010-0013-5 (cited on page 106).

[57] Gurobi Optimization, Inc. *Gurobi*. Version 6.4. Oct. 1, 2015. URL: http://www.gurobi.com (cited on pages 59, 210).

[58] Jørgen Haahr. "Reactive Robustness and Integrated Approaches for Railway Optimization Problems". PhD thesis. 2015 (cited on page 17).

[59] Jørgen Thorlund Haahr, Richard Martin Lusby, Jesper Larsen, and David Pisinger. *A Branch-and-Price Framework for Railway Rolling Stock Rescheduling During Disruptions*. Tech. rep. 2014 (cited on pages 16, 17, 157).

[60] Jørgen Haahr, Joris Wagenaar, Lucas Veelenturf, and Leo Kroon. *A Comparison of Two Exact Methods for Passenger Railway Rolling Stock (Re)Scheduling*. ERS-2015-007-LIS. June 2015. URL: http://hdl.handle.net/1765/78317 (cited on page 17).

[61] Steven Harrod and Thomas Schlechte. "A Direct Comparison of Physical Block Occupancy Versus Timed Block Occupancy in Train Timetabling Formulations". In: *Transportation Research Part E: Logistics and Transportation Review* 54 (2013), pp. 50–66. DOI: 10.1016/j.tre.2013.04.003 (cited on page 26).

[62] Olga Heismann. "The Hypergraph Assignment Problem". PhD thesis. Technische Universität Berlin, 2014. DOI: 10.14279/depositonce-4083 (cited on pages 18, 19, 29, 30, 197).

[63] Olga Heismann and Ralf Borndörfer. "Minimum Cost Hyperassignments with Applications to ICE/IC Rotation Planning". In: *Operations Research Proceedings 2011*. (URN links to ZIB report). 2012, pp. 59–64. DOI: 10.1007/978-3-642-29210-1_10. URN: urn:nbn:de:0296-matheon-10781 (cited on page 30).

[64] Keld Helsgaun. "General k-opt submoves for the Lin–Kernighan TSP heuristic". English. In: *Mathematical Programming Computation* 1.2-3 (2009), pp. 119–163. ISSN: 1867-2949. DOI: 10.1007/s12532-009-0004-6 (cited on pages 71, 86, 104).

[65] Dennis Huisman. "A column generation approach for the rail crew re-scheduling problem". In: *European Journal of Operational Research* 180.1 (2007), pp. 163–173. DOI: 10.1016/j.ejor.2006.04.026 (cited on page 157).

[66] IBM ILOG. *CPLEX*. Version 12.6.3. Apr. 1, 2016. URL: http://www.ibm.com/software/integration/optimization/cplex-optimizer (cited on pages 30, 60, 210).

[67] Stefan Irnich, Birger Funke, and Tore Grünert. "Sequential search and its application to vehicle-routing problems". In: *Computers & Operations Research* 33.8 (2006), pp. 2405–2429. DOI: 10.1016/j.cor.2005.02.020 (cited on page 86).

[68] Paris-C. Kanellakis and Christos H. Papadimitriou. "Local Search for the Asymmetric Traveling Salesman Problem". In: *Operations Research* 28.5 (1980), pp. 1086–1099. DOI: 10.1287/opre.28.5.1086 (cited on page 70).

[69] Marika Karbstein. "Line Planning and Connectivity". PhD thesis. Technische Universität Berlin, 2013. ISBN: 978-3-8439-1062-0 (cited on page 25).

[70] H. W. Kuhn. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97. ISSN: 1931-9193. DOI: 10.1002/nav.3800020109 (cited on pages 29, 57, 69, 71, 74, 75, 95, 210).

[71] Christian Liebchen and Rolf Möhring. "The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables — and Beyond". English. In: *Algorithmic Methods for Railway Optimization*. Ed. by Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, and ChristosD. Zaroliagis. Vol. 4359. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 3–40. ISBN: 978-3-540-74245-6. DOI: 10.1007/978-3-540-74247-0_1 (cited on page 143).

[72] S. Lin and B. W. Kernighan. "An Effective Heuristic Algorithm for the Traveling-Salesman Problem". In: *Operations Research* 21.2 (1973), pp. 498–516. DOI: 10.1287/opre.21.2.498 (cited on pages 70, 71, 87).

[73] Shen Lin. "Computer Solutions of the Traveling Salesman Problem". In: *Bell System Technical Journal* 44.10 (1965), pp. 2245–2269. ISSN: 1538-7305. DOI: 10.1002/j.1538-7305.1965.tb04146.x (cited on pages 71, 88).

[74] Andreas Löbel. *Optimal Vehicle Scheduling in Public Transit*. English. Ph.D. thesis, Technische Universität Berlin. Aachen: Shaker Verlag, 1997 (cited on pages 16, 33, 34, 47, 120).

[75] M.E. Lübbecke and J. Desrosiers. "Selected Topics in Column Generation". In: *Operations Research* 53.6 (2005), pp. 1007–1023. DOI: 10.1287/opre.1050.0234 (cited on pages 37, 43).

[76] Gábor Maróti and Leo Kroon. "Maintenance Routing for Train Units: The Transition Model". In: *Transportation Science* 39 (4 Nov. 2005), pp. 518–525. ISSN: 1526-5447. DOI: 10.1287/trsc.1050.0116 (cited on page 16).

[77] Gábor Maróti and Leo G. Kroon. "Maintenance routing for train units: The interchange model". In: *Computers & OR* (2007), pp. 1121–1140. DOI: doi:10.1016/j.cor.2005.05.026 (cited on page 16).

[78] Silvano Martello and Paolo Toth. "Lower bounds and reduction procedures for the bin packing problem". In: *Discrete Applied Mathematics* 28.1 (1990), pp. 59–70. ISSN: 0166-218X. DOI: 10.1016/0166-218X(90)90094-S (cited on page 98).

[79] Julika Mehrgardt. "Kreiseinbettungen in Hypergraphen und ihre Anwendung in der Umlaufoptimierung". German. Masters's Thesis. Technische Universität Berlin, Feb. 2013 (cited on pages 40, 41).

[80] Taïeb Mellouli and Leena Suhl. "Rotation Planning of Locomotive and Carriage Groups with Shared Capacities". In: *Algorithmic Methods for Railway Optimization*. Ed. by Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, and ChristosD. Zaroliagis. Vol. 4359. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 276–294. ISBN: 978-3-540-74245-6. DOI: 10.1007/978-3-540-74247-0_15 (cited on page 16).

[81] C. E. Miller, A. W. Tucker, and R. A. Zemlin. "Integer Programming Formulation of Traveling Salesman Problems". In: *Journal of the Association for Computing Machinery* 7.4 (Oct. 1960), pp. 326–329. ISSN: 0004-5411. DOI: 10.1145/321043.321046 (cited on page 69).

[82] Karl Nachtigall and Jens Opitz. "Solving Periodic Timetable Optimisation Problems by Modulo Simplex Calculations". In: *ATMOS'08*. Vol. 9. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2008. ISBN: 978-3-939897-07-1. DOI: 10.4230/OASIcs.ATMOS.2008.1588 (cited on page 70).

[83] Lars Kjaer Nielsen. "Rolling Stock Rescheduling in Passenger Railways: Applications in short-term planning and in disruption management". PhD thesis. E, Feb. 2011. URL: http://hdl.handle.net/1765/22444 (cited on page 17).

[84] OpenMP Architecture Review Board. *OpenMP Application Program Interface*. Version 4.5. Feb. 26, 2016. URL: http://openmp.org (cited on page 210).

[85] Diego Pecin, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. "Improved Branch-Cut-and-Price for Capacitated Vehicle Routing". English. In: *Integer Programming and Combinatorial Optimization*. Ed. by Jon Lee and Jens Vygen. Vol. 8494. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 393–403. ISBN: 978-3-319-07556-3. DOI: 10.1007/978-3-319-07557-0_33 (cited on pages 91, 92).

[86] Luigi Di Puglia Pugliese and Francesca Guerriero. "A survey of resource constrained shortest path problems: Exact solution approaches". In: *Networks* 62.3 (2013), pp. 183–200. ISSN: 1097-0037. DOI: 10.1002/net.21511 (cited on page 70).

[87] T. Ralphs. *Branch Cut and Price Resource Web*. June 2014. URL: http://www.branchandcut.org/ (visited on 08/02/2014) (cited on pages 101, 102).

[88] C. Raphael. "Coarse-to-Fine Dynamic Programming". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.12 (2001), pp. 1379–1390. ISSN: 0162-8828. DOI: 10.1109/34.977562 (cited on pages 39, 42).

[89] G. Reinelt. *TSPLIB - A T.S.P. Library*. Tech. rep. 250. Augsburg, 1990, p. 18 (cited on pages 101, 102, 104).

[90] Jacques Renaud, Gilbert Laporte, and Fayez F. Boctor. "A tabu search heuristic for the multi-depot vehicle routing problem". In: *Computers & Operations Research* 23.3 (1996), pp. 229–235. ISSN: 0305-0548. DOI: 10.1016/0305-0548(95)00026-P (cited on page 34).

[91] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-key Cryptosystems". In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342 (cited on page 6).

[92] David F. Rogers, Robert D. Plante, Richard T. Wong, and James R. Evans. "Aggregation and Disaggregation Techniques and Methodology in Optimization". In: *Operations Research* 39.4 (1991), pp. 553–582. DOI: 10.1287/opre.39.4.553 (cited on pages 39, 40).

[93] Edward Rothberg. "An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions". In: *INFORMS Journal on Computing* 19.4 (2007), pp. 534–541. DOI: 10.1287/ijoc.1060.0189 (cited on page 70).

[94] Thomas Schlechte. "Railway Track Allocation - Models and Algorithms". PhD thesis. Technische Universität Berlin, 2012. DOI: 10.14279/depositonce-3124 (cited on pages 14, 26, 114, 123, 179).

[95] Thomas Schlechte, Ralf Borndörfer, Berkan Erol, Thomas Graffagnino, and Elmar Swarat. "Micro-Macro Transformation of Railway Networks". English. In: *Journal of Rail Transport Planning & Management* 1.1 (2011). (URN links to ZIB report), pp. 38–48. DOI: 10.1016/j.jrtpm.2011.09.001. URN: urn:nbn:de:0297-zib-11881 (cited on pages 39, 40).

[96] Nicola Secomandi and François Margot. "Reoptimization Approaches for the Vehicle-Routing Problem with Stochastic Demands". In: *Operations Research* 57.1 (2009), pp. 214–230. DOI: 10.1287/opre.1080.0520 (cited on page 157).

[97] Yossi Shiloach. *The Two Paths Problem is Polynomial*. Tech. rep. Stanford, CA, USA, 1978 (cited on page 96).

[98] Bjarne Stroustrup. *The C++ Programming Language*. 4th. Addison-Wesley Professional, 2013. ISBN: 9780321563842 (cited on pages 5, 100).

[99] J. Tang, S. MacLachlan, R. Nabben, and C. Vuik. "A Comparison of Two-Level Preconditioners Based on Multigrid and Deflation". In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (2010), pp. 1715–1739. DOI: 10.1137/08072084X (cited on page 38).

[100] Per Thorlacius, Jesper Larsen, and Marco Laumanns. "An integrated rolling stock planning model for the Copenhagen suburban passenger railway". In: *Journal of Rail Transport Planning & Management* 5.4 (2015), pp. 240–262. ISSN: 2210-9706. DOI: 10.1016/j.jrtpm.2015.11.001 (cited on page 17).

[101] P. Toth and D. Vigo. *Vehicle Routing*. Ed. by Daniele Vigo and Paolo Toth. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014. DOI: 10.1137/1.9781611973594 (cited on pages 17, 69, 71, 91, 92, 104).

[102] Marcel Turkensteen, Diptesh Ghosh, Boris Goldengorin, and Gerard Sierksma. "Tolerance-based Branch and Bound algorithms for the ATSP". In: *EJOR* 189.3 (2008), pp. 775–788. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2006.10.062 (cited on page 93).

[103] Anke Uffmann. "Das Kanalmodell zur Effizienzsteigerung in der Fahrzeugumlaufplanung". German. Diplomarbeit. Universität Göttingen, 2010 (cited on page 120).

[104] Robindro Ullah. "Rolling Stock Rostering – Mathematische Modellierung und Lösungsansätze mittels Heuristiken am Beispiel der Deutschen Bahn AG". German. (URL visited on 2016-02-19). Diplomarbeit. Universität Hamburg, 2005. URL: http://www.math.uni-hamburg.de/projekte/optag/downloads/ullah.pdf (cited on page 119).

[105] Thibaut Vidal. "Approches générales de résolution pour les problèmes multi-attributs de tournées de véhicules et confection d'horaires". PhD thesis. Université de Montréal & Université de Technologie de Troyes, 2013. URL: http://hdl.handle.net/1866/9699 (cited on pages 35, 104).

[106] Joris Wagenaar, Leo Kroon, and Marie Schmidt. *Maintenance Appointments in Railway Rolling Stock Rescheduling*. ERS-2016-001-LIS. Jan. 2016. URL: http://hdl.handle.net/1765/79441 (cited on page 17).

[107] Steffen Weider. "Integration of Vehicle and Duty Scheduling in Public Transport". PhD thesis. Technische Universität Berlin, 2007. DOI: 10.14279/depositonce-1672 (cited on page 189).

[108] Abdelouahab Zaghrouti, François Soumis, and Issmail El Hallaoui. "Integral Simplex Using Decomposition for the Set Partitioning Problem". In: *Operations Research* 62.2 (2014), pp. 435–449. DOI: 10.1287/opre.2013.1247 (cited on page 70).

[109] Koorush Ziarati, François Soumis, Jacques Desrosiers, Sylvie Gélinas, and André Saintonge. "Locomotive assignment with heterogeneous consists at CN North America." English. In: *European Journal of Operations Research* 97.2 (1997), pp. 281–292. DOI: 10.1016/S0377-2217(96)00198-1 (cited on page 16).