Models and Algorithms for Railway Line Planning Problems

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Maastricht, op gezag van de Rector Magnificus, Prof. mr. G.P.M.F. Mols volgens het besluit van het College van Decanen, in het openbaar te verdedigen op woensdag 24 november 2004 om 16.00 uur

door

Jan-Willem Henricus Mathijs Goossens
Promotoren:
Prof. dr. ir. C.P.M. van Hoesel
Prof. dr. L.G. Kroon (Erasmus Universiteit Rotterdam)

Beoordelingscommissie:
Prof. dr. ir. A.W.J. Kolen (voorzitter)
Dr. J.J. van de Klundert
Prof. dr. A.P.M. Wagelmans (Erasmus Universiteit Rotterdam)
Prof. dr. D. Wagner (Universität Karlsruhe)

Models and Algorithms for Railway Line Planning Problems
©2004, Jan-Willem H.M. Goossens
Proefschrift Universiteit Maastricht
ISBN 90-9018605-0

Het in dit proefschrift beschreven onderzoek werd mede mogelijk gemaakt door NS Reizigers B.V. en het onderzoeksnetwerk AMORE van de Europese Unie.
Acknowledgements

On the first of November 1998, now almost six years ago, I started my research as a PhD candidate at the University of Maastricht. Within these very interesting six years I presented at a range of international conferences, taught courses for students, obtained my LNMB diploma, moved to Konstanz in Germany—and back again—and, of course, spent a lot of time doing research, and writing this thesis. My position was made possible by a research agreement between the Dutch passenger railway operator NS Reizigers and the University of Maastricht. This agreement was the brainchild of my supervisors Stan van Hoesel and Leo Kroon. In the first place, I would like to express my gratitude to them for making all of this possible.

Through Leo’s involvement in the EU project for Algorithmic Methods for Optimizing the Railways in Europe, I came into contact with a number of other participating groups, and visited several of their workshops. On one of these meetings, after a nice evening with pizzas, good wines and a discussion with Dorothea Wagner of the Universität Konstanz, Leon Peeters and I reached the conclusion that a long visit to the group in Konstanz would be a good opportunity to do research, work on our theses, learn to speak German, and, on the side, go snowboarding on winter weekends. On April 1st 2002 this plan came into realisation when, with an eleven months leave from the University of Maastricht, I moved to the beautifully small city of Konstanz. I would like to thank Dorothea for giving me this opportunity. In addition, I wish to thank all the members of the Algorithms and Datastructures Group for their warm welcome and the interesting time I spent there.

Furthermore, I am grateful to all of those who took the time to proofread different parts of my thesis over and over again. For this I would particularly like to thank Leo Kroon, Stan van Hoesel, Leon Peeters and Anton van der Kraaij.

I am also grateful to all the members of the Department of Quantitative Economics. With the numerous questions that I’ve had over the last six years, the mix of colleagues within the department proved to be very valuable.

Finally, I would like to express a lot of thanks to my family and friends, in particular to my parents and to Nathalie for their unconditional and generous support.

Maastricht, September 2004
Jan-Willem Goossens
## Contents

Acknowledgements .......................... i

Contents .................................. iii

List of Tables .......................... vii

List of Figures .......................... ix

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

1.1 Past to present situation in the Netherlands .......................... 1

1.2 Future developments ............................................ 2

1.2.1 Increasing reliability ............................................ 3

1.2.2 Increasing capacity ............................................ 3

1.3 Planning problems for passenger railways .......................... 4

1.3.1 Estimating passenger demand .................................... 5

1.3.2 Line planning ............................................ 5

1.3.3 Timetabling ............................................ 5

1.3.4 Platform and track assignment .................................... 6

1.3.5 Rolling stock planning ............................................ 7

1.3.6 Crew scheduling ............................................ 9

1.3.7 Shunting and maintenance planning .................................... 10

1.4 The overall planning process at NSR .................................... 11

1.4.1 Current planning stages ............................................ 11

1.4.2 Proposed planning process ............................................ 12

1.5 An introduction to line planning ............................................ 14

1.6 Research objective ............................................ 15

1.7 Outline of this thesis ............................................ 17

I Foundations ............................ 19

2 Combinatorial foundations .......................... 21

2.1 Polyhedral combinatorial optimisation ............................................ 21

2.1.1 Combinatorial optimisation problems ............................................ 21
## CONTENTS

2.1.2 Polyhedral theory .................................................. 22
2.1.3 Integer linear programming problems .......................... 23
2.1.4 Solution approaches ............................................. 24
2.2 Multi-commodity flows ............................................. 29
2.3 Theoretical overview .............................................. 31

3 Foundations in line planning modelling ......................... 33
  3.1 Line plans and cyclic timetables .............................. 33
  3.2 Rolling stock ..................................................... 34
  3.3 Passenger demand .............................................. 35
    3.3.1 Passenger demand and capacity planning ............... 35
    3.3.2 Passenger demand across the network .................. 39
    3.3.3 Passenger demand variations over time ................. 39
  3.4 Infrastructure .................................................. 40
    3.4.1 Station capacity .......................................... 40
    3.4.2 Homogenising the train system .......................... 41
    3.4.3 New railway safety systems ............................. 42
  3.5 Modelling the railway system .................................. 44
    3.5.1 Infrastructure ............................................. 44
    3.5.2 Lines ...................................................... 44
    3.5.3 Rolling stock .............................................. 46

II Applications .......................................................... 49

4 Branch-and-cut for line planning problems .................... 51
  4.1 Introduction ..................................................... 51
  4.2 Model formulation ............................................... 52
    4.2.1 Formulation ............................................ 53
  4.3 Branch-and-cut method for CLP ............................... 54
    4.3.1 Preprocessing ............................................ 55
    4.3.2 Cutting planes .......................................... 60
    4.3.3 Tree search ............................................ 65
  4.4 Implementation issues .......................................... 67
    4.4.1 Preprocessing implementation .......................... 67
    4.4.2 Cutting planes implementation ........................ 68
    4.4.3 Branching rules implementation ......................... 70
  4.5 Computational results .......................................... 71
    4.5.1 Unbounded computation times ........................... 76
  4.6 Summary and conclusions ....................................... 76

5 The multiple-type line planning problem ....................... 79
  5.1 Modelling ....................................................... 79
    5.1.1 Definitions and notation .............................. 80
  5.2 Formulating the multi-type line planning problem ........ 82
  5.3 Formulations for the edge capacity problem ............... 83
| 5.3.1 The multi-commodity flow formulation | 83 |
| 5.3.2 The mixed integer programming formulation | 88 |
| 5.3.3 The integer programming formulation | 92 |
| 5.4 Computational results | 97 |
| 5.5 Summary and conclusions | 100 |
| **6 The station type optimisation problem** | 103 |
| 6.1 Introduction | 103 |
| 6.2 Modelling | 104 |
| 6.2.1 The line-event graph | 104 |
| 6.2.2 Valid paths in the line-event graph | 108 |
| 6.3 Problem formulation | 109 |
| 6.3.1 Extending the STOP(f) formulation | 111 |
| 6.4 Lagrangian relaxation | 112 |
| 6.5 The branch-and-bound algorithm | 114 |
| 6.5.1 Preprocessing | 114 |
| 6.5.2 Bounding methods | 119 |
| 6.5.3 Tree search | 122 |
| 6.6 Computational results | 123 |
| 6.6.1 Comparing line plans | 130 |
| 6.7 Summary and conclusions | 130 |
| **A Appendix for Chapter 1** | 133 |
| **B Appendix for Chapter 4** | 135 |
| **C Appendix for Chapter 5** | 137 |
| **D Appendix for Chapter 6** | 139 |
| Bibliography | 147 |
| Index | 155 |
| Samenvatting (Summary in Dutch) | 159 |
| Curriculum Vitae | 163 |
List of Tables

4.1 Characteristics of the instances. ........................................... 71
4.2 Preprocessing results for variable and constraint reduction. ............... 72
4.3 Statistics for the individual classes of cutting planes. ........................ 72
4.4 Statistics for the branching rules. ......................................... 72
4.5 Computational results. ..................................................... 75
4.6 Cut-and-branch without time restrictions. ..................................... 76

5.1 Variable and constraint statistics per formulation. ............................. 97
5.2 Characteristics of the instances. ........................................... 98
5.3 Statistics for the different instances and formulations. ......................... 98
5.4 Computational results. ..................................................... 100
5.5 Comparing CLP and MCLP solutions. ....................................... 100

6.1 Characteristics of the instances. ........................................... 123
6.2 Initial statistics before preprocessing. ....................................... 125
6.3 Statistics after preprocessing. ............................................. 125
6.4 Statistics for the root node of the branch-and-bound tree. .................... 126
6.5 Branch-and-bound statistics for the branching rules. ......................... 128
6.6 Results of applying the MAA and RVA techniques at every node. ............. 129
6.7 Comparing the current and new total travel times. ........................... 129
6.8 Statistics for the MCLP line plans. ....................................... 130

D.1 Statistics for the original STOP formulation. ............................... 142
D.2 Solution statistics. ....................................................... 143
List of Figures

1.1 Planning problems .............................................. 4
1.2 Time-space diagram for the tracks between Rotterdam and Utrecht .................. 6
1.3 An example of a platform assignment for the station Utrecht ....................... 7
1.4 An example of a rolling stock assignment .................................. 8
1.5 A Gannt chart with duties ...................................... 9
1.6 An example of a station layout of the station Zwolle ............................... 10
1.7 The current planning process .................................... 11
1.8 The new planning process concept .................................. 13
1.9 Line plan of the underground system in London .................................. 14
1.10 How to get from Gloucester Road to Westbourne Park .......................... 15

2.1 Two types of hulls for the two points \( x_1 = (1,2) \) and \( x_2 = (3,1) \) from \( \mathbb{R}^2 \) ................................. 22
2.2 The feasible points, their convex hull, and the LP polytope ........................ 25

3.1 Different types of rolling stock .................................... 36
3.2 An example origin-destination matrix .................................. 38
3.3 The percentage of all travellers grouped by travel distance ........................ 40
3.4 The number of travellers at different points in time ................................ 41
3.5 The current situation and four proposed modifications ............................ 43
3.6 The current situation and the moving block safety system ......................... 44
3.7 The Dutch and German railway networks ................................... 45
3.8 Trains versus frequency ............................................ 46

4.1 The Dutch network, and the network graph for the IC instance ..................... 53
4.2 Example of a CLP instance ........................................ 55
4.3 Example of constraint dominance ..................................... 61
4.4 Example of a flow cover ............................................ 64
4.5 Example instance of 2-Cover cuts ...................................... 69
4.6 The network graphs for the instances SP97AR and SP97IC ......................... 73

5.1 The network graph \( G \) and three train lines ..................................... 81
5.2 The type graph, based on the network graph from Figure 5.1 ....................... 81
5.3 The directed graph \( D^T \) based on the network in Figure 5.2 ....................... 85
5.4 Travellers are possibly assigned to underlying type edges ........................ 89
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5</td>
<td>The network graph (5.5(a)) and the type graph (5.5(b)).</td>
<td>92</td>
</tr>
<tr>
<td>5.6</td>
<td>The type graphs for the instances.</td>
<td>99</td>
</tr>
<tr>
<td>6.1</td>
<td>The network graph and two lines.</td>
<td>106</td>
</tr>
<tr>
<td>6.2</td>
<td>The line-event graph.</td>
<td>106</td>
</tr>
<tr>
<td>6.3</td>
<td>The line-event graph, given that station v is of type 1.</td>
<td>106</td>
</tr>
<tr>
<td>6.4</td>
<td>The highlighted path is not valid.</td>
<td>108</td>
</tr>
<tr>
<td>6.5</td>
<td>Setting station b to type 1 causes line l2 not to halt at b.</td>
<td>116</td>
</tr>
<tr>
<td>6.6</td>
<td>The shortest path from a to d can contain two non-halt arcs at b.</td>
<td>117</td>
</tr>
<tr>
<td>6.7</td>
<td>The network graphs for the instances.</td>
<td>124</td>
</tr>
<tr>
<td>6.8</td>
<td>The improvement of the lower bound $z_{LR}(\lambda)$ over time.</td>
<td>127</td>
</tr>
<tr>
<td>A.1</td>
<td>The station layout of Utrecht.</td>
<td>133</td>
</tr>
<tr>
<td>B.1</td>
<td>The graphs for the instances SP98AR, SP98IR and SP98IC.</td>
<td>135</td>
</tr>
<tr>
<td>B.2</td>
<td>Nodes in the enumeration tree of SP98IC.</td>
<td>136</td>
</tr>
<tr>
<td>C.1</td>
<td>The instances placed in a map of the Netherlands.</td>
<td>137</td>
</tr>
<tr>
<td>C.2</td>
<td>Example instance of a type graph.</td>
<td>138</td>
</tr>
<tr>
<td>C.3</td>
<td>The layered, directed in-tree $T(e)$ for the edge e of type 1.</td>
<td>138</td>
</tr>
<tr>
<td>D.1</td>
<td>The line-event graph of NS3600.</td>
<td>142</td>
</tr>
<tr>
<td>D.2</td>
<td>Performance at the root node of combinations of RVA and MAA.</td>
<td>142</td>
</tr>
<tr>
<td>D.3</td>
<td>Example of an enumeration tree.</td>
<td>143</td>
</tr>
<tr>
<td>D.4</td>
<td>Visualisation of the passenger flows around station Dvd.</td>
<td>144</td>
</tr>
<tr>
<td>D.5</td>
<td>Nodes in the enumeration tree of NS3600.</td>
<td>144</td>
</tr>
<tr>
<td>D.6</td>
<td>Nodes in the enumeration tree of NSNH.</td>
<td>145</td>
</tr>
<tr>
<td>D.7</td>
<td>Nodes in the enumeration tree of NSRandstad.</td>
<td>145</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The growing demand for mobility, together with the change from state owned to privately owned railways has led to an increase in research on railway planning problems. This thesis studies mathematical models and solution methods for railway line planning problems. The line planning problem of a railway operator is to decide between which pairs of stations to operate train lines, and where the lines should halt along their routes.

This chapter provides some background information on past, current and future developments in public rail transportation in the Netherlands.

1.1 Past to present situation in the Netherlands

In the early 1990s, the first steps were made by the European Community to introduce the regulated opening-up of the European rail transport markets. With the European Council's directive 91/440/EEC in 1991, the foundation was laid for separating the provision of railway transport services from infrastructure management. This opened the way for future competition between railway operators. Now, more than a decade later, the European railways still face big challenges before the liberalisation is fully completed (see EC [22]).

With the first steps taken in 1992, the privatisation of the main Dutch railway operator Nederlandse Spoorwegen¹ (NS) was well under way by 1995, when the infrastructure management division was split from the core of NS. As of January 2003, all infrastructure in the Netherlands is managed by ProRail, a governmental organisation that is responsible for issues ranging from the building and maintaining of new infrastructure and stations, to the assignment of rail infrastructure capacity to the operators, and traffic control (see ProRail [71]). The competition on the railway tracks in the Netherlands has led to around fifteen railway operators that are currently conducting services on the Dutch network. Of these fifteen operators, five are authorised for passenger transport (see EC [23]). The passenger railway operator

¹In English Netherlands Railways.
NS Reizigers\(^2\) (NSR) is the main operator on the Dutch network.

Over the last two decades, public transportation in the Netherlands has taken up around 10-13\% of the total number of passenger kilometers travelled per year. Around two thirds of this is done by train. The market share of public transportation is positively correlated with the travelled distance. In V&W [83], it is reported that public transportation takes only 1\% of the share of distances below 5 kilometers, yet this share grows to 20\% for 50 kilometers or more. The average distance travelled by Dutch railway passengers is around 35–40 kilometers per trip. For more information on passenger demand for railway transportation, see §3.3.

One of the largest criticisms of the railways in the Netherlands in recent years has been the reliability of the train services. The lowest punctuality was reached in 2001, with an average of 79.9\% of trains arriving on time. Partly, the disruptions in the train services were due to failures in the infrastructure. Serious cutbacks in the governmental investments in the 1970s and 1980s led to a build up of delayed maintenance. It took until the early 1990s for the investments to pick up, but by then the problems had already started (see NRC [58]). The problems with the infrastructure led to many disruptions, caused by problems with power lines, railway junctions, signposts, etc.

The infrastructure was not the only source of problems. A postponement of investments at NSH decreased their expenses, but it also led to shortages in rolling stock. Maintenance was postponed in order to make more train units available to meet the increased demand. Nevertheless, the train capacities were often exceeded. Inevitably, this in turn led to more breakdowns of trains, and thus to more delays.

Delayed trains may also cause indirect delays for other trains. Not only can a delayed train cause its passengers to miss their connections, also the crew and the rolling stock may be too late for their next trip. Therefore, tight crew schedules—due to a general shortage in personnel—can propagate delays indirectly. Combined, these indirect, or secondary delays make up around 80\% of all delays, as mentioned by Project B&B [70]. Increasing the reliability of the train services is one of the issues discussed in the next section.

1.2 Future developments

The outlook for the Dutch public transport sector is good. Public transport, and railway transport in particular is expected to grow significantly over the next two decades. The main growth markets are those where the travel times for public transport can compete with those of car travel, and where the facilities for getting to and from railway stations are good, e.g., by car parks or good connections with public transport to and from the stations (see V&W [82]). As a result, a high growth for public transportation is expected for both the short range trips of less than 10 kilometers within cities, and for medium range distances between city centers. Forecasts predict that for future travels this could lead to a growth of almost 100\% of railway travel among, and to and from the city centers of the four agglomerations Amster-

\(^2\)In English *Netherlands Railways Travellers.*
1.2. FUTURE DEVELOPMENTS

dam, The Hague, Rotterdam, and Utrecht by 2020. The market share of public transportation within the cities is expected to grow by around 20%. High reliability is a necessary condition for realising further growth of the capacity of the railway system to accommodate the increasing demand. However, as mentioned previously, over the last decade the reliability of the infrastructure and rolling stock has proven to be a problem.

1.2.1 Increasing reliability

Secondary delays form the majority of the delays in railway transport in the Netherlands. One of the proposed solutions to prevent secondary delays are process simplifications.

One element of the process simplifications concerns the operated train services. In the long run, to reduce the interdependencies among train lines, the number of scheduled transfer connections is reduced, and compensated for by more frequent train lines. In addition, the number of direct connections is reduced, and focussed more on connections with large passenger flows.

Another element of the simplifications is the binding of both the crew and the rolling stock to specific train lines. To realise this, several recruitment campaigns and investments were made to assure the additional necessary personnel and rolling stock. The introduction in 2001 of an initial version of this system for the operating crew—the infamous so-called Rondje om de Kerk\(^3\)—caused much protest from the crew unions.

The process simplifications are aimed at creating conflict-free and dependency-free corridors within the railway network, thereby limiting the impact of disturbances (see Project B&B [70]).

The punctuality data made available by railway operators in other countries is limited. Nevertheless, as mentioned in NYFER [61], the recent 2001 low of 79.9% of trains on time is not too bad when compared internationally. The definition of "on time" varies per operator. In the Netherlands a train is considered on time if it is delayed by no more than 3 minutes. Many other operators, however, apply a less strict 5 minute rule. This latter definition would imply that over the last years more than 90% of the trains were on time, putting the Netherlands at the top of the European railways, together with Switzerland.

1.2.2 Increasing capacity

Recent studies by V&W [84] and Project B&B [70] argue that the increasing demand for mobility in general can be met by more efficiently utilising the current infrastructure, and, if necessary, by investing in new infrastructure at bottlenecks. Building new infrastructure on a large scale is, however, not considered to be a plausible solution. Not only would this take too much time; it would also require high investments. Therefore, for all transport methods (see V&W [84]), and for the railway system in particular (see NS [59], Project B&B [70], Railforum [72]), it is necessary to achieve a

\(^3\)In English Trip around the church.
to achieve this it is necessary to homogenise the operated train services per track. By not operating train services with very different halting patterns, such as intercity trains and local trains, but instead deploying train lines with more similar patterns, the throughput capacity of the tracks can be increased. The introduction of a new safety system for trains based on high-tech ICT is expected to further increase the number of trains that can be operated per hour on a track. For more details, see §3.4.2 and §3.4.3.

Other studies, such as NYFER [61], argue that further increasing the train kilometers per network kilometer will not be cost-efficient. This alternative conclusion is founded on the observation that, with a yearly average of 50,000 passing trains per network kilometer, the Dutch railway network is the most intensively operated network in the world. This is almost 10% more than for Switzerland and Japan, the numbers two and three on this list. A further increase in this utilisation rate does not necessarily lead to a decrease in the total cost per train kilometer. The current utilisation rate is already 10% above the cost optimum. When using the currently available types of rolling stock, a further increase of the utilisation will not only lead to higher net cost, but may also jeopardise the safety on the tracks. Therefore, according to NYFER [61], meeting the higher passenger demand should not be achieved by increasing the utilisation of the infrastructure, but by operating the available rolling stock more efficiently.

### 1.3 Planning problems for passenger railways

A large range of planning problems needs to be solved before railway travellers can come to a railway station and take a train to their destination. An overview of these problems is given in Figure 1.1.

Due to the complexity of the various problems, we consider these problems to be solved sequentially, in coherence with many other authors, such as Bussieck et al. [14], Lindner [48], Zwaneveld [88], and Peeters [66]. In practice, there are many feedback loops between the different planning stages. If, for example, the chosen line plan does not allow for a feasible timetable, then the line plan is adjusted. After reviewing these problems, we consider the overall planning process. The next sections describe the various planning problems and briefly discuss some recent work.
1.3. Estimating passenger demand

At the basis of customer oriented planning problems lies the problem of estimating the passenger demand. This demand data is usually given for pairs of origins and destinations in a so-called origin-destination matrix. In this matrix, every row refers to an origin station, and every column to a destination station. More on the problem of describing the passenger demand can be found in §3.3.

Another problem is how to estimate future demand. Studies of the development of transportation demand are published in many countries. For example, in Germany this is done by the Bundesministerium für Verkehr, Bau- und Wohnungswesen which publishes the Bundesverkehrswegeplan (see BMVBW [10]), or in the Netherlands by the Ministerie van Verkeer en Waterstaat4 (V&W) publishing the Nationaal Verkeers- en Vervoersplan (see V&W [84]). Both use sophisticated econometric models that estimate the developments of passenger mobility in the near future, e.g., for 2010 or 2020.

1.3.2 Line planning

The line planning problem is the main topic of this thesis. The problem will be introduced in more detail in §1.5 and Chapter 3. In short, it involves the selection of paths in the railway network on which train lines are operated. Besides the paths, a line plan also specifies the stops and hourly frequencies of the chosen lines. The problem of choosing such a set of lines is called the line planning problem.

1.3.3 Timetabling

After the line plan has been built, a schedule for arrival and departure times for all trains at every station—the timetable—can be constructed. Timetables can be divided into cyclic (or periodic) timetables and noncyclic timetables. A cyclic timetable describes departure and arrival events of trains that are repeated every cycle time, which is often one hour. One of the advantages of a cyclic timetable is that it is transparent to the traveller, and easy to memorise. For a literature review on cyclic timetables see Peeters [66], Odijk [62], Schrijver and Steenbeek [78] and Nachtigall [54]. A noncyclic timetable, on the other hand, can be adjusted more to the variations in demand during the day. For a survey on noncyclic timetables, see Cordeau et al. [19] and Caprara et al. [15].

The constructed timetable must meet many requirements. Most importantly, the safety regulations on the tracks and stations must be respected. Also, the driving and dwelling behaviour of trains, connections between trains, and, to a lesser extent, circulation of rolling stock and operating crews are taken into account. Objectives typically consider the travel time or waiting time, delay sensitivity, and cost efficiency, i.e., the required rolling stock or workforce to operate the timetable.

Although the developed mathematical models and techniques are in principle generally applicable for various planning horizons, they are mostly used for medium

---

4The Dutch Ministry of Traffic
CHAPTER 1. INTRODUCTION

and long term planning purposes. In practice, most of the planning of operational timetables is still done manually.

Figure 1.2 shows an example of one cycle time of a cyclic timetable in the form of a time-space diagram. The horizontal axis shows the stations that are passed on the route from Rotterdam on the left, to Utrecht on the right. The vertical axis shows the time, in this case for one cycle time of sixty minutes. The lines in the diagram show the various locations of trains on this track at a given point in time; the numbers correspond to particular trains. For example, consider train 21700 represented by the dashed lines. Within the displayed time frame this train travels once from Utrecht to Rotterdam, and back. At eight minutes past the hour it leaves Rotterdam, as can be seen in the lower left part of the diagram. The train arrives in Utrecht at forty-two minutes past the hour.

1.3.4 Platform and track assignment

Not all the infrastructure details are taken into account in the construction of the timetable. The exact routing of trains at junctions for example, or the routes through stations are not considered. In particular, the routing of the in-station traffic can be a complex problem (see Zwaneveld [88] and Zwaneveld et al. [89]). Here, the problem is to find routes for trains to pass through a station, given the layout of the station, and a timetable. Similar to the timetabling problem, there is a set of hard safety constraints on, for example, the train movements and the distances between trains. Additionally, the routing should retain as much of the service issues of the timetable
as possible, such as connections and preferred dwell times. The platform assignment of trains to platforms follows from the routes assigned to the trains.

Figure 1.3 shows some of the available platforms at Utrecht along the left vertical axis, while the horizontal axis shows the times at which a platform is used. The lines in the figure indicate trains that are halting for specified periods of time at available platforms. The schematic station layout of the station Utrecht can be seen in Figure A.1 on page 133.

We consider the platform and track assignment as a separate problem from the timetable construction. In practice, however, one is not considered without the other, and thus a timetable is also assumed to describe a platform and track assignment.

1.3.5 Rolling stock planning

Although the platform and track assignments take the infrastructure details into account, an accurate assignment of rolling stock to the train lines is not made. The type and quantity of the engines and carriages to be used, and the more detailed routing of these units is made in the rolling stock planning. The efficient use of the rolling stock is important, especially for commercially oriented railway operators, since it is one of their largest cost sources. The rolling stock planning can be divided into two phases. First, the total amount of available rolling stock is distributed over the operated train lines (see Abbink et al. [3]), without making a detailed circulation plan. Second, a detailed rolling stock circulation is constructed for every line, given the total number of units assigned to this line in the first phase (see Alfieri et al.
Figure 1.4 shows an example of a rolling stock assignment for the train units operated on the railway tracks between Gvc (Den Haag) and Ut (Utrecht), Rtd (Rotterdam) and Utrecht, and Utrecht and Es (Enschede). This time-space diagram is similar to Figure 1.2, but with the space and time axes interchanged. Instead of showing the positions of trains, it shows the positions of rolling stock units. For readability, only the start and end segments of the diagonal lines are drawn. The line colours indicate the kind of rolling stock to be operated.

Consider, for example, the two trains 1731 and 21731. Train 1731 leaves Den Haag a few minutes after 9:00 with three train units headed for station Utrecht, as shown in the upper left corner of Figure 1.4. Around the same time, train 21731 departs from Rotterdam, also with three units, and also heading towards Utrecht. Both trains arrive in Utrecht at approximately 9:45. The three units of the 1731 train
are coupled with two of the three units of the 21731 train. The combination continues as train 1731 to Dv (Deventer), where two units are decoupled and transferred to train 636. The coupling and decoupling of train units is also referred to as combining and splitting.

### 1.3.6 Crew scheduling

The journeys of the trains, also of the empty trains or equipment between stations, are split into sequences of trips. A trip is a segment of a train journey that must be serviced by the same crew, without rest periods. The problem of crew scheduling deals with the construction of duties from a given collection of trips. Each duty has to satisfy several constraints, such as a maximum length of 9:30 hours. An example of seven duties is given in Figure 1.5. The trips are represented by the boxes in the chart. The numbers in the boxes correspond to the trains of which the according trip is part.

Due to the large number of trips—around 10,000 for some instances reported in Kroon and Fischetti [43]—and side constraints on the duties, practical crew scheduling problems are highly complex to solve. The general approach for making a crew schedule is, therefore, not based on the individual members of the workforce, but on these generic duties. The duties are later assigned to the actual crew members in the crew rostering phase. For more information on crew rostering and scheduling, see Caprara et al. [16] and Kroon and Fischetti [43]. It is clear that the crew scheduling/rostering problem and the earlier planning problems such as rolling stock planning are correlated. However, due to the complexity, these problems are mostly considered separately, although some researchers considers an integrated approach (see Freling [31], Freling et al. [32] and Wren and Fares Gualda [87]).
1.3.7 Shunting and maintenance planning

At the end of the planning process, after the timetable and rolling stock circulation are known, comes the problem of finding suitable shunting and maintenance plans. Let us first consider the shunting plan.

During the rush hours, most of the rolling stock is in use. At other times of the day, and especially during the night, not all train units are needed, and, therefore, they have to be parked. The processes involved with moving the rolling stock units to and from their parked positions are called shunting, or dispatching (see Gallo and Di Miele [34], Winter and Zimmermann [86] and Freling et al. [33]). The shunting movements are restricted by the available railway infrastructure, and by time and safety constraints. The available shunt tracks are not all identical. While some can be approached from both sides, others must be called in a last-in, first-out fashion. This is especially a complicating factor if the rolling stock units are also not all identical.

The shunting problem is not only the problem of matching incoming and outgoing rolling stock units, and parking them on available shunt tracks, but also to schedule the routing on the station infrastructure, and for cleaning and short term maintenance. For an example of the available tracks around a station—the station layout—see Figure 1.6.

Maintenance planning is generally done when the rolling stock circulation is known. It considers the deviations from the circulation that are necessary to carry out the maintenance of the individual carriages. The rolling stock schedule, as discussed in §1.3.5, only contains information about anonymous units. Not only would a more detailed planning make the scheduling problem much more difficult to solve, but due to possible disturbances the location of rolling stock is uncertain long before the date of operation. Several days before the maintenance of units is due, their location is known and the operated detailed circulation is altered to allow these units to visit one of the available maintenance facilities in the network. This may have consequences for the shunting plans. For recent research on this topic, see Kroon et al. [44].

As mentioned before, the input/output relations between the problems described above are not as strictly sequential as shown in Figure 1.1. Through feedback loops in the overall planning process, the occurrence of a critical problem at one stage can
call for a replanning at another, earlier stage. The overall planning process specifies the general structure and order in which the different problems are tackled. In the next section we review two overall planning processes: first with an emphasis on early detailed planning, and second a structure in which the planning of details is postponed as much as possible.

1.4 The overall planning process at NSR

The planning process at NSR can be divided into several chronological stages and into several organisational structures. Every stage deals with a subset of the planning problems of §1.3. Though possibly with some differences, the planning process at NSR exists also for other railway operators. Let us consider both the current planning process, and a conceptual planning process to be used in the near future.

1.4.1 Current planning stages

The current planning stages are schematically shown in Figure 1.7. These processes are described in detail in Prins [69] and Peeters [66]. The stages differ in two dimensions. First, the three stages are ordered chronologically, and differ moderately in the level of detail. Second, there is the distinction between how the chronological stages are treated by both the central and local planning departments.

At the most abstract level in the planning process lies the construction of the basic one-hour timetable (BOT), consisting of the basic one-hour pattern (BOP), and the basic platform assignment (BPA). The BOP gives a cyclic one-hour time-space
diagram for every track in the network (see also §1.3.3). The BPA adds in-station
details such as the platform assignments of the various trains in the BOT. The
BOT is constructed centrally at NSR, but is checked at the different local planning
departments. There is a feedback loop between the central and local planning de-
partments to arrive at a feasible BOP and BPA. For railway operators such as NSR,
that operate a cyclic timetable, this is the main step in the overall planning process.
In order to complete the other planning stages, the BOP and BPA must be ready
approximately nine months before the new timetable is scheduled to begin.

At the next stage, the one-hour timetable is extended to a timetable for one
standard week by the central planners. This stage takes roughly seven months, and
ends a few months prior to operations. It incorporates most of the planning details.
Both the rolling stock and crews are planned centrally (see also §1.3.5 and §1.3.6). In
addition, the detailed shunting plans are constructed. These plans are tested locally
for their feasibility. If changes to the standard week plan are necessary, then this
is reported to the central planning department, which proposes modifications and
restarts the feedback cycle. The standard week plans are adjusted several times per
year, using so-called adjustment sheets\(^5\).

Ultimately, daily plans are constructed throughout the year to fill in the details
of the standard week plan. For individual days of the year, the daily plans take into
account the planned maintenance of the infrastructure, but also, for example, extra
train services offered at large events such as soccer matches. Note that this also
requires altering the timetable.

The daily plans are designed in the same way as the standard week plan. Again,
the central planners plan the rolling stock and crew, and the local planners test the
feasibility of the necessary shunting operations. If the constructors of the daily plans
identify structural problems in the standard week plan, then the standard week plan
is adjusted by the central planners.

As can be seen on the timeline of Figure 1.7, most of the time is used at the
second stage, the planning for one standard week. Due to the high level of detail
in the standard week plan, the necessary correction cycles between central and lo-
cal planners often take significant time. To reduce the overall makespan, NSR has
launched a large-scale redesign project for the planning process. These proposals are
discussed in the next section.

1.4.2 Proposed planning process

To identify possible capacity problems with the daily planning and scheduling, the
original standard week plan of Figure 1.7 describes many details, such as the rolling
stock schedules and the shunting plans. However, the share of these details that
are critical for the feasibility of the standard week plan is only small. In addition,
the usefulness of these details is limited in the construction of the daily plans. For
example, due to maintenance of the infrastructure, the planned rolling stock schedules
need to be altered, which in turn influence the shunting plans, etc. Thus, much of the
time spent on making the detailed plans can be saved. The main goal for redesign is

\(^5\) In Dutch *Wijzigingsbladen*
1.4. THE OVERALL PLANNING PROCESS AT NSR

The overall planning process at NSR is characterized by process coordination, aiming to reduce the makespan of the planning process.

The proposed planning process is described in Figure 1.8. Note the absence of separate local and central planning departments. By investing in information and communication technology, these planning departments are virtually integrated to decrease the time lag caused by feedback loops. Still, any request or demand for changes in the plans is coordinated between the involved planning stages.

Similar to the first stage in Figure 1.7, the process starts with the construction of the BOT. Now, however, the planning of infrastructure details is advanced to this first phase, but only for so-called critical shunting and train activities. For a discussion on how to recognise a priori whether the required shunting movements for a train line will be critical, see Reinartz and Fassaert [73] and Van Eck van der Sluijs et al. [24]. Other preliminary studies, aimed at making rough capacity estimates, are also made for the rolling stock and crew plans. This causes the first planning stage to be longer than the original three months.

At the second stage, the BOT is used to construct the complete timetable for a set of basic days. Although the planning for the basic days is comparable to the current planning of a standard week, the makespan is considerably shorter. This is achieved by delaying the planning of uncritical details. Especially by the elimination of several correcting feedback cycles between local and central planning, this could considerably shorten the lag times and thus the makespan.

At the final stage, all the detailed plans for several weeks are made in parallel. These plans are made a number of weeks ahead of operation, using a rolling horizon. Thus, for example, in week 33, the new rolling stock circulation and crew schedule are made for the weeks 38-43, while in week 34 the planners are constructing the schedules for the weeks 39-44. This is represented in Figure 1.8 by the stacked Specific Days.
CHAPTER 1. INTRODUCTION

1.5 An introduction to line planning

A line plan lies at the heart of almost all scheduled public transport, whether it concerns bus or subway systems, regular trains, high speed trains, or airline services. All of these line plans specify a list of operated services between pairs of locations that are operated at a given frequency. This frequency can range from several times per hour for subways, to bimonthly services for a stagecoach line from the mid eighteen hundreds. An example of a modern day line plan is given in Figure 1.9.

The underground system in London dates back to 1863, when the world's first underground railway opened. The fourteen lines in this system have all been given names, such as the Hammersmith & City Line, the Picadilly Line, and the Victoria Line. A line plan lies at the basis of the operated timetable. While for the stagecoach example at best a specific day was known, a timetable in general specifies the times at which the operated means of transport calls at a station. On the other hand, for a system with high frequency connections, such as the London Underground, a detailed timetable is—at least for the passengers—no longer necessary. Instead, the fastest way by underground for an example trip from Gloucester Road to Westbourne Park is described as "At Gloucester Road, wait for Picadilly Line train. At Hammersmith, wait for Hammersmith & City Line train. At Westbourne Park, arrival" (see Figure 1.10).

For the public railway service in the Netherlands, a growing political interest in
1.6 Research objective

The aim of this thesis is to explore mathematical models and solution techniques for aiding the construction of railway line plans, with an emphasis on the efficient use of the available resources. This research topic is of high importance both for the service

---

6In Dutch Prestatiecontract
to the traveller, as well as for cost efficiency of the railway operator.

The lag time of investments for expanding the railway infrastructure or the rolling stock is too long to handle the growing passenger demand in the short run. Therefore, to meet the service standards and the traveller's growing mobility requirements, the currently available capacity must be used at high efficiency. This efficient use of resources is in line with the objectives of modern railway operators, as a consequence of the privatisation and the resulting competition on the tracks.

The line plan holds a pinnacle position within the planning process of railway operators. Reasons for adding or altering lines in the existing system vary. For the customer, the addition of new lines can directly decrease the travel time. Additionally, increasing the line frequencies may be preferable since it increases the traveller's planning freedom, and may reduce the expected travel time. The introduction of new lines can also reduce the system's vulnerability to delays. As an example, consider introducing a combination of short lines to replace a geographically long train line. A delay in the service of a long line can easily propagate through the network, since the long route has many blocking opportunities for other train lines. For the railway operator, focussing on the design of the line plan is important for the efficient use of the available resources—both of the infrastructure, and of the rolling stock and personnel. Efficient use of the rolling stock, especially at times of capacity shortages, cuts both ways. It is not only of value to the operator, but also increases the service to the passengers. The efficient use of the infrastructure and personnel, even though not directly interesting for the customer, can influence the level of service through, for example, lower operating costs and thus lower ticket prices.

The importance of research into the efficient use of resources, starting with line planning, has also been recognised by several independent studies, such as Project B&B [70], V&W [84] and NYFER [61]. For an overview of early literature on line planning, consider Bussieck [11]. The main objective in early work was to construct line plans that provide a high number of direct train connections for the travellers. A drawback of the direct travellers approach is that it often yields lines with geographically long routes. Besides the delay sensitivity of long lines mentioned before, there are also capacity related implications. A train line covering a geographically long route often has a significant variation in the number of passengers along its route. Depending on the available capacity, this leads to underutilisation or overutilisation, both of which are undesirable.

After the first steps in the mid 1990s by the European Union to come to privately owned railway operators, the research trend has moved to cost efficiency objectives. This lead to publications such as Claessens et al. [18], Zwaneveld [88] and Bussieck [11]. Of course, focussing on minimising the operating costs can also have negative effects. The most important disadvantage is that it can go at the expense of the service to the customer, e.g., by causing an increase in the number of passenger transfers.
1.7 Outline of this thesis

This thesis is structured as follows. First, the two chapters of Part I address various foundations. Chapter 2 gives an introduction into combinatorial optimisation problems and techniques. Chapter 3, on the other hand, focuses on issues in line planning modelling. We address topics ranging from the estimation of demand data, to topics involved with the throughput capacity of the railway network.

In the second part of this thesis we discuss three applications of mathematical models for railway line planning problems. In Chapter 4 we develop and analyse a branch-and-cut algorithm for solving cost-minimising line planning problems. We discuss new preprocessing techniques for these problems in §4.3, and also discuss a range of cutting plane classes and tree search issues such as branching rules and primal heuristics. Then, in §4.4, the implementational details of these techniques are discussed. For example, in §4.4.2 we describe the algorithms that are used to find violated cutting planes of the various classes that were introduced before. The proposed methods are then put to the test using a number of real-life line planning problems in §4.5.

The line planning model of Chapter 4 considers a separate line planning problem for each of the different types of trains, such as regional trains and intercity trains. Chapter 5 introduces the multi-type line planning problem. We develop three model formulations for this problem in §5.3 using a simplified sub-problem called the edge capacity problem. Equivalence proofs for these formulations are given. Yet, their performance in solving practical instances differs significantly. This is tested in §5.4.

The line planning models that were described in Chapter 4 and Chapter 5 design a new line plan. For every proposed line the halting stations are dictated by the type of the line, and the types of the stations along its route. However, these station types are part of the input of the line planning problem.

Alternatively, Chapter 6 presents a model that reconsider the stations at which the trains stop for a given line plan. This model is then used to determine the halting stations in such a way that the total travel time of passengers is minimised. First, in §6.2, we introduce new concepts, such as the line-event graph. Then, in §6.3 we show how to formulate this problem as a multi-commodity network flow problem with additional constraints and variables. Using Lagrangian relaxation, we show in §6.4 how to find lower bounds for this problem. To effectively use these bounds in a branch-and-bound framework, we introduce a number of preprocessing and tree search techniques, together with a problem-specific multiplier adjustment algorithm. This is discussed in §6.5. In §6.6 we, once again, describe a computational study based on instances of the Dutch passenger railway operator NSR.
The following is a passage from the document:

"...In this respect, it is clear that the introduction of these measures has not been without its challenges. The implementation of the new regulations has been met with resistance from some stakeholders, particularly those who are accustomed to the previous system. Despite these challenges, the benefits of the new system are clear, with improved efficiency and fairness. The success of these changes will depend on continued collaboration and support from all parties involved."
Part I

Foundations
Chapter 2

Combinatorial foundations

The results in the following chapters build on well-known foundations of combinatorial optimisation. In this chapter we recall the basics of such topics as (multi-commodity) network flows, integer programming and branch-and-bound. In the discussion of these topics, some familiarity with the basic notation and definitions of graph theory is assumed. For a general introduction into combinatorial optimisation, see Nemhauser and Wolsey [55], Papadimitriou and Steiglitz [65] and Schrijver [75, 77].

2.1 Polyhedral combinatorial optimisation

This section gives a short introduction into the field of combinatorial optimisation, and in particular to polyhedral methods.

2.1.1 Combinatorial optimisation problems

A typical description of a combinatorial problem is given using a finite ground set $E$ and a family $\mathcal{F}$ of subsets of $E$ called the feasible sets. Now, given a cost $c_e$ for every element $e$ of $E$, a combinatorial optimisation problem can in general be written as

$$\max \sum_{e \in F} c_e \quad \text{subject to } F \in \mathcal{F} \quad (2.1)$$

An example of a classical combinatorial optimisation problem is the maximum cardinality matching (MCM) problem: Given a graph with a set of vertices and edges connecting them, what is the largest subset of edges such that every vertex is part of at most one of the edges in this subset. For the MCM problem, an edge in the graph represents a matching between its two vertices. The ground set is built up out of all the edges, while the feasible sets consist of all feasible matchings. Since the MCM problem is concerned with finding the largest number of edges, all the weights of elements of the ground set are equal to one. A variation of this problem is the
Figure 2.1: Two types of hulls for the two points $x_1 = (1, 2)$ and $x_2 = (3, 1)$ from $\mathbb{R}^2$.

maximum weighted matching problem, where every edge is given a specific weight to indicate how valuable the edge is.

A relaxation of a maximisation problem is a problem

$$\max \ c_R(F)$$

subject to $F \in \mathcal{F}_R$ \hspace{1cm} (2.2)

where the feasible set $\mathcal{F}_R$ contains $\mathcal{F}$, and for which every solution $F$ of the original problem yields at least the same objective function value, i.e., $\sum_{e \in F} c_e \leq c_R(F)$.

Even though maximisation problems are considered here, these ideas carry over to minimisation problems with only minor changes. For example, relaxations of maximisation problems yield upper bounds, while for minimisation problems, they result in lower bounds. We will refer to the objective function value of a relaxation as a dual bound, and call a feasible solution to the original problem a primal bound.

### 2.1.2 Polyhedral theory

Polyhedral theory links combinatorial optimisation to integer and linear programming, discussed in the next section. Therefore, let us first give a short theoretical introduction.

Fundamental in polyhedral theory are the concepts of linear combinations and convex combinations. We denote the Euclidean linear space of dimension $n$ by $\mathbb{R}^n$. Consider a set of $k > 0$ points $S = \{x_1, \ldots, x_k\} \subseteq \mathbb{R}^n$. Some point $y \in \mathbb{R}^n$ is called a linear combination of the points in $S$, if for some scalars $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$ it holds that $y = \sum_{i=1}^{k} \lambda_i x_i$. If it also holds that $\sum_{i=1}^{k} \lambda_i = 1$, then $y$ is called an affine combination of the points in $S$. An affine combination with $\lambda_i \geq 0$ for all $i \in 1, \ldots, k$ is referred to as a convex combination. The set of convex (affine) combinations of all points of $S$ is called its convex (affine) hull. The convex hull of some set $S$ is often written as conv($S$). See also the examples in Figure 2.1 for $n = 2$.

A hyperplane in $\mathbb{R}^n$ is defined as the set of points $\{x \in \mathbb{R}^n\}$ that satisfy the nontrivial equation $ax = b$, where $a \in \mathbb{R}^n \setminus \{0\}$ and $b \in \mathbb{R}$. Alternatively, the set $\{x \in \mathbb{R}^n : ax \leq b\}$ is called a halfspace. A polyhedron is the intersection of a finite set of halfspaces. It is a well-known result in combinatorial theory that a bounded
2.1. POLYHEDRAL COMBINATORIAL OPTIMISATION

A polyhedron in n-dimensional real space is structurally equivalent to the convex hull of a finite set of points in $\mathbb{R}^n$ (see Minkowski [51] and Weyl [85]). Therefore, not all the definitions of a polytope in the literature are the same; some define it as simply a bounded polyhedron, while others use the convex hull description.

An inequality $ax \leq b$ is said to be valid for a set $S \subseteq \mathbb{R}^n$ if $S \subseteq \{x \in \mathbb{R}^n : ax \leq b\}$, i.e., if $S$ is contained within the halfspace defined by $ax \leq b$. Alternatively, a point $x^* \in \mathbb{R}^n$ violates the inequality if $ax^* > b$.

2.1.3 Integer linear programming problems

The link between combinatorial optimisation problems and polyhedra can be made, for example, by introducing variables $x_e$ that take the value 1 if $e$ belongs to a feasible set $F$, and 0 otherwise. These variables thus describe a feasible set $F \in \mathcal{F}$ as a point in $\mathbb{R}^n$, where $n$ is the size of the ground set. Using these variables, the optimisation problem of (2.1) can also be described as

$$\max \sum_{e \in E} c_e x_e \quad \text{subject to } x \in S$$

(2.3)

where the set $S$ is defined as the union of all feasible incidence vectors

$$x_e^F = \begin{cases} 1 & \text{if } e \in F \\ 0 & \text{otherwise} \end{cases}$$

which make up the feasible set $S = \{x^F \in \mathbb{R}^n : F \in \mathcal{F}\}$. If this set is not empty, then there always exists an optimal vertex solution to

$$\max \sum_{e \in E} c_e x_e \quad \text{subject to } x \in \text{conv}(S)$$

(2.4)

that is also in $S$. Following the equivalence between polytopes and bounded polyhedra, there exists a finite set of inequalities for which the intersection of the defined halfspaces is equal to conv$(S)$. Therefore, the optimisation problem given above can be solved using linear programming (LP) techniques. Finding the convex hull of a set of feasible solutions is, however, often not trivial, since the number of necessary linear constraints may be very large.

The feasible sets of combinatorial optimisation problems are frequently defined using integer linear descriptions, such as

$$S = \{x \in \mathbb{Z}^n : Ax \leq b\}$$

(2.5)

where $b$ is a vector, and $A$ is a matrix of appropriate dimensions. Note that this integer linear description can easily be extended to also incorporate continuous variables.

Consider again the maximum cardinality matching problem on a graph with $m$ vertices and $n$ edges. The constraints $Ax \leq b$ and $x \in \mathbb{Z}^n$ must enforce that every vector $x \in \mathbb{R}^n$ that satisfies them is an incidence vector of a matching, and vice versa.
This can be achieved by using \( Ax \leq b \) to enforce that every vertex is incident to at most one of the edges in the matching:

\[
\sum_{e \in E : v \in e} x_e \leq 1 \quad \text{for all vertices } v. \tag{2.6}
\]

Combined with the integrality restriction \( x \in \mathbb{Z}^n \), the restrictions

\[
0 \leq x_e \leq 1 \quad \text{for all edges } e \tag{2.7}
\]

complete the integer programming formulation of the feasible set of MCM.

For proving the correctness of this integer programming formulation, first consider a feasible matching \( F \), and the solution \( x_e = 1 \) if \( e \in F \), and \( x_e = 0 \) otherwise.

Clearly, since every vertex is part of at most one edge, the first class of constraints is satisfied. By construction the solution is integer and all \( x_e \) are between 0 and 1. Second, consider a feasible integer solution \( x_e \). From \( 0 \leq x_e \leq 1 \) it follows that \( x_e \in \{0, 1\} \). By contradiction, if the solution \( F = \{e \in E : x_e = 1\} \) is not a feasible matching, then apparently there exists a vertex \( v \) that is part of two edges \( e \) and \( e' \) in the matching \( F \). This, in turn, implies \( x_e = x_{e'} = 1 \), and thus that the constraint of the class (2.6) for vertex \( v \) is violated, since it has a left-hand side value of at least 2. The vector \( x \) can therefore not be a feasible solution, which contradicts our initial assumption.

### 2.1.4 Solution approaches

There are no known methods for solving general integer linear programming problems efficiently. Moreover, many core combinatorial problems, including general integer linear programming, belong to the class of \( \mathcal{NP} \)-hard problems (see Garey and Johnson [35]). For this class of problems it is strongly believed that no algorithm exists with a worst-case running time that is bounded by a polynomial in the size of the input.

**Cutting plane method**

For linear programming problems, on the other hand, there exist polynomial time algorithms. A well-known property of linear programs is that for every problem that is defined over a bounded polyhedron and for which an optimal solution exists, there also exists an optimal solution that is a vertex of the associated polyhedron. Consider solving the linear programming relaxation of (2.3) for \( S \) defined as in (2.5)

\[
\max \sum_{e \in E} c_e x_e \quad \text{subject to } x \in P_{LP},
\]

where \( P_{LP} = \{x \in \mathbb{R}^n : Ax \leq b\} \). In case this linear program results in an optimal solution that is integer, then this solution is also optimal for the problem defined on (2.5) since \( S \subseteq P_{LP} \). Notably, if \( \text{conv}(S) = P_{LP} \), then every optimal solution to the LP relaxation is also feasible, and thus optimal for the original problem on \( S \). 
However, in general the convex hull of $S$ and the LP relaxation polytope are not equal. This is illustrated in the following example.

Consider the feasible region in Figure 2.2. The feasible set $S$ is defined as

$$S = \{x \in \mathbb{Z}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1, 3x_1 + 4x_2 \leq 5\},$$

or equivalently, $S = \{(0,0), (1,0), (0,1)\}$. Since the convex hull of $S$ is strictly contained within the LP polytope

$$P_{LP} = \{x \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1, 3x_1 + 4x_2 \leq 5\},$$

the LP relaxation can yield solutions that are not part of $\text{conv}(S)$. However, consider the inequality $x_1 + x_2 \leq 1$. This restriction is valid for $\text{conv}(S)$, because $\text{conv}(S)$ is contained in the halfspace defined by this inequality, i.e., $x_1 + x_2 \leq 1$ is satisfied by all solutions of $S$. Yet, adding this constraint to the problem formulation will cut off part of the original $P_{LP}$ polytope. Therefore, it is also referred to as a cutting plane, i.e., a valid inequality $ax \leq b$ for which $P_{LP} \not\subseteq \{x \in \mathbb{R}^n : ax \leq b\}$. Adding $x_1 + x_2 \leq 1$ to the description of the problem makes the linear programming relaxation coincide with $\text{conv}(S)$. Thus, the original problem can be solved using this new linear programming relaxation. This method is also known as the cutting plane method. It was first introduced by Dantzig et al. [20], and is outlined below.

1. Given are a feasible set $S = \{x \in \mathbb{Z}^n : Ax \leq b\}$ and its linear programming relaxation $P_{LP} = \{x \in \mathbb{R}^n : Ax \leq b\}$
2. Solve the LP relaxation, and let $x^*$ be the optimal solution.
3. If $x^* \in S$, then $x^*$ is optimal for $S$. Hence, stop.
4. Otherwise, identify a cutting plane $ax \leq b$ that is violated by $x^*$.
5. Add the inequality to $P_{LP}$, and goto step (2).

Step (4) is the most important step of the method. The problem of identifying whether a solution $x^*$ is feasible, or else to find a violated valid inequality—a cutting
In general, the separation problem is polynomially solvable if and only if the original optimisation problem is polynomially solvable. Note, however, that this does not exclude the existence of polynomial time separation algorithms for specific classes of valid inequalities. Over the last decades, many classes of valid inequalities, and separation algorithms have been found for a wide range of combinatorial problems (see Nemhauser and Wolsey [55] and Aardal and Van Hoesel [1]). We also refer to §4.3.2 and §4.4.2 for classes of cutting planes and separation algorithms specifically for line planning problems. For the maximum cardinality matching problem, Edmonds [25] showed that the two classes of inequalities given before, together with the class of odd-set constraints

\[ \sum_{e \in E(U)} x_e \leq \left\lfloor \frac{|U|}{2} \right\rfloor \]

for all odd-sized subsets \( U \subseteq E \)
describe the complete convex hull of matchings. Even though there is an exponential number of odd-set inequalities, the separation problem can be solved in polynomial time (see Padberg and Rao [64]). This is in line with the existence of a polynomial time combinatorial algorithm for MCM described in Edmonds [26]. In general, however, the cutting plane method can fail in step (4), ending with a solution that is not in \( S \), e.g., a solution that is fractional. In such cases, an enumerative algorithm such as 'branch-and-bound' can be used.

**Branch-and-bound-and-cut**

The classical principal of branch-and-bound is best described as *divide and conquer.* The principle is that, if optimising over \( S \) is too difficult, perhaps the problem can be solved by optimising over subsets of \( S \), and then taking the best of these solutions. For the MCM problem, for example, the set \( S \) can be split into two subproblems \( S_{x_e=0} \) and \( S_{x_e=1} \) for some edge \( e \). Obviously, this division can also be done recursively, e.g., for various edges. The subproblems are also called the *nodes* of the branch-and-bound enumeration tree.

To prevent complete enumeration, a dual bound is computed at every node of the branch-and-bound tree. The value of this relaxation is a bound on the best solution of the subproblem associated with the current node. At this point, there are four possible outcomes:

1. The relaxation is infeasible.
2. The optimal solution to the relaxation is feasible for the original problem.
3. The optimal solution to the relaxation is worse than the best-known primal bound.
4. None of the above.

Clearly, the value of a feasible solution for the original problem is a primal bound on its optimal value. Throughout the search process, the best feasible solution is
stored in memory. In the first three cases, further investigation of the current node can be stopped. Otherwise, the problem at the current node is split into subproblems for new child nodes. All the nodes are stored in a list of active nodes. When a node has been evaluated, it is removed from this list. Thus, as soon as the list of active nodes is empty, the original problem is solved.

In most branch-and-bound implementations, the dual bound on the subproblems is calculated by solving the LP relaxation. In addition, to implement the branch-and-bound algorithm, we have to solve the problem of how to select the next subproblem (node selection strategy), and how to create new subproblems (branching rule). For a comprehensive study of these topics, see Linderoth and Savelsbergh [47]. We also refer to §4.3.3 and §6.5.3 for specific node selection and branching rules for line planning problems.

As can be seen, the quality of the derived bounds influences the size of the enumeration tree. The integration of the cutting plane method in the branch-and-bound algorithm is an obvious next step. In early work, the branch-and-bound algorithm was used after no more cutting planes could be found by the cutting plane algorithm. This is also called cut-and-branch. The alternative is to apply the cutting plane algorithm at every branch-and-bound node to improve the value of the LP relaxation. This is referred to as branch-and-cut. Several branch-and-cut software libraries have been developed in recent years. For a short overview, see Ladányi et al. [46].

Lagrangian relaxation

The idea to use a relaxation of the original problem to obtain a dual bound on the objective value can be taken further than the linear programming relaxation. The observation that relaxing some of the constraints, e.g., the integrality restrictions, makes the underlying problem easier to solve, is also used in a technique called Lagrangian relaxation (LR). For two classical studies of the use of Lagrangian relaxation for integer programming, see Geoffrion [36] and Fisher [29]. In addition, Ahuja et al. [5] gives a good overview of the Lagrangian relaxation technique, and several applications in network optimisation. As an example, consider the integer linear problem IP

\[
\begin{align*}
  z_{IP} &= \max \quad cx \\
  \text{s.t.} & \quad A^1 x \leq b^1 \quad \text{(complicating constraints)} \\
  & \quad A^2 x \leq b^2 \quad \text{(nice special structure)} \\
  & \quad x \in \mathbb{Z}^n_+,
\end{align*}
\]

where \( n = m_1 + m_2 \), and \( m_1 \) and \( m_2 \) are the number of rows of \( A^1 \) and \( A^2 \) respectively. For notational purposes, let us combine the integrality restrictions and the class of nice constraints in the feasible set \( Q = \{ x \in \mathbb{Z}^n_+ : A^2 x \leq b^2 \} \). The idea is now that the problem of optimising solely over \( Q \) is much easier to solve than IP. Therefore, by dualising the complicating constraints, we obtain a problem that is easier to solve, and whose optimal value is a dual bound on the optimal value of the original problem. This is explained in the next paragraphs.
For the problem IP, as defined above, the Lagrangian relaxation LR(\lambda) for any vector \lambda \in \mathbb{R}_+^{m_1}, is defined as

$$z_{LR}(\lambda) = \max_{x \in Q} cx + \lambda(b^1 - A^1 x)$$

The complicating constraints \(A^1 x \leq b^1\) are not part of the feasible set \(Q\). These constraints have been replaced by a penalty term \(\lambda(b^1 - A^1 x)\) in the objective function. Thus, since the element \(\lambda_i\) for every relaxed constraint \(i\) in \(1, \ldots, m_1\) is nonnegative, any violation of these constraints \(A^1 x \leq b^1\) worsens the objective function value. The vector \(\lambda\) is referred to as the vector of Lagrangian multipliers.

The Lagrangian relaxation LR(\lambda) is a relaxation of IP for any \(\lambda \in \mathbb{R}_+^{m_1}\). Any feasible solution \(x\) of IP clearly satisfies \(x \in Q\), and is therefore also feasible for LR(\lambda). For the objective function value of LR(\lambda) it is easy to see that \(cx + \lambda(b^1 - A^1 x) \geq cx\) holds for any feasible solution \(x\) of IP, since \(\lambda \geq 0\) and \(A^1 x \leq b^1\). Thus, \(z_{LR}(\lambda) \geq z_{IP}\) for any \(\lambda \geq 0\).

The Lagrangian dual problem LD is the problem of finding the vector \(\lambda\) that results in the tightest (lowest) upper bound, i.e.,

$$z_{LD} = \min_{\lambda \in \mathbb{R}_+^{m_1}} z_{LR}(\lambda). \quad (2.8)$$

The vector of multipliers at which this minimum is attained, is denoted by \(\lambda^*\).

There are several reasons for considering Lagrangian relaxation techniques. A first reason is found in the following theorem.

**Theorem 2.1.** If, for a given vector \(\lambda\), the solution vector \(x\) satisfies the three conditions

1. \(x\) is optimal for LR(\(\lambda\))
2. \(A^1 x \leq b^1\)
3. \(\lambda(b^1 - A^1 x) = 0\)

then the vector \(x\) is also optimal for the original problem IP. If only the last condition is not satisfied, then \(x\) is at most \(\epsilon\) away from the optimal solution of IP, with \(\epsilon = \lambda(A^1 x - b^1)\).

These are the well-known conditions under which a solution of a Lagrangian relaxation is also optimal for IP (see Geoffrion [36]). The condition \(\lambda(b^1 - A^1 x) = 0\) is also called the complementary slackness condition. Note that, since all elements of \(\lambda\) are nonnegative, these conditions thus state that for every element \(i \in \{1, \ldots, m_1\}\) either \(\lambda_i\) is zero, or \((b^1 - A^1 x)_i\) is zero (or both). Also note that \(\epsilon\) is nonnegative, since the solution vector \(x\) satisfies \(A^1 x \leq b^1\) by the second assumption.

The optimality conditions given in Theorem 2.1 show that Lagrangian relaxation is not only useful for obtaining dual bounds, but may also yield optimal solutions to the underlying problem IP.
2.2 MULTI-COMMODITY FLOWS

Another reason for considering Lagrangian relaxation is the quality of the Lagrangian dual, as compared to the linear programming relaxation of IP. As was shown in Geoffrion [36]

\[ z_{LP} \leq z_{LD} \] (2.9)

for a general maximisation problem like IP, where \( z_{LP} \) is the optimal value of the linear programming relaxation of IP. In other words, the optimal Lagrangian bound \( LR(\lambda^*) \) is at least as good as the bound obtained from the linear programming relaxation. However, Geoffrion [36] also gives conditions under which the two bounds are equal. This is stated in Nemhauser and Wolsey [55] as follows.

**Theorem 2.2.** If all the extreme points of \( \{ x \in \mathbb{R}^n_+ : A^2 x \leq b^2 \} \) are integral, then \( z_{LP} = z_{LD} \) for any cost vector \( c \).

Thus, under the mentioned conditions, the Lagrangian bound is not better than the LP bound. In such cases, it can still be preferable to use Lagrangian relaxation for dual bounding, if the Lagrangian bound is easier to compute (see also §6.4).

So far, we have not considered the problem of optimising the Lagrangian dual in (2.8). It is essential that for a given vector \( \lambda \), the problem structure of the relaxed problem \( LR(\lambda) \) is easy to solve. Now, first observe that \( z_{LR}(\lambda) \) is piecewise linear and convex (see, for example, Geoffrion [36] and Nemhauser and Wolsey [55]). The methods used to solve the problems of the Lagrangian dual can be divided into two categories.

The most widely used methods use subgradient optimisation algorithms. Even though many versions of such algorithms exist, the property that is used is that for some vector \( \lambda \) and the according optimal solution \( \bar{x} \) of \( LR(\lambda) \), the vector \( b^1 - A^1 \bar{x} \) is a subgradient of \( z_{LR} \) at \( \lambda \). The idea in these algorithms is that the optimal vector \( \lambda^* \) for LD can be found by sequentially updating \( \lambda \leftarrow \lambda + s(b^1 - A^1 x) \) with a decreasing step size \( s \). For examples of generic subgradient optimisation algorithms, see Ahuja et al. [5], Bazaraa et al. [8] and Nemhauser and Wolsey [55].

Alternatives to subgradient optimisation are multiplier adjustment methods. Here, the combinatorial structure of the particular problem is used to find the optimal multiplier vector. Several examples in the literature are known where this method outperforms generic subgradient based algorithms. Fisher [29] presents a small overview and some empirical data for these methods. For an example of multiplier adjustment methods for the generalised assignment problem, see Fisher et al. [30]. In Volgenant and Jonker [81], a detailed description is given of adjustment techniques for the symmetric travelling salesman problem, based on the 1-tree relaxation. Again, more information on Lagrangian relaxation can be found in §6.4.

2.2 Multi-commodity flows

A basic problem in the class of network flow problems is the minimum cost ("min-cost") flow problem. For an in-depth coverage of network flow problems, see Ahuja et al. [5]. In min-cost flow problems, the question is how to transport a commodity through a network as cheaply as possible, while satisfying demand at certain nodes.
from available supplies at other nodes. Most instances also take into account the limited availability of capacity on the links (arcs) between the nodes.

Consider a directed graph \( D = (V, A) \) with vertices \( V \), and directed arcs \( A \) connecting the vertices. The demand or supply at a vertex \( i \in V \) is given by \( b(i) \). The quantity \( b(i) \) is also called the net supply of vertex \( i \). If \( b(i) \) is negative, then \( b(i) \) is considered a demand vertex, whereas \( b(i) > 0 \) implies that \( i \) is a supply vertex. On every arc \((i, j)\) between two vertices \( i \) and \( j \), an upper bound \( u_{ij} \) and lower bound \( l_{ij} \) is given on the total amount of flow across arc \((i, j)\). Now, given a cost \( c_{ij} \) per unit of flow for all arcs \((i, j) \in A\), the min-cost flow problem can be formulated as

\[
\begin{align*}
\min & \quad \sum_{(i,j) \in A} c_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{j: (i,j) \in A} x_{ij} - \sum_{j: (j,i) \in A} x_{ji} = b(i) \quad \text{for all } i \in V \\
& \quad l_{ij} \leq x_{ij} \leq u_{ij} \quad \text{for all } (i,j) \in A
\end{align*}
\]  

(2.10a)  

(2.10b)  

(2.10c)

where a variable \( x_{ij} \) represents the amount of flow on an arc \((i, j)\). The objective function (2.10a) sums up the total cost of the flow vector \( x \). The constraints (2.10b) are referred to as the mass balance constraints. These restrictions ensure that for every node \( i \), the total flow leaving \( i \), minus the total flow entering \( i \), is equal to the net supply of \( i \). For example, for nodes with a net supply of zero, i.e., with \( b(i) = 0 \), the constraints (2.10b) ensure that the total amount of inflow equals the total amount of outflow. Very often, these restrictions are also stated in matrix notation as \( Nx = b \), where \( N \) is the node-arc incidence matrix, and \( x \) and \( b \) represent the flow vector and net supply vector. The constraints (2.10c) reflect the lower and upper bounds on the amount of flow on the arc from \( i \) to \( j \).

The min-cost flow problem is well-known to be polynomially solvable. This accounts for why, as was mentioned in §2.1.4, Lagrangian relaxation techniques have many applications in network optimisation.

One of the properties of min-cost flow problems is the integrality property: If all arc capacities, supplies, and demands of nodes are integer, then there always exists an integer minimum cost flow, if the instance is feasible.

If there is only one vertex \( i \) with \( b(i) > 0 \), then the problem is called a single source problem.

**Observation 2.3.** The single source min-cost flow problem without capacity restrictions can be solved by finding a shortest path tree rooted at the source vertex.

Because the problem is uncapacitated, every unit of flow that is sent from the source vertex \( s \) to some target vertex \( t \) can follow a shortest (cheapest) path through the network.

Multi-commodity flow (MCF) problems arise when several distinct commodities use the same underlying network. The ways in which the commodities differ can be, e.g., their physical characteristics, or simply their origin and destination in the network. Since they use the same network, the commodities compete for the available capacity on the arcs. These capacity restrictions bind the commodities together.
For an MCF problem, let $x^k_{ij}$ and $c^k_{ij}$ denote, respectively, the amount of flow of commodity $k$ on arc $(i,j)$, and its cost per unit. Similar to the linear programming formulation of the single-commodity flow problem, the MCF problem can be formulated as

$$\min \sum_k c^k x^k$$

s.t. $N x^k = b^k$

$$\sum_k x^k_{ij} \leq u_{ij} \quad \text{for all } (i,j) \in A$$

where $x^k$ and $c^k$ are the flow vector and per unit cost vectors. Note that the objective function allows for different costs per arc for units of different commodities. The restrictions in $N x^k = b^k$ are the mass balance constraints for every commodity. The second class introduces for every arc $(i, j)$ in the network an upper bound $u_{ij}$ on the total flow of all commodities using $(i, j)$.

If the flows through the network are allowed to take non-integral values, then the min-cost multi-commodity flow problem is polynomially equivalent to linear programming, and thus solvable in polynomial time (see Itai [42] and Grötschel et al. [40]). However, the integrality property of single-commodity flows does not hold for multi-commodity flows. The MCF problem with integral flows is $\mathcal{NP}$-hard, even in the case of only two commodities (see Even et al. [28]).

**Observation 2.4.** If there are no capacity restrictions on the arcs in an MCF problem, then the MCF problem can be solved by considering a separate single-commodity flow problem for every commodity.

This observation is also used in the application of Lagrangian relaxation to MCF problems. By dualising the capacity restrictions, the LR problem for a given value of the multiplier vector $\lambda$ does not involve capacity restrictions and can thus be solved efficiently. For a complete example, see Ahuja et al. [5].

### 2.3 Theoretical overview

The techniques and models discussed in the previous sections are used in various chapters throughout this manuscript. Chapter 4 presents an integer programming formulation for modelling the line planning problem. For this formulation, we describe in detail a branch-and-cut algorithm. This includes several new classes of cutting planes, and accompanying separation techniques. Using a set of practical instances, it is shown that the use of branch-and-cut for this problem can yield optimal, or at least very good, primal and dual bounds in short time.

The results in Chapter 5 depend to a large extent on multi-commodity flow models. By considering passengers as commodities that want to be transported through the railway network, an MCF model is used to prove the validity of a number of alternative formulations by showing their equivalence. These (mixed) integer linear formulations, presented in §5.3.2 and §5.3.3, are used for solving three real-life instances.
Chapter 6 combines multi-commodity flow models with the theory of Lagrangian relaxation. The problem at hand is modelled as an MCF problem with additional restrictions. However, for real-world instances, the resulting problems are too large to handle. Therefore, using Lagrangian relaxation techniques on the additional constraints, the overall problem is split into a series of subproblems that can each be solved efficiently.
Chapter 3

Foundations in line planning modelling

This chapter bridges the gap between the real-world environment described in the introduction, and the models discussed in the next chapters. Here, we focus on practical aspects of line planning, and on how they are modelled, or left out. This chapter also introduces the notation and mathematical concepts that are specific for the railway models presented in later chapters.

3.1 Line plans and cyclic timetables

The line planning problem, as presented in §1.5, is concerned with determining on which routes to operate trains, and at what hourly frequencies. This definition implies the construction of cyclic timetables. Such timetables are repeated, more or less identically, every hour of the day (see also Peeters [66] and §1.3.3).

The passenger demands, however, vary significantly throughout the different hours of the day. The morning and afternoon rush hours, for example, are the busiest parts of the day (more on this topic in §3.3.3). The alternative, an a-cyclic timetable, offers more opportunities to the railway operator to fine-tune the timetable to match these demand fluctuations. Such a line plan could contain many trains during the rush hours, but only few trains during the rest of the day. A line plan and cyclic timetable can only adapt to the fluctuating market demand by adjusting the operated capacity of the train lines. Typically, during rush hours, high capacity trains are used, such as longer and/or double-deck trains. In off-peak hours, trains may consist of less carriages.

In practise, a hybrid concept is possible, based on a cyclic core timetable. This allows extra trains to be operated during rush hours that are not available during low-traffic hours. However, the amount of available infrastructure and rolling stock is often a binding factor in railway services during peak hours.
3.2 Rolling stock

The main cost factors that determine the operating costs of a line plan are the costs of using the infrastructure, and the costs of the required personnel and rolling stock. Both the number of trains needed to operate the lines at given frequencies, as well as the necessary number of conductors and drivers, depend on the circulation of the rolling stock, and therefore on the timetable. For this reason, some authors consider an integrated approach to construct a timetable and a rolling stock circulation (see Nachtigall [52, 53] and Lindner [48]).

However, neither the timetable, nor the rolling stock circulation are known when the line plan is constructed. Therefore, to estimate the amount of necessary rolling stock, we assume a circulation schedule in which all rolling stock is dedicated to specific lines. So, the switching of rolling stock between lines is kept to a minimum, and thus the interdependencies among the train lines are reduced (see also the reliability issues in §1.2.1).

We assume that every train line is operated in both directions. This is standard policy for most railway operators, since it offers a more transparent service to the passenger. We also suppose that the composition of rolling stock units of trains is the same in each direction. In addition, the trains used to operate a single line are assumed to be identical, i.e., pulling the same number of carriages. Now, given the line’s hourly frequency and the number of carriages per train, the costs of operating a specific line can be determined using the circulation time (see also §3.5.3).

Splitting and combining of trains during their trips is assumed to be not allowed. In practice this splitting and combining is sometimes done for two lines that have a significant part of their route in common: two trains can be combined at the station where their routes meet, and split where the trains leave the common route. An example of this is given in §1.3.5. Although the splitting and combining of trains may have a negative impact on the punctuality, it also allows for more direct connections for passengers. At the same time, the capacity on the tracks taken up by the combined trains is less than for the two separate trains. This is, for example, due to minimum safety distances between trains (see also Peeters [66]).

Other restrictions are not due to the availability of the rolling stock, but because of the rolling stock itself. In particular, consider the combining of train units to build trains. The train units of NSR are classified according to classes and subclasses. Figure 3.1 shows units of three different classes, and for every class two subclasses. The class Koploper consists of two subclasses: a subclass with three carriages called Koploper 3, and a subclass with four carriages referred to as Koploper 4. Such units cannot be split into smaller parts. Trains can only consist of units of one class, although possibly of different subclasses. For example, a Koploper train combines units of 3 and 4 carriages. As a consequence, it is not possible to make a Koploper train with 5 carriages in total. Similarly, the Regiorunner class has units of 4 and 6 carriages, while the Mat’64 class can combine units of 2 and 4 carriages. In general, all these train units can move in two directions by themselves, and, thus, do not need

\[1\] To avoid confusion with the types that are used to define the halting patterns of lines, we do not use the common types and subtypes here.
additional locomotives.

Note that the bottleneck for the maximum number of rolling stock units of a train is often the platform length of the visited stations, rather than the physical and technical limitations of the units.

3.3 Passenger demand

In this section, we discuss passenger demand and service to the passengers in connection with line planning. The topic of estimating the demand was already briefly discussed in §1.3.1. We also address issues concerning cost-oriented line planning and the service to passengers, as mentioned before in §1.6.

Most railway operators offer their travellers a number of differently priced comfort alternatives. Similar to airline companies, two such levels are often used: first class and second class. As mentioned by NS [59], next to short travel times, passengers value a high level of service and comfort. By offering, e.g., more comfortable seats in much less crowded carriages, a modest percentage of the travellers is willing to pay around 60% more, compared to the price of a second class ticket (see NS [60]). With insufficient train capacity, some travellers—often those who travel second class—question whether the two-class system should remain. Throughout this thesis we do not make a distinction between classes, and consider only aggregated data. This is, however, merely a strategic modelling decision.

3.3.1 Passenger demand and capacity planning

There are many different alternatives for representing the passenger demand. Consider, for example, cross-section data. In railway transport terms, the cross-section data describes the numbers of passengers in the trains that run in parallel at a particular moment in time. In the time-space diagram of Figure 1.2 on page 6, the lines in the figure mark the positions of trains at different points in time. This time-space diagram describes the basic one-hour timetable for a specific part of the network, but in general such a schedule is available for a complete day, and for every part of the network. Now, the trains in the eight o'clock cross-section, for example, are those trains that cross the imaginary horizontal line of eight o'clock.

Cross-section data is often used in the allocation of rolling stock. It is clear that, at least from a service perspective, the number of units that is assigned to every train in the timetable must meet the demand of passengers as well as possible at every moment of the day—in particular for the 8 a.m. cross-section (see Abbink et al. [3]). This is an example of bottleneck planning, since the morning rush-hour is usually the busiest time of the day. More on the variation of demand over time can be found in §3.3.3.

The problem with using cross-section data for line planning is obvious: the data is defined on an existing set of lines, and thus not suitable for designing a new line plan. An example of a method that can describe the passenger demand without using a line plan or timetable, is the origin-destination (OD) matrix. The rows and columns of this matrix represent the different stations in the network. Thus, a cell
Figure 3.1: Different types of rolling stock (Pipers [68]).
of the matrix refers to a particular OD pair, and contains the total number of people that, within a given time period, want to make this trip. The problem of estimating OD matrices, for example as part of overall traffic modelling, has received a lot of attention in the last fifty years (see, e.g., Abrahamsson [4], Bierlaire [9] and Sherali and Park [79]). The OD matrix, in general, refers to the number of trips between any two geographic locations. The splitting of all trips over the available travel modes, such as car, bus, train, underground or on foot, is called a modal split.

An example of an origin-destination matrix between a set of railway stations is shown in Figure 3.2. The origin stations are given on the vertical axis. The size of the bubble for a pair of stations \((v, w)\) represents the number of travellers from \(v\) to \(w\), within a period of one hour.

Train travellers often have a number of combinations of train lines and connections available to take them from their origin to their destination. These connections may even involve several geographically different routes. As mentioned before, a passenger’s choice is influenced by the different travel times, ticket prices, necessary train changes, and comfort and service levels. The problem of estimating how users travel through the network is known as the traffic assignment problem. The traffic assignment problem is another important step in the analyses of transportation demand, besides the estimation of the OD matrix (see Safwat and Magnanti [74]).

A traffic assignment simulation model specific for railway systems is described by Oltrogge [63].

In order to make capacity assignments to lines in a proposed line plan, we have to be able to estimate, or to make assumptions on, how passengers will traverse the network and which train lines they will take. Since finding such a traffic assignment is not trivial, we make assumptions on the route choices of passengers through the network. One possibility is to let the line planning model determine a traffic assignment, that is ideal for minimising the operating costs of the line plan. Since the model thus dictates the trips of passengers, this solution can be far from the true traffic assignment made by passengers themselves. In particular, it can involve excessively large travel times for certain travellers, if this would have a positive effect on the total amount of assigned capacity.

A first alternative is to assume an a priori assignment of passengers to geographical routes through the network. From the passenger’s perspective, this is comparable to the restrictions enforced by the ticket regulations. The freedom of the line planning model is now restricted to assigning the passengers to train lines along the prescribed route. This assumption has two important implications. Firstly, it can be used to exclude unrealistic traffic assignments in which some travellers are expected to use unnecessarily long routes. Secondly, this assumption on the routes of passengers can be used to simplify the line planning models considerably (see Chapter 5). The prescribed routes could be the shortest routes in the network, or the observed routes taken by travellers according to the currently operated timetable. The essence is that the route is known for every traveller.

The above approach is used in the models that are presented in Chapter 4 and Chapter 5. The capacity that is allocated to the lines of the new line plan must suffice to transport all passengers. The difference in the assumptions in this respect,
Figure 3.2: An example origin-destination matrix.
between Chapter 4 and Chapter 5, is that in the former we assume that the traffic assignment has been split into assignments for the different train systems, such as intercity trains and regional trains. Such a *system split* is used by many authors (see Bussieck et al. [12], Claessens et al. [18]).

Another alternative is the traffic assignment model used in Chapter 6. Here we use, e.g., estimates of the amount of time needed to change from one train to the next to predict the shortest path of passengers with respect to travel time. Different from the first traffic assignment, this assignment therefore depends on the operated line plan. Such an assignment model can be used since Chapter 6 is involved with determining halting patterns of trains given a line plan. This model is then used in §6.6 for an analysis of the line plans that were proposed in Chapter 5.

The problem with all these approaches is clear: the assumed traffic assignment—split or not—and the OD matrix need not be realistic when the new line plan is put into operation. The newly-determined line plan can in turn influence the behaviour of potential travellers. For example, a line plan that offers more frequent and direct connections between stations might well encourage travellers currently using their car to switch to using the train. For a more detailed discussion on the substitution effect between public transportation and car travel, see V&W [82, 83].

The approach applied in practice is to evaluate the resulting line plan using detailed traffic assignment models. The resulting demand data is used to generate a new line plan, and so on, hopefully obtaining a consistent passenger allocation within a small number of iterations.

### 3.3.2 Passenger demand across the network

The concern of §1.6 is that a line plan that aims to minimise the operating costs could have a negative impact on the service to the customer, if the new line plan results in an increase in the travel times or the necessary number of train changes. This effect depends on the structure of the passenger demand across the network. If, for example, the length of the trip of a passenger is small compared to the lengths of the tracks in the network, then the number of necessary train changes will also be low.

An example distribution of the trip lengths of passengers is visualised in Figure 3.3. This figure is based on morning rush-hour data from NSR. The recorded distances are the lengths of the shortest paths in the railway network. As mentioned in §1.1, the average trip length in the Dutch network is around 35–40 kilometers, while nearly 50% of the passengers travel less than 25 kilometers. However, the average distance between consecutive intercity stations is 27.3 kilometers, and 19.0 and 6.5 kilometers between interregional and regional stations respectively.

### 3.3.3 Passenger demand variations over time

The passenger demand varies over time from one hour to the next, and from day to day. This behaviour is illustrated in Figure 3.4, which shows the total number of travellers in the system at different points in time. The distribution peaks at the morning rush hour around eight o'clock and in the afternoon at roughly half past
Figure 3.3: The (cumulative) percentage of travellers grouped by travel distance. Distance group 0 includes all trips with a length of \([0, 5)\) kilometers, etc.

five. The morning rush hour is generally busier, while the evening rush hour is spread over a larger time interval. The trips in the evening are often the return trips of the morning journeys.

The line plan is by definition a plan for one standard hour. Thus, we assume that the OD matrix on which the line plan is based portrays a steady state situation. An alternative would be to determine a core line plan, including a subset of lines that are not operated in off-peak hours, similar to the hybrid timetable concept discussed in §3.1.

### 3.4 Infrastructure

In this section we discuss issues in line planning related to the stations and the tracks in the network.

#### 3.4.1 Station capacity

The number of trains a station can handle per hour depends, among other things, on the available platforms and the lengths of the platforms, the possible simultaneous routes through the station, and the available tracks and crew for shunting movements.

Every train that visits a station, whether it halts there or passes through without stopping, occupies the tracks and platforms at that station for a certain amount of time. The necessary amount of time depends on the type of the train and on the type of rolling stock. Turning a train around can take, e.g., between 10 and 30 minutes.
3.4 INFRASTRUCTURE

Figure 3.4: The number of travellers at different points in time.

In addition, there are also service aspects, such as a preference for cross-platform connections, that influence the utilisation of platforms. The number of routes through a station can, in a similar way, be a binding factor in the throughput capacity of a station (see also §1.3.4). Another important restrictive factor is concerned with the parking of train units and related processes, i.e., the shunting. As mentioned in §1.3.7, the shunting of trains is restricted by the available railway infrastructure, available personnel, and by time and safety constraints.

When more and more trains visit a station, any of these three factors can become critical in the operational process. Therefore, the station utilisation is an important factor in the investigation of how to increase the throughput capacity of the railway network.

3.4.2 Homogenising the train system

As mentioned in §1.2.2, several recent studies propose a homogenisation of the train system to create more throughput capacity on the railway network (see NS [59], Railforum [72] and Project B&B [70]). There are several approaches for doing so, as shown in Figure 3.5. The current timetable is shown in Figure 3.5(a). There are two lines being operated: an intercity line departing every 15 minutes from station a and going directly to station g, and a regional line, stopping at every station, and also leaving a every 15 minutes, but three minutes after the intercity line. Given that the trains have no opportunity to pass each other on the tracks and that there must always be a minimum distance of three minutes between subsequent trains, the maximum number of connections between a and g in this instance is 8 connections.
CHAPTER 3. FOUNDATIONS IN LINE PLANNING MODELLING

A first alternative approach considers shortening the lengths of the train lines. An example is shown in Figure 3.5(b). By splitting the regional line into three separate lines, and letting one depart before the feeder line arrives, the travel times of the two original lines become more homogenous. Therefore, the throughput capacity can increase. This option would, however, be inconvenient for the travellers that enter and leave the system in the regional stations. These passengers have to change trains more often.

Another option, debated in NS [59], Railforum [72] and Project B&B [70], is to abandon the strict hierarchical pattern of ordered station and line types. Figure 3.5(c) demonstrates the effect of introducing a system where one line halts at a, c, e and g, and the other at a, b, d, f and g. Similar to the first approach, this also homogenises the travel times of both lines, thereby creating the possibility to stack the lines closer together in the timetable. Again, the number of lines per hour can increase.

A third approach, consider clustering the intercity trains and regional trains. This option is discussed in Peeters [66]. By homogenising the traffic order and velocity, a higher number of connections per hour can be achieved. However, this setting is not likely to be implemented, since it is not attractive for travellers.

A last option—that requires additional investments—is the building of new tracks to increase the possibilities for faster trains to overtake slower ones. This alternative is visualised in Figure 3.5(e).

By using a combination of building new tracks for overtaking, and by relaxing the halting patterns, NS [59], Railforum [72] and Project B&B [70] conclude that the throughput capacity can be increased by roughly 30% to 40%. However, this would be achieved at the expense of an increase in the delay sensitivity of the timetable. This is also emphasised in NYFER [61].

3.4.3 New railway safety systems

Supported by the European Commission, a new European railway traffic management system (ERTMS) has been designed to increase the track capacity, and to improve the safety and the interoperability between railway operators internationally. This system consists of two parts: the European Train Control System (ETCS) and the Global System for Mobile communications - Railways (GSM-R). ETCS is the new automatic control system for which the communication requirements, including voice and data communication, are provided by GSM-R (see ERTMS [27]).

The ETCS includes a new train safety system by means of moving safety blocks. The current safety system, called a fixed block system, divides every track into a number of fixed blocks (see Figure 3.6). If, for example, one block is occupied by a train, then any oncoming train is given a red signal before entering this block, or even one block before. The sizes of the blocks are such that even the longest and heaviest train can stop within one block.

With the new moving block system that is part of ETCS, these fixed blocks are replaced by safety blocks that move with the trains. The sizes of such blocks are based on the individual speeds, sizes and breaking powers of the trains. Using the
Figure 3.5: The current situation with alternating intercity and regional trains (left), and four proposed modifications.
GSM-R system, the drivers of the trains can now be given specific information on how to alter their driving speeds. Thus, the moving block system can allow trains to drive closer together, thereby increasing the throughput capacity of the infrastructure.

3.5 Modelling the railway system

3.5.1 Infrastructure

The railway infrastructure of the stations and tracks is modelled as an undirected graph $G = (V, E)$ of stations (vertices) and tracks (edges). This graph is also called the network graph. Every track $e = \{v, w\} \in E$, by definition represents the track connecting stations $v$ and $w$ of $V$. As an example, two network graphs are shown in Figure 3.7.

3.5.2 Lines

A line is described by a path in the railway network along which trains are operated at a given hourly frequency (see §1.3.2). The route that a line $l$ follows through the network graph is given by a simple path of edges. For ease of notation, we define this route simply as $l \subseteq E$.

The line planning problems discussed here are all based on selecting a set of operated lines from a large set of candidates. Therefore, an important part of the input consists of the set of candidate lines. Not every route between a pair of stations is a feasible candidate. Many infrastructural and operational restrictions have to be taken into account, such as the shunting possibilities for turning at the end stations and the maximum distance covered by a line. For every feasible pair of end stations, at most three possible connecting paths, i.e., candidate lines, are considered. These paths are, in general, the three shortest paths for the combination of begin and end
Figure 3.7: The Dutch and German (Bussieck [11]) railway networks.
station, where the length of the longest of these paths is not allowed to be more than twice the length of the shortest (see also Claessens et al. [18], Härte [41]).

The stations at which a line stops, also called the line's halts or dwells, are given by the type of the line. This type comes from a set of types \( T = \{1, \ldots, T_{\text{max}}\} \). Not only is every line \( l \) associated with a type \( t(l) \), also every station \( v \in V \) is of a type \( t_v \) from \( T \). The halting pattern of \( l \), i.e., the stations at which \( l \) halts, are defined by the line type \( t(l) \), and the types of the stations along the line's route. The types often reflect the sizes of the stations: type 1 for stations in villages, up to type \( T_{\text{max}} \) for stations serving large metropolitan areas. Most real-life railway instances consider three types of stations and lines. These are referred to as regional (R) for type 1, interregional (IR) for type 2, and intercity (IC) for type 3. Note that some authors refer to regional lines and stations as "aggloregional".

As is common in the Netherlands, but also in most other countries, the halting patterns of train lines follow a simple ordering. Train lines of type 1 halt at all stations they pass. Lines of type 2 skip the small stations of type 1, but halt at all stations of type 2 and higher, etc. Throughout this thesis we assume that the halting stations of lines follow this strict hierarchical pattern in which a line of type \( t \) halts at all stations along its route that are of type \( t \), or higher. In practice, however, exceptions are sometimes made (see §3.4.2).

### 3.5.3 Rolling stock

It takes a number of trains to operate a line at a given frequency. To estimate the necessary number of trains without a timetable or a rolling stock schedule, we use the total time needed to operate the line in both directions and the time needed at both end stations to prepare the train for the reverse journey.

Consider a line with a travel time of 60 minutes from the line's origin to its destination station, as shown in Figure 3.8. At either end station, at least 15 minutes are needed to prepare the train for the reverse journey. This amounts to a total circulation time of 150 minutes. If the line is operated once per hour, this would require (at least) \( \lceil 150/60 \rceil = 3 \) trains (Figure 3.8(a)). However, operating the line at a frequency of two (Figure 3.8(b)), departing every 30 minutes, will call for \( \lceil 150/30 \rceil = 5 \) trains. In general, we let \( cp_l \) denote the fractional number of necessary trains per
hour for some line \( l \).

The rolling stock is one of the largest cost drivers of a railway operator. The hourly costs of operating a line plan can be described by a function of fixed and variable costs per used train and per carriage (see also Claessens et al. [18]). The fixed costs are incurred for the availability of the individual trains and carriages and involve, for example, depreciation costs and fixed maintenance costs. Examples of variable costs per kilometer are cost sources such as ticket collectors costs, energy costs and maintenance costs. Now, the total cost of operating a line \( l \) at frequency \( f \), and with \( c \) carriages per train can be calculated as

\[
[cp_l \cdot f] \cdot (w_{\text{fix}}^{\text{tr}} + c \cdot w_{\text{fix}}^{\text{car}}) + d \cdot f \cdot (w_{\text{var}}^{\text{tr}} + c \cdot w_{\text{var}}^{\text{car}}).
\]  

(3.1)

This formulation thus incorporates four classes of resource costs:

- \( w_{\text{fix}}^{\text{tr}} \) the hourly costs per train.
- \( w_{\text{fix}}^{\text{car}} \) the hourly costs per carriage.
- \( w_{\text{var}}^{\text{tr}} \) the hourly costs per train per kilometer.
- \( w_{\text{var}}^{\text{car}} \) the hourly costs per carriage per kilometer.

The distance in kilometers covered by \( l \) is given by the parameter \( d \). Note that the total number of kilometers that is progressed in one hour by the trains of a line does \textit{not} depend on the number of trains, but only on the frequency \( f \) and the distance \( d \).

To estimate the required amount of rolling stock and the involved costs, the line planning models of Chapter 4 and Chapter 5 not only decide on which lines to operate, and at what frequency, but they also assign a number of rolling stock units to every train line.

The costs described above are for a period of one hour. As mentioned, this planning period is often taken to be the morning rush-hour, the busiest part of the day. However, the costs of one hour at peak-time are not representative for the costs at other hours, or at different days. Therefore, as proposed by Claessens [17], scaled estimates are used that reflect the total \textit{annual} cost of operating a train for one hour during rush-hour.

This concludes this introduction chapter on the railway related issues. The next part of this thesis considers three line planning applications.
Part II

Applications
Application
Chapter 4

A branch-and-cut approach for solving railway line planning problems

This chapter considers the problem of determining a cost-optimal line plan for a single train type. We present an integer linear programming model, and solve it using branch-and-cut. For this, we give a variety of valid inequalities and reduction methods. A computational study of five real-life problems based on instances of NSR is included. The results in this chapter are based on Goossens et al. [38].

4.1 Introduction

Traditionally, the objective when constructing a line plan has been to find a set of lines that maximises the number of direct travellers, i.e., the number of travellers that do not have to change trains during their journey (see also §1.6). This is an obvious target from a customer service point of view. However, it ignores important issues, such as the involved operating costs and the efficient use of the rolling stock. As an alternative objective, similar to Claessens et al. [18] and Bussieck [11], this chapter focusses on the problem of minimising the operating costs of a line plan.

There are several approaches for formulating the cost-optimising line planning problem (CLP). In Claessens et al. [18], an integer nonlinear programming model is presented as a natural model for CLP. They transform this model into a linear programming problem on binary variables (BLP), and attempt to solve it using branch-and-bound. In Bussieck [11], an alternative formulation using general integer variables (ILP) is presented. In recent work, Bussieck et al. [13] reconsider the original nonlinear model. They study a relaxation of the nonlinear formulation to obtain good primal solutions of the original problem.

Besides presenting the ILP formulation, Bussieck [11] also compares the BLP and ILP formulations using a cutting plane algorithm with cut-and-branch. He concludes
52 CHAPTER 4. BRANCH-AND-CUT FOR LINE PLANNING PROBLEMS

that the more compact ILP formulation is preferable for generating good feasible solutions. However, the lower bounds provided by BLP are superior to the lower bounds of ILP, therefore the BLP formulation is more suitable for proving optimality of a feasible solution. By extending the known reduction methods and classes of cutting planes presented, this chapter shows that the BLP formulation can be used to solve large instances of the problem using branch-and-cut, or at least to obtain excellent upper bounds and lower bounds in reasonable time.

The model formulation is presented in the next section, recalling some of the details about line planning from Chapter 3. In §4.3 we describes the solution methodology. The implementational issues are described in §4.4. Finally, §4.5 presents a computational study based on instances of the Dutch passenger railway operator NSR.

4.2 Model formulation

A line specifies a route between an origin and a destination station and the subsequent stops, combined with the operated hourly frequency. The line plan is the set of operated lines. It does not incorporate any timetable information for the operated lines, though we assume that the timetable is cyclic with a cycle time of one hour, i.e., that the line plan is repeated every hour. This reflects the situation in the Netherlands, as well as in many other European countries. The model described here focuses on finding a line plan that minimises the induced operating costs, given the railway infrastructure with the accompanying stations and the number of travellers between stations.

For a given network graph \( G = (V, E) \), as defined in §3.5.1, the travellers determine their route through the network based on the associated travel time. In general, this implies that they prefer to switch to lines of higher types at the earliest opportunity. As described in Oltrogge [63], this idea can be used to decompose the overall problem into separate line planning problems for the different types. Throughout this chapter, we therefore assume that the line planning problem considers exactly one type of lines. Such a system split is used by many authors (consider Bussieck et al. [12] and Claessens et al. [18]). Line planning models with multiple line and station types are discussed in Chapter 5. For an instance of type \( t \), the network graph can be simplified in a straightforward way by removing all the stations at which lines of type \( t \) do not halt. The edges in \( G \) are modified accordingly. For an example, consider Figure 4.1.

As discussed in §3.3, the demand data for the line planning problem are given by an OD matrix for which the route through the network is known for every traveller. Moreover, we assume that the flow of passengers is symmetric. Therefore, for every edge we can calculate the number of passengers traversing it. As a service to the passengers, we assume that each line is always operated in both directions. Thus, as in Claessens et al. [18], we can regard lines as being undirected.

The level of service that the proposed line plan offers to the passenger is ensured by two conditions. The capacities of the lines should suffice to transport all passengers, and the line plan is enforced to guarantee a high number of connections between
4.2. MODEL FORMULATION

Figure 4.1: The Dutch network, and the network graph for the according IC instance (SP97IC).

every pair of consecutive stations in the network. We consider this issue in more detail in the next section.

4.2.1 Formulation

Our formulation of CLP is similar to the formulation used by Claessens et al. [18], and later in Bussieck [11]. Given is the undirected network graph $G = (V, E)$ with vertex set $V$, edges $E$ and a set of potential lines $L$. Every line $l \in L$ corresponds to a set of edges that make up a simple path between the end vertices of $l$. The consecutive stops of line $l$ are given by the stations corresponding to the vertices along its path, i.e., the stations $v$ for which there is an edge $\{v, w\} \in l$. Recall that we consider only line planning problems with exactly one type of lines. For every line we have to decide whether to deploy it, and, if so, at what hourly frequency and with how many carriages. The set of possible frequencies for the lines is denoted by $F \subseteq \mathbb{N}$, the possible number of carriages by $C \subseteq \mathbb{N}$. As mentioned before, the resulting line plan must meet two types of service restrictions. For every edge $e$ in the network, we are given a minimum number $f^e$ of passing trains per hour, and a necessary number of carriages $h^e$ needed to meet the demand of all passengers who want to traverse $e$ per hour.

For formulating the line planning problem as an integer linear programming problem, we introduce a binary variable for every $(l, f, c) \in N$, with the set of triples as $N = L \times F \times C$. Every $i \in N$ is used for referring to a particular $(l_i, f_i, c_i)$ combination. For convenience, we also introduce the set $N(e) \subseteq N$ for every edge $e$ as $N(e) = \{i \in N : e \in l_i\}$. The line planning problem CLP can now be formulated
where the decision variables $x_i$ for $i = (l_i, f_i, c_i) \in N$ are used to model

$$x_i = \begin{cases} 1 & \text{if line } l_i \text{ is operated at frequency } f_i, \text{ with } c_i \text{ carriages per train} \\ 0 & \text{otherwise} \end{cases}$$

The objective function coefficients $w_i$ represent the costs of operating the line associated with variable $x_i$, as discussed in more detail in equation (3.1) in §3.5. The constraints (4.1b) enforce the minimum number of passing trains per hour on every edge $e \in E$. Restrictions (4.1c) impose the lower bound $h^e$ on the number of carriages crossing edge $e$ in one hour. Typically, $h^e$ is the smallest number of carriages necessary for transporting all passengers across edge $e$. Note again that we consider all units of rolling stock to be identical. The third group of constraints ensures that every line is operated in at most one configuration. Contrary to Claessens et al. [18] and Bussieck [11], we have no upper bound restriction on the number of passing trains per hour for every track. Including these restrictions to account for possible bottlenecks in the infrastructure would be part of a more operational study.

Claessens et al. [18] prove that finding a cost-optimal allocation of trains using only capacity restrictions is $\mathcal{NP}$-hard. Since in general, that problem is a special case of CLP, it follows that CLP is also $\mathcal{NP}$-hard.

Before describing our branch-and-cut method, we first introduce some general notation and visualisation of CLP instances throughout the remainder of this chapter. Consider Figure 4.2. The instance consists of three stations, $V = \{u, v, w\}$, and two tracks, $E = \{e = \{u, v\}, f = \{v, w\}\}$. The pairs of numbers in brackets above the edges show the frequency and capacity lower bounds $f^e$ and $h^e$ respectively. The overall set of candidate lines $L$ for this network consists of three lines $L = \{l_1, l_2, l_3\}$. We refer to an individual variable $x_i$ as being of line $l_i$ with a frequency of $f_i$ trains per hour, consisting of $c_i$ carriages, and therefore with an hourly capacity of $f_i \cdot c_i$ carriages.

### 4.3 Branch-and-cut method for CLP

The branch-and-cut techniques are divided into three sections: preprocessing (§4.3.1), cutting planes (§4.3.2) and tree search (§4.3.3). The preprocessing section focusses
on reducing the initial size of the problem. This is done by removing superfluous variables and constraints, and by tightening the model restrictions. Next, we discuss a variety of classes of cutting planes, both generally applicable, and cutting planes specifically derived for CLP. Finally, the tree search section covers branching rules and primal heuristics.

### 4.3.1 Preprocessing

The preprocessing techniques described in this section are designed to strengthen the initial formulation of the problem. Strengthening, in this context, covers both coefficient reduction techniques, and methods for reducing the number of variables and constraints used to model a given CLP instance. Claessens et al. [18] and Bussieck [11] describe several preprocessing techniques, specifically for CLP. We briefly recall these techniques and then review them in a more general context.

#### Coefficient reduction

The main constraints of the model—the capacity constraints (4.1c) and the frequency constraints (4.1b)—are strengthened using the methods described below. The coefficient strengthening or reduction techniques used here are extensions of ideas and techniques described in Dietrich and Escudero [21].

The coefficient strengthening techniques are described using a generalised line planning problem GLP with only two types of constraints.

\[
\text{z}_{\text{GLP}} = \min \sum_{i \in N} w_i x_i \quad \text{subject to } x \in S. \tag{4.2}
\]

The feasible set \( S \) is defined as all \( x \) for which

\[
Ax \geq b \tag{4.3}
\]

\[
Cx \leq 1 \tag{4.4}
\]

\[
x \in \{0, 1\}^n \tag{4.5}
\]

where \( N \) is the set of variables with \( n = |N| \). The matrix \( A \) is an \( m \times n \) matrix with nonnegative integer entries \( a_{ij} \), and \( b \) is an \( m \)-dimensional vector of positive integers. Every variable \( x_i \) is associated with a unique line \( l \in L \), denoted \( l_i \). For every \( l \in L \), there is a constraint \( \sum_{i \in l} x_i \leq 1 \) in the second constraint matrix \( C \). Note that CLP is contained in this class, where the constraints (4.3) contain the service constraints on the tracks.
We derive several methods for strengthening problems of this form. Using notation similar to Dietrich and Escudero [21], the coefficient \( a_{kj} \) of row \( k \) and column \( j \) can in general be strengthened to

\[
\tilde{a}_{kj} = \max\{0, a_{kj} - \Delta_{kj}\} = \max\{0, b_k - Q_{kj}\}
\]

(4.6)

where \( \Delta_{kj} = a_{kj} - b_k + Q_{kj} \geq 0 \). Here, \( Q_{kj} \) represents a lower bound on the left-hand side value of constraint \( k \) in any feasible solution \( \bar{x} \in S \) with \( \bar{x}_j = 1 \). To maintain the nonnegativity property for \( a_{kj} \), we have added the lower bound on \( \tilde{a}_{kj} \). Note that an initial strengthening can be obtained by setting \( \tilde{a}_{kj} = b_k \) for all \( a_{kj} \geq b_k \) since we have assumed \( a_{kj} \geq 0 \) and all variables are binary. This reasoning also implies \( Q_{kj} \geq 0 \).

Next, we present two techniques for determining valid values for \( Q_{kj} \). Both techniques use an additional row from the constraint matrix \( A \) to strengthen its coefficients. Note that if \( Q_{kj} \) is valid for \( S \), then also any \( \bar{Q}_{kj} \leq Q_{kj} \) is valid. Alternatively, for any valid, though non-integer \( Q_{kj} \), it follows that also \( \bar{Q}_{kj} \) is valid. Both techniques give lower bounds for the value of the following minimisation problem.

**Theorem 4.1.** Consider an instance of GLP as defined by (4.2)-(4.5). Strengthening the coefficient \( a_{kj} \) of the constraint matrix \( A \) according to (4.6) with

\[
Q_{kj}(h) = \min \sum_{i \neq j} a_{ki} x_i \quad \text{subject to} \quad a_{hj} + \sum_{i \neq j} a_{hi} x_i \geq b_h, Cx \leq 1, x \in \{0,1\}^n
\]

(4.7)

if \( a_{hj} < b_h \), and \( Q_{kj}(h) = 0 \) otherwise, is valid for \( S \) for any constraint \( h \).

_Proof_. We prove that for all \( x \in S \) it holds that \((b_h - Q_{kj}(h))x_j + \sum_{i \neq j} a_{ki} x_i \geq b_k\). This is obviously true for all \( x \in S \) for which \( x_j = 0 \), since then the new coefficient of \( x_j \) is unimportant. For all \( x \in S \) for which \( x_j = 1 \) we have to show that \( \sum_{i \neq j} a_{ki} x_i \geq b_k - (b_k - Q_{kj}(h)) = Q_{kj}(h) \). Clearly, any \( x \in S \) with \( x_j = 1 \) satisfies \( a_{hj} + \sum_{i \neq j} a_{hi} x_i \geq b_h \) and \( x_j + \sum_{i \neq j} a_{ki} x_i \leq 1 \), making \( x \) a feasible solution to the minimisation problem in (4.7). \( \square \)

Solving the integer covering problem in (4.7) is in general too complex to use it for coefficient strengthening. We discuss two relaxations of this problem that give good bounds for \( Q_{kj}(h) \).

**Example 4.2.** Consider the set \( S \) of integer points \( x \in \{0,1\}^5 \) for which

\[
\begin{align*}
3x_1 + 4x_2 + 7x_3 + 4x_4 + 6x_5 & \geq 7 \\
x_1 + x_2 + x_3 + x_4 + 2x_5 & \geq 2 \\
x_1 + x_2 + x_3 & \leq 1 \\
x_4 + x_5 & \leq 1
\end{align*}
\]

Consider strengthening the coefficient of \( x_3 \) in the first constraint. Although it is equal to the right-hand side, any feasible solution \( x \in S \) with \( x_3 = 1 \) must still...
have at least one of the variables \( x_4 \) or \( x_5 \) set to 1 for the second constraint to be satisfied. Meeting the second constraint would therefore imply an additional left-hand side of \( Q_{13} = \min\{4, 6\} = 4 \) units in the first constraint and therefore

\[
3x_1 + 4x_2 + (7 - 4)x_3 + 4x_4 + 6x_5 \geq 7
\]

is valid for \( S \). Note that for instance the fractional solution \( \{0, 0, \frac{1}{4}, 0, \frac{7}{8}\} \) to the LP relaxation of \( S \) is cut off by the strengthened constraint.

The following corollary generalises the idea of the previous example.

**Corollary 4.3.** Consider an instance of GLP as defined by (4.2)-(4.5). Take two not necessarily distinct rows from the constraint matrix \( A \), say rows \( h \) and \( k \). Strengthening the coefficient \( a_{kj} \) according to (4.6) with

\[
Q'_{kj}(h) = \begin{cases} a_{kr} & \text{if } a_{hj} < b_h, \\ 0 & \text{otherwise} \end{cases}
\]  

(4.8)

where \( r = \arg\min_{i \mid l_i \neq l_j, a_{hi} > 0} a_{ki} \), is valid for \( S \).

**Proof.** We prove the validity of \( Q'_{kj}(h) \) by showing that it is the solution to a relaxation of the optimisation problem in (4.7). Given that \( a_{hj} < b_h \), then \( \sum_{i \mid l_i \neq l_j} a_{hi}x_i > 0 \) is a relaxation of the restriction in (4.7). Now, for binary \( x \), this new constraint is satisfied only if at least one \( x_i \) with \( l_i \neq l_j \) and \( a_{hi} > 0 \) has value 1, and thus

\[
a_{kr} = \min \sum_{i \neq j} a_{ki}x_i \quad \text{subject to} \quad \sum_{i \mid l_i \neq l_j} a_{hi}x_i > 0, x \in \{0, 1\}^n.
\]

Note that rows \( h \) and \( k \) need not be different. This method for finding valid \( Q'_{kj} \) works well on constraints with moderately sized \( b_h - a_{hj} \). If this number is larger, we may consider another relaxation of (4.7), where we drop the integrality constraints imposed on the variables in the covering problem. As described later, the resulting continuous covering problem can be solved to optimality using a greedy algorithm.

**Example 4.4.** Consider the feasible set \( S \) as introduced in Example 4.2. We again strengthen the coefficient of \( x_3 \) in the first constraint. In any feasible solution \( x \) with \( x_3 = 1 \), equation (4.7) tells us that strengthening \( a_{13} \) with

\[
Q_{13} = \min 4x_4 + 6x_5 \quad \text{subject to} \quad x_4 + 2x_5 \geq 1, x_4 + x_5 \leq 1, x_4, x_5 \in \{0, 1\}
\]

is valid for \( S \). If we drop the integrality restrictions on \( x_4 \) and \( x_5 \), we can solve the remaining problem using a greedy algorithm. The optimal solution is \( x_4 = 0, x_5 = 0.5 \), which shows us that \( Q_{13} \geq 3 \). From this it follows that

\[
3x_1 + 4x_2 + (7 - 3)x_3 + 4x_4 + 6x_5 \geq 7
\]

is valid for \( S \).
The following corollary generalises this idea.

**Corollary 4.5.** Consider GLP, taking two rows from the constraint matrix $A$, say rows $h$ and $k$. Strengthening the coefficient $a_{kj}$ of column $j$ in row $k$ according to (4.6) is valid for $S$, with

$$Q''_{kj}(h) = \min \sum_{i \neq j} a_{ki}x_i \quad \text{subject to} \quad a_{hj} + \sum_{i \neq j} a_{hi}x_i \geq b_h, \quad Cx \leq 1, \quad x \in [0,1]^n \quad (4.9)$$

if $a_{hj} < b_h$, and $Q''_{kj}(h) = 0$ otherwise.

**Proof.** Clearly, (4.9) is a relaxation of the optimisation problem in Theorem 4.1 and thus $Q''_{kj}(h) \leq Q_{kj}(h)$. 

The continuous covering problem with non-overlapping constraints (4.4) is closely related to the multiple-choice knapsack problem (see Martello and Toth [50]). It can be solved using a greedy algorithm similar to the one for continuous knapsack problems. Consider finding the optimal solution to

$$\min \sum_{i=1}^{n} w_i y_i \quad \text{subject to} \quad \sum_{i=1}^{n} a_i y_i \geq b, \quad Cy \leq 1, \quad y \in [0,1]^n$$

in which $w, a \in \mathbb{Z}_+^n$ and $b \in \mathbb{Z}_+$. The constraint matrix $C$ describes the non-overlapping group constraints of (4.4) over all $y$ variables. Without loss of generality, assume the variables are ordered so that $w_1/a_1 \leq \ldots \leq w_n/a_n$. Now construct an optimal solution $y^*$ as follows. Consider all variables in the given order, starting at $y_1$, and skip the variable $y_j$ if it is in the same group as an already selected variable. Otherwise, if $\sum_{i<j} a_i y_i^* < b$, then

$$y_j^* = 1 \quad \text{or} \quad y^* = \left(b - \sum_{i<j} a_i y_i^*\right)/a_j$$

depending on the size of the gap that still has to be satisfied. Once $\sum_{i<j} a_i y_i^* \geq b$, then it follows that all remaining $y_j^*$ will be 0.

The strengthening methods presented above all build on the assumption that all coefficients are nonnegative. This property is guaranteed by $\bar{a}_{kj} = \max\{0, b_k - Q_{kj}\}$ in (4.6).

**Variable reduction**

We reduce the number of variables by deriving dominance relations between groups of variables. This is a preprocessing technique that uses an exchange argument between variables in feasible solutions. The essence of this technique is also described in Claessens et al. [18]. The variable reduction of Bussieck [11] uses similar techniques. Both parties, however, do not directly link their variable reduction methods to coefficient strengthening procedures. By introducing this link, we improve upon their results and give a more transparent description of these techniques. Note that,
4.3. BRANCH-AND-CUT METHOD FOR CLP

as stated in Bussieck [11], their variable reduction is done without strictly preserving
the feasibility of the elimination.

If in any feasible solution \( x \) with \( x_j = 1 \), this solution can be altered to \( x_j = 0 \)
and \( x_i = 1 \) for some \( i \), while maintaining feasibility and without worsening the
objective value, then \( x_j \) is said to be dominated and can be removed from the model.
If there is a fixed \( x_i \) with this property, then \( x_i \) is said to dominate \( x_j \).

**Lemma 4.6.** Given an instance of GLP as defined by (4.2)-(4.5), consider two
variables \( i, j \in \mathbb{N} \), with \( i \neq j \). Variable \( x_i \) dominates variable \( x_j \), and can
therefore be removed from the problem, if

\[
\begin{align*}
\frac{w_i}{a_{ki}} & \leq \frac{w_j}{a_{kj}} & \text{and} & \\
\frac{a_{ki}}{a_{kj}} & > 1 & \text{for all } k \in \{1, \ldots, m\}
\end{align*}
\]

**Proof.** Straightforward. □

The variable dominance relations described above are transitive, i.e., if \( i \) dominates \( j \) and \( j \) dominates \( k \) then \( i \) dominates \( k \). Variable dominance is tested between
all variables of a line \( l \in L \). The dominance technique described above is also used
in Claessens et al. [18], but without linking it to coefficient reduction techniques.
Without coefficient strengthening, the model may not contain many dominated variables.
The next example demonstrates the effect of coefficient strengthening on the
dominance relations between variables.

**Example 4.7.** Consider the following minimisation problem

\[
\begin{align*}
\text{min} & \quad 15x_1 + 20x_2 + 25x_3 + 18x_4 + 24x_5 + 30x_6 \\
& \quad 3x_1 + 4x_2 + 5x_3 + 3x_4 + 4x_5 + 5x_6 \geq 5 \\
& \quad x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 1 \\
& \quad x_1 + x_2 + x_3 \leq 1 \\
& \quad x_4 + x_5 + x_6 \leq 1
\end{align*}
\]

with \( x \in \{0, 1\}^6 \). By applying the techniques from the previous section, we strengthen
the coefficients in the first two constraints. Let us denote the coefficient of variable \( x_j \)
in the \( i \)-th constraint in this problem by \( a_{ij} \). We use the order \( \{a_{11}, a_{21}, a_{12}, a_{22},
\ldots, a_{26}\} \) for strengthening the coefficients, which is equivalent to variable by vari-
able, first constraint 1, followed by constraint 2. For this coefficient order, the first
two constraints can be strengthened to

\[
\begin{align*}
2x_1 + 2x_2 + 5x_3 + 3x_4 + 3x_5 + 5x_6 & \geq 5 \\
0x_1 + 0x_2 + x_3 + x_4 + x_5 + x_6 & \geq 1
\end{align*}
\]

Now it is clear that \( x_2 \) is dominated by \( x_1 \) and \( x_5 \) by \( x_4 \).

**Constraint reduction**

This section derives dominance rules for constraints. Again, we assume the problem
to be given in the form of GLP. We introduce conditions under which constraints
are redundant for the description of the set of feasible solutions.
Theorem 4.8. Consider an instance of GLP and a pair of constraints $h$ and $k$. Constraint $k$ is redundant for the description of GLP if $b_k \leq Q_k(h)$, with

$$Q_k(h) = \min \sum_{i \in N} a_{ki} x_i \quad \text{subject to } x \in P$$

(4.10)

where $P = \{x \in [0,1]^n : \sum_{i \in N} a_{hi} x_i \geq b_h, Cx \leq 1\}$.

Proof. Clearly, the feasible region $P$ is a superset of the solution set $S$ of the original problem, since it is defined by only one constraint of the constraint matrix $A$, and for $P$ we have $x \in [0,1]^n$. From this definition, it follows that for every solution $x \in P : \sum_{i \in N} a_{ki} x_i \geq Q_k(h)$. Since $S \subseteq P$, this also holds for feasible solutions $x \in S$.

Theorem 4.8 can also be used to strengthen the right-hand side $b_k$ to $[Q_k(h)]$, since all coefficients $a_{ki}$ are integer.

Example 4.9. Consider the feasible set $S$ defined as

$$S = \{x \in \{0,1\}^5 : 3x_1 + 4x_2 + 7x_3 + 4x_4 + 7x_5 \geq 7, \quad x_1 + x_2 + 2x_3 + x_4 + x_5 \geq 2, \quad x_1 + x_2 + x_3 \leq 1, \quad x_4 + x_5 \leq 1\}.$$ 

If we define the minimisation problem (4.10) on the first two constraints, then $Q_{10}(2) = 7$ from the solution $(1,0,0,1,0)$. Thus, the second constraint induces a left-hand side value for the first constraint of 7 in the LP relaxation. Therefore, the first constraint is redundant.

Note that the constraint reduction or strengthening techniques are not independent of one another. Similar to the coefficient strengthening and variable reduction techniques described earlier, they should be applied sequentially, updating the optimisation problem in (4.10) accordingly every time.

As we will see in the computational results of §4.5, constraint dominance frequently occurs in CLP instances, both between the capacity and frequency constraints of some track, as well as between the service constraints on connecting tracks. Especially for dead-end tracks, frequently all lines covering such a track also cover an adjacent track, causing overlapping non-zero elements in the service constraints of both tracks. If the required frequency and capacity for the dead-end track are higher than for the neighbouring track, then the service constraints of the other track are redundant.

Example 4.10. Assume that all lines crossing some track $e$ also use track $f$. Therefore, all the capacity that is made available on $e$ will also be available on $f$. Thus, if the required capacity on $e$ is larger than on $f$, i.e., $h_e \geq h_f$, then the capacity restriction on $f$ is redundant.

### 4.3.2 Cutting planes

Besides preprocessing, we describe several classes of cutting planes. The separation algorithms for these classes are discussed in §4.4.2.
4.3. BRANCH-AND-CUT METHOD FOR CLP

Figure 4.3: An example of constraint dominance where the capacity restriction of track $f$ is redundant.

Probing cuts

This class of cutting planes is described on GLP problems, as introduced in (4.2)-(4.5). We derive valid inequalities from information that is obtained from variable probing.

Example 4.11. Assume that $\bar{x}$ is the optimal solution to the LP relaxation of an instance of GLP. Here $\bar{x}_1 = 1$, $\bar{x}_2 = 0.8$ and $\bar{x}_i = 0$ for all $i \notin \{1, 2\}$. One constraint in the associated problem is

$$5x_1 + 10x_2 + \sum_{i \in N \setminus \{1, 2\}} a_{ki}x_i \geq 13.$$  

Fixing $x_1$ to 1 in this constraint yields

$$10x_2 + \sum_{i \in N \setminus \{1, 2\}} a_{ki}x_i \geq 8 \quad \Rightarrow \quad 8x_2 + \sum_{i \in N \setminus \{1, 2\}} a_{ki}x_i \geq 8.$$  

The left inequality is also valid in case $x_1 = 0$, since it is weaker than the original inequality. Now, by using for example the coefficient strengthening techniques we can reduce the coefficient of $x_2$, yielding the right tightened constraint. The solution $\bar{x}$ violates this new valid inequality.

The idea described in the example is generalised in the following lemma. The way in which these cuts are constructed preserves the problem-specific structure of the initial inequality. Therefore, they can be used to generate new cuts of known classes of valid inequalities. Note that the probing inequalities themselves will never be violated.

Lemma 4.12. Consider an instance of GLP and some variable $x_j$ for $j \in N$. The inequality

$$\sum_{i|l_i = l_j, a_{ki} > a_{kj}} a_{ki}x_i + \sum_{i|l_i \neq l_j} a_{ki}x_i \geq b_k - a_{kj}$$  

is valid for $S$ for any constraint $k$ of the constraint matrix $A$.  


Proof. For any feasible solution it follows that \( \sum_{i \mid l_i = l_j} a_{ki} x_i \leq 1 \), and thus
\[
b_k \leq \sum_{i \in N} a_{ki} x_i = \sum_{i \mid l_i = l_j} a_{ki} x_i + \sum_{i \mid l_i \neq l_j} a_{ki} x_i \leq a_{kj} + \sum_{i \mid l_i = l_j, a_{ki} > a_{kj}} a_{ki} x_i + \sum_{i \mid l_i \neq l_j} a_{ki} x_i.
\]

Since the inequality (4.11) is valid for the original problem, it is also possible to recursively apply this technique for a set of variables. Note that the resulting inequality is order independent as long as at most one variable per line is used for probing, since the new coefficient \( a_{jk} = 0 \).

2-Cover cuts

This class of cutting planes, called 2-cover (2C) cuts, is based on the covering constraints in the CLP for a given track \( e \). If variable \( x_i \) alone does not satisfy both service constraints, then every feasible solution with \( x_i = 1 \) will contain at least one more line across \( e \). This observation is made in the following example.

Example 4.13. Consider the following polytope \( S \) of a CLP instance
\[
S = \{ x \in \{0,1\}^5 : x_1 + x_2 + x_3 + 2x_4 + 2x_5 \geq 3 \}.
\]

Since at least two of these variables have a nonzero value in any feasible solution
\[
x_1 + x_2 + x_3 + x_4 + x_5 \geq 2
\]
is valid for \( S \).

Lemma 4.14. Let the set \( C \) and its complement \( \bar{C} \) be defined as
\[
\bar{C} = \{ \{ i \in N(e) \} \mid f_i \geq f^c, f_i c_i \geq h^c \}
\]
and \( C = N(e) \setminus \bar{C} \). For any instance of CLP, the inequality
\[
\sum_{i \in C} x_i + \sum_{i \in \bar{C}} 2x_i \geq 2 \tag{4.12}
\]
is a valid inequality for every \( e \in E \).

Proof. Clearly, every feasible solution \( \bar{x} \) for which \( \bar{x}_i = 1 \) for \( i \in \bar{C} \) satisfies (4.12). Alternatively, for any solution with \( \bar{x}_i = 0 \) for all \( \bar{C} \), by definition there must exist at least two distinct variables \( i, j \) in \( C \) with \( \bar{x}_i = \bar{x}_j = 1 \) for \( \bar{x} \) to be feasible. \( \square \)
The next corollary describes an extension of Lemma 4.14 in which the variables
are divided into two or more parts. In particular, the resulting so-called multi-cover
(MC) cuts consider \( n + 1 \) subsets.

**Corollary 4.15.** Consider an arbitrary instance of CLP. Given a track \( e \), and a
positive integer \( n \) for which \( 2 \leq n + 1 < h^e \), then the inequality

\[
\sum_{s=1}^{n} \sum_{i \in C^s} s x_i + \sum_{i \in \bar{C}} (n+1)x_i \geq n+1
\]  

(4.13)

is valid for CLP, with \( C^s \) and \( \bar{C} \) given as \( C^s = \{ i \in N(e) | h^e(s-1) \leq f_i c_i n < h^e s \} \),
and \( \bar{C} = N(e) \setminus \bigcup_s C^s \).

**Proof.** First, let us denote \( f_i c_i \) by \( a_i \) and \( h^e \) by \( b \). Note that all numbers are integers.
Thus, \( s_i = \min \{ s \in \mathbb{N} | (b - 1) \leq a_i n < b s \} \) is equal to \( s_i = \min \{ s \in \mathbb{N} | (b - 1) \leq a_i n < (b - \epsilon) s \} \) for suitably small \( \epsilon > 0 \). To prove that (4.13) is valid, we
show that it is a Gomory cut (see Dietrich and Escudero [21]) obtained from (4.1c)
with multiplication factor \( n/(b - \epsilon) \). This is clear for the left-hand side coefficients.
It remains to be shown that \( \lceil bn/(b - \epsilon) \rceil = n + 1 \) for all applicable values of \( b \)
and \( n \). Clearly, \( \lceil nb/(b - \epsilon) \rceil \geq n + 1 \). It is also easy to show that \( \lceil nb/(b - \epsilon) \rceil \leq \lceil nb/(b - 1) \rceil \leq n + 1 : \lceil nb/(b - 1) \rceil - (n + 1) = \lceil nb/(b - 1) - (n + 1) \rceil \leq \lceil n/(b - 1) \rceil - 1 \leq 1 - 1 = 0 \). \( \square \)

**Example 4.16.** Consider the following polytope \( S \) of a CLP instance.

\[
S = \{ x \in \{0, 1\}^5 : 3x_1 + 5x_2 + 7x_3 + 10x_4 + 14x_5 \geq 14 \}.
\]

If we consider \( n = 2 \), then \( C^1 = \{1, 2\}, \ C^2 = \{3, 4\} \) and \( \bar{C} = \{5\} \), and thus from
Corollary 4.15 it follows that the inequality

\[
x_1 + x_2 + 2x_3 + 2x_4 + 3x_5 \geq 3
\]

is valid for \( S \).

**Flow cover cuts**

The flow cover (FC) cuts are described in Bussieck [11]. These cuts use the line
structure of CLP, in which lines typically cover more than one track.

**Example 4.17.** Consider the CLP instance shown in Figure 4.4. Assume \( \bar{x} \) is the
optimal solution to the LP relaxation. If we divide the three tracks into two sets,\n\( \{e_0\} \) and \( C = \{e_1, e_2\} \), then we can distinguish two types of feasible solutions:
those containing at least one line using only \( e_0 \), and those in which all lines passing \( e_0 \)
also pass some tracks of \( C \). Thus, either a valid line plan contains a line that uses
only \( e_0 \), or validity implies that there are at least \( h^{e_0} \) carriages per hour being pulled
along the tracks of \( C \).

Conversely, if we extend all lines passing some tracks of \( C \) to also pass \( e_0 \), then
lines that only pass \( e_0 \) merely fill the gap between \( h^{e_0} \) and \( \sum_{e \in C} h^e \). Since this is
Figure 4.4: Example of a flow cover.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$(l_i, f_i, c_i)$</th>
<th>$\bar{x}_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(l_1, 1, 6)$</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>$(l_2, 1, 7)$</td>
<td>1.0</td>
</tr>
<tr>
<td>3</td>
<td>$(l_3, 1, 12)$</td>
<td>0.5</td>
</tr>
<tr>
<td>4</td>
<td>$(l_4, 1, 3)$</td>
<td>0.34</td>
</tr>
</tbody>
</table>

\[(19 - 7 - 8)\bar{x}_3 + 1 \cdot 6\bar{x}_1 + 1 \cdot 7\bar{x}_2 + 2 \cdot 3\bar{x}_4 = 2 + 6 + 7 + 2 = 17 < 19\]

Lemma 4.18. Let $C \subseteq E$, $e_0 \in E \setminus C$ for which $h^{e_0} > \sum_{e \in C} h^e$. Then, the inequality

$$(h^{e_0} - \sum_{e \in C} h^e) \sum_{i \in N(e_0)} x_i + \sum_{e \in C} \sum_{i \in N(e)} f_i c_i x_i \geq h^{e_0}$$

(4.14)

is valid.

Proof. First, suppose $\sum_{i \in N(e_0)} x_i \geq 1$. Since adding the capacity constraints for all tracks in $C$ gives $\sum_{e \in C} \sum_{i \in N(e)} f_i c_i x_i \geq \sum_{e \in E} h^e$, validity is obvious. Next, consider $\sum_{i \in N(e_0)} x_i = 0$. This implies that all lines that pass $e_0$ also use at least one track in $C$, and therefore

$$h^{e_0} \leq \sum_{i \in N(e_0)} f_i c_i x_i \leq \sum_{e \in C} \sum_{i \in N(e)} f_i c_i x_i.$$

Corollary 4.19 identifies redundant choices for $e_0$ and $C$, which is useful for the separation algorithm, as discussed in §4.4.2.

Corollary 4.19. Consider an FC inequality defined by the pair $(e_0, C)$ with $e_0 \in E$ and $C \subseteq E \setminus e_0$, where the capacity constraint for $e_0$ has been strengthened using the techniques described in §4.3.1. Flow cover inequalities without lines crossing both $e$ and part of $C$, i.e., $\{l : e_0 \in l, l \cap C \neq \emptyset\} = \emptyset$, cannot cut off any feasible solution of the LP relaxation.

Proof. Strengthening the capacity constraint for $e_0$ implies that for all variables $i$ using $e_0$, $f_i c_i \leq h^e$ holds. Therefore, $\sum_{i \in N(e_0)} x_i \geq 1$. Now, since $\{l : e_0 \in l, l \cap C = \emptyset\} = \{l : e_0 \in l, l \cap C \neq \emptyset\}$, this implies that $\sum_{i \in N(e_0)} x_i \geq 1$. We can thus prove this corollary along the same lines as used in the proof of Lemma 4.18.
4.3.3 Tree search

In this section we consider several problem-specific branching rules and primal heuristics.

Branching

As reported in Linderoth and Savelsbergh [47], implementing problem-specific branching rules can significantly reduce the size of the branching tree and speed up the solution process. In general, the applied branching rule for binary problems is to branch on a single variable, creating two new subproblems. In this section we discuss several alternative rules.

Branching rules are used to split a problem into several, usually two, new subproblems. Similar to cutting planes in a class of cuts, every class of branching rules contains many possible branching instances. For example, for the class of variable branching, a branching instance is a variable. The quality of a branching instance in our minimisation problem is measured by the lowest bound of the new subproblems. The branching instance for which this lowest bound is maximal is considered best (see §4.4.3).

Variable branching Using the solution to the LP relaxation $\bar{x}$ at a subproblem, the choice of a branching variable is made by taking the variable for which $x_j$ is closest to $\frac{1}{2}$. In general, ties are broken by considering the variable for which the objective coefficient is largest.

Generalised upper bounds and special ordered set branching An alternative class of branching rules for problems containing generalised upper bound (GUB) constraints of the form

$$\sum_{i \in C} x_i \leq 1 \quad (4.15)$$

for $C \subseteq N$ with $x \in \{0, 1\}^{|N|}$, is to branch on such a GUB constraint (see Linderoth and Savelsbergh [47]).

For any feasible solution, (4.15) ensures that at most one of the variables in $C$ can be set to 1. Therefore, in general, a valid branching scheme would be to create $K$ branches in which subsequently for $k \in \{1, \ldots, K\}$ the variables in subsets $\hat{C}_k$ are forced to zero, or

$$x_i = 0 \quad \text{for all } i \in \hat{C}_k \quad (4.16)$$

The sets $\hat{C}_k$ are subsets of $C$ and must be chosen such that $\bigcup_{k \in K} C \setminus \hat{C}_k \supseteq C$. In most applications of GUB branching, only two new subproblems are created ($K = 2$). Now, instead of fixing a variable to zero in all branches, we can use (4.15) to split the problem as follows

$$\sum_{i \in \hat{C}} x_i = 1 \quad \text{versus} \quad x_i = 0 \text{ for all } i \in \hat{C} \quad (4.17)$$
This splitting for $K = 2$ is stronger than using (4.16) and can be used for the set $\mathcal{C}$ and $\mathcal{C} \setminus \mathcal{C}$. The problem is to determine the set $\mathcal{C}$.

In special ordered set (SOS) branching, an ordering of the variables in $\mathcal{C}$ is used to determine $\mathcal{C}$. The variables in the GUB constraints (4.1d) for every line $l$ can be ordered according to their capacity. Then, a subproblem for which the LP solution $\tilde{x}$ is fractional is split into two new problems as in (4.17) using $\mathcal{C} = \{i | l_i = l\}$ and

$$ \mathcal{C} = \{i \in \mathcal{C} | f_i c_i \leq \delta\} \quad \text{where} \quad \delta = \sum_{i \in \mathcal{C}} f_i c_i \tilde{x}_i. \quad (4.18) $$

The value of the parameter $\delta$ is the so-called branching value of the SOS.

**Example 4.20.** Consider a subproblem of the branching tree in which the optimal LP solution $\tilde{x}$ is fractional for some variables of line $l$. More specifically, suppose $\tilde{x}_{(l,1,8)} = \tilde{x}_{(l,1,12)} = \frac{1}{2}$. So, the capacity of line $l$ in this LP solution is $\sum_{i | l_i = l} f_i c_i \tilde{x}_i = 1 \cdot 8 \cdot \frac{1}{2} + 1 \cdot 12 \cdot \frac{1}{2} = 10$ carriages per hour.

**Line branching** Similar to generalised upper bound branching rules, the line branching rule also splits a subproblem into two new subproblems using the GUB constraints (4.1d) on the lines. In line branching, we take $\mathcal{C} = \mathcal{C}$. Thus, the branching dichotomy for some line $l$ is

$$ \sum_{i | l_i = l} x_i = 0 \quad \text{versus} \quad \sum_{i | l_i = l} x_i = 1. $$

The effect of line branching on the new subproblems is two-sided. On one hand, the added restriction itself has a direct effect on the solution to the LP relaxation. On the other hand, the LP problem in the left subproblem becomes smaller because we can remove any variable of line $l$ from the model. The upward branch allows us to obtain a higher value for $Q_{kj}$ for strengthening the coefficients of the constraint matrix. When $\sum_{i | l_i = l} x_i = 1$, then for all variables $j$ that do not belong to line $l$

$$ Q_{kj} = \min_{i | l_i = l} a_{ki} $$

is clearly valid.

**Capacity branching** A subproblem is split into two new problems by taking a set of integer variables $\mathcal{C}$ and enforcing

$$ \sum_{i \in \mathcal{C}} x_i \leq g \quad \text{versus} \quad \sum_{i \in \mathcal{C}} x_i \geq g + 1 $$

for some integer $0 \leq g < |\mathcal{C}|$. The set $\mathcal{C}$ is constructed to contain variables with an identical capacity. For a given track $e$, a capacity $b$ and an integer $g$, the current problem is split using $\mathcal{C} = \{i \in N(e) | f_i c_i = b\}$. Since $x$ is restricted to integer values, the two branches cover the complete solution space.
4.4 IMPLEMENTATION ISSUES

Subproblem selection

For the selection of the next open subproblem to be processed we use the well-known best-first rule. Selecting the best subproblem from the list of open subproblems is done by reviewing the estimates for the objective function value of the LP relaxation. In the simplest case, these estimates are the value of the LP relaxation in the parent problem. These estimates, however, postpone the fathoming of subproblems that may turn out to have an LP lower bound above the value of the current best-known solution. To prevent this, we use the strong branching bounds calculated in the previous section for finding the best branching rule.

Primal heuristic

Our primal heuristic is based on the LP relaxation in a subproblem. From this LP solution \( \hat{x} \), a new instance of CLP is constructed using only the lines \( L' = \{ l \in L \mid \sum_{i \in l} \hat{x}_i > 0 \} \). This significantly smaller problem is given to an off-the-shelf IP solver, while bounding the available CPU time. Clearly, integer solutions to these problems are also feasible integer solutions to the original problem. The algorithm can be terminated when the lower bounds of the reduced problem are higher than the current best-known primal solution of the overall problem, since we are only interested in solutions that are better. Together with the small problem size, this causes the heuristic to terminate well before the time limit is reached in most of our experiments. Our primal heuristic is by default applied at the root node, and every tenth node with a depth in the enumeration tree of at least five.

4.4 Implementation issues

We now discuss the implementation of the preprocessing, cutting planes and branching techniques of the previous sections.

4.4.1 Preprocessing implementation

All preprocessing techniques of §4.3.1 are applied at every subproblem of the enumeration tree. The implementation of the coefficient reduction techniques is not described here, but in §4.4.2.

Variable reduction

The dominance rules for variable reduction are tested between all pairs of variables belonging to the same line. For identifying dominance relations, it suffices to compare the coefficients of both variables only in all (strengthened) service constraints present in the model, not the added cutting planes. By definition, this is sufficient for the validity of the reduction in any optimal integer solution.

Instead of using the strengthened constraints as given by the coefficient reduction techniques to identify variable dominance, we use the following, stronger approach. Consider strengthening the coefficients for every line separately. Strengthening all
service constraints only for some given line will result in valid inequalities that are at least as tight as the previous restrictions. Thus, we can derive dominance relations based on these new constraints. Since removing these variables is also valid for the original system, we can repeat this individual approach for all lines, thereby circumventing the order dependence of the strengthening procedure.

**Constraint reduction**

At the root problem, the constraint reduction technique is applied to all pairs of service constraints \( h \) and \( k \) for tracks that have one vertex in common. Recall from Theorem 4.8 that constraint \( k \) is redundant for the description of GLP if \( b_k \leq Q_{k0}(h) \) for some constraint \( h \). If \( b_k < Q_{k0}(h) \), then constraint \( k \) is not removed, but \( b_k \) is set to \( \lfloor Q_{k0}(h) \rfloor \). In all other subproblems of the enumeration tree, we only apply this technique on constraints \( h \) and \( k \) belonging to the same track.

### 4.4.2 Cutting planes implementation

Next, we describe the separation algorithms to find violated inequalities of the proposed classes of cutting planes. At every iteration of the cutting plane algorithm, the separation algorithms of all classes of cutting planes are called and violated inequalities are added to the LP relaxation. The LP is then re-optimised. These two steps are repeated until no more cuts are found. For all classes of cutting planes we impose a minimum violation of 1% for the valid inequality to be added to the system.

**Coefficient strengthening cuts**

The results of the coefficient strengthening techniques of §4.3.1 depend on the order of strengthening. The later a coefficient is strengthened, the smaller the effect of the strengthening. We describe a heuristic for determining a permutation of the variables. In the root problem, this order is used to alter the coefficients in the constraints. In subsequent subproblems a new cut built up from strengthened coefficients is added to the problem description, replacing the initial constraint. We refer to these cuts as coefficient strengthening (CoS) cuts.

Recall that strengthening the coefficient \( a_{kj} \) of variable \( x_j \) for constraint \( k \) involves finding a valid value for \( Q_{kj}(h) \), for some constraint \( h \). For these \( h-k \) pairs, we only consider the frequency and capacity restrictions on a track \( e \). The reduced coefficient \( \tilde{a}_{kj} \) is then determined by

\[
\tilde{a}_{kj} = \min\{a_{kj}, \max\{0, b_k - Q_{kj}\}\} \quad \text{where} \quad Q_{kj} = \max\{Q'_{kj}(k), Q'_{kj}(h), Q''_{kj}(h)\}.
\]

For every track \( e \), we first strengthen the coefficient in the frequency constraint, then in the capacity constraint.

The strengthening is done in two phases. First, the coefficients of variables with nonzero values in the current solution of the LP relaxation \( \tilde{x} \) are strengthened. The ordering of the variables is done in blocks of the same line. These line blocks are sorted according to \( \sum_{|t_i = t} \tilde{x}_i \), strengthening first those lines for which this number is high. The order of the cuts is given by the sequence of the tracks that are passed by
the line from its origin to its destination station. In the second phase, we reuse this ordering of the lines, but now strengthen all coefficients of the remaining variables.

The resulting CoS cutting planes are added to the LP of the current subproblem. Since all coefficients are either not changed, or strictly smaller than the coefficients in the original constraint, we can safely remove the original constraints from the current LP. From that point on, these new constraints are considered to be the service constraints of the various tracks.

2-Cover cuts

The separation algorithm for the class of 2-cover cuts is straightforward, since there is only one 2C inequality for every track. However, substituting for fixed variables in the service constraints, gives rise to new 2C inequalities. We also use the probing techniques from §4.3.2 for finding new violated 2C inequalities.

Example 4.21. Consider the instance with two tracks shown in Figure 4.5. The displayed fractional solution $\bar{x}$ has two variables at nonzero values, $\bar{x}_1 = 1$ and $\bar{x}_2 = \frac{1}{2}$.

The 2C inequality for track $e$ is $x_1 + 2 \cdot x_2 + \ldots \geq 2$. Clearly, $\bar{x}$ satisfies this 2C cut. Now consider probing on $x_1$. This shows that substituting for $x_1$ in the capacity constraint results in a valid inequality with $h^e = 10 - 5 = 5$. The corresponding 2C cut $2 \cdot x_2 + \ldots \geq 2$ is violated by $\bar{x}$.

For every track $e$, we probe on a set of variables. This set is built in two steps. First, all variables $j$ for which $\bar{x}_j = 1$ and $e \in l_j$ are added. Second, we add one additional variable that results in the maximally violated new 2C cut by considering all $i$ for which $e \in l_i$ and $0 < \bar{x}_i < 1$.

Multi-cover cuts

The probing set $\{i | e \in l_i, \bar{x}_i = 1\}$ is used for finding violated multi-cover cuts for every track $e$. Furthermore, we limit the search for the number of cover intervals to $3 \leq n + 1 \leq \min\{6, h^e - 1\}$. For every track we only add the MC constraint for $n$ for which the violation is maximal.

Flow cover cuts

In general, no efficient separation algorithm is known for the class of flow cover cuts (4.14) (see Bussieck [11]). Therefore, the separation problem for FC cuts is...
solved heuristically. Corollary 4.19 shows that the edge $e_0$ and the edges in the set $C$ should not be too far apart in the graph $G$. Given the solution $\tilde{x}$ to the LP relaxation, it is clear from the proof of Lemma 4.18 that violated FC cuts can only be found for tracks $e_0$ and $C$ for which $0 < \sum_{i \in N(e_0) \cap C = \emptyset} \tilde{x}_i < 1$. Our separation algorithm considers all possible $e_0$ and $C$ that are subsets of $\delta(v)$ for all stations $v \in V$, where $\delta(v)$ is the set of edges for which $v$ is one of the two endpoints. For every combination of $v$ and $e_0$ we add the cut that is violated most by $\tilde{x}$.

### 4.4.3 Branching rules implementation

As mentioned in §4.3.3, we compare different branching instances by using estimates for the resulting increase of the lower bound in the new subproblems. Determining these lower bounds can be done exactly by explicitly solving the new LP relaxations, so-called strong branching, or heuristically by determining estimates for the new objective values. In our implementation we heuristically find one candidate instance for every class of rules, and then use strong branching to decide among these candidates.

#### Special ordered set branching

From all the lines in a given CLP instance, we select the line $l$ with the largest number of fractional variables as the line to branch on. Similar to Linderoth and Savelsbergh [47], we only consider SOS branching on lines with at least three variables at a fractional value. If there is more than one line that attains the largest number of fractional variables, we break ties by choosing the line with the highest weighted objective value $\sum_{i | l_i = l} w_i \tilde{x}_i$. The set $\bar{C}$ is set to be the largest of $\bar{C}$ as in (4.18) and $C \setminus \bar{C}$. Note the possibility of a branching cycle, i.e., that the current solution $\tilde{x}$ remains feasible in one of the new subproblems. This would imply that for this new problem the previously chosen branching rule would again be best. The latter is prevented by ensuring that $\emptyset \neq \bar{C} \neq \{i \in C | \tilde{x}_i > 0\}$.

#### Line branching

We pick the candidate line by calculating estimates for the new LP lower bounds of the new problems arising if this line is chosen to be branched on. These degradation estimates for branching on line $l$ are based on the objective function coefficients for a fractional solution $\tilde{x}$:

$$D^- = \sum_{i | l_i = l} w_i \tilde{x}_i$$

$$D^+ = \frac{1 - \sum_{i | l_i = l} \tilde{x}_i}{\sum_{i | l_i = l} \tilde{x}_i} \sum_{i | l_i = l} w_i \tilde{x}_i$$

where $D^-$ and $D^+$ are the estimates for the left and right branch respectively. From the lines for which $\sum_{i | l_i = l} \tilde{x}_i$ is fractional, we select the line with the highest value for $\min\{D^-, D^+\}$. This rule represents the multiple-variable variant of the standard objective function based degradation estimate.
4.5 Computational results

The effectiveness of the techniques described in the previous sections is examined using five test instances. Characteristics for these instances can be found in Table 4.1. Figure 4.6 shows the networks for SP97AR and SP97IR. The visual representations of the other three instances can be found in Figure B.1 on page 135. The network reduction methods as described in Claessens et al. [18] were already applied to these instances. Note that Bussieck [11] reports on similarly named instances that are, however, not identical to ours. Therefore, comparing results is not useful. The last two characters in the names of the instances give the train type that is considered. All techniques have been implemented in the branch-and-cut framework ABACUS (see Thienel [80] and ABACUS [2]). The linear programming relaxations arising in the subproblems of the branch-and-cut tree are solved by CPLEX 6.6.1. All computations where done on an Intel Pentium III 866 Mhz PC with 128 MB internal memory running Windows 98SE.

First, we consider the effect of the different techniques on the root problem. The results of the preprocessing techniques for reducing the number of variables and constraints are given in Table 4.2. The initial number of constraints reflects the two service constraints for every track.

### Table 4.1: Characteristics of the instances.

<table>
<thead>
<tr>
<th></th>
<th>SP97AR</th>
<th>SP97IC</th>
<th>SP98AR</th>
<th>SP98IR</th>
<th>SP98IC</th>
</tr>
</thead>
<tbody>
<tr>
<td># Stations</td>
<td>141</td>
<td>40</td>
<td>118</td>
<td>44</td>
<td>41</td>
</tr>
<tr>
<td># Tracks</td>
<td>177</td>
<td>52</td>
<td>134</td>
<td>44</td>
<td>46</td>
</tr>
<tr>
<td># Lines</td>
<td>1212</td>
<td>831</td>
<td>913</td>
<td>420</td>
<td>627</td>
</tr>
<tr>
<td>Set F</td>
<td>{1,2,3,4}</td>
<td>{1,2}</td>
<td>{1,2,3,4}</td>
<td>{1,2}</td>
<td>{1,2}</td>
</tr>
<tr>
<td>Set C</td>
<td>{1,...,5}</td>
<td>{3,...,15}</td>
<td>{2,...,10}</td>
<td>{3,...,12}</td>
<td>{3,...,15}</td>
</tr>
</tbody>
</table>

Capacity branching

Given an LP solution \(\bar{x}\) in a subproblem, we choose the track \(e\) with the largest number of fractional variables \(\{|i|e \in l_i, 0 < \bar{x}_i < 1\}\) to be used for capacity branching. We only consider capacity branching for tracks with at least \(f > \) fractional variables. Given a track \(e\) and a branching capacity \(b\), the parameter \(g\) is completely specified, since it must satisfy \(g < \sum_{i \in C} \bar{x}_i < g+1\) to prevent a branching cycle. The estimates for the increase in the objective function value for the left and right branch are now calculated for every capacity \(b\) as

\[
D^- = \frac{\sum_{i \in C} \bar{x}_i - g}{\sum_{i \in C} \bar{x}_i} \sum_{i \in C} w_i \bar{x}_i,
\]

\[
D^+ = \frac{g + \sum_{i \in C} \bar{x}_i}{\sum_{i \in C} \bar{x}_i} \sum_{i \in C} w_i \bar{x}_i,
\]

for both branches respectively. The branching capacity \(b\) is selected to be such that the value \(\min\{D^-, D^+\}\) is highest.
CHAPTER 4. BRANCH-AND-CUT FOR LINE PLANNING PROBLEMS

Table 4.2: Preprocessing results for variable and constraint reduction.

<table>
<thead>
<tr>
<th></th>
<th>SP97AR</th>
<th>SP97IC</th>
<th>SP98AR</th>
<th>SP98IR</th>
<th>SP98IC</th>
</tr>
</thead>
<tbody>
<tr>
<td># Var. initially</td>
<td>24240</td>
<td>21606</td>
<td>32868</td>
<td>8400</td>
<td>16320</td>
</tr>
<tr>
<td># Con. initially</td>
<td>354</td>
<td>104</td>
<td>268</td>
<td>88</td>
<td>92</td>
</tr>
<tr>
<td># Var. after §4.4.1</td>
<td>14101</td>
<td>12497</td>
<td>15065</td>
<td>3651</td>
<td>10894</td>
</tr>
<tr>
<td># Con. after §4.4.1</td>
<td>181</td>
<td>60</td>
<td>191</td>
<td>65</td>
<td>63</td>
</tr>
<tr>
<td>Size reduction</td>
<td>70%</td>
<td>67%</td>
<td>67%</td>
<td>68%</td>
<td>54%</td>
</tr>
</tbody>
</table>

Table 4.3: The objective function values of the best solution along with the different LP relaxations at the root problem when applying only a specific class of cutting planes.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Mean</th>
<th>St.Dev.</th>
<th>After 1h.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>3.8</td>
<td>4.8</td>
<td>4465 (42%)</td>
</tr>
<tr>
<td>SOS</td>
<td>2.0</td>
<td>4.0</td>
<td>4458 (29%)</td>
</tr>
<tr>
<td>Line</td>
<td>3.6</td>
<td>3.4</td>
<td>4464 (40%)</td>
</tr>
<tr>
<td>Cap</td>
<td>16.5</td>
<td>27.2</td>
<td>4451 (15%)</td>
</tr>
</tbody>
</table>

Table 4.4: Statistics for the increase in the lower bound of SP98IC for different branching rules.
In Table 4.3 we see the effect of the different classes of cutting planes on the lower bound at the root node. The percentages between brackets indicate what fraction of the gap between the best-known upper bound and the initial lower bound is closed. The best-known upper bounds are obtained from the branch-and-cut process. It is clear that at least for the root node the coefficient strengthening cuts are superior. The effects of the other classes of cuts are comparable. The combined results—for all classes of cuts together—still show a large increase in the percentage of gap closed compared to the individual results.

Before presenting the branch-and-cut results, let us also review the three proposed branching rules of §4.4.3 together with the standard variable branching. To analyse the effect of a specific branching rule, we keep track of the increase in the objective function value due to the application of the branching rule, i.e., before adding new cutting planes. All rules have been tested on SP98IC, with a maximum computation time of one hour and applying all classes of cutting planes. Table 4.4 shows both the average increase and the standard deviation when applying the variable branching rule ("Variable"), the special ordered set branching ("SOS"), the line branching ("Line"), or the capacity branching rule ("Cap"). The total effect of a branching rule in the tree search is measured by the increase of the overall lower bound at the end of the hour, relative to the best lower bound from the branch-and-cut process (see Table 4.5).

A low standard deviation compared to the average increase is necessary for obtaining a balanced enumeration tree (see Linderoth and Savelsbergh [47]). Table 4.4 illustrates this. The capacity branching rule has by far the highest average increase. However, since the standard deviation of the increase for this rule is also high, the tree will be far from balanced. As predicted, the line branching rule combines both a high mean increase with a relatively low standard deviation. The slightly better performance of the variable branching rule can be explained by the higher number...
of processed subproblems, due to the simplicity of this branching rule.

The branch-and-cut algorithm was tested on the five instances using both different classes of cutting planes, and different branching rules. The results for all 55 problems are shown in Table 4.5. The cutting planes are tested using all classes, only the coefficient strengthening cuts and the coefficient strengthening cuts combined with either the FC cuts, the 2C cuts or the MC cuts. All combinations of cuts are tested using only variable branching ("V"), and using variable branching combined with line branching ("V & L"). When applying all classes of cuts, we additionally tested all four branching rules ("All") at the same time. The last column shows the performances of the ILP solver of CPLEX 6.6.1 for every instance, without applying any of the techniques described in this chapter.

We enforced a maximum computation time of two hours. Table 4.6 shows results that were obtained without imposing any time constraints. We will come back to these results at the end of this section. Computation times are only reported when an instance is solved within this time bound. For every problem the table shows the best feasible solution that was found ("UB"), the remaining overall lower bound ("LB"), the implied gap ("Gap"), the total number of created nodes in the tree ("# Nodes") and the number of nodes that has already been processed ("# Done"). The best upper and lower bounds for every instance are typed in bold. The only instance that can be solved to optimality is SP98IR. An optimal solution for his version of this instance was already found by Bussieck [11].

From Table 4.5 it is clear that the best lower bounds within two hours are obtained using all classes of cuts simultaneously, even though the number of subproblems that can be processed in this time is lower compared to using only a subset of the classes. Again we see that the effect of using line branching on the balance of the enumeration tree does not outweigh the larger number of subproblems that can be processed when using only variable branching. This is illustrated best by SP98IR. Solving this problem to optimality using variable branching requires more than 20% more nodes, yet the solution time is lowest.

The differences in the reported gaps after two hours are not only a result of the differences in the overall lower bounds. The best-known upper bounds also differ significantly for some of the instances. This can best be explained by the fact that these solutions are often found using the primal heuristic of §4.3.3. Since the LP solution is input for the heuristic, results can vary when different classes of cutting planes are used.

Finally, the results in Table 4.5 show that our branch-and-cut algorithm outperforms the off-the-shelf ILP solver of CPLEX. For all 5 instances and for all the tested combinations of cutting planes, both the upper and lower bounds are significantly better than those obtained with CPLEX. For example, instance SP98IR, which can be solved to optimality in around 3 minutes, cannot be solved by CPLEX within the time limit of two hours.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SP97AR</td>
<td>6924</td>
<td>6841</td>
<td>6924</td>
<td>6919</td>
<td>6901</td>
<td>6728</td>
<td>6949</td>
<td>6949</td>
<td>6728</td>
</tr>
<tr>
<td></td>
<td>6547</td>
<td>6551</td>
<td>6547</td>
<td>6534</td>
<td>6531</td>
<td>6550</td>
<td>6537</td>
<td>6536</td>
<td>6536</td>
</tr>
<tr>
<td>Gap</td>
<td>5.76%</td>
<td>4.43%</td>
<td>5.76%</td>
<td>5.89%</td>
<td>5.67%</td>
<td>2.72%</td>
<td>4.92%</td>
<td>6.30%</td>
<td>8.02%</td>
</tr>
<tr>
<td># Nodes</td>
<td>633</td>
<td>923</td>
<td>665</td>
<td>1485</td>
<td>1285</td>
<td>1293</td>
<td>1027</td>
<td>1005</td>
<td>803</td>
</tr>
<tr>
<td># Done</td>
<td>321</td>
<td>461</td>
<td>332</td>
<td>742</td>
<td>643</td>
<td>646</td>
<td>514</td>
<td>502</td>
<td>402</td>
</tr>
<tr>
<td>SP97IC</td>
<td>4308</td>
<td>4304</td>
<td>4337</td>
<td>4314</td>
<td>4302</td>
<td>4325</td>
<td>4309</td>
<td>4319</td>
<td>4316</td>
</tr>
<tr>
<td></td>
<td>4244</td>
<td>4244</td>
<td>4244</td>
<td>4228</td>
<td>4230</td>
<td>4236</td>
<td>4232</td>
<td>4232</td>
<td>4232</td>
</tr>
<tr>
<td>Gap</td>
<td>1.51%</td>
<td>1.41%</td>
<td>2.19%</td>
<td>2.03%</td>
<td>1.70%</td>
<td>2.10%</td>
<td>1.72%</td>
<td>2.06%</td>
<td>1.82%</td>
</tr>
<tr>
<td># Nodes</td>
<td>1176</td>
<td>1519</td>
<td>1029</td>
<td>2503</td>
<td>2473</td>
<td>2291</td>
<td>2425</td>
<td>1633</td>
<td>1271</td>
</tr>
<tr>
<td># Done</td>
<td>616</td>
<td>760</td>
<td>514</td>
<td>1252</td>
<td>1236</td>
<td>1145</td>
<td>1213</td>
<td>817</td>
<td>636</td>
</tr>
<tr>
<td>SP98AR</td>
<td>5438</td>
<td>5380</td>
<td>5311</td>
<td>5308</td>
<td>5307</td>
<td>5312</td>
<td>5317</td>
<td>5439</td>
<td>5459</td>
</tr>
<tr>
<td></td>
<td>5262</td>
<td>5263</td>
<td>5261</td>
<td>5243</td>
<td>5244</td>
<td>5252</td>
<td>5255</td>
<td>5252</td>
<td>5254</td>
</tr>
<tr>
<td>Gap</td>
<td>3.34%</td>
<td>2.22%</td>
<td>0.95%</td>
<td>1.24%</td>
<td>1.20%</td>
<td>1.14%</td>
<td>1.18%</td>
<td>3.56%</td>
<td>3.90%</td>
</tr>
<tr>
<td># Nodes</td>
<td>635</td>
<td>929</td>
<td>707</td>
<td>2074</td>
<td>1655</td>
<td>1819</td>
<td>1479</td>
<td>1019</td>
<td>757</td>
</tr>
<tr>
<td># Done</td>
<td>340</td>
<td>464</td>
<td>354</td>
<td>1039</td>
<td>828</td>
<td>910</td>
<td>739</td>
<td>510</td>
<td>378</td>
</tr>
<tr>
<td>SP98IR</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
</tr>
<tr>
<td></td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
<td>2182</td>
</tr>
<tr>
<td>Gap</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.23%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td># Nodes</td>
<td>236</td>
<td>293</td>
<td>239</td>
<td>5821</td>
<td>9033</td>
<td>665</td>
<td>689</td>
<td>1601</td>
<td>2827</td>
</tr>
<tr>
<td># Done</td>
<td>236</td>
<td>293</td>
<td>239</td>
<td>5821</td>
<td>6503</td>
<td>665</td>
<td>689</td>
<td>1601</td>
<td>2827</td>
</tr>
<tr>
<td>Time</td>
<td>0:03:00</td>
<td>0:02:38</td>
<td>0:03:23</td>
<td>0:54:12</td>
<td>-</td>
<td>0:03:00</td>
<td>0:03:59</td>
<td>0:14:28</td>
<td>0:31:52</td>
</tr>
<tr>
<td>SP98IC</td>
<td>4501</td>
<td>4499</td>
<td>4495</td>
<td>4518</td>
<td>4511</td>
<td>4506</td>
<td>4508</td>
<td>4510</td>
<td>4503</td>
</tr>
<tr>
<td></td>
<td>4469</td>
<td>4470</td>
<td>4469</td>
<td>4441</td>
<td>4438</td>
<td>4465</td>
<td>4465</td>
<td>4446</td>
<td>4445</td>
</tr>
<tr>
<td>Gap</td>
<td>0.72%</td>
<td>0.65%</td>
<td>0.58%</td>
<td>1.73%</td>
<td>1.64%</td>
<td>0.92%</td>
<td>0.96%</td>
<td>1.44%</td>
<td>1.30%</td>
</tr>
<tr>
<td># Nodes</td>
<td>1488</td>
<td>2567</td>
<td>1685</td>
<td>2833</td>
<td>2923</td>
<td>2827</td>
<td>2921</td>
<td>2493</td>
<td>1941</td>
</tr>
<tr>
<td># Done</td>
<td>785</td>
<td>1300</td>
<td>844</td>
<td>1418</td>
<td>1461</td>
<td>1428</td>
<td>1467</td>
<td>1249</td>
<td>970</td>
</tr>
</tbody>
</table>

Table 4.5: Computational results.
4.5.1 Unbounded computation times

Even though the remaining integrality gaps for four of the tested instances are very small, the imposed time bound of two hours is too tight for them to be solved to optimality. To know the true optimal values for these problems, we have tested them also without time restrictions. These tests were done using a cut-and-branch approach. We first applied all preprocessing and strengthening methods described in the previous sections, and then let CPLEX 7.5\textsuperscript{1} attempt to solve them. In addition, the best-known solutions from the branch-and-cut process were given to the solver before starting. The results are presented in Table 4.6.

All instances, except SP98IC, were tested on a hyperthreaded Intel XEON 2.20GHz machine, with 2 Gigabyte RAM running Linux kernel version 2.4.19. The instance SP98IC was run on an AMD Athlon XP 2700+ with 1 Gigabyte of RAM, running Linux kernel 2.4.18. As shown, only the instance SP97AR was not solved to optimality, even after 477 CPU hours. The branching tree for this instance was still growing rapidly, and therefore the process was stopped. For all other instances, the best-known solutions from our branch-and-cut algorithm turned out to be within 0.6% of optimality.

4.6 Summary and conclusions

In this chapter we have outlined a branch-and-cut approach for solving the problem of allocating lines to passenger flows. Two integer programming models were given for this problem in Claessens et al. [18] and Bussieck [11]. Where these previous papers use branch-and-bound and cut-and-branch respectively, to solve the models, we have switched to branch-and-cut. The algorithm is described by introducing several classes of preprocessing rules (§4.3.1), cutting planes (§4.3.2) and branching rules (§4.3.3).

The developed techniques have been tested on five real-life instances of NSR. From these tests we conclude that the described techniques perform well on practical instances, and significantly better than the ILP solver of CPLEX 6.6.1. The preprocessing techniques considerably reduce the size of the initial problem and the mentioned classes of cutting planes effectively strengthen the LP lower bounds. Combined with the primal heuristic, we are not only able to obtain excellent lower bounds.

\textsuperscript{1}At the time of these experiments CPLEX was upgraded to version 7.5.
but also to find good primal solutions in reasonable time. Of the five test instances, we can prove to have the optimal solution of one instance. Of the remaining 4 instances, two were provably solved to within 1%, and two to within 3% of optimality within two hours. Without time restrictions, all instances except SP97AR could be solved to optimality.

Chapter 5

The multiple-type line planning problem

The line planning models described in the literature, including Chapter 4, consider decomposed line planning problems in which all lines are of only one type. In such models it can be assumed without loss of generality that the lines built at all stations along their routes. Solving problems with more than one type was done by a priori assigning the passengers to the different train types (see Citrogge [27]). In this chapter we introduce a generalisation of these models, for simultaneously solving mixed-optimising line planning problems for multiple train types.

The next section introduces new definitions for the multi-type model. In §5.2 we discuss how to formulate the multi-type model by using an intermediate problem, called the edge capacity problem. For this problem we consider a number of model formulations in §6.3. Apart from a multi-commodity flow formulation, we develop two alternative mathematical formulations and prove their equivalence. In §5.4 we describe a computational study, based on instances of NSR. The results in this chapter are based on Goossens et al. [37].

§6.1 Modelling

The essential difference between CLP and the multi-type problem MCLP is the interaction of the different systems, and thus the freedom of the model to distribute the passengers over the trains of the different types. Clearly, this can lead to better solutions. However, as mentioned in Chapter 4, CLP is an NP-hard optimisation problem. Since MCLP is a generalisation of CLP, it follows that MCLP is also NP-hard.

Although CLP and MCLP are very similar in concept, there are several important new aspects that have to be covered. As an example, a new modelling issue is MCLP is to ensure that passengers are not assigned to trains that pass their destination station without stopping there.
Table 4.6: Cut-and-branch without time restrictions.

4.5.1 Unbounded computation times

Even though the remaining integrality gaps for four of the tested instances are very small, the imposed time bound of two hours was too tight for them to be solved to optimality. To know the true optimal values for these problems, we have tested them also without time restrictions. These tests were done using a cut-and-branch approach. We first applied all preprocessing and strengthening methods described in the previous sections, and then let CPLEX 7.5 attempt to solve them. In addition, the best known solutions from the branch-and-cut process were given to the solver before starting. The results are presented in Table 4.6.

All instances, except SP955C, where tested on a hyperthreaded Intel XEON 2.2GHz machine with 2 Gigabytes RAM running Linux kernel version 2.4.19. The instance SP955C was run on an AMD Athlon XP 2700+ with 1 Gigabyte of RAM running Linux kernel version 2.4.19. As shown, only the instance SP977R was not solved to optimality even after 477 CPU hours. The branching time for this instance was 477 hours, compared to 11 hours for the rest of the instances. In all other instances, the best known solutions from our branch-and-cut algorithm turned out to be within 0.0% of optimality.

4.6 Summary and conclusions

In this chapter we have continued a branch-and-cut approach for solving the problem of allocating lines to passenger cases. Two integer programming models were given for this problem in Ciesielski et al. [10] and Biesiadecki [11]. Where these previous papers form branch-and-bound and cut-and-bound respectively, to solve the models we have developed to branch-and-cut. The algorithm is described by introducing several classes of preprocessing rules (4.5.1), cutting planes (4.5.2) and branching rules (4.6.1).

The development techniques have been tested on five real-life instances of NSE. From these tests we conclude that the described techniques perform well on practical instances, and significantly better than the LP solver of CPLEX 7.5.1. The preprocessing techniques considerably reduce the size of the initial problem and the mentioned classes of cutting planes effectively strengthen the LP lower bounds. Combined with the primal heuristic, we are not only able to obtain excellent lower bounds.

1 At the time of these experiments CPLEX was upgraded to version 7.5.
Chapter 5

The multiple-type line planning problem

The line planning models described in the literature, including Chapter 4, consider decomposed line planning problems in which all lines are of only one type. In such models, it can be assumed without loss of generality that the lines halt at all stations along their route. Solving problems with more than one type was done by a priori assigning the passengers to the different train types (see Oltrogge [63]). In this chapter we introduce a generalisation of these models, for simultaneously solving cost-optimising line planning problems for multiple train types.

The next section introduces new definitions for the multi-type model. In §5.2 we discuss how to formulate the multi-type model by using an intermediate problem, called the edge capacity problem. For this problem we consider a number of model formulations in §5.3. Apart from a multi-commodity flow formulation, we develop two alternative mathematical formulations and prove their equivalence. In §5.4 we describe a computational study, based on instances of NSR. The results in this chapter are based on Goossens et al. [37].

5.1 Modelling

The essential difference between CLP and the multi-type problem MCLP is the integration of the different systems, and thus the freedom of the model to distribute the passengers over the trains of the different types. Clearly, this can lead to better solutions. However, as mentioned in Chapter 4, CLP is an $\mathcal{NP}$-hard optimisation problem. Since MCLP is a generalisation of CLP, it follows that MCLP is also $\mathcal{NP}$-hard.

Although CLP and MCLP are very similar in concept, there are several important new aspects that have to be covered. As an example, a new modelling issue in MCLP is to ensure that passengers are not assigned to trains that pass their destination station without stopping there.
To arrive at a formulation for MCLP, we are going to present a simplified version of this problem, called the edge capacity problem. Where MCLP decides on the provided capacities for the lines, in the edge capacity problem the decision variables model the supplied capacities for individual edges. First, we introduce several new concepts and notations.

5.1.1 Definitions and notation

As before, consider a network graph $G = (V, E)$. To later distinguish between different kinds of edges, we refer to the edges in the network graph as network edges. On this graph we define commodities $k = (v^k, w^k) \in V \times V$ that can be seen as travellers that want to travel from their source station $v^k$ to their destination $w^k$. A commodity is not allowed to use just any arbitrary path through the network. Instead, every commodity $k$ is restricted to use the network edges of a given simple path $P_k \subseteq E$ between $v^k$ and $w^k$. This is comparable to the restriction enforced by the ticket regulations. In general this route is the shortest path. The assumption that there is exactly one fixed route is not important. The essence is that the route is known for every traveller. The demand for commodity $k$, i.e., the number of travellers that want to travel from $t^k$ to $t^*$, is given by its entry $H_{t^k t^*}$ in the square demand matrix $H$. This demand for commodity $k$ is also denoted by $H^k$. The OD matrix $H$ is assumed to be symmetric. The developed models can, however, easily be adapted to suit instances for which this assumption does not hold.

Every vertex in the network graph is of a certain type. A similar categorisation is also made for the train lines that will be operated on the network. The route of a train line through this network is a path-shaped collection of connected tracks. The type of a train line determines the stations along the line's route at which the line halts. Train lines of type 1, for example, halt at all stations they pass. Lines of type 2 skip the small stations of type 1, etc. In general, a train line of type $t$ halts at all stations $n$ along its route with a type $t_n > t$.

Example 5.1. Consider the network graph in Figure 5.1. The network is described by the connected graph $G = (V, E)$, where $V = \{a, b, c, d, v, w\}$, and $E = \{(a, b), (b, c), (c, d), (d, v), (v, w)\}$. The type of a station is given by the number below its vertex. Thus, $t_a = 1$, $t_b = 3$, etc. For this graph $G$, we have defined three lines of different types. Line 1 of type 1, going from station $a$ to station $w$, halting at all stations in between. Line 2 of type 2, from station $b$ to station $w$, halting only at stations $c$ and $v$. Line 3 of type 3 that does not halt at any station, apart from its origin station $b$ and destination station $v$. The halting stations are also shown by the vertical dashes in the lines that represent the routes of the train lines.

Notice in Example 5.1 that travellers using line 3 of type 3 to travel from $b$ to $v$ will not halt at any of the stations in between. We could thus introduce an edge $\{b, v\}$ of type 3 to show that, due to the types of the stations in between $b$ and $v$, train lines of type 3 will not stop at any of the stations between $b$ and $v$. That is, they will use edge $\{b, v\}$ instead. In general, we construct from the network graph $G$ its "type" graph $G^T = (V, E^T)$. With an identical set of vertices, the difference between $G$
and $G^T$ lies in the set of edges. In the type graph we introduce $T_{\text{max}}$ sets of edges, and we refer to these edges as type edges.

**Example 5.2.** The network graph $G$ given in Figure 5.1 can be associated with the type graph $G^T = (V, E^T)$ displayed in Figure 5.2. Note that the structure of the type graph depends on the types of the stations, not of the lines.

The mapping of the edges in the type graph $G^T$ to the original network edges in $G$ is done through the definition of the route of a type edge $e$. The route $R(e) \subseteq E$ is the simple path in the original network graph that is covered by the type edge $e$. So, in the example above, with type 3 edge $\{b, v\}$, we have that $R(\{b, v\}) = \{\{b, c\}, \{c, d\}, \{d, v\}\}$. The edge set $E^T_t$ contains all type edges of type $t$. The overall set of type edges $E^T$ of the type graph is the union of all the sets of edges of a type $t$, so $E^T = \bigcup_{t \in T} E^T_t$. We assume that the lowest set of type edges is equal to the set of network edges, i.e., $E^T_1 = E$. Every edge $e$ of type $t$, i.e., every $e \in E^T_t$ satisfies that its route $R(e)$ contains only internal vertices $i$ for which $t_i < t$. We say that a type edge $g$ is covered by a type edge $f$ if $R(g) \subseteq R(f)$. Again, e.g., in Example 5.2 the type edge $\{c, v\}$ is covered by the type edge $\{b, v\}$.

The sets $E^T_t$ for all $t \in T$, together with the corresponding routes of the type edges, are part of the problem input. In most cases the route for any pair of nodes $v$ and $w$ describes the shortest path from $v$ to $w$. The sets $E^T_t$ need not be exhaustive, i.e., not every pair of vertices $v, w \in V$ for which there exists a simple path has to be present in $E^T_t$. The graph $G^T$ can be a multigraph in the sense that some type edge $\{v, w\} \in E^T_t$ and $\{v, w\} \in E^T_{t'}$ for $t \neq t'$. This is shown in Example 5.2 by the type edges $\{b, c\}$ and $\{v, w\}$. 
We assume that the route definitions are consistent, i.e., if the route \( R(e) \) of type edge \( e = \{i, j\} \in E^T \) contains two vertices \( v \) and \( w \) for which there exists a type edge \( f = \{v, w\} \in E^T \), then also \( R(f) \subseteq R(e) \). This is illustrated in Figure 5.2 by the edge \( e = \{b, v\} \) of type 3, and the type 2 edge \( f = \{c, v\} \). In addition, we assume that if there exists an edge \( g \) of type \( t > 1 \) whose route contains a network edge \( e \), then there is also a type edge \( f \) of type \( t - 1 \) whose route is contained in that of \( g \), and that also covers \( e \): \( e \subseteq R(f) \subseteq R(g) \). It would be the same to assume that the complete route \( R(g) \) of \( g \) can be covered by the routes of edges of type \( t - 1 \), that are all contained in \( R(g) \). Thus, for the type edge \( g = \{b, v\} \) we assume the presence of the type 2 edges \( \{b, c\} \) and \( \{c, v\} \). Both of these assumptions are not very restrictive.

Using the edges of the type graph, we introduce the set \( P_k^T \subseteq E^T \) of type edges for every commodity \( k \). These paths consist of the type edges that make up the best (highest type) possible route across a commodity’s path \( P_k \) from \( v^k \) to \( w^k \). Formally, for all type edges \( e \in E^T_k \) it should hold that

\[
e \in P_k^T \iff R(e) \subseteq P_k \text{ and } \#t' > t : \exists f \in E^T_{t'} : R(e) \subseteq R(f) \subseteq P_k
\]  

(5.1)

Hence, in the graph in Figure 5.2 the best-edge path for \( k = (a, w) \) is \( P_k^T = \{\{a, b\}, \{b, v\}, \{v, w\}\} \) where \( \{a, b\} \in E^T_3 \), \( \{b, v\} \in E^T_2 \) and \( \{v, w\} \in E^T_2 \).

### 5.2 Formulating the multi-type line planning problem

This section describes the first step in extending the formulation for CLP to model multiple line types simultaneously in the MCLP.

Again, every line \( l \in L \) corresponds to a simple path through the network graph \( G = (V, E) \). The halting pattern of a line \( l \) according to its type \( t(l) \in T \) follows the type edges of the same type in the type graph \( G^T \). Therefore, \( l \) is said to “use” a simple path of type edges, namely the path of all type edges \( e \in E^T_t \) for which \( R(e) \subseteq l \). As in CLP, we have to decide for every line whether to deploy it and, if so, at what hourly frequency, and with how many carriages per train. This is done using the decision variables \( x_i \) for \( i = (l_i, f_i, c_i) \in N \) that model

\[
x_i = \begin{cases} 
1 & \text{if line } l_i \text{ is operated at frequency } f_i, \text{ with } c_i \text{ carriages per train} \\
0 & \text{otherwise.} 
\end{cases}
\]  

(5.2)

Now, however, the possible frequencies and number of carriages of a line depend on its type. Valid frequencies and capacities of lines of type \( t \) are given by \( F(t) \subseteq \mathbb{N} \), and \( C(t) \subseteq \mathbb{N} \) respectively. The set \( N \) of triples is now defined as \( N = \{(l, f, c)|l \in L, f \in F(t(l)), c \in C(t(l))\}\).

On every type edge \( e \) in the network, the total capacity provided by all lines that use a type edge \( e \) is given by

\[
\sum_{i \in N|l_i \text{ uses } e} \lambda(t_{l_i}) f_i c_i x_i,
\]  

(5.3)
where $\lambda(t)$ represents the capacity, in number of passengers, of one carriage of lines of type $t$. If a line is selected to be in the line plan at a certain configuration, then it thus provides capacity along all the edges in the type graph $G^T$ that it uses. Note that, in contrast to CLP, we will not consider any lower bound restrictions on the number of train connections between adjacent stations.

Instead of formulating MCLP directly, we consider a simplified problem called the edge capacity problem (ECP). The ECP is described on the network graph $G$ and the associated type graph $G^T$. The problem is to assign enough capacity to individual edges in the type graph $G^T$, such that all commodities can be transported simultaneously, while minimising some objective function of the allocated capacity. As such, MCLP is a generalisation of ECP.

We are going to present several formulations of ECP in which variables $x(e) \in C$ are used to represent the amount of capacity that is assigned to type edge $e$. Once we have formulated ECP, then MCLP can be formulated by substituting (5.3) of MCLP for every variable $x(e)$ of ECP. In addition, the domain restrictions $x(e) \in C$ have to be replaced by

$$\sum_{i \in N | \lambda_i = l} x_i \leq 1 \quad \text{for all } l \in L$$

$$x_i \in \{0, 1\} \quad \text{for all } i \in N$$

as in (4.1d) and (4.1e) in Chapter 4. The objective function $\sum_{e \in E^T} f(x(e))$ of ECP can be replaced by $\sum_{i \in N} w_i x_i$ for MCLP, where the weights $w_i$ are defined as in (4.1a).

### 5.3 Formulations for the edge capacity problem

We now present three different formulations for ECP.

#### 5.3.1 The multi-commodity flow formulation

Let us introduce two directed graphs, similar to the network graph and the type graph. First, $D = (V, A)$ is constructed from the network graph $G$ using the arc set $A$ which contains a forward arc $(i, j)$ and a backward arc $(j, i)$ for every network edge $\{i, j\} \in E$. Second, the directed graph $D^T = (V, A^T)$ is built similarly from the undirected type graph $G^T$ by replacing every type edge in $E^T$ by two opposing arcs in $A^T$. For dealing with these directed graphs we define $R(a) \subseteq A$ as the directed simple path for an arc $a = (i, j) \in A^T$ similar to $R(e)$ for the corresponding type edge $e = \{i, j\} \in E^T$. The prescribed path $P_k \in E$ for commodity $k$ in the original graph is represented by the directed simple path $\tilde{P}_k \subseteq A$.

In general, a feasible multi-commodity flow satisfies the flow conservation constraints

$$\sum_{j \mid (i,j) \in A^T} F^k_{ij} - \sum_{j \mid (j,i) \in A^T} F^k_{ji} = b^k_i \quad \text{for all } i \in V \text{ and } k \in V \times V$$
where the flow variables $F^k_{ij}$ represent the number of passengers of the commodity $k$ that use arc $(i, j) \in A^T$ through the directed type graph $D^T$. The right-hand sides $b^k_i$ are chosen such that

$$b^k_i = \begin{cases} H v^k w^k & \text{if } i = v^k, \\ 0 & \text{if } v^k \neq i \neq w^k, \\ -H v^k w^k & \text{if } i = w^k. \end{cases}$$

The MCF can thus be formulated as

$$\min \sum_{e \in E^T} f(x(e))$$

s.t. $x(e) \geq \sum_k F^k_{ij}$ for all $t \in T$ and $(i, j) \in A^T, e = \{i, j\} \in E^T$ (5.4b)

$$\sum_{j \mid (j, i) \in A^T} F^k_{ij} - \sum_{j \mid (j, i) \in A^T} F^k_{ji} = b^k_i$$ for all $i \in V$ and $k \in V \times V$ (5.4c)

$$F^k_{ij} = 0$$ for all $k \in V \times V$ and $(i, j) \in A^T : \bar{R}((i, j)) \notin \bar{P}_k$ (5.4d)

$$F^k_{ij} \in \mathbb{N}$$ for all $k \in V \times V$ and $(i, j) \in A^T$ (5.4e)

$$x(e) \in C$$ for all $e \in E^T$. (5.4f)

From the construction of the directed type graph $D^T$ it is evident that there is an exact 1-to-2 relation between an edge $e = \{i, j\} \in E^T$ for some type $t$, and a pair of arcs $(i, j)$ and $(j, i)$, both in $A^T$ (and vice-versa). This relation is used in constraints (5.4b) to enforce that the capacity assigned to type edge $e, x(e)$, is at least as large as the flow across both related arcs. The combined capacity of all lines connecting two stations should be at least as large as the flow in either direction. Consider, for example, a network with two stations $v$ and $w$. If 50 people want to travel from $v$ to $w$, and 60 from $w$ to $v$, then the combined capacity of the lines that connect $v$ and $w$ should be at least $\max\{50, 60\} = 60$.

The restrictions (5.4c) are the flow conservation constraints for every vertex. Restrictions (5.4d) enforce that travellers between $a$ and $b$ have to travel using arcs that are within their predetermined path $\bar{P}_{ab}$. In the directed type graph $D^T$, we thus restrict $k$ to use only arcs $(i, j)$ for which $\bar{R}((i, j)) \subseteq \bar{P}_k$. Finally, the set of feasible values for $x(e)$ is given by the set $C \subseteq \mathbb{N}$, which represents the possible capacities of edges.

We will now describe two lemmas that will be used to preprocess problem instances, and to prove the equivalence of alternative formulations. Let us first show that a commodity $k = (n, m)$ can be split into a number of partial commodities if its path $\bar{P}_k^T$ consists of more than one arc. Every feasible flow for these partial commodities can be recombined to a feasible flow for the original commodity $k$, while the reverse also holds.

**Example 5.3.** Let us preview the commodity decomposition principle on the network graph $G$ and the type graph $G^T$ used in Example 5.2. Figure 5.3 first of all shows
the directed graph $D^T$ based on $G^T$. In addition, it also shows how the commodity $k = (a, w)$ and its best path $\vec{P}_k^T$ are decomposed from $\vec{P}_k^T = \{(a, b), (b, v), (v, w)\}$ to three separate commodities $k_{(a,b)}$, $k_{(b,v)}$ and $k_{(v,w)}$ and the three best paths $\vec{P}_{k_{(a,b)}}^T = \{(a, b)\}$, $\vec{P}_{k_{(b,v)}}^T = \{(b, v)\}$ and $\vec{P}_{k_{(v,w)}}^T = \{(v, w)\}$. The commodity decomposition Lemma 5.4 shows that if we have a feasible flow for the three separate commodities, then it is possible to recombine it into a feasible flow for the original commodity $k$, and vice versa.

**Lemma 5.4 (Commodity Decomposition).** Given a commodity $k$ with demand $H^k$ and arc set $\vec{P}_k^T$, consider the following decomposition. Every feasible flow for a commodity $k$ with demand $H^k$ can be split into a feasible flow for $|\vec{P}_k^T|$ new commodities $k_f$, with $f \in \vec{P}_k^T$, for which the demand is $H^{k_f} = H^k$ and with best-arc set $\vec{P}_{k_f}^T = \{f\}$. The reverse—combining the flows—results again in a feasible flow for $k$.

**Proof.** For all type arcs $g = (i, j) \in A^T$ that can be used by $k$, i.e., for which $\vec{R}(g) \subseteq \vec{P}_k$, there exists a type arc $f = (n, m) \in \vec{P}_k^T$ in which $g$ is contained ($\vec{R}(g) \subseteq \vec{R}(f)$). We will prove this lemma by showing that setting the flows equal to

$$F_{k_f}^{k_i} = F_{ij}^k$$

for all $f$ and $(i, j) \mid \vec{R}((i, j)) \subseteq \vec{R}(f)$

and vice versa, satisfies the flow balance restrictions of both instances.

Let us start with proving the decomposition. Note that, constructed in this way, it is sufficient to show that the flow balance constraints for commodity $k_f$ are satisfied at both endpoints of an arc $f = (n, m) \in \vec{P}_{k_f}^T$:

$$\sum_{i|(n, i)\in A^T} F_{ni}^{k_f} = H^k = H^{k_f}$$

and

$$\sum_{j|(j, m)\in A^T} F_{jm}^{k_f} = H^k = H^{k_f}$$

Equality holds in both cases because $f$ is one of the best-path arcs for $k$. This, in turn, implies that the total flow of $k$ (and of $k_f$) uses $f$ or arcs covered by $f$. For internal nodes in $\vec{R}(f)$, the flow balance constraints are already satisfied because $F_{ij}^k$ is a feasible flow.

Next, consider the reverse, i.e., that the combined flow is a feasible flow for $k$. This is true since the flow balance constraints are satisfied because both the total
amount of incoming flow via $f_1(H^{k_1})$ and outgoing flow via $f_2(H^{k_2})$ are equal to $H^k$ by construction.

The application of the decomposition part of Lemma 5.4 for all commodities, will result in many commodities with the same origin, destination, and prescribed path. The following lemma shows that these similar commodities can be aggregated, thereby reducing the total number of commodities in the system.

**Lemma 5.5 (Commodity Aggregation).** Consider two commodities $k_1 = (n,m)$ and $k_2 = (n,m)$ with identical prescribed paths $P_{k_1} = P_{k_2}$. The demands for the commodities are given by $H^{k_1}$ and $H^{k_2}$. Both commodities can be replaced by a new commodity $k$ with demand $H^k = H^{k_1} + H^{k_2}$ and path $P_k = P_{k_1} = P_{k_2}$. Conversely, every feasible flow for $k$ can be disaggregated into feasible flows for $k_1$ and $k_2$.

**Proof.** Consider a feasible flow for commodity $k$ with demand $H^k = H^{k_1} + H^{k_2}$. Construct two separate flows $k_1$ and $k_2$ by labelling $H^{k_1}$ of the leaving flow units in red, and $H^{k_2}$ of them blue. Clearly, these flows are still feasible flows. The reverse is shown by removing the labels from both commodities.

By the previous two lemmas, we can assume that all commodities $k = (n,m)$ in an instance of MCF have the property that $P^k = \{(n,m)\}$. Note that this does not imply that an arc $(n,m)$ can only be used by one commodity, since commodities are still allowed to be routed using all arcs in their prescribed path.

The results of both lemmas hold because the ECP is only interested in finding a capacity assignment that minimises the total cost. Any information about the flows according to the original routes of the passengers is lost by decomposing and aggregating the commodities.

**Example 5.6.** Let us review the MCF problem on the graph displayed in Figure 5.2 on page 81. Originally, this problem contained $6 \times 5 = 30$ different commodities, i.e., one for every pair of vertices. After applying both of the lemmas above, we are left with at most $|A^T| = 2|E^T| = 18$ commodities. However, the type 1 edges $\{b,c\}$ and $\{v,w\}$ can never be part of a best path because of the similar type 2 edges. Therefore, the number of commodities can be reduced to 14.

Next, we use the previous two lemmas to show that we can assume that there exists an optimal flow that is symmetric. This is shown by using induction on the number of train types $T_{\text{max}}$. In the induction step, where we assume that we can construct a symmetric solution for $T_{\text{max}} = t^*$, we show how to transform a nonsymmetric flow across the arcs of type $T_{\text{max}} = t^* + 1$ into a symmetric flow.

**Corollary 5.7.** Consider an arbitrary instance of MCF. If, for some arc $(i,j)$ of type $T_{\text{max}}$ there exists a commodity $k$ for which $(i,j) \in P^k$, then $k$ is also the only commodity with this property.

**Proof.** Recall the definition of $P^k$, and thus of $P^T_k$, in (5.1). Since arc $(i,j)$ is of type $T_{\text{max}}$, there do not exist any arcs $(i',j') \neq (i,j)$ of any type $t > T_{\text{max}}$. 

\[\Box\]
5.3. FORMULATIONS FOR THE EDGE CAPACITY PROBLEM

Theorem 5.8 (Symmetric Flow). If the demand matrix \( H \) is symmetric, then, for any solution \((X^*, F^*)\) of MCF there exists a solution \((X^*, F)\) with the same objective function value, and with the property that \( F_{ij}^{nm} = F_{ji}^{mn} \), i.e., that \( F \) is a symmetric flow.

Proof. We prove this theorem using induction on the number of types \( T_{\text{max}} \). Initially, consider \( T_{\text{max}} = 1 \). Since in this case there is only one type, and the prescribed path is simple, every commodity has one unique path in the type graph from its origin to its destination. Therefore, in case \( T_{\text{max}} = 1 \), \( F^* \) will be symmetric, given that \( H \) is symmetric.

Next, assume that the theorem holds for \( T_{\text{max}} = t^* \). We show that this implies that it also holds for \( T_{\text{max}} = t^* + 1 \). From Corollary 5.7 it follows that for every arc of type \( T_{\text{max}} \), there is at most one commodity that is allowed to use this arc. If such a commodity does not exist, then we are done. Hence, assume that there exists one commodity for arc \((i, j)\) and one for arc \((j, i)\). Thus, for the type edge \( \{i, j\} \in E_{t^*+1}^T \), equation (5.4b) tells us

\[
x(|i, j|) \geq \sum_k F_{ij}^k = F_{ij}^{t^*} \quad \text{and} \quad x(|i, j|) \geq \sum_k F_{ji}^k = F_{ji}^{t^*}
\]

Suppose the two opposing flows defined for this type edge are not symmetric. So, without loss of generality, assume \( F_{ij}^{t^*} < F_{ji}^{t^*} \). We can now find \( F_{ji}^{t^*} - F_{ij}^{t^*} \) units of flow of commodity \((i, j)\) and, according to Lemma 5.5, reassign them to the arc \((i, j)\), making the flow on \((i, j)\) and \((j, i)\) equal. The capacity restriction for \( x(|i, j|) \) in (5.4b) will still be satisfied. Since we have only redirected flow away from the other arcs that could be used by \((i, j)\), this also holds for the type edges below \(|i, j|\). The resulting flow is feasible for MCF, and is symmetric on all edges of type \( t^* + 1 \) by repetition. Now, let us construct a new MCF instance with only \( t^* \) types from which the arcs of type \( t^* + 1 \) have been removed and the demands for commodities \((i, j)\) have been decreased by \( F_{ij}^{t^*} \) for every \((i, j) \in A_{t^*+1}^T \). Clearly, the previous flow on all but the arcs of type \( t^* + 1 \) is a feasible flow for this new instance. Therefore, by the induction hypothesis, this MCF can be made symmetric. The overall cost will now be

\[
\sum_{e \in E_{t^*+1}^T} f(x^*(e)) + \sum_{t \leq t^*} \sum_{e \in E_t^T} f(x^*(e)) = \sum_{e \in E_t^T} f(x^*(e)).
\]

In view of Theorem 5.8 we will no longer distinguish the commodities \((n, m)\) and \((m, n)\), or the arcs \((i, j)\) and \((j, i)\), since we have shown that we can assume that \( F_{ij}^{nm} = F_{ji}^{mn} \). Therefore, we will no longer use the directed graphs \( D \) and \( D^T \).

Before we introduce alternative model formulations for ECP, let us first make some general remarks about the structure of the undirected network graph and the type graph.

Lemma 5.9. Consider a network graph \( G = (V, E) \) that is a path. Now, for every network edge \( e \in E \) and type \( t \in T \), there is at most one type edge \( f \) of type \( t \) for which \( e \in R(f) \).

Proof. Without loss of generality, we can rename the vertices and network edges of \( G \) such that \( V = \{1, \ldots, n\} \) and \( E = \{(v, v+1) | v \in \{1, \ldots, n-1\}\} \), since \( G \) is a path. The proof is by contradiction. Assume that, for an arbitrary type \( t \), there are two
distinct type edges $f = \{v, w\}$ and $g = \{i, j\}$ in $E^T_t$ that cover $e$. Without loss of
generality, we can assume not only that $v < w$ and $i < j$ and $v \leq i < w$, but also
that either $w < j$ (crossing) or $j < w$ (non-crossing). Note that if $w = j$, then
we could reverse the numbering of the vertices. The first case implies that $t_w \geq t$
since $f \in E^T_t$, while the fact that $w$ is an internal vertex of $R(g)$ implies that it is of
type less than $t$. Similar reasoning can be applied in the second case.

**Corollary 5.10.** For any two distinct type edges $f$ and $g$ both of type $t$ with $R(f) \cap R(g) \neq \emptyset$, the graph induced by $R(f) \cup R(g)$ is not a path.

**Proof.** Assume that the graph induced by the network edges in $R(f) \cup R(g)$ is a
path. Since $R(f) \cap R(g) \neq \emptyset$, it follows that there is at least one network edge $e$ for
which $e \in R(f)$ and $e \in R(g)$. This is not possible according to Lemma 5.9.

**Corollary 5.11.** There does not exist a type edge $h$ of type $t'$ that covers two type
edges $f$ and $g$ of type $t < t'$ for which $R(f) \cap R(g) \neq \emptyset$.

**Proof.** By construction, $R(h)$ is a path. Assuming that such type edges $f$ and $g$
exist, then this implies that $R(f) \cup R(g)$ is also a path. However, this contradicts
Corollary 5.10.

**Lemma 5.12.** Consider a network edge $e \in E$, and an arbitrary type edge $g$ of type $t$
with $e \in R(g)$. Now, for every type $t' < t$ there exists a unique type edge $f$ of type $t'$
with $e \in R(f) \subseteq R(g)$

**Proof.** First, consider the case where $t' = t - 1$. Now, existence is immediate from the
assumptions. Uniqueness follows from Corollary 5.11. Moreover, since the existence
and uniqueness also hold for this type edge of type $t'$, there thus exists a unique type
edge for every type $t' < t$, by repetition.

Thus, we have shown that there cannot exist a type edge $h$ that covers two
overlapping type edges $f, g \in E^T_t$ of the same type, simply because the original
network edges covered by $f$ and $g$ cannot be a path according to Corollary 5.10.

### 5.3.2 The mixed integer programming formulation

Solving ECP instances using the MCF formulation requires a large number of vari-
ables and restrictions. It introduces a flow variable for all the available arcs in the
path for every commodity, requiring flow conservation constraints for all the nodes
along this path. We will now describe an integer programming formulation that uses
fewer variables, and show the equivalence of both formulations.

Compared to the MCF formulation with its completely disaggregated flow, the
IP$_{XY}$ formulation is based on constraining only the capacities of the edges in the type
graph. For ease of notation, let us introduce $\bar{H}(e)$ as the number of travellers for
whom type edge $e$ is part of their best path $P^T_k$. Thus, we define $\bar{H}(e) = \sum_{k \in P^T_k} \bar{H}^k$
for every type edge $e$. We also introduce additional variables $y_e$ for every type
edge $e \in E^T_t$ with $t > 1$. They represent the number of travellers over all pairs $(a, b)$
that could have used type edge $e$ across this particular part of their path $P_{ab}$, but
do not. Capacity will be reserved for them on the underlying type edges.
5.3. FORMULATIONS FOR THE EDGE CAPACITY PROBLEM

Figure 5.4: Travellers are possibly assigned to underlying type edges.

Example 5.13. Consider the type graph in Figure 5.4. The $(b, v)$-travellers can either use the type 3 edge $f$ from $b$ to $v$, or they are assigned to the two underlying type 2 edges $\{b, c\}$ and $c = \{c, v\}$ using the variable $y_f$. Whether they will actually use these type 2 edges depends on the individual values of the variables $y_{\{b, c\}}$ and $y_e$ through which they can be assigned to the underlying type 1 edges. In this example, the capacity restrictions for the type 3 edge $f$ and for the type 2 edge $e$ will be

$$x(f) \geq \tilde{H}(f) - y_f$$

and

$$x(e) \geq \tilde{H}(e) + y_f - y_e$$

The idea illustrated in Example 5.13 can be generalised to the following formulation, referred to as IP$_{xy}$.

$$\min \sum_{e \in E^t} f(x(e))$$

s.t. $$x(e) \geq \tilde{H}(e) + \sum_{f \in E^T_{t+1} \mid R(e) \subseteq R(f)} y_f$$

for all $e \in E^T_t$ (5.6)

$$x(e) \geq \tilde{H}(e) + \sum_{f \in E^T_{t+1} \mid R(e) \subseteq R(f)} y_f - y_e$$

for all $1 < t < T_{\max}$ and $e \in E^T_t$ (5.7)

$$x(e) \geq \tilde{H}(e) - y_e$$

for all $e \in E^T_{T_{\max}}$ (5.8)

$$x(e) \in \mathcal{C}$$

for all $e \in E^T_t$ (5.9)

$$y_e \in \mathbb{N}$$

for all $1 < t \leq T_{\max}$ and $e \in E^T_t$ (5.10)

If we enforce that all $y_e = 0$, then all capacities $x(e)$ must suffice to transport all travellers using only the type edges in their best path. The model, however, can decide to use different type edges (still part of the prescribed path) through the use of the variables $y_e$. These $y_e$ variables model the number of people that were assigned to use type edge $e$ of type $t$, but instead will be assigned to underlying type edges of type $t - 1$. Note that in this way, these travellers may then again be reassigned to edges of type $t - 2$, etc. The structure of the constraints for the type edges depends on the types of the edges. For an edge $e$ of type $t = 1$ in (5.6), there are no possibilities for rerouting passengers through $y_e$, since there are no edges of lower type. A similar
argument for edges of type $T_{\text{max}}$ in (5.8) makes it clear that we can only reassign passengers from these edges, not to them.

Next, we will prove the equivalence between MCF and $\text{IP}_{XY}$. To do so, let us first make the following observations concerning feasible flows.

**Observation 5.14.** For an arbitrary commodity $k$ and an arbitrary type edge $e \in E^T_k$ that is allowed for this commodity, i.e., with $R(e) \subseteq P_k$, exactly one of the following holds:

\[ e \in P^T_k \quad \text{or} \quad \exists t' > t : \exists f \in E^T_{t'} : R(e) \subseteq R(f) \subseteq P_k \]

Thus, either a type edge $e$ is part of the best path for commodity $k$, or there exists a type edge $f$ of higher type that can also be used by $k$ at this part of his path.

Next, Observation 5.15 considers the sum of the demand for all commodities that are allowed to use some network edge $e$. It is easy to see that, for any feasible flow $F$, this is equal to the sum of all the flows across $e$, i.e., to the sum of the flows on the network edge $e$, and on all type edges $f$ of a type $t > 1$ for which $e \in R(f)$.

**Observation 5.15.** For every feasible flow $F$ of MCF the following holds for every edge $e$ of type 1:

\[ \hat{H}(e) + \sum_{f \in E^T_{t > 1} | e \subseteq R(f)} \hat{H}_f = \sum_{k | R(e) \subseteq P_k} F^k_e + \sum_{f \in E^T_{t > 1} | e \subseteq R(f)} F^k_f \text{ for all } e \in E^T_1 \]

**Lemma 5.16.** Every solution $(X, F)$ of MCF can be transformed into a solution $(X, Y)$ of $\text{IP}_{XY}$ with the same objective function value.

**Proof.** We prove this lemma by induction on $T_{\text{max}}$. First, consider the case in which $T_{\text{max}} = 1$. From Observation 5.15 it follows that $\hat{H}(e) = \sum_{k | R(e) \subseteq P_k} F^k_e$ for all type edges $e \in E^T_1$, and thus that

\[ x(e) \geq \sum_{k} F^k_e = \sum_{k | R(e) \subseteq P_k} F^k_e = \hat{H}(e) \quad \text{for all } e \in E^T_1 \]

As induction hypothesis, let us now assume that the lemma holds for $T_{\text{max}} = t^*$, and consider the case with $T_{\text{max}} = t^* + 1$. For any edge $e$ of type $t^* + 1$ we construct $y_e = \hat{H}(e) - \sum_k F^k_e$. Note that $e \in E^T_{t^*+1}$ implies that $y_e$ is nonnegative. Clearly, now

\[ x(e) \geq \sum_k F^k_e = \hat{H}(e) - y_e \quad \text{for all } e \in E^T_{t^*+1} \]

The final step is to reduce the problem from $t^* + 1$ types to $t^*$ types by removing all the type edges of type $t^* + 1$ and the associated variables from the problem. The original solution $(X, F)$ is now also feasible for the MCF formulation of the reduced problem with $T_{\text{max}} = t^*$. Therefore, we can apply the induction hypothesis and thus prove this lemma.

**Lemma 5.17.** Every solution $(X, Y)$ of $\text{IP}_{XY}$ can be transformed into a solution $(X, F)$ of MCF with the same objective function value.
Proof. From Lemma 5.4 and Lemma 5.5 it is clear that we should show that feasible flows can be constructed from \((X, Y)\) for artificial commodities \(k = \{v, w\}\) for type edges \(\{v, w\} \in E_T\), with demand \(\vec{H}(k)\). We will prove this lemma using induction on \(T_{\text{max}}\). First, note that for \(T_{\text{max}} = 1\) all constraints of (5.6) are of the form \(x(e) \geq \vec{H}(e)\) for all \(e \in E_T\). Since there is only one type of edges, we can thus set

\[ F^k_e = \vec{H}(e) \quad \text{for all} \quad k \in E_T, \quad e \in P^T_k = \{k\} \]

Thus, every commodity corresponds to an edge in \(E_T\), and \(F^f_e = 0\) for all \(f \neq k\). Obviously, all flow restrictions (5.4b)-(5.4f) are satisfied.

Next, assume that we can construct feasible flows for \(T_{\text{max}} = t^*\). Now we show that it is also possible to construct feasible flows for \(T_{\text{max}} = t^* + 1\). The constraints (5.8) for the type edges \(e \in E_{t^* + 1}\) are

\[ x(e) \geq \vec{H}(e) - y(e) \quad \text{for all} \quad e \in E_{t^* + 1} \]

The total flow across type edge \(e\) can thus be found by taking \(F^k_e\) such that

\[ F^k_e = \vec{H}(e) - y_e \]

The remaining demand \(y_e\) will be routed along the other possible edges: the type edges \(f \in E^T_{t^*}\) for which \(R(f) \subseteq R(e)\). The last part of this proof is to show that we are now not only able to construct a feasible flow for the edges of type \(t^* + 1\), but additionally, also for all other edges. To show that this is possible, note that by restriction (5.7) it follows that

\[ x(f) \geq (\vec{H}(f) + y(e)) - y(f) \quad \text{for all} \quad f \in E^T_{t^*} \]

This implies that, by our induction hypothesis, we can also find a feasible flow for the remaining edges of types \(t \leq t^*\).

The previous two lemmas imply the following theorem.

**Theorem 5.18.** The formulations MCF and IP\(_{XY}\) are equivalent.

**Lemma 5.19.** Consider the relaxation of IP\(_{XY}\) in which all \(y_e \in \mathbb{R}_+\). Every solution \((X, Y)\) of this relaxation, where \(Y\) is the vector containing all \(y_e\), can be transformed into an integer solution \((X, \tilde{Y})\) with the same objective function value.

**Proof.** We show that setting all \(y_e\) to the rounded down value \(\tilde{y}_e = \lfloor y_e \rfloor\) results in a feasible solution \((X, \tilde{Y})\). First, consider this rounding scheme for type edges \(e\) of type \(t = T_{\text{max}}\). The integrality of \(x(e)\) and \(\vec{H}(e)\) ensures that \(x(e) \geq \vec{H}(e) - \lfloor y_e \rfloor\). Since \(\tilde{y}_e \leq y_e\), all other restrictions also remain satisfied. Next, consider the edges of type \(t = T_{\text{max}} - 1\). Clearly, in (5.7) the sum of all \(\tilde{y}_f\) of the type edges \(f \in E^T_{t+1}\) is integer. Therefore, by applying similar reasoning as before, we see that the rounding scheme preserves the feasibility of the constructed solution for the remaining edges of types \(T_{\text{max}} - 1\) through \(t = 1\). \(\square\)
5.3.3 The integer programming formulation

In the IP_{XY} formulation we introduced additional y_e variables to model the rerouting of commodities over other edges in the type graph. In this section, we describe an alternative formulation that does not use the rerouting variables y_e, but instead guides the routing by imposing additional restrictions.

Let us first review an example of this formulation for \( T_{\text{max}} = 2 \).

**Example 5.20.** Consider the network displayed in Figure 5.5. For every edge in the network graph, we consider all combinations of type edges that can be used to cross this edge. In this light, the following constraints are necessary, and, as we show later, also sufficient to formulate ECP:

- For the network edge e, and the crossing type edges \( f_1 \) and \( f_2 \):
  
  \[
  \begin{align*}
  x(e) &\geq \tilde{H}(e) \\
  x(e) + x(f_1) &\geq \tilde{H}(e) + \tilde{H}(f_1) \\
  x(e) + x(f_1) + x(f_2) &\geq \tilde{H}(e) + \tilde{H}(f_1) + \tilde{H}(f_2) \\
  x(e) + x(f_2) &\geq \tilde{H}(e) + \tilde{H}(f_2)
  \end{align*}
  \]

- For the network edge g, and the crossing type edge \( f_1 \):
  
  \[
  \begin{align*}
  x(g) &\geq \tilde{H}(g) \\
  x(g) + x(f_1) &\geq \tilde{H}(g) + \tilde{H}(f_1)
  \end{align*}
  \]

- For the network edge h, and the crossing type edge \( f_2 \):
  
  \[
  \begin{align*}
  x(h) &\geq \tilde{H}(h) \\
  x(h) + x(f_2) &\geq \tilde{H}(h) + \tilde{H}(f_2)
  \end{align*}
  \]

For example, the restriction \( x(g) + x(f_1) \geq \tilde{H}(g) + \tilde{H}(f_1) \) enforces that the combined capacity of the type edges g and \( f_1 \) must suffice to transport all commodities that can at best use g, plus all that can at best use \( f_1 \). Since the only possibility to arrive at c is to use either g or \( f_1 \), it is clear that this is a necessary restriction.
5.3. FORMULATIONS FOR THE EDGE CAPACITY PROBLEM

The integer programming formulation $IP_X$ for ECP reads as follows

$$\min \sum_{e \in E^T} f(x(e))$$

subject to

$$x(e) + \sum_{t>1} \sum_{f \in S^e_t} x(f) \geq \tilde{H}(e) + \sum_{t>1} \sum_{f \in S^e_t} \tilde{H}(f) \quad \text{for all } e \in E^T_t$$

$$x(e) \in C \quad \text{for all } e \in E^T$$

where for a given edge $e$ of type 1 the sets $S^e_t$ are such that $S^e_t \subseteq \bigcup_{j \in S^e_{t-1}} \{g \in E^T_t | R(f) \subseteq R(g)\}$ with $S^e_1 = \{e\}$. Thus, $S^e_2$ is a subset of all the type 2 edges that cover $e$. Next, $S^e_3$ is then a subset of all the type 3 edges that cover some edge in the current $S^e_2$, etc. Restriction (5.12) enforces sufficient capacity on type edge $e$ together with the type edges in the sets $S^e_t$ for $t = 2, \ldots, T_{\max}$. Note that a constraint is added for all possible sets $S^e_2, \ldots, S^e_{T_{\max}}$. For ease of reference, we refer to these restrictions as the capacity-subset (CS) constraints. Finally, feasible values for $x(e)$ are enforced by the set $C \subset \mathbb{N}$, which represents the valid capacities that can be assigned to a type edge.

We will now show that the edge capacities $X$ of any feasible solution of $IP_{XY}$ are also a feasible solution for $IP_X$.

**Lemma 5.21.** Every solution $(X, Y)$ of $IP_{XY}$ can be transformed into a solution $X$ for $IP_X$ with the same objective function value.

**Proof.** We will show that all restrictions of $IP_X$ are valid for $IP_{XY}$. Consider an arbitrary restriction of $IP_X$ for type edge $e \in E^T_1$, and with the sets $S^e_1, \ldots, S^e_{T_{\max}}$. Recall the CS restriction (5.12) for type edge $e$:

$$x(e) \geq \tilde{H}(e) + \sum_{f \in E^T_1 | R(e) \subseteq R(f)} y_f$$

Now, consider an arbitrary collection $S^e_2, \ldots, S^e_{T_{\max}}$ of subsets, i.e., an arbitrary CS constraint. Since all values $y_e$ are nonnegative, the following holds:

$$x(e) + \sum_{t>1} \sum_{f \in S^e_t} x(f) \geq \tilde{H}(e) + \sum_{f \in E^T_1 | R(e) \subseteq R(f)} y_f + \sum_{t>1} \sum_{f \in S^e_t} x(f)$$

$$\geq \tilde{H}(e) + \sum_{f \in S^e_1} y_f + \sum_{t>1} \sum_{f \in S^e_t} \left( \tilde{H}(f) + \sum_{g \in S^e_{t+1}} y_g - y_f \right)$$

$$\ldots$$

$$\geq \tilde{H}(e) + \sum_{t>1} \sum_{f \in S^e_t} \tilde{H}(f)$$

To prove the last step we need to show that the $y_e$ variables with a negative sign
cancel out against the other $y_e$ variables. This is shown as follows.

$$\{ f \in S_2^e | R(e) \subseteq R(f) \} \cup \bigcup_{f \in S_2^e} \{ g \in S_{t+1}^e | R(f) \subseteq R(g) \} = \{ f \in S_2^e \} \cup \bigcup_{f \in S_2^e} \{ g \in S_{t+1}^e \} \supseteq \bigcup_{f \in S_2^e} \{ f \in S_2^e \}$$

The equality holds because, by definition, every $g \in S_{t+1}^e$ has some $f \in S_2^e$ for which $R(f) \subseteq R(g)$. Since we take the union over all $f \in S_2^e$, the equality follows immediately. This completes the proof. \qed

The inverse, extending a solution $X$ of IP$_X$ with appropriately chosen values for $Y$, gives a feasible solution to IP$_{XY}$, as is shown in Lemma 5.22.

**Lemma 5.22.** Every solution $X$ of IP$_X$ can be transformed into a solution $(X, Y)$ for IP$_{XY}$ with the same objective function value.

**Proof.** As in the proof of Lemma 5.16, we will again provide a scheme for constructing a suitable vector $Y$. Consider an arbitrary solution $X$ of IP$_X$. Let us choose the value $y_e$ for any type edge $e \in E_t^T$ as

$$y_e = (\bar{H}(e) - x(e) + \sum_{f \in E_{t+1}^T} \min \{ R(e) \subseteq R(f), y_f \})^+ \quad (5.14)$$

where $(x)^+ \equiv \max\{0, x\}$. Recursively, we can thus construct all $y_e$ starting at type edges $e$ of type $t = T_{\text{max}}$ (for which $E_{t+1}^T = \emptyset$), and ending at $e \in E_2^T$. We are now left to prove that

$$x(e) \geq \bar{H}(e) + \sum_{f \in E_3^T} \min \{ R(e) \subseteq R(f), y_f \} \quad \text{for all } e \in E_1^T$$

Thus, substituting (5.14) for all $y_f$, we have to prove the validity of

$$x(e) \geq \bar{H}(e) + \sum_{f \in E_1^T} \min \{ R(e) \subseteq R(f), \sum_{g \in E_3^T} \min \{ R(f) \subseteq R(g), \bar{H}(g) - x(g) + \ldots \}^+ \}$$

To show this, consider the values of the different max-plus parts, i.e., the $y_f$. Being either zero or positive, we introduce the sets $S^*_t \subseteq E_t^T$ such that $S^*_t = \{ f \in E_t^T | y_f > 0 \}$ for our arbitrary solution $X$. Since the capacity restrictions (5.12) of the IP$_X$ formulation contain all possible combinations of sets $S_t^e$, it follows that all sets $S^*_t$
are among them. Therefore

\[ \tilde{H}(e) + \sum_{f \in E_T^1 \backslash R(e) \subseteq R(f)} \left( \tilde{H}(f) - x(f) + \sum_{g \in E_T^1 \backslash R(f) \subseteq R(g)} (\tilde{H}(g) - x(g) + \ldots)^+ \right) \]

\[= \tilde{H}(e) + \sum_{f \in S_T^1} (\tilde{H}(f) - x(f)) + \sum_{g \in S_T^1} (\tilde{H}(g) - x(g)) + \ldots \]

\[= \tilde{H}(e) + \sum_{t > 1} \sum_{f \in S_T^t} \tilde{H}(f) - \sum_{t > 1} \sum_{f \in S_T^t} x(f) \leq x(e) \]

As in the proof of Lemma 5.16, replacing the nested summations by the separate summations in the second equation can be done using the results from Lemma 5.12. Since this construction is valid for an arbitrary solution \( X \), this completes the proof.

The previous two lemmas imply the following theorem.

**Theorem 5.23.** The formulations IP\( X \) and IP\( XY \) are equivalent.

The number of CS restrictions (5.12) is exponential in the number of type edges. However, as we will show now, we can find the maximally violated CS constraint for every edge \( e \) of type 1 in polynomial time.

To solve this separation problem, consider an edge \( e \) of type 1, and construct the directed graph \( T(e) \) that contains a node for every type edge that covers \( e \), i.e., a node for every \( f \in E_T^1 \) with \( e \in R(f) \). The nodes can be layered according to the types of the associated arcs. The graph contains only arcs between nodes of layer \( t \) to layer \( t - 1 \). Let us say that the node in layer \( t \) is related to type edge \( g \in E_T^t \) while the node in layer \( t - 1 \) is related to type edge \( f \in E_T^{t-1} \). There exists an arc between these two nodes if the type edge \( f \) is the unique type edge—according to Lemma 5.12—for which \( e \in R(f) \subseteq R(g) \). Thus, every node in \( T(e) \) has exactly one outgoing arc, though possibly many incoming arcs. It is easy to see that \( T(e) \) is a directed in-tree rooted at the node associated with the type edge \( e \). See Figure C.2 and Figure C.3 in Appendix C for an example of a type graph and its corresponding layered tree.

Next, consider a solution \( \bar{X} \) with a value \( \bar{x}(e) \) for every \( e \in E_T^1 \). For simplicity, let us refer to the node in \( T(e) \) that is associated with type edge \( f \) as node \( f \), etc. With every node \( f \) in \( T(e) \) we associate a revenue (violation) \( \tilde{H}(f) - \bar{x}(f) \). The problem is to find a subtree of \( T(e) \) that is rooted at \( e \) for which the sum of the revenues of the nodes in the subtree is maximal. Such a subtree that has a positive revenue, corresponds to a violated CS inequality.

**Lemma 5.24.** Consider a layered tree \( T(e) \) for an edge \( e \) of type 1. Every subtree of \( T(e) \) that is rooted at \( e \) corresponds to a CS inequality of (5.12) for type edge \( e \).
CHAPTER 5. THE MULTIPLE-TYPE LINE PLANNING PROBLEM

Algorithm 5.1 Determine the maximally violated CS constraint on type edge $e$.

Input: A directed in-tree $T(e)$ that is layered, and rooted at node $e$.

A solution $\tilde{x}$.

Algorithm:

1. Initialise $d(f) = \tilde{H}(f) - \tilde{x}(f)$ for each node $f$ in $T(e)$.
2. Initialise $t = t^*$, where $t^*$ is the highest type for which there are type edges represented in $T(e)$.
3. while $t > 1$ do
4.   for all nodes $g$ in layer $t$ do
5.   if $d(g) > 0$ then
6.     Let $f$ be the unique node in layer $t - 1$ to which $g$ is connected.
7.     Set $d(f) = d(f) + d(g)$
8.   else
9.     Delete the subtree rooted at $g$ from $T(e)$.
10. end if
11. end for
12. Set $t = t - 1$.
13. end while

and with the set $S_t^e$ equal to the nodes in the subtree that are in layer $t$, for $t \in \{2, \ldots, T_{\text{max}}\}$.

Proof. We have to show that

$$S_t^e \subseteq \bigcup_{f \in S_{t-1}^e} \{g \in E_t^T \mid R(f) \subseteq R(g)\}$$

with $S_1^e = \{e\}$ holds. By the construction of $T(e)$, and the fact that the subtree is connected, this result follows immediately.

To solve the problem of finding the subtree that maximises the total node revenue we add a label $d(f)$ to every node $f$, and initialise it to $d(f) = \tilde{H}(f) - \tilde{x}(f)$. In Algorithm 5.1, we examine the tree $T(e)$ starting at the nodes in the layer of the highest type for which there are type edges represented in $T(e)$, say type $t = t^*$. Note that $t^*$ can be smaller then $T_{\text{max}}$. For every node $g$ in layer $t$ that has a positive label $d(g)$, let $f$ be the unique node in layer $t - 1$ to which $g$ is connected. For these nodes $g$ with $d(g) > 0$, set $d(f) = d(f) + d(g)$. Otherwise, if $d(g) \leq 0$, delete the subtree rooted at $g$ from $T(e)$. Note that the remaining graph $T(e)$ is still a directed in-tree rooted at $e$. Now, set $t = t - 1$, and proceed with the next layer as before.

After $t - 1$ iterations, we arrive at layer 1 that contains only node $e$. All that is left of $T(e)$ is the subtree of the original tree, rooted at $e$ that has a total revenue of $d(e)$. Note that the running time of the algorithm is $O(T_{\text{max}}|E_T^T|)$, and therefore polynomial in the size of the input.

Lemma 5.25. When processing layer $t$ in Algorithm 5.1, the value of $d(f)$ for any node at layer $t$ is the maximum revenue of any subtree of the original $T(e)$ that is rooted at $f$. 
5.4. COMPUTATIONAL RESULTS

<table>
<thead>
<tr>
<th></th>
<th>MCF</th>
<th>$\text{IP}_{XY}$</th>
<th>$\text{IP}_{X}$</th>
</tr>
</thead>
<tbody>
<tr>
<td># Vars.</td>
<td>$</td>
<td>E</td>
<td>^2 + O(</td>
</tr>
<tr>
<td># Cons.</td>
<td>$</td>
<td>E</td>
<td>^2 + O(</td>
</tr>
</tbody>
</table>

Table 5.1: Variable and constraint statistics for the MCF, $\text{IP}_{XY}$ and $\text{IP}_{X}$ formulations of ECP.

Proof. We show this by induction on the current layer $t$. For the initial layer at $t = t^*$, the value $d(f) = \bar{H}(f) - \bar{x}(f)$ is exactly equal to the total revenue of the subtree rooted at $f$, i.e., only node $f$. Now, assume that at some layer $t$ the claim holds. For an arbitrary node $f$ of layer $t - 1$, the algorithm takes all, and only those subtrees rooted at nodes $g$ in layer $t$ with $d(g) > 0$. Clearly, if $d(g) \leq 0$, then the total revenue would become no better than if the subtree rooted at $g$ were deleted. On the other hand, if a subtree with $d(g) > 0$ were deleted, then this would strictly worsen the overall revenue of the subtree rooted at $f$. Therefore, $d(f)$ is also the maximal revenue of any subtree rooted at $f$ for any node $f$ in layer $t - 1$. This completes the proof. 

Corollary 5.26. The remaining subtree of the original tree $T(e)$ is the maximal subtree that is rooted at $e$, with respect to the total revenue over all nodes.

Proof. This follows immediately from the previous lemma.

If $d(e)$ is positive, then, according to Corollary 5.26 and Lemma 5.24 we have found a violated CS constraint. Alternatively, if $d(e) \leq 0$, then all CS constraints for the edge $e$ of type 1 are satisfied.

Theorem 5.27. The separation problem for the exponentially many capacity-subset constraints of (5.12) can be solved in polynomial time in the size of the input.

Proof. The separation problem for the CS constraints for one edge $e$ of type 1 can be solved in polynomial time. Since we only have to test this for every edge of type 1, we can thus solve the overall separation problem for the CS constraints in polynomial time.

5.4 Computational results

The three formulations MCF, $\text{IP}_{XY}$, and $\text{IP}_{X}$ differ with respect to the necessary number of variables and constraints for ECP. Table 5.1 shows worst-case statistics for all formulations.

We have used three real-life instances of MCLP to compare the practical performance of solving them using the $\text{IP}_{XY}$ and $\text{IP}_{X}$ formulations, after applying the substitutions mentioned in §5.2. The instances all concern different parts of the Dutch railway network. We have chosen these instances because of the different structures
of the associated network graphs. The characteristics of the instances can be found in Table 5.2.

The three networks are visualised in the graphs in Figure 5.6. The first two instances are rather small with respect to the numbers of nodes in the network. We have chosen these instances to compare the practical use of the two proposed formulations. At first glance, one might expect that there will be a computational tradeoff between the model structure of IP_{XY} and the additional CS constraints in IP_{X}. The network graph of NS3600 is a path. This influences the number of type edges in \( E^T \). From Lemma 5.9, it is easy to see that, in the case of a path, the number of edges in \( E^T \) is at most \( T_{\max} \cdot |E| \). But even more important, the number of possible subsets for IP_{X} is also at most \( T_{\max} \cdot |E| \), i.e., at most \( T_{\max} \) per network edge \( e \in E \). To test the behaviour of the number of subset restrictions, we have also included two instances that introduce stations with a degree higher than two in the network graph.

Every instance was tested using both the IP_{XY} and the IP_{X} formulation. The numerical results were obtained using CPLEX 7.5 on an AMD Athlon 800 MHz with 512 MB internal memory running Linux, kernel 2.4.8. All instances where tested with all the CPLEX parameters at their default values. The model statistics and computational results are shown in Table 5.3 and Table 5.4 respectively.

The most striking statistics of Table 5.3 are the numbers of CS constraints for IP_{X}. The table shows that, for these real-life instances, the structure and size of the networks call for a number of subsets that is roughly only linear in the number of type edges \( |E^T| \). Therefore, we did not implement the separation algorithm described in §5.3.3, but simply added all CS constraints at the root node.

The computational results in Table 5.4 show that none of the IP_{XY} instances could be solved to optimality within one hour. One possible explanation for this
5.4. COMPUTATIONAL RESULTS

(a) NS3600

(b) NSNH

(c) NSRandstad

Figure 5.6: The type graphs for the instances NS3600 (5.6(a)), NSNH (5.6(b)) and NSRandstad (5.6(c)).
Table 5.4: Computational results. The asterisk (*) indicates that the time limit of 3600 seconds was reached.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Model</th>
<th>Best</th>
<th>Root LP</th>
<th>Best LP</th>
<th>Gap</th>
<th># Sec.</th>
<th># Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS3600</td>
<td>IP_*</td>
<td>7520</td>
<td>7213</td>
<td>7520</td>
<td>0.0%</td>
<td>0.81</td>
<td>60</td>
</tr>
<tr>
<td>NS3600</td>
<td>IP_{XY}</td>
<td>7520</td>
<td>6430</td>
<td>7173</td>
<td>4.6%</td>
<td>*</td>
<td>1207k</td>
</tr>
<tr>
<td>NSNH</td>
<td>IP_*</td>
<td>1376</td>
<td>1313</td>
<td>1376</td>
<td>0.0%</td>
<td>6.66</td>
<td>407</td>
</tr>
<tr>
<td>NSNH</td>
<td>IP_{XY}</td>
<td>1376</td>
<td>1250</td>
<td>1331</td>
<td>3.3%</td>
<td>*</td>
<td>748k</td>
</tr>
<tr>
<td>NSRandstad IP_*</td>
<td>5248</td>
<td>4888</td>
<td>5051</td>
<td>5248</td>
<td>3.8%</td>
<td>*</td>
<td>25k</td>
</tr>
<tr>
<td>NSRandstad IP_{XY}</td>
<td>5536</td>
<td>4615</td>
<td>4873</td>
<td>5536</td>
<td>12.0%</td>
<td>*</td>
<td>104k</td>
</tr>
</tbody>
</table>

Table 5.5: A comparison between the MCLP solutions, and solutions of using CLP with a system split. Could be the significantly lower root LP values. With NS3600 for example, the root LP value of the IP_\* formulation is 7213 (4.08% gap), whereas one hour, or 1.2 million nodes, of branching on IP_{XY} gives a best lower bound of only 7173 (4.61% gap). Similar conclusions can also be drawn from the other results. It can be concluded that the results for IP_\* are more promising than those for IP_{XY}. We have also tested the effect of using a system split and solving a separate line planning problem for every type as in CLP, compared to solving the line planning problem using MCLP. The system split was made according to the best-paths for every commodity. Thus, the capacities must allow all passengers to use lines of their best (highest) possible type. Using such a system split is equivalent to forcing all \( y_e \) variables in an IP_{XY} instance to zero. The optimal results for this system split are shown in Table 5.5. The reduction in the objective function value by using MCLP is between 15.7% and 21.8% for the tested instances. Note once more that these models only minimise the operated capacities and operating costs, and do not measure, for example, the necessary number of train changes of passengers that travel through the network. For an analysis of the MCLP line plans with the more detailed traffic assignment used in Chapter 6, see \( §6.6.1 \).

5.5 Summary and conclusions
In this chapter we have described different integer programming formulations for modelling the MCLP. Where previous work, e.g., Bussieck [11], Claessens et al. [18] focused on modelling CLP for exactly one type of trains and stations, we present
generalisations of these models within a cost-minimising setting. First, the general multi-commodity flow formulation is introduced in §5.3.1. This formulation is then used to prove the validity of the two main formulations IP_{XY} (see §5.3.2) and IP_{X} (see §5.3.3). Even though the number of restrictions of IP_{X} can be exponential in the size of the instance, we show in §5.3.3 how to identify violated constraints in polynomial time.

Using three real-life instances we compare the computational results for both formulations. From these tests, we can conclude that the IP_{X} formulation outperforms IP_{XY} in all of the chosen instances. Furthermore, the solutions that are obtained using the multi-type formulations are better than the results from solving a sequence of single-type problems.

#### 6.1 Introduction

In many real-life passenger railway networks, the types of stations and lines characterize the stopping patterns of the lines. This chapter considers the problem of altering the halt of lines by both upgrading and downgrading stations, such that the results in less total travel time for all passengers considered. We propose a combination of exact methods, Lagrangian relaxation, and a polynomial sub-multiplicative adjustment algorithm to solve the proposed generalised revenue maximisation problem. A computational study of several real-world instances of the problem is presented. The results in this chapter are based on the work of Geraerts et al.

In many real-life passenger railway networks, the types of stations and lines characterize the stopping patterns of the lines. This chapter considers the problem of altering the halt of lines by both upgrading and downgrading stations, such that the results in less total travel time for all passengers considered. We propose a combination of exact methods, Lagrangian relaxation, and a polynomial sub-multiplicative adjustment algorithm to solve the proposed generalised revenue maximisation problem. A computational study of several real-world instances of the problem is presented. The results in this chapter are based on the work of Geraerts et al.

In many real-life passenger railway networks, the types of stations and lines characterize the stopping patterns of the lines. This chapter considers the problem of altering the halt of lines by both upgrading and downgrading stations, such that the results in less total travel time for all passengers considered. We propose a combination of exact methods, Lagrangian relaxation, and a polynomial sub-multiplicative adjustment algorithm to solve the proposed generalised revenue maximisation problem. A computational study of several real-world instances of the problem is presented. The results in this chapter are based on the work of Geraerts et al.

In many real-life passenger railway networks, the types of stations and lines characterize the stopping patterns of the lines. This chapter considers the problem of altering the halt of lines by both upgrading and downgrading stations, such that the results in less total travel time for all passengers considered. We propose a combination of exact methods, Lagrangian relaxation, and a polynomial sub-multiplicative adjustment algorithm to solve the proposed generalised revenue maximisation problem. A computational study of several real-world instances of the problem is presented. The results in this chapter are based on the work of Geraerts et al.

In many real-life passenger railway networks, the types of stations and lines characterize the stopping patterns of the lines. This chapter considers the problem of altering the halt of lines by both upgrading and downgrading stations, such that the results in less total travel time for all passengers considered. We propose a combination of exact methods, Lagrangian relaxation, and a polynomial sub-multiplicative adjustment algorithm to solve the proposed generalised revenue maximisation problem. A computational study of several real-world instances of the problem is presented. The results in this chapter are based on the work of Geraerts et al.

In many real-life passenger railway networks, the types of stations and lines characterize the stopping patterns of the lines. This chapter considers the problem of altering the halt of lines by both upgrading and downgrading stations, such that the results in less total travel time for all passengers considered. We propose a combination of exact methods, Lagrangian relaxation, and a polynomial sub-multiplicative adjustment algorithm to solve the proposed generalised revenue maximisation problem. A computational study of several real-world instances of the problem is presented. The results in this chapter are based on the work of Geraerts et al.
could be the significantly lower cost LP value. With 833620 for example, the cost LP value of the LP on formulation 2270 (4.001% gap), whereas the lower, or 1.2 million modes, of branching on IP $xy$ gives a best lower bound of only 7177 (4.61% gap). Similar conclusions can also be drawn from the other results. It can be concluded that the results for IP $x$ are more promising than those for IP $xy$.

We have also treated the effect of using a system split and solving a separate line planning problem for every type as in CLP, compared to solving the line planning problem using MCLP. The system split was made according to the best-path for every commodity. Thus, the capacities must allow all passengers in the line of their best (highest) possible type. Using such a system split is equivalent to fixing all $u$-variables in an IP $xy$ instance to zero. The optimal results for this system split are shown in Table 5.6. The reduction in the objective function value by using MCLP is between 15.7% and 21.8% for the tested instances. Note once more that these values only consider the operational capacities and operating costs, and do not measure, for example, the necessary number of train changes of passengers that travel through the network. For an analysis of the MCLP line plans with the more detailed traffic assignment used in Chapter 6, see §6.8.1.

### 5.5 Summary and Conclusions

In this chapter, we have described different integer programming formulations for modelling the MCLP. Where previous work, e.g., Bussieck [41], Consetten et al. [48] focused on modelling CLP for exactly one type of trains and stations, we present

<table>
<thead>
<tr>
<th>Instance</th>
<th>CLP</th>
<th>MCLP</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>833620</td>
<td>8923</td>
<td>8559</td>
<td>-363 (-3.8%)</td>
</tr>
<tr>
<td>8338</td>
<td>1790</td>
<td>1875</td>
<td>-85 (-4.5%)</td>
</tr>
<tr>
<td>Standard</td>
<td>6224</td>
<td>5218</td>
<td>-976 (-15.7%)</td>
</tr>
</tbody>
</table>
Chapter 6

The station type optimisation problem

In many real life passenger railway networks, the types of stations and lines characterise the stopping patterns of the lines. This chapter considers the problem of altering the halts of lines by both upgrading and downgrading stations, such that this results in less total travel time for all passengers combined. We propose a combination of reduction methods, Lagrangian relaxation, and a problem-specific multiplier adjustment algorithm to solve the presented mixed integer linear programming formulation. A computational study of several real-life instances based on problem data of NSR is included. The results in this chapter are based on Goossens et al. [39].

6.1 Introduction

The line planning problems in Chapter 4 and Chapter 5 describe the decision problem of finding routes in the railway network on which trains are to be operated. However, the stations at which these lines halt are dictated by the types of the stations and lines. These types are part of the problem input. In contrast, this chapter considers the line plan to be given, but concentrates on altering the stops of the lines along their route to decrease the total travel time of passengers through the network.

The next sections show how to formulate this problem as a multi-commodity network flow problem with additional constraints and variables. Using Lagrangian relaxation, §6.4 shows how to obtain lower bounds for this problem. These bounds are then applied in a branch-and-bound framework in §6.5. In §6.6 we describe a computational study based on instances of the Dutch passenger railway operator NSR.
6.2 Modelling

As defined in Chapter 3, a line specifies a route between an origin and a destination station, the subsequent stops and an operated hourly frequency. Even though the subsequent halts of a line could be any arbitrary set of the stations it passes along its route, we consider the halts to follow strict patterns. As explained in §3.4, we assume the halting pattern to be such that a line of type \( t \) halts at all stations along its route that are of type \( t \), or of a higher type.

For most planning problems, the types of lines and stations in the network are given as part of the input. In contrast, this chapter considers the types of the lines to be given, and concentrates on finding a type for every station. Since the type of a station determines the halts of the passing lines, these types also influence the travel times through the network.

As an example, consider some station \( v \). The type of this station is important not only for the people that want to travel to and from \( v \), but also for the people that pass \( v \) on their journey. Because some of the passing lines may not halt there, the travellers starting or ending their trip at \( v \) are restricted in their possibilities by the type of \( v \). On the other hand, upgrading \( v \) to a higher station type can have a negative influence on the travel times of people on a train line passing \( v \), if the new type of \( v \) causes their train to halt there.

The models we describe for the station type optimisation problem focus on finding an assignment of types to stations that minimises the total travel time of all passengers through the network. We assume that all passengers choose their route through the network in such a way that they minimise their individual travel time. To allow the passengers this freedom, we also assume that the capacities of the train lines are sufficient. This yields the traffic assignment for this system.

6.2.1 The line-event graph

Consider a railway network graph \( G = (V, E) \) built up from the set of vertices (stations) \( V \) and connecting edges (tracks) \( E \). In addition, we are given a set of lines \( L \) and a set of types \( T \). Note that \( L \) is the set of operated lines, i.e., the line plan. We also define the subset of lines \( L(v) \) as all the lines that pass some station \( v \in V \). For every operated line \( l \in L \), the line plan gives a type \( t(l) \in T \) and an operated hourly frequency \( f(l) \in \mathbb{N} \).

Without knowing the halting stations of the lines, the travel times of train lines and, therefore, of the passengers are clearly not known. Yet, the passengers choose their own shortest path through the network. To model this, we construct the line-event graph. In this graph, every possible halt, train change, etc, is represented. For example, the line-event graph contains two nodes for every station that a line passes, with two pairs of opposing arcs between these nodes: a pair with a positive length representing a halting event (the halt arcs), and a pair with length zero (for not halting). It also contains arcs representing the trips between consecutive stations, as well as arcs modelling the changing between trains at a station (consider already Figure 6.2). In this line-event graph, depending on an assignment of types to stations, only a subset of the arcs is available. The length of an arc represents the amount of
time needed for the event. This concept is formalised below.

Consider the network graph $G = (V, E)$ and a set of lines $L$. The line-event graph $G^L = (V \cup V^L, A^L)$ is a directed graph that is defined on the node sets $V$ and $V^L$. The set of line-event nodes $V^L$ is defined as

$$V^L = \{(l, e, v) | l \in L, e \in E, v \in V : v \in e, e \in l\}.$$  

For every train line the set $V^L$ contains two nodes for every station it passes. The event tuple $i \in V^L$ is defined as the triple $i = (l, e, v_i)$. The nodes in the set $V$ are used in $G^L$ as general source/sink nodes for every station. We refer to these nodes as station nodes.

The overall arc set $A^L$ of line-event arcs is defined as the union of five sets:

$$A^L_d(e) = \{(i, j) | i, j \in V^L : i = l, e = e_i = e_j, v_i \neq v_j\}$$

$$A^L_c(v) = \{(i, j) | i, j \in V^L : i \neq l, v = v_i = v_j\}$$

$$A^L_n(v) = \{(i, j) | i, j \in V^L : i = l, e_i \neq e_j, v = v_i = v_j\}$$

$$A^L_p(v) = \{(i, v), (v, i) | i \in V^L : v = v\}$$

(6.1)

for all tracks $e \in E$ and stations $v \in V$. The set $A^L_d(e)$ models the possible driving trips across track $e$. We refer to these arcs as driving arcs. The changing of trains at $v$ is modelled by the change arcs in $A^L_c(v)$. The possible halting events and non-halting events at a station $v$ are given by the arcs in the sets $A^L_n(v)$ and $A^L_p(v)$ respectively. Finally, the possibilities for passengers to enter and leave trains at $v$ are modelled using the passenger arcs in $A^L_p(v)$. The overall set of all change arcs, i.e., the union of the sets $A^L(v)$ for all $v$, is denoted $A^L_v$. Analogously, we define the sets $A^L_d, A^L_n, A^L_p$ for the other line-event arcs.

Note that the set of non-halt arcs $A^L_n(v)$ is identical to the set of halt arcs $A^L_h(v)$. Note also the existence of arcs implied by these definitions. For example, the existence of two change arcs $(i, j)$ and $(j, k)$, both at some station $v$ and between lines $l_i \neq l_k$ implies that there also exists a change arc $(i, k)$ at $v$.

Example 6.1. Consider the network graph $G = (V, E)$ with stations $V = \{u, v, w\}$ and tracks $E = \{e, f\}$ with $e = \{u, v\}$ and $f = \{v, w\}$ as shown in Figure 6.1. There are two lines $l_1$ and $l_2$ operated on this network. The first line uses only edge $f$ and thus goes from $v$ to $w$ and vice versa, and is of type 1. The second line uses both tracks $e$ and $f$ and is of type 2.

The line-event graph $G^L = (V \cup V^L, A^L)$ for this instance is shown in Figure 6.2. The larger nodes are the nodes from $V = \{u, v, w\}$, the others are the line-event nodes from $V^L$.

For simplicity we have not shown the triplets $(l_i, e_i, v_1)$ but we have also numbered these nodes. The arcs between nodes 1 and 2 represent the driving of line $l_1$ from station $v$ to $w$ and vice versa. There are two pairs of arcs between nodes 4 and 5. One pair represents the halting of line $l_2$ at $v$ with a length of e.g., 2 minutes, either coming from $u$ and going to $w$, or the other way around. The other two arcs between vertices 4 and 5 are the non-halt arcs of length 0. The arcs between
vertices 1 and 4 represent the possibility for passengers to change from line \( l_1 \) to \( l_2 \) at \( v \) and vice versa. Finally, the arcs between node \( v \) and node 4 are there to model the possibilities for passengers to enter and to leave line \( l_2 \) at \( v \).

All commodities use the line-event graph to find their (shortest) path through the network. However, as discussed earlier, the availability of the arc pairs in the line-event graph depends on the types of the stations.

**Example 6.2.** Whether an event can occur—an arc can be used—depends on a given assignment of types. Consider the line-event graph of the previous example and the type-assignment vector \( t \) with \( t_u = 2 \), \( t_v = 1 \) and \( t_w = 2 \).

Since line 2 is of type 2, this line does not halt at \( v \) when \( v \) is of type 1. Therefore, neither the pairs of change arcs, nor the two pairs of halt arcs are available. Thus, for station \( v \), this assignment allows only the arcs \( (v, 1) \) and \( (1, v) \) and the non-halt arcs \( (4, 5) \) and \( (5, 4) \) to be used, as can be seen in Figure 6.3.

![Figure 6.3: The line-event graph, given that station \( v \) is of type 1.](image-url)
To model the availability of arcs, we introduce the subgraph $G^L(\bar{t})$ of the line-event graph $G^L$ for a given type-assignment vector $\bar{t}$ of types to the stations. The available arcs at station $v$, in case $v$ is of type $t = \bar{t}_v$, are given by the sets

\begin{align*}
A^L_c(v, t) &= \{(i, j) \in A^L_c(v) : t(l_i) \leq t, t(l_j) \leq t\} \\
A^L_h(v, t) &= \{(i, j) \in A^L_h(v) : t(l_i) \leq t\} \\
A^L_n(v, t) &= \{(i, j) \in A^L_n(v) : t(l_i) > t\} \\
A^L_p(v, t) &= \{(i, v) \in A^L_p(v) : t(l_i) \leq t\}
\end{align*}

(6.2)

If station $v$ is of type $t$, then the halt arcs, change arcs and passenger arcs of lines of type $t$ and lower are available. On the other hand, only the non-halt arcs of lines of types strictly larger than $t$ can be used. We also define the complementing sets of arcs, i.e., those arcs that are unavailable at a type strictly lower than $t$. For halt arcs, for example, these sets are given by $A^L_h(v, t) = A^L_h(v) \setminus A^L_h(v, t)$.

Using these sets, we define the graph $G^L(\bar{t}) = (V \cup V', A^L(\bar{t}))$. This graph contains only the arcs of $A^L$ that are available for a given assignment vector. In this definition $A^L(\bar{t})$ is the union of the above arc sets for $t = \bar{t}_v$ for all $v \in V$, together with the drive arcs of $A^L_d$. An assignment vector $\bar{t}$ is called feasible if between any pair of stations $v$ and $w$, there exists a path from $v$ to $w$ in $G^L(\bar{t})$.

From these definitions we can make the following observations that are used throughout this chapter.

**Observation 6.3.** The sets of change arcs, halt arcs and passenger arcs are nested in $t$ in the sense that for any station $v$ it holds that

$$A^L_r(v, t') \subseteq A^L_r(v, t)$$

for all $t' \leq t$ and $r \in \{c, h, p\}$.

The opposite relation is true for the non-halt arcs:

$$A^L_n(v, t') \supseteq A^L_n(v, t)$$

for all $t' \leq t$.

**Observation 6.4.** The sets of halt and non-halt arcs at some station $v$ are each other’s complements:

$$A^L_n(v, t) = A^L_h(v) \setminus A^L_h(v, t) = \bar{A}^L_h(v, t)$$

for all $t \in T$.

For a driving arc $(i, j) \in A^L_d(e)$, the length depends on the involved line $l_j (= l_i)$ and edge $e = \{v_i, v_j\}$ of the network graph $G$. Thus, the lengths of pairs of driving arcs are symmetric. The driving times are part of the input.

A change arc $(i, j) \in A^L_c(v)$ has a length that only depends on the line $l_j$ that is changed to. Given that line $l = l_j$ is operated at an hourly frequency of $f(l)$, we assume that these $f(l)$ trains per hour are evenly distributed across the hour, i.e., every $\frac{60}{f(l)}$ minutes. This leads to an expected change time of $\frac{60}{2f(l)}$.

The length of a halt arc $(i, j) \in A^L_h(v)$ of line $l$ at station $v$ is assumed to depend only on this line and station. The length of arc $(i, j)$ is assumed to be less than $\frac{60}{2f(l)}$. All the arcs of $A^L_n$ and $A^L_p$ have length 0.
6.2.2 Valid paths in the line-event graph

Given a type assignment \( \bar{t} \), a shortest path in \( \mathcal{G}^L(\bar{t}) \) is not necessarily a realistic path for real life travellers. Using the graph of Figure 6.1, consider the path shown in Figure 6.4. All the highlighted arcs are available for the assignment \( \bar{t} \). Still, the path for travellers from \( u \) to \( w \) is invalid because it describes a path in which the passengers first change from line \( l_2 \) to \( l_1 \) using arc \((6,2)\), and then use the leaving arc \((2,w)\) to exit the system. We define valid paths as follows.

**Definition 6.5.** A directed path \( P = (a_1, \ldots, a_{2m+1}) \subseteq A^L \) from station \( v \in V \) to station \( w \in V \) is a valid path if and only if

\[
\begin{align*}
a_1, a_{2m+1} &\in A^L_p \\
a_{2i} &\in A^L_d \\
a_{2i+1} &\in A^L_c \cup A^L_h \cup A^L_n
\end{align*}
\]

for all \( 1 \leq i \leq m \)

The arcs \( a_1 \) and \( a_{2m+1} \) are passenger arcs of \( A^L_p \) since, by construction, these are the only arcs incident to the station nodes \( v \in V \).

Finding a shortest valid path is polynomially equivalent to finding a shortest path. Consider Algorithm 6.1. After Step 1, the only remaining passenger arcs are those that are outgoing arcs at \( v \), and those that are incoming arcs at any station node other than \( v \). The algorithm used in Step 2 to find the shortest path with the smallest number of arcs requires only a simple modification of the well-known Dijkstra algorithm. Of all paths with minimal length, this algorithm returns the one with the smallest number of arcs. We now show that the shortest path \( P \) is also valid.
Lemma 6.6. The shortest path \( P = \{a_1, \ldots, a_{2m+1}\} \) as calculated by Algorithm 6.1 is a shortest valid path.

**Proof.** Hiding all outgoing passenger arcs at nodes other than \( v \) ensures the absence of internal passenger arcs in the shortest path from some station \( v \) to \( w \). The assumption that our path \( P \) is a shortest path with the smallest number of arcs, allows us to show that both arcs \( a_2 \) and \( a_{2m} \) can only be drive arcs of \( A^L_d \). Since similar reasoning holds for arc \( a_2 \) and arc \( a_{2m} \), we only discuss the latter case. Given that the arc \( a_{2m+1} = (j, w) \) is a passenger arc, consider the arc \( a_{2m} = (i, j) \), with \( v_i = v_j \). Since, by construction, there also exists a passenger arc \((i, w)\) of the same length as \((j, w)\), we could replace the arcs \(a_{2m}\) and \(a_{2m+1}\) by \((i, w)\) and thereby decrease the number of arcs in \(P\).

The combination of a halt arc \((i, j)\) and a change arc \((j, k)\) in \(P\) can be ruled out by similar reasoning: by construction there exists a change arc \((i, k)\) of the same length as \((j, k)\). Replacing the sequence \((i, j)\), \((j, k)\) by \((i, k)\) thus decreases the length of \(P\) by the (positive) length of \((i, j)\).

For a sequence of two change arcs, the argumentation is very similar. Now, however, we use the property that the length of a change arc depends only on the line that is changed to, not on the feeder line. Thus, two change arcs \((i, j)\) and \((j, k)\) from line \(l_i\) to \(l_j\) and from line \(l_j\) to \(l_k\) can always be replaced by the arc \((i, k)\).

To show that a pair of halt and non-halt arcs cannot occur consecutively in any path \(P\) in some \(G^L(\bar{t})\), observe that for some line \(l\) at a station \(v\) both pairs of arcs cannot be present simultaneously. This is, by construction, also true for non-halt arcs and change arcs.

\(\square\)

Lemma 6.7. For a line \(l\) and a station \(v\), every valid shortest path contains at most one arc of \(l\) at station \(v\), i.e., at most one arc \((i, j) \in A^L(v)\) for which \(l = l_i\) or \(l = l_j\).

**Proof.** If a halt arc or a non-halt arc of \(l\) is used, then no other arcs of \(l\) can be used at \(v\), since both line-event nodes have already been visited. In addition, the definition of a valid path tells us that at most one passenger arc at \(v\) can be used. Therefore, we are left with only two possibilities: a combination of a passenger arc and a change arc, or a combination of two change arcs. The first combination cannot be present in a shortest path, since also the passenger arc (of the same length) of the line that is changed from could have been used to exit the system. The usage of two change arcs can be ruled out by a similar exchange argument, since this valid path can be shortened by immediately changing from the first line to the last.

\(\square\)

6.3 Problem formulation

The number of travellers that want to travel from \(v\) to \(w\), i.e., the demand for commodity \(k = (v, w)\), is given by its entry \(H^{vw}\) in the origin-destination matrix \(H\). By using the directed graph \(G^L(\bar{t})\) for some type assignment \(\bar{t}\), we define the multi-commodity flow problem \(\text{KSTOP}(\bar{t})\) for routing the passengers (commodities)
through the network as follows:

$$z_{\text{KSTOP}(i)} = \min \sum_{k \in V \times V} dF^k$$

(6.3a)

s.t. $$NF^k = b^k$$ for all $$k \in V \times V$$

(6.3b)

$$0 \leq F^k_{ij}$$ for all $$k \in V \times V$$ and $$i, j \in A^L(i) : i \notin V \setminus \{v^k\}, j \neq v^k$$

(6.3c)

where the variables $$F^k_{ij}$$ represent the amount of flow of commodity $$k$$ across arc $$(i, j)$$ in the network. In this sense, $$F^k$$ is the vector of all flow variables for all arcs of commodity $$k$$. The vector $$d$$ gives the travel times for every line-event arc in $$A^L(i)$$.

The matrix $$N$$ is the node-arc incidence matrix of $$A^L(i)$$. Together with the column vector $$b^k$$ of size $$|V \cup V^L|$$, it builds the flow balance constraints (6.3b). Note that for every commodity $$k = (v, w)$$, the vector $$b^k$$ of size $$|V|$$ contains only two nonzero entries: a positive entry $$H^v$$ at $$v$$, and $$-H^w$$ at $$w$$. The restrictions (6.3c) impose that the commodity $$k = (v^k, w^k)$$ can use arcs that are not outgoing from a station node, except $$v^k$$, and that are also not incoming arcs at $$v^k$$. The flow on arcs for which these conditions do not hold is set to 0. These constraints are identical to the restrictions in Algorithm 6.1.

Note the absence of capacity restrictions on the amounts of flow in (6.3). As mentioned in §6.2, we assume that the capacities of the train lines are sufficient to meet the passenger demand. This is in contrast with the capacity shortages as remarked in §1.1. However, the station type optimisation problem is designed to focus on the passengers and their travel times.

**Lemma 6.8.** For every feasible solution $$F^k_{ij}$$ of KSTOP(i) there exists a solution with the same cost per commodity, but for which for every commodity the arcs with positive flow make up a valid path through the network.

**Proof.** Since KSTOP(i) does not enforce capacity restrictions on the arcs of the flow problem, it follows that Observation 2.4 and Observation 2.3 hold. The arcs-usage restrictions in (6.3c) allow the same arcs to be used as in Step 1 of Algorithm 6.1. It follows from Lemma 6.6 that if there exists a shortest path between a pair of nodes in this restricted graph of Step 1, then there exists a shortest valid path of equal length.

As the model formulation lacks restrictions that tie the different commodities together, the minimisation problems for the different commodities are independent. Every commodity chooses the cheapest route(s) according to the arc lengths $$d$$ through the network. Consequently, these paths make up a feasible and optimal solution.

Lemma 6.8 shows that an optimal solution to the multi-commodity flow problem of KSTOP(i) can be found by solving a shortest path problem in $$G^L(i)$$ for every commodity $$k \in V \times V$$, considering only the available arcs in (6.3c). Thus, the set of arcs with positive flow of the commodities with source $$u$$ form a shortest path tree rooted at $$u$$. This leads to the following equivalent reformulation of KSTOP(i) called STOP(i). This formulation uses the flow variables $$F^u_{ij}$$ for $$u \in V$$ that represent the
6.3. PROBLEM FORMULATION

flow across arc \((i,j)\) that originates at \(u\).

\[
z_{\text{STOP}(\bar{t})} = \min \sum_{u \in V} dF^u
\]

s.t. \(NF^u = b^u\) for all \(u \in V\)

\[
0 \leq F^u_{ij} \leq b^u_{ij} \quad \text{for all } (i,j) \in A\bar{L}(\bar{t}) : i \notin V \setminus \{u\}, j \neq u
\]

where again the vector \(d\) contains the costs per arc in \(A\bar{L}(\bar{t})\). We use this reformulation to describe a model for the station type optimisation problem, i.e., optimising \(\text{STOP}(\bar{t})\) over the type assignments \(\bar{t}\).

The next section introduces a formulation for solving \(\min_{\bar{t}} \text{STOP}(\bar{t})\) as a mixed integer programming problem.

6.3.1 Extending the \(\text{STOP}(\bar{t})\) formulation

To model the variable type \(t\) of a station \(v\), we introduce additional binary variables \(x_t^v\), where

\[
x_t^v = \begin{cases} 
1 & \text{if station } v \text{ is of type } t, \text{ and} \\
0 & \text{otherwise.}
\end{cases}
\]

Two new classes of constraints are used to link the station type variables to the flow variables, allowing only a flow that is consistent with the value of the \(x_t^v\) variables. This formulation is called \(\text{STOP}\).

\[
\begin{align*}
\text{min} & \quad \sum_{u \in V} dF^u \\
\text{s.t.} & \quad NF^u = b^u \quad \text{for all } u \in V \quad (6.4a) \\
& \quad \sum_{u \in V} \sum_{a \in A^L_h(v,t)} F^u_{a,i} \leq M \sum_{t' < t} x_{t'}^v \quad \text{for all } v \in V, t \in T \text{ and } r \in \{h,c,p\} \quad (6.4c) \\
& \quad \sum_{u \in V} \sum_{a \in A^L_c(v,t)} F^u_{a,i} \leq M \sum_{t' < t} x_{t'}^v \quad \text{for all } v \in V, t \in T \quad (6.4d) \\
& \quad \sum_{t \in T} x_t^v = 1 \quad \text{for all } v \in V \quad (6.4e) \\
& \quad 0 \leq F^u_{ij} \quad \text{for all } (i,j) \in A\bar{L}(\bar{t}) : i \notin V \setminus \{u\}, j \neq u \quad (6.4f) \\
& \quad x_t^v \in \{0,1\} \quad \text{for all } v \in V, t \in T \quad (6.4g)
\end{align*}
\]

We refer to the optimal objective function value of (6.4) as \(z_{\text{STOP}}\). Constraint (6.4c) for all arcs enforces that if a station \(v\) is of some type \(t\) or lower, in which case \(\sum_{t' > t} x_{t'}^v\) is zero, then no flow is allowed across the arcs in \(A^L_h(v) \setminus A^L_c(v,t) = \bar{A}^L_h(v,t)\), i.e., those that are unavailable at type \(t\) or lower.

**Theorem 6.9.** The formulations \(\text{STOP}\) and \(\min_{\bar{t}} \text{STOP}(\bar{t})\) are equivalent.
Proof. The additional restrictions of (6.4c)-(6.4d) are necessary restrictions to hold for any feasible solution $F$ of STOP($\bar{t}$) for a vector $\bar{t}$. Thus, the flow $F$ automatically satisfies the additional restrictions of STOP with $x^v_i = 1$ for $t = \bar{t}_v$ and zero otherwise.

Vice versa, consider some feasible solution $(F, x)$ of STOP. We construct a vector $\bar{t}$ such that $F$ is feasible for STOP($\bar{t}$). Because of (6.4e), we set $\bar{t}_v = t$ for those $x^v_i = 1$. It remains to be shown that only for arcs $a \in A^L(\bar{t})$ the flow $F^u_a$ is strictly positive for any station $u$. By contradiction, assume that the solution $(F, x)$ has some $F^u_a > 0$ for an arc $a$ at some station $v$ for which $x^v_i = 1$, but with $a \notin A^L(v, t)$. The following argument can similarly be applied to (non-)halt, change and passenger arcs. Consider the case where $a$ is a non-halt arc. It follows that $a \in A^L(v, t)$ since $a \notin A^L(v, t)$. However, since by assumption $x^v_i = 1$, this implies that the right-hand side of (6.4c) is equal to zero, and thus $F^u_a \leq 0$. Clearly, this contradicts that $(F, x)$ is feasible in STOP.

The problem formulation STOP allows us to model the station type optimisation problem. The difficulty, however, lies in the number of variables needed in the modelling. With the number of arcs in the line-event graph $G^L$ of order $O(mk + nk^2)$, the number of flow variables $F^u_{ij}$ is of order $O(n \times (mk + nk^2))$, where $n = |V|$, $m = |E|$ and $k = |L|$. Even though this is still polynomial in $n$, $m$ and $k$, the number of variables needed to model practical instances is enormous (see §6.6). The next section considers Lagrangian relaxation on the complicating constraints (6.4c)-(6.4d). This allows us to solve the STOP formulation by considering only small independent subproblems.

### 6.4 Lagrangian relaxation

Consider dualising the complicating additional constraints (6.4c)-(6.4d). Since these are the only constraints linking the flow variables of different origin stations, this makes the remaining problem separable. Therefore, let us study the following relaxation of STOP, referred to as LR($\lambda$).

\[
\begin{align*}
    z_{LR}(\lambda) = \min & \sum_{u \in V} dF^u + \lambda \left( \sum_{u \in V} A(u)F^u - \sum_{v \in V} B(v)x^v \right) \\
    \text{s.t.} & \quad NF^u = b^u & \text{for all } u \in V \\
    & \quad Cx^v = 1 & \text{for all } v \in V \\
    & \quad 0 \leq F^u_{ij} & \text{for all } u \in V \text{ and } (i, j) \in A^L(\bar{t}) \setminus \{u\}, j \neq u \\
    & \quad x^v_i \in \{0, 1\} & \text{for all } v \in V, t \in T
\end{align*}
\]

(6.5)

where the matrix $A$ is an $m$ by $|V| \cdot |A^L|$ matrix, and $B$ is an $m \times (|V| \cdot T_{\text{max}})$ matrix, with $m$ equal to the number of additional constraints. We define $A(v)$ and $B(v)$ as the submatrices of $A$ and $B$ that contain only the columns for some station $v \in V$. Thus, for example, $B(v)$ is of size $m$ by $T_{\text{max}}$, and holds for station $v$ the big $M$ values of the restrictions that contain the $x^v_i$ variables. Note that the only nonzero
elements that \( B(v) \) contains are at the restrictions for station \( v \). The class of constraints \( Cx^v = 1 \) represents the constraints for choosing exactly one type for every station as in (6.4e). The vector \( \lambda \in \mathbb{R}_+^m \) is a multiplier vector with one element for every of the \( m \) additional constraints. Lagrangian theory tells us that for any non-negative vector \( \lambda \), the value \( z_{LR}(\lambda) \) is a lower bound for the objective value \( z_{\text{STOP}} \) of the original problem. In particular, recall the following well-known theorem in optimisation theory.

**Theorem 6.10 (Kuhn and Tucker [45]).** For a given vector \( \lambda \), if the optimal solution of LR(\( \lambda \)) is feasible in the original problem, and if \( \lambda \) and the dualised constraints satisfy the complementary slackness conditions, then the solution is also optimal for the original problem.

We use this lower bound in a branch and bound scheme, and we are therefore interested in the vector \( \lambda \) that maximises \( z_{LR}(\lambda) \). For a review of Lagrangian relaxation and duality, we refer to Nemhauser and Wolsey [55, Page 323]. First, consider solving LR(\( \lambda \)) for a given \( \lambda \). We can rewrite and solve LR(\( \lambda \)) as

\[
z_{LR}(\lambda) = \min \ z_{LRF}(\lambda) - z_{LRX}(\lambda)
\]

with the two subproblems LR\( F(\lambda) \) and LR\( X(\lambda) \) defined as

\[
z_{LRF}(\lambda) = \min_{v \in V} \sum_{u \in V} (d + \lambda A(u))F^u
\]

s.t. \( NF^u = b^u \) for all \( u \in V \)

\( 0 \leq F^u_{ij} \) for all \( u \in V \) and

\( (i, j) \in A_L(i) : i \notin V \setminus \{u\}, j \neq u \)  \hspace{1cm} \text{(6.6)}

and

\[
z_{LRX}(\lambda) = \max_{v \in V} \sum_{v \in V} \lambda B(v)x^v
\]

s.t. \( Cx^v = 1 \) for all \( v \in V \)

\( x^v_t \in \{0, 1\} \) for all \( v \in V, t \in T \)  \hspace{1cm} \text{(6.7)}

First consider the maximisation problem (6.7). Since the constraints \( Cx^v = 1 \) impose that for every station \( v \) we choose exactly one station type, this problem can be solved to optimality by inspection:

\[
x^v_t = \begin{cases} 
1 & \text{if } t = \text{argmax}_{t' \in T} \lambda B(v, t'), \\
0 & \text{otherwise.}
\end{cases}
\]

\hspace{1cm} \text{for all } v \in V \hspace{1cm} \text{(6.8)}

where \( \lambda B(v, t') \) is the objective function coefficient of variable \( x^v_t \).

The subproblem (6.6) is also much easier to solve than the original problem. Given a vector \( \lambda \), this problem is similar to \text{STOP}(t), but with all arcs in \( A_L^t \). The objective function coefficients \( d + \lambda A(u) \) make up a new vector of the arc costs per unit of flow.
originating from station \( v \). However, the matrix \( A(u) \) is based on the arc-presence in the sets \( \mathcal{A}_L^u(v, t) \). Thus, for all pairs of stations \( u \) and \( u' \) the constraint matrices are equal: \( A(u) = A(u') \). This is easy to see, since a flow variable \( F^u_a \) appears in the constraints based on the arc \( a \) for which it is defined, but irrespective of its origin station \( u \).

If we define the new arc costs depending on \( \lambda \) as \( d(\lambda) = d + \lambda A(\cdot) \), then the optimal routing in the multi-commodity flow problem can be found by solving one all-pairs shortest path problem using \( d(\lambda) \) as arc costs. Given the shortest paths \( P_{uv} \subseteq A^L \), the flows \( F \) can be reconstructed as \( F^u_a = \sum_{w \in V : a \in P_{uw}} H^{uw} \).

In §6.5.2, we consider a standard subgradient based optimisation method, and a problem-specific method to find good multiplier vectors. First, however, consider the following well-known result by Geoffrion [36].

**Theorem 6.11 (Geoffrion [36]).** If for all vectors \( \lambda \geq 0 \) the optimal value of \( z_{LR}(\lambda) \) is not altered by dropping the integrality conditions on its variables, then the value of the Lagrangian dual problem (LD)

\[
\max_{\lambda \in \mathbb{R}_+^m} z_{LR}(\lambda)
\]

is equal to the optimal value of the linear programming relaxation of the original problem.

For the integrality restriction of (6.7), it is easy to see that all extreme points of \( \{ x \in \mathbb{R}^{|V|T_{\text{max}}} : Cx = 1, 0 \leq x \leq 1 \} \) are integer, since every variable is only part of exactly one of the constraints. Therefore, the following is true for the Lagrangian dual LD.

**Corollary 6.12.** The value of \( z_{LD} \) is equal to the value of the linear programming relaxation of STOP.

This bound offers a way to test the quality of algorithms for finding a good multiplier vector \( \lambda \).

### 6.5 The branch-and-bound algorithm

As mentioned in the previous section, STOP cannot be solved by only optimising the Lagrangian dual problem of (6.9). Therefore, we apply branch-and-bound on the \( x^v_l \) variables, using the Lagrangian relaxation LR(\( \lambda \)) for calculating lower bounds. The details of the branch-and-bound algorithm are described in three sections. First, we discuss preprocessing techniques. Then, the details of using the Lagrangian relaxation for deriving good dual bounds are presented. Finally, the section on tree search covers branching rules and primal heuristics.

#### 6.5.1 Preprocessing

Strengthening the initial problem formulation is done on three levels. The next section discusses techniques for reducing the size of the line-event graph.
6.5. THE BRANCH-AND-BOUND ALGORITHM

Line-event graph reduction

Identifying redundant lines in \( L \) can significantly reduce the size of the line-event graph.

**Lemma 6.13.** Consider two distinct lines \( l \) and \( l' \) for which the path of line \( l' \) in the network graph \( G \) is contained in the path of \( l \). If for every arc \( a' = ((l', e_1, v_1), (l', e_2, v_2)) \) of line \( l' \) the following holds

\[
d_a \leq d_{a'}
\]

for all \( a = ((l, e_1, v_1), (l, e_2, v_2)) \), then line \( l' \) is dominated by line \( l \), and can be removed from the problem description.

**Proof.** We show that every arc of line \( l' \) can be replaced by an arc of \( l \) that is not longer. This follows immediately from the fact that the path of \( l' \) is contained in the path of \( l \), and the presence of arcs as implied by definition (6.1). Any solution \((F^*, x)\) of STOP, that uses arcs of \( l' \), can thus be replaced by a solution \((F, x)\), where the flow on arcs of \( l' \) is replaced by arcs of \( l \). \( \square \)

Due to the structure of the line-event graph, and the assumptions we have made concerning the lengths of its arcs, sufficient conditions for line dominance can easily be derived.

**Corollary 6.14.** Consider two distinct lines \( l \) and \( l' \) for which the path of \( l' \) in the network graph \( G \) is contained in the path of \( l \). If \( t(l) = t(l') \) and \( f(l) \geq f(l') \), then \( l' \) is dominated by \( l \).

Reducing the number of dualised constraints

The Lagrangian relaxation formulation presented in §6.4 is defined on the line-event graph, and the dualised constraints. The number of these constraints, i.e., the number of constraints that is relaxed, depends on the possible types \( T(v) \) for the stations \( v \) in the network. These sets can be taken equal to the set of all available types \( T \). However, the sizes of these sets can be reduced by using several intuitive methods. We discuss three such ideas.

Consider a station \( v \) and the lines \( L(v) \subseteq L \) for which \( v \) is part of their route. Making \( v \) of a type that is higher than any of the lines is not useful. Similarly, a type of \( v \) that is lower than the lowest type of the lines in \( L(v) \) would induce that no line halts at \( v \).

The given set of operated lines implies that the start and end stations are available for these lines. In other words, the type of a station must be at least equal to the type of the lines that start or end there.

In case a station \( v \) is fixed to type \( t \in T(v) \), then by definition the arcs of \( A^L(v, t) \) are available at \( v \). If, for some other type \( t' \in T(v) \), the available set \( A^L(v, t') \) is a subset of \( A^L(v, t) \), then type \( t' \) can be removed from the set of available types of \( T(v) \).

Fixing a station \( v \) at some type \( t \in T(v) \) can make \( G^L \) unconnected. In this case, any assignment \( t \) with \( v \) at this type would be infeasible, and therefore \( t \) can be removed from \( T(v) \). This holds in particular for the smallest type in \( T(v) \).
Figure 6.5: Setting station b to type 1 causes line \( l_2 \) not to halt at b.

Example 6.15. Consider the following example with four stations \( V = \{1, 2, 3, 4\} \), three tracks \( E = \{\{1, 2\}, \{2, 3\}, \{2, 4\}\} \) and two lines, \( l_1 \) of type 1, operated between stations b and d, and \( l_2 \) of type 2 operated between a and c. Using the rules given above, we can set \( T(2) = \{1, 2\} \). However, if we consider the case where station b is of type 1, then this causes line \( l_2 \) not to halt at b. Thus, the travellers from b and d cannot reach a or c, and vice versa. Thus, type 1 is not a valid option for b.

Coefficient reduction

We propose a formulation specific preprocessing rule to strengthen the big M constraints of (6.4c)-(6.4d):

\[
\sum_{u \in V} \sum_{a \in A^u_L(v,t)} F^u_a \leq M \sum_{t' > t} x^v_{t'} \quad \text{for all } v \in V, t \in T(v) \tag{6.10}
\]

\[
\sum_{u \in V} \sum_{a \in A^u_L(v,t)} F^u_a \leq M \sum_{t' > t} x^v_{t'} \quad \text{for all } v \in V, t \in T(v) \tag{6.11}
\]

\[
\sum_{u \in V} \sum_{a \in A^u_L(v,t)} F^u_a \leq M \sum_{t' > t} x^v_{t'} \quad \text{for all } v \in V, t \in T(v) \tag{6.12}
\]

\[
\sum_{u \in V} \sum_{a \in A^u_L(v,t)} F^u_a \leq M \sum_{t' < t} x^v_{t'} \quad \text{for all } v \in V, t \in T(v) \tag{6.13}
\]

Given a station \( u \) and type \( t \), note that the right-hand sides of the first three constraints are identical. In order to tighten the values of \( M \), let us recall Lemma 6.6, i.e., the valid path lemma. From this we can show that non-halt arcs are the only arcs of which travellers originating from some station \( v \) can possibly use more than one arc at any station in the network. This is formalised in the following lemma.

Lemma 6.16. A shortest valid path of a commodity contains at most one arc \( a \in A^L(v) \setminus A^L_{n_0}(v) \) for every station \( v \in V \).

Proof. By contradiction, assume that two or more of these arcs are used. From Lemma 6.7 it follows that at least two lines are involved. Since at some station, \( v \) or other, the path changes from one line to another, this change could also have been
made immediately at \( v \), since the assumption of §6.2.2 tells that the time needed to change lines depends only on the line that is changed to. Thus, a valid path cannot be a shortest valid path if two or more arcs of \( A^L(v) \backslash A^L_n(v) \) are used at \( v \in V \). □

This lemma shows that any passenger can use at most one change or halt arc at any station. Therefore, we can merge (6.10) and (6.11) to

\[
\sum_{u \in V} \sum_{a \in A^L_n(v,t)} F^u_a + \sum_{u \in V} \sum_{a \in A^L_n(v,t)} F^u_a \leq M \sum_{t' > t} x^v_{t'} \text{ for all } v \in V, t \in T(v) \tag{6.14}
\]

where, for a station \( v \), for the value of \( M \) we can use the total number of travellers in the system except the people entering or leaving the system at \( u \):

\[
M \leftarrow \sum_{u, w \in V \backslash \{v\}} H^{uw} \text{ for all } v \in V, t \in T(v) \tag{6.15}
\]

Passengers entering or leaving at \( v \) use a passenger arc, and can thus not use any change or halt arcs according to Lemma 6.16.

For the passenger arcs at \( u \), we already know from Lemma 6.16 that at most

\[
M \leftarrow \sum_{u \in V} (H^{vu} + H^{uv}) \text{ for all } v \in V, t \in T(v) \tag{6.16}
\]

units of flow can pass on the arcs entering and leaving \( v \) in the line-event graph. Thus, the \( M \) for station \( v \) in constraint (6.12) can be replaced by this quantity.

The number of non-halt arcs used at a station can be more than one. Consider the following example.

**Example 6.17.** Let us study an instance with four stations and two lines as shown in Figure 6.6. In case the type of station \( b \) is less than the type of \( l_1 \) and of \( l_2 \), then travellers that want to travel from \( a \) to \( d \) have to use two non-halt arcs at \( b \).

From Lemma 6.7 it follows that at most one of the non-halt arcs of a line at a station can be used. Therefore, the total flow across the arcs in \( A^L_n(v,t) \) is at most

\[
M \leftarrow |L(v)| \left( \sum_{u \in V} (H^{vu} + H^{uv}) + \sum_{u, w \in V \backslash \{v\}} H^{uw} \right) \text{ for all } v \in V, t \in T(v) \tag{6.17}
\]
Note that the right-hand side sum of elements of $H$ is equal to the total number of travellers in the system.

We now derive bounds for the maximum amount of flow via station $v$, i.e.,

$$\sum_{u, w \in V \setminus \{v\}} H_{uw}.$$ 

The technique derives guaranteed maximum travel times for traversing the network, and uses these to determine if stations in the network can ever be passed by travellers of a commodity in any feasible solution. First, we derive lower bounds on the travel times between stations in $G^L(t)$ for any assignment $t$.

**Lemma 6.18.** The shortest valid path distance between two nodes $v$ and $w$ in $G^L$ is a lower bound for the shortest valid path distance between $v$ and $w$ in $G^L(t)$ for any assignment $t$.

**Proof.** Trivial, since $G^L(t)$ is a subgraph of $G^L$. \hfill \square

Next, consider the shortest path distance in the following graph that is derived from the line-event graph. The graph $D$ is equal to $G^L(t)$ for $t_v = \min\{t \in T(v)\}$. Thus, $D$ contains the arcs that are available in case all stations are assigned to their lowest type. In contrast to the arcs in the line-event graph, the non-halt arcs in $D$ are of a length that is equal to the length of their (unavailable) halt counterpart in $A^L_h(v, t_v)$.

**Lemma 6.19.** Consider two nodes $v$ and $w$ for which a shortest valid path in $D$ exists, with length $s$. The length of the shortest valid path between $v$ and $w$ in $G^L(t)$ for any feasible assignment $t$ is at most $s$.

**Proof.** We prove this lemma by contradiction. Define $P$ to be the shortest valid path from $v$ to $w$ in $D$. If a shortest valid $v$-$w$ path $P'$ in $G^L(t)$ is longer than $s$, then either some arc lengths in $G^L(t)$ are higher than in $D$, or $P$ contains arcs that are not present in $G^L(t)$. From Observation 6.3 it is clear that, by construction, the only arcs of $D$ that can be unavailable in $G^L(t)$ are non-halt arcs. However, the complementarity relation of Observation 6.4 shows that for every unavailable non-halt arc there is an available halt arc. Since, in $D$, these arcs are of the same length it follows that $P'$ cannot be longer than $P$. \hfill \square

This lemma leads immediately to the following corollary.

**Corollary 6.20.** Consider two nodes $v$ and $w$ for which a shortest valid path in $D$ exists, with length $s$. Any $v$-$w$ path in some $G^L(t)$ that is strictly longer than $s$ is not a shortest valid path in this graph.

Using the lower and upper bounds on the lengths of the shortest valid paths, we can prove the following theorem.

**Theorem 6.21.** Consider two nodes $v$ and $w$ for which a shortest valid path in $D$ exists, with length $s$. If the shortest valid path in $G^L$ from $v$ to $u$, combined with the shortest valid path from $u$ to $w$ is strictly longer than $s$, then $u$ cannot be part of the shortest valid path from $v$ to $w$ in $G^L(t)$ for any assignment $t$. 
Proof. From the assumption, and Lemma 6.18 it follows that a similar path from \( v \) to \( w \) via \( u \) in any \( G^L(t) \) is at least as long as that in \( G^L \), and thus also longer than \( s \). The remainder follows from Corollary 6.20. \( \square \)

Thus, we can bound the number of travellers passing some station in any type assignment. Only travellers for whom the conditions in Theorem 6.21 for some station \( v \) do not hold can possibly travel via \( v \). The total number of these passengers for station \( v \) is used in (6.15) and (6.17) to replace the previous bound \( \sum_{u,w \in V \setminus \{v\}} H_{uw}^v \) on the number of travellers via \( v \).

6.5.2 Bounding methods

The Lagrangian relaxation formulation is used to find lower bounds for the optimal objective value in a branch-and-bound setting. Since the size of the enumeration tree strongly depends on the quality of these lower bounds, this section describes a problem-specific algorithm for finding good multiplier vectors for the Lagrangian relaxation. This method is based on a sensitivity analysis of the arc costs. We have also implemented the subgradient-based Revised Volume Algorithm (RVA) as described in Bahiense et al. [7]. The algorithm can be found in Algorithm D.1 on page 140 in Appendix D. At the end of this section we discuss several issues related to using Lagrangian relaxation for bounding in general.

Problem-specific multiplier adjustment algorithm

As we have seen previously, solving the Lagrangian relaxation problem for a given multiplier vector \( \lambda \) can be done by solving two polynomially solvable problems. This section describes a technique for finding good values for \( \lambda \), that is based on a sensitivity analysis of the shortest paths resulting from optimising the flow problem LRF(\( \lambda \)) in (6.6). We refer to this multiplier adjustment algorithm as MAA.

First, recall the structure of the dualised constraints. In the original formulation of (6.4), each restriction relates the flow across a set of arcs to the variables modelling the station types. Following the dualisation, the multiplier \( \lambda_r \) of constraint \( r = 1, \ldots, m \) thus contributes to the cost of a group of arcs in the problem (6.5). For simplicity, we refer to this set of arcs as \( A^L_r \). Conversely, a specific arc can be contained in the arc set of more than one constraint. The relation between the multipliers \( \lambda \) of the constraints and the cost per unit per arc is given by the vector \( d(\lambda) \). The set of constraints that are part of the restrictions of some station \( u \) are given by the set \( R(u) \).

For suitably small changes in \( \lambda \), and so for small changes in the arc costs \( d(\lambda)_a \), the optimal flow \( F \) of LRF(\( \lambda \)) does not change. Therefore, we propose to do a sensitivity analysis on the arc costs \( d(\lambda)_a \). This analysis results in a lower bound \( \delta^-_a \), and an upper bound \( \delta^+_a \) for every arc \( a \). If we consider one arc \( a \), then the flow solution \( F \) is optimal for any cost of \( a \) between these two bounds, given that all other arc costs remain unchanged.

A sensitivity analysis on the cost of an arc \( a \) with respect to an optimal flow \( F \), is equivalent to an arc cost analysis for shortest paths. This is a consequence of
CHAPTER 6. THE STATION TYPE OPTIMISATION PROBLEM

Observation 2.3 for min-cost flow problems in uncapacitated networks. We present a specialised algorithm for performing this analysis. The special structure of the underlying problem gives us the opportunity to derive better (wider) bounds on the allowed changes in arc costs. Of particular use is the fact that the costs of arcs are not changed individually, but are altered as a consequence of a change in a constraint multiplier that affects a set of arcs at the same time. A second important aspect is the valid path property that must hold for the paths of travellers through the network. The proposed sensitivity analysis algorithm is outlined in Algorithm D.2 on page 141 in Appendix D.

We assume that the optimal flow \( F \) does not change if the implied changes in all the arc costs remain within the sensitivity bounds. This allows us to model the effects of changes of the multiplier vector to changes in the value of the Lagrangian dual. The following linear program determines a good update \( \lambda \leftarrow \lambda + \Delta \) by building the vector \( \Delta \) station by station.

Given a flow \( F \), a change \( \Delta_r \) in the multiplier of some constraint \( r \) has a two-sided effect on the objective function value of (6.5). Firstly, through (6.6), it results in a change of

\[
\Delta_r(AF)_r = \Delta_r \sum_{a \in A^L(r)} F(a) = \Delta_r c_r
\]

where \( F(a) \) is the total flow of all origin stations on arc \( a \).

Secondly, via (6.7) there can be a change in the optimal solution \( x \). The new vector \( \Delta \) appears in the objective function of (6.7). As shown in (6.8), the optimal solution to this problem is found by station-wise setting the \( x^*_t \) to one for which the objective function coefficient is highest.

The overall effect of a variable \( \Delta_r \) on the two subproblems can be expressed using linear restrictions. For the constraints of a station \( u \), the problem of finding the vector elements \( \Delta \) that maximise the objective value of the Lagrangian relaxation is modelled as follows

\[
\text{max} \sum_{r \in R(u)} c_r \Delta_r - Z \quad (6.18a)
\]

\[
\text{s.t.} \quad \delta^-_a \leq \sum_{r, a \in A^L(r)} \Delta_r \leq \delta^+_a \quad \text{for all } a \in A^L(u) \quad (6.18b)
\]

\[
(\lambda + \Delta)B(u, t) \leq Z \quad \text{for all } t \in T \quad (6.18c)
\]

\[
\Delta_r \in \mathbb{R}_+ \quad \text{for all } r \in R(u) \quad (6.18d)
\]

\[
Z \in \mathbb{R}_+ \quad (6.18e)
\]

The variable \( Z \) is used to model the optimal objective function value \( z_{LRX}(\lambda) \) of (6.7). The constraints (6.18b) link the change in the multiplier of every constraint \( r \) to the upper and lower bounds \( \delta^+_a \) and \( \delta^-_a \) for arcs on which \( r \) is defined. According to (6.8), the optimal solution for station \( u \) takes the type \( t \) that maximises \( (\lambda + \Delta)B(u, t) \). The restrictions (6.18c) model this by ensuring that \( Z \) is at least as large as \( (\lambda + \Delta)B(u, t) \) for any type \( t \) of station \( u \). Since \( Z \) has a negative coefficient in the
objective function of the maximisation problem here, this ensures that $Z$ actually attains the correct value.

The bounds derived by the sensitivity analysis are only valid for changes in the costs of individual arcs. Changing the costs of many arcs, as we propose here, can therefore not guarantee an improvement. However, consider only the arcs $R(u)$ at station $u$. Lemma 6.16 shows that every valid shortest path can contain at most one of them, except for the non-halt arcs. Therefore, we could propose to iteratively solve $LR(\lambda)$ for the new vector $\lambda \leftarrow \lambda + \Delta$, for changes made to the arcs of only one station. However, extensive initial experiments have shown that solving for all changes at the same time gives better results. This procedure is shown in Line 30 and Line 34 of Algorithm D.2.

Using Lagrangian relaxation for branch-and-bound

A solution $(F, x)$ of $LR(\lambda)$ for some vector $\lambda$ is used in the primal heuristic of §6.5.3. Alternatively, in case the solution $(F, x)$ is primal feasible, then such a solution is only optimal for the subproblem, if $(F, x)$ is an optimal solution of $LR(\lambda)$ and if the complementary slackness conditions hold, according to Theorem 2.1. The optimality condition is satisfied automatically, by the algorithms used to solve $LR(\lambda)$. In case the complementary slackness conditions are not satisfied, then this means that the Lagrangian relaxation lower bound resulted in a feasible solution, but there is no guarantee that this solution is the best feasible solution for this subproblem.

When some station $u$ is branched on, then its $x_u^v$ variables and the constraints $R(u)$ can be removed from the problem description. The remaining question is what happens to the lower bound in the child node. Let us compare the value of the lower bound for a vector $\lambda^*$ in the parent node and in the child node.

Lemma 6.22. For a given vector $\lambda \geq 0$, removing any redundant dualised restriction $r$ from $LR(\lambda)$ increases the value of the Lagrangian bound from $z_{LR}(\lambda)$ to $z_{LR}(\lambda')$ for $\lambda'$ equal to $\lambda$ with element $\lambda'_r = 0$.

Proof. The objective function of $LR(\lambda')$ is

$$z_{LR}(\lambda') = \min_{F, x} \sum_{v \in V} dF^v + \lambda' (AF - Bx)$$

$$= \min_{F, x} \sum_{v \in V} dF^v + \lambda (AF - Bx) - \lambda_r (AF - Bx)_r.$$ 

Since $r$ is redundant, it follows that $(AF)_r \leq (Bx)_r$ for any solution $(F, x)$, and thus $z_{LR}(\lambda) \leq z_{LR}(\lambda')$ for any $\lambda \geq 0$. \hfill $\square$

This lemma shows that removing any obsolete restrictions from the dualised constraints by fixing their multiplier to zero does not lower the optimal solution value of the minimisation problem for the Lagrangian relaxation.

Corollary 6.23. Consider a parent and child node in the branch-and-bound tree for which a vector $\lambda$ results in a lower bound of $z_p \equiv z_{LR}(\lambda)$ in the parent node. In the child node, after fixing the variables and removing the redundant constraints by setting the elements of $\lambda$ to zero, the Lagrangian relaxation lower bound is at least $z_p$. 
6.5.3 Tree search

Next, we discuss the construction of the enumeration tree and the search process through it.

Branching

Branching rules are used to split a problem into several new subproblems that are easier to solve than the original problem. Normal variable branching for binary variables creates two new subproblems, one for every possible value of the binary variable. Alternatively, we propose to create at most $T_{\text{max}}$ new subproblems, where in every new node a chosen station $u$ is assigned to be of type $t$ with $t \in T(u)$. Clearly, the station to use for branching can be chosen in an number of ways.

Branching rules influence the number of nodes of the branch-and-bound tree. Because the size of the enumeration tree strongly depends on how quickly the upper and lower bounds converge, we prefer a branching station that improves these bounds (see also Linderoth and Savelsbergh [47]). A difficulty of the STOP problem is that many of the variables—in particular, all $x$ variables, and the flows on the non-halt arcs and passenger arcs—do not appear in the objective function (6.4a). We consider three branching rules, given a solution $(F, x)$ of the Lagrangian relaxation problem for the multiplier vector $\lambda$.

Maximum unavailable flow branching For every station $u$, we consider the current amount of flow of $F$ that would become unavailable if $u$ were to be of type $t$ in $T(u)$. The quality of branching on $u$ is given by the smallest amount of unavailable flow over the types in $T(u)$. Thus, we branch on the station for which

$$\min_{t \in T(u)} \sum_{a \in A^t(u, t)} F(a)$$

is maximal.

Constraint based branching Consider the vector $AF - Bx$ that represents the dualised constraints. A positive entry shows a violated constraint $AF \leq Bx$ of (6.5). Every constraint can be contributed to exactly one station. This branching rule proposes to branch on the station with the maximally violated constraint, i.e., the station $u$ for which

$$(AF - Bx)_r$$

is maximal.

Degradation estimated branching Throughout the branch-and-bound tree, it is likely that a station is branched on more than once. To estimate the increase in the lower bound of the new child problems when branching on some station $u$, this branching rule uses the average increases over previous nodes where $u$ was used for branching (see also Linderoth and Savelsbergh [47]). For every station $u$, we store the sum of the differences between the lower bound at the parent node, and the lower bound at the current node after fixing $u$ to a certain type. This total improvement for $u$ is reset after a fixed number of times $u$ was branched on. Then, this estimate $P_u$ is used to weigh the amount of unavailable flow, as in the first branching rule. We
choose the station $u$ for which

$$\min_{t \in T(u)} P_u \sum_{a \in A^L(u, t)} F(a)$$

is highest. To initialise the estimate $P_u$, we iteratively fix every station $u$ to the types $t \in T(u)$.

**Primal heuristics**

A primal feasible solution $(F, x)$ is completely described by an assignment vector $x$ of types to stations. We propose two algorithms to construct good assignments according to a solution $(F, x)$.

The first primal heuristic is based on the flow solution $F$ of $LRF(\lambda)$. Every station is assigned the type $t$ for which the amount of unavailable flow of $F$ is smallest: $\tilde{t}_u = \arg\min_t \sum_{a \in A^L(u, t)} F(a)$. This assignment is then used with $STOP(\tilde{t})$ to find a primal feasible flow and the total travel time needed.

Second, consider the assignment variables $x^u_t$ that are produced by $LRX(\lambda)$. From the construction of this solution in (6.8) it is clear that the solution $x^u_t$ can also be represented by an assignment vector $\tilde{t}$. Again, a primal feasible flow can easily be constructed from this assignment by solving $STOP(\tilde{t})$.

These fast primal heuristics are used at every node in the branch-and-bound tree to find good primal solutions.

### Computational results

The proposed Lagrangian relaxation based branch-and-bound algorithm of §6.5 is tested using three real life instances of NSR. Table 6.1 shows some basic statistics for these instances, such as the number of stations, tracks and available types, as well as the number of lines in the instance. The same networks were also used in Chapter 5 (see Table 5.2). Figure 6.7 shows the network graphs for these instances. All computations were obtained on an AMD Athlon XP 2700+ with 1 GB internal memory running Linux, kernel 2.4.18, using CPLEX 8.1.

<table>
<thead>
<tr>
<th></th>
<th>NS3600</th>
<th>NSNH</th>
<th>NSRandstad</th>
</tr>
</thead>
<tbody>
<tr>
<td># Stations</td>
<td>28</td>
<td>36</td>
<td>122</td>
</tr>
<tr>
<td># Tracks</td>
<td>27</td>
<td>37</td>
<td>138</td>
</tr>
<tr>
<td># Types</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td># Lines</td>
<td>14</td>
<td>14</td>
<td>64</td>
</tr>
</tbody>
</table>

Table 6.1: Characteristics of the instances.
Figure 6.7: The networks for the instances NS3600 (6.7(a)), NSNH (6.7(b)) and NSRandstad (6.7(c)).
Table 6.2: Initial statistics before preprocessing. The numbers of nodes and arcs refer to the line-event graph. The value $z_{LR}(0)$ is the Lagrangian bound for the zero multiplier vector.

<table>
<thead>
<tr>
<th></th>
<th>NS3600</th>
<th>NSNH</th>
<th>NSRandstad</th>
</tr>
</thead>
<tbody>
<tr>
<td># LR cons</td>
<td>120</td>
<td>176</td>
<td>432</td>
</tr>
<tr>
<td># Lines</td>
<td>14</td>
<td>14</td>
<td>64</td>
</tr>
<tr>
<td># Nodes</td>
<td>168</td>
<td>288</td>
<td>1308</td>
</tr>
<tr>
<td># Arcs</td>
<td>1296</td>
<td>3074</td>
<td>19812</td>
</tr>
<tr>
<td>$z_{LR}(0)$</td>
<td>4326</td>
<td>3221</td>
<td>3404</td>
</tr>
</tbody>
</table>

Table 6.3: Statistics after preprocessing. The first row shows the number of stations for which the type could be fixed a priori. The numbers of nodes and arcs refer to the line-event graph. The value $z_{LR}(0)$ is the Lagrangian bound for the zero multiplier vector.

<table>
<thead>
<tr>
<th></th>
<th>NS3600</th>
<th>NSNH</th>
<th>NSRandstad</th>
</tr>
</thead>
<tbody>
<tr>
<td># LR cons</td>
<td>60 (-50%)</td>
<td>126 (-28%)</td>
<td>321 (-26%)</td>
</tr>
<tr>
<td># Fixed</td>
<td>10 (+36%)</td>
<td>6 (+17%)</td>
<td>57 (+47%)</td>
</tr>
<tr>
<td># Lines</td>
<td>7 (-50%)</td>
<td>9 (-36%)</td>
<td>40 (-37%)</td>
</tr>
<tr>
<td># Nodes</td>
<td>130 (-23%)</td>
<td>230 (-20%)</td>
<td>922 (-30%)</td>
</tr>
<tr>
<td># Arcs</td>
<td>660 (-49%)</td>
<td>1773 (-42%)</td>
<td>9502 (-52%)</td>
</tr>
<tr>
<td>$z_{LR}(0)$</td>
<td>4939 (+14%)</td>
<td>3353 (4%)</td>
<td>3717 (+9%)</td>
</tr>
</tbody>
</table>

The number of dualised constraints in the Lagrangian problem ("# LR cons"), and the values of the Lagrangian relaxation $z_{LR}(\lambda)$ for $\lambda = 0$.

Using the preprocessing techniques described in §6.5.1, the sizes of the line-event graphs can be reduced considerably, as can be seen in Table 6.3. By eliminating redundant lines from the problem description, the number of nodes and arcs is reduced on average by around 27% and 50% respectively. The strength of the reduction techniques for the available types of stations is shown by the number of dualised constraints for the Lagrangian relaxation. For some of the stations only one available type remains. These stations can thus be fixed to this type a priori ("# Fixed"). Note also the improvements in the values for $z_{LR}(\lambda)$.

The line-event graphs for these instances are too large to have a useful visual representation. Still, as an example, Figure D.1 on page 142 presents the line-event graph of NS3600.

We start our analysis of the reduced problems at the root node of the branch-and-bound tree. To test the effectiveness of the two techniques mentioned in §6.5.2, several combinations of them are tried at the first node. Apart from the two algorithms, we also test the zero multiplier vector, as an obvious trivial "algorithm". The statistics
Table 6.4: Statistics for the root node of the branch-and-bound tree.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Zero</th>
<th>MAA</th>
<th>RVA</th>
<th>MAA, RVA</th>
<th>RVA, MAA</th>
<th>LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS3600</td>
<td>4939</td>
<td>5019</td>
<td>5024</td>
<td>5019</td>
<td>5025</td>
<td>5026</td>
</tr>
<tr>
<td>ZLR</td>
<td>5497</td>
<td>5497</td>
<td>5497</td>
<td>5497</td>
<td>5497</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td>11.30%</td>
<td>9.52%</td>
<td>9.41%</td>
<td>9.52%</td>
<td>9.39%</td>
<td></td>
</tr>
<tr>
<td>Gap</td>
<td>0.24</td>
<td>0.47</td>
<td>4.14</td>
<td>1.59</td>
<td>4.32</td>
<td>32.95</td>
</tr>
<tr>
<td>NSNH</td>
<td>3353</td>
<td>3368</td>
<td>3353</td>
<td>3368</td>
<td>3368</td>
<td>3370</td>
</tr>
<tr>
<td>ZLR</td>
<td>4252</td>
<td>4252</td>
<td>4252</td>
<td>4252</td>
<td>4252</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td>26.81%</td>
<td>26.25%</td>
<td>26.81%</td>
<td>26.25%</td>
<td>26.25%</td>
<td></td>
</tr>
<tr>
<td>Gap</td>
<td>0.31</td>
<td>0.87</td>
<td>2.37</td>
<td>3.01</td>
<td>3.09</td>
<td>165.95</td>
</tr>
<tr>
<td>Time</td>
<td>3717</td>
<td>3717</td>
<td>3717</td>
<td>3717</td>
<td>3717</td>
<td></td>
</tr>
<tr>
<td>NSRandstad</td>
<td>4200</td>
<td>4200</td>
<td>4200</td>
<td>4200</td>
<td>4200</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td>2.4</td>
<td>5.95</td>
<td>127.73</td>
<td>131.41</td>
<td>129.81</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.4: Statistics for the root node of the branch-and-bound tree.

for the revised volume algorithm, the multiplier adjustment algorithm and the zero-vector ("Zero") are shown in Table 6.4. The column "MAA, RVA" gives the results for first applying the MAA, and then using the resulting vector as input for the RVA. The column "RVA, MAA" is similar, but for the reverse order. The last column shows the value of the linear programming lower bound, obtained from constructing and solving the LP relaxation of the original STOP model of (6.4). As was shown in Corollary 6.12, these LP bounds are tight upper bounds on the best possible values of the Lagrangian relaxation. The LP relaxation for the last problem, NSRandstad, turned out too large to construct. More information on the number of variables and constraints of STOP can be found in Table D.1 in Appendix D.

For the first instance, the improvement of the lower bound ("zLR"), compared to using just the zero-vector, is considerable. Here RVA takes much more time, but also slightly outperforms MAA. Combining RVA and MAA, in this order, further enhances the quality of the lower bound. This is shown graphically in Figure 6.8. The graph shows the value of the so far best lower bound as a function of time. The different nodes in the graph indicate the iterations of the two algorithms. For the RVA, the nodes represent only the iterations at which an improvement was made. The final bound is very close to the theoretical limit imposed by the linear programming relaxation. This is also true for the second instance, NSNH. However, for this instance RVA cannot improve upon the trivial lower bound, nor on the vector of MAA. Based on these findings, we use the combination of RVA, followed by MAA to find the multiplier vector at the root node. For a graphical representation, see Figure D.2 on page 142 in Appendix D.

The results of applying the primal heuristics of §6.5.3 are shown in Table 6.4 ("zSTOP"). The remaining gaps between the lower bounds and these upper bounds ("Gap") are still considerable. However, as we shall see in Table 6.6, these initial solutions are within a few percent of the optimal solutions.
Next, we investigate the different branching rules presented in §6.5.3. Each of the three instances was tested using one of the branching rules. For a maximum computation time of one hour, the results of these nine tests are reported in Table 6.5.

An important difference in the results is the fact that, using the maximum unavailable flow rule ("Flow"), the instance NSNH could not be solved within the given time bound, while the two other rules solve within a few hundred seconds. Based on these results, we propose to use the estimated degradation rule ("Degradation") as the default branching rule.

As an illustration of the branching process, consider Figure D.3 in Appendix D. This figure shows a graphical representation of the first 66 branch-and-bound nodes in the process of solving NS3600 using the Degradation based branching rule. At every node, it is indicated what station was used for branching, and also to which type it was fixed. Note the recurrent order of stations throughout the tree. On the left side, e.g., the station Tbw is first fixed to type 1, then station 0 is set to 1, and Est to 0. Almost the same order, although with different types, is repeated in different parts of the tree. Similar behaviour was found for the other branching rules, and also for the different instances.

Using the Degradation based branching rule, and strengthening the root node as described earlier, we have tested all three instances against three different scenarios for applying the bound improvement techniques. To investigate the computational tradeoff between better bounds and complex improvement techniques, we first tried using only the zero-vector at every node. This considers the zero-vector, and the multiplier vector of the parent, and chooses the one that gives the best bound. Through
128  CHAPTER 6. THE STATION TYPE OPTIMISATION PROBLEM

<table>
<thead>
<tr>
<th>Instance</th>
<th>Flow</th>
<th>Constraint</th>
<th>Degradation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS3600</td>
<td>5489</td>
<td>5489</td>
<td>5489</td>
</tr>
<tr>
<td>zLR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zSTOP</td>
<td>5489</td>
<td>5489</td>
<td>5489</td>
</tr>
<tr>
<td>Gap</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td># Nodes</td>
<td>389</td>
<td>377</td>
<td>331</td>
</tr>
<tr>
<td>Time</td>
<td>6.8</td>
<td>6.68</td>
<td>6.51</td>
</tr>
<tr>
<td>NSNH</td>
<td>4088</td>
<td>4139</td>
<td>4139</td>
</tr>
<tr>
<td>zLR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zSTOP</td>
<td>4203</td>
<td>4139</td>
<td>4139</td>
</tr>
<tr>
<td>Gap</td>
<td>2.81%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td># Nodes</td>
<td>240489</td>
<td>28333</td>
<td>18500</td>
</tr>
<tr>
<td>Time</td>
<td>*</td>
<td>507.15</td>
<td>320.14</td>
</tr>
<tr>
<td>NSRandstad</td>
<td>3972</td>
<td>3949</td>
<td>3979</td>
</tr>
<tr>
<td>zLR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zSTOP</td>
<td>4135</td>
<td>4135</td>
<td>4138</td>
</tr>
<tr>
<td>Gap</td>
<td>4.10%</td>
<td>4.71%</td>
<td>4.00%</td>
</tr>
<tr>
<td># Nodes</td>
<td>12900</td>
<td>12978</td>
<td>13752</td>
</tr>
<tr>
<td>Time</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

Table 6.5: Branch-and-bound statistics for using the different branching rules on the three instances. An asterisk (*) indicates that the time limit of 3600 seconds was exceeded.

its simplicity, this technique is very fast at every node. Alternatively, we tried the MAA and RVA individually, but we have not tested a combination of these two.

The findings of applying different improvement techniques are shown in Table 6.6. A maximum computation time of two CPU hours is used. The first two instances could be solved to optimality, regardless of the configuration. However, the necessary amount of time and the number of enumeration nodes differs significantly. For these instances, the number of nodes is smallest when using the MAA at every node, even though the difference is only modest. The RVA does not outperform the trivial zero-vector method. For NSNH, the number of nodes needed to solve the instance is much higher. With respect to the required amount of time, the zero-vector method excels due to its simplicity. The time spent at a node is roughly a factor of four lower than for MAA. Even though the quality of the individual lower bounds is less, the overall computation times are lower.

The last column in Table 6.6 shows the findings for solving the complete STOP formulation of (6.4) as a mixed integer programming problem using CPLEX 8.1. As mentioned before, the NSRandstad instance is too large to construct in this way. The remaining two instances can be solved. However, the computation times are far worse than using, e.g., the zero-vector method, or MAA. This shows that the techniques proposed in this chapter perform much better on practical instances than the off-the-shelf solver CPLEX.

In Table 6.7 we compare the total travel time using the currently operated station types ("Current"), with the best solution of Table 6.6 ("New"). The improvements of between 2.1% and 4.4% are indications that a reassignment of station types can
Table 6.6: Computational results of applying the MAA and RVA techniques at every node of tree. The results of the first column are obtained using only the zero-vector at every node. An asterisk (*) indicates that the time limit of 7200 seconds was exceeded. The NSRandstad could not be constructed using the original STOP formulation.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Zero</th>
<th>MAA</th>
<th>RVA</th>
<th>STOP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NS3600</strong></td>
<td>5489</td>
<td>5489</td>
<td>5489</td>
<td>5489</td>
</tr>
<tr>
<td><strong>NSNH</strong></td>
<td>4139</td>
<td>4139</td>
<td>4139</td>
<td>4139</td>
</tr>
<tr>
<td><strong>NSRandstad</strong></td>
<td>3996</td>
<td>3954</td>
<td>3835</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.7: Comparing the current and new total travel times.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Current</th>
<th>New</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NS3600</strong></td>
<td>5607</td>
<td>5489</td>
<td>-118</td>
</tr>
<tr>
<td><strong>NSNH</strong></td>
<td>4331</td>
<td>4139</td>
<td>-192</td>
</tr>
<tr>
<td><strong>NSRandstad</strong></td>
<td>4281</td>
<td>4138</td>
<td>-143</td>
</tr>
</tbody>
</table>

The table shows a comparison of travel times between the current and new solutions. The differences indicate improvements made by applying the MAA and RVA techniques.
Appendix A

Appendix for Chapter 1

Figure A.1: The station layout of Utrecht (taken from NS [60]).
Appendix A

Appendix for Chapter I
Appendix B

Appendix for Chapter 4

Figure B.1: The graphs for the instances SP98AR, SP98IR and SP98IC.
Figure B.2: The number of open and done (processed) nodes at various depths in the enumeration tree of SP98IC (All cuts, variable and line branching rules).
Appendix C

Appendix for Chapter 5

Figure C.1: The instances NS3600, NSNH and NSRandstad placed in a map of the Netherlands.
APPENDIX C. APPENDIX FOR CHAPTER 5

Figure C.2: Example instance of a type graph.

Figure C.3: The layered, directed in-tree $T(e)$ for the edge $e$ of type 1 in the instance in Figure C.2.
Appendix D

Appendix for Chapter 6

Algorithm D.2 Multiplier Adjustment Algorithm

Require: $\lambda$ vector from previous iteration

A distance matrix with $D_{i,j}$ as the distance between variables $i$ and $j$. The vector $\lambda$ contains the current multipliers. Initialize the dual variable vector $\lambda$ with zero. For each variable $i$, set $\lambda_i = H_i$ where $H_i$ is the heuristic value of variable $i$.

1. Set $\lambda = \lambda$ (the best vector so far).
2. Initialize the dual variable set $\{i \in \lambda: \lambda_i > 0\}$.
3. While the dual variable set has elements do
   4. If $\lambda_i < 0$ for any $i$ then break and return.
   5. For $k = 1$ to $n$ do
      6. If $\lambda_k > 0$ then break.
   7. For $k = 1$ to $n$ do
      8. If $\lambda_k \neq 0$ then continue.
      9. Compute the new dual variable $\lambda_k'$.
     10. For $i = 1$ to $n$ do
         11. If $\lambda_i > 0$ then continue.
         12. Compute the new dual variable $\lambda_i'$.
     13. If $\lambda_k' > 0$ then break.
     14. End if
   15. End for
   16. End if
   17. End while

Output: the optimal multiplier vector $\lambda$.
Algorithm D.1 Revised Volume Algorithm

Input: A vector of Lagrangian multipliers \( \lambda_0 \geq 0 \), and a primal bound \( z^* \).
- A tolerance \( m_1 \in (0, 1) \), and a relaxation factor \( \mu \in (0, 2) \).
- A maximum \# of iterations \( T \), and a \# of consecutive null steps \( N \).
- Two threshold values \( \delta_\omega \) and \( \delta_\epsilon \).

1: Solve LR(\( \lambda_0 \)), and let \((F, x)\) be the optimal solution.
2: Initialise \( \nu_0 = (AF - Bx) \), \( z_1 = F \), \( \lambda_1 = p_1 = \lambda_0 \), \( \omega_0 = \omega_1 = \nu_0 \), and \( \epsilon_1 = 0 \).
3: Calculate the new stepsize
   \[ s_{t+1} = \mu \frac{z^* - z_{LR}(\lambda_t)}{\|\omega_t\|}. \] (D.1)

4: Set \( k = t = 1 \).
5: while \( t \leq T \) and \( z_{LR}(\lambda_k) \leq z^* \) do
   6: Make the move \( \lambda_t = \lambda_k + s_t \omega_t \).
   7: Compute the ascent measure \( \delta_t = \delta_t \|\omega_t\|^2 + \|\omega_t, \lambda_k - p_t\| + \epsilon_t. \)
   8: if \( \|\omega_t\|^2 \leq \delta_t^2 \) or \( \|\omega_t, \lambda_k - p_t\| + \epsilon_t \leq \delta_t \) then
      9: stop
   10: end if
   11: Solve LR(\( \lambda_t \)) as described in §6.4, and let \((F, x)\) be the optimal solution.
   12: Set \( \nu_t = (AF - Bx) \).
   13: if \( z_{LR}(\lambda_t) \geq z_{LR}(\lambda_k) + m_1 \delta_t \) then \{serious step\}
      14: Reset the counter \( n = 0 \), and set \( k = k + 1 \).
      15: Set \( \lambda_k = \lambda_t \)
      16: if \( \mu < 1 \) then \( \mu = 1.1 \cdot \mu \) end if
   17: else \{null step\}
      18: \( n \leftarrow n + 1 \)
      19: if \( n \geq N \) then \{too many null steps\} Set \( n \leftarrow 0 \) and \( \mu \leftarrow 0.5 \mu \) end if
   20: end if
   21: Calculate the new stepsize \( s_{t+1} \) as in (D.1) above.
   22: Set \( E_t = \nu_t(\lambda_k - \lambda_t) \), and \( E_t = \omega_t(\lambda_k - p_t) \).
   23: Set \( \alpha_t \in [0, 1] \) to be the value closest, or equal to
   \[ \frac{E_t - E_t}{\nu_t \omega_t} + \frac{\|\omega_t\|^2}{\|\nu_t\|^2 - \|\nu_t \omega_t\| + \|\omega_t\|^2}. \] (D.2)

24: Compute
   \[ z_{t+1} = \alpha_t F + (1 - \alpha_t)z_t \]
   \[ \omega_{t+1} = \alpha_t \nu_t + (1 - \alpha_t)\omega_t \]
   \[ p_{t+1} = \alpha_t \lambda_t + (1 - \alpha_t)p_t \]
   \[ \epsilon_{t+1} = \alpha_t \epsilon_t + (1 - \alpha_t)\epsilon_t \]
   where \( \sigma_t = (1 - \alpha_t)(\nu_t - \omega_t, p_t - \lambda_t) \)
25: Set \( t \leftarrow t + 1 \).
26: end while

Output: the proposed multiplier vector \( \lambda_k \)
Algorithm D.2 Multiplier Adjustment Algorithm

Input: A vector of Lagrangian multipliers \( \lambda \), and a tolerance \( m_1 \).
A distance matrix with \( D_{vi} \in \mathbb{R}_+ \) for \( v \in V \) and \( i \in V \cup V^L \).
A predecessor matrix with \( P_{vi} \in A^L \) for \( v \in V \) and \( i \in V \cup V^L \).
The set \( P_v \) contains the arcs of the shortest path tree rooted at \( v \).
A maximum # of iterations \( T \), and a # of consecutive null steps \( N \).
A factor \( c > 0 \) for widening the derived sensitivity bounds.

1: Set \( \bar{\lambda} = \lambda \) \{the best vector is \( \bar{\lambda} \}\}
2: while \( t \leq T \) and \( s_{LR}(\bar{\lambda}) \leq z^* \) do
3:   for all stations \( v \) do
4:     for all \( a = (i, j) \in A^L : a \notin P_v \) do \{loop all non-tree arcs\}
5:       if \( P_{vi} \notin A^L_d \) then break end if
6:       Set \( d = D_{vj} - (D_{vi} + d(\lambda)_a) \) \{d is non-positive\}
7:       Set \( a' = (i,j') = P_{vj} \)
8:       if \( a' \) is not in the same constraints as \( a \) then \{consider decreasing the cost of \( a \}\}
9:         \( \delta^-_a = \max(\delta^-_a, d) \)
10:    end if
11:   end for
12:   while \( i \neq j \) do
13:     if \( D_{vi} > D_{vj} \) or \( t = v \) then \{consider decreasing the cost of arcs to \( i \}\}
14:       \( a' = (i', j') = P_{vj} \) \{\( a' \) is on the path to the tail of \( a \), but not to head.\}
15:       Update \( \delta^-_{a'} = \max(\delta^-_{a'}, d) \)
16:     end if
17:     \( i \leftarrow i' \)
18:     else \{consider increasing the cost of arcs to \( j \}\}
19:       \( a' = (i', j') = P_{vj} \) \{\( a' \) is on the path to the head of \( a \), but not to tail.\}
20:       if \( a' \) is not in the same constraints as \( a \) then
21:         Update \( \delta^+_{a'} = \min(\delta^+_{a'}, -d) \)
22:       end if
23:     \( j \leftarrow i' \)
24:   end if
25: end while
26: end for
27: end for
28: Stretch the bounds: \( \delta^* \leftarrow c \cdot \delta^- \), and \( \delta^+ \leftarrow c \cdot \delta^+ \).
29: if \( \|\delta^+ - \delta^-\| \leq m_1 \) then stop end if
30: for all stations \( v \) that are not fixed do
31:   Solve the improvement problem (6.18) for \( v \), giving the vector \( \Delta \)
32:   Set \( \lambda \leftarrow \lambda + \Delta \).
33: end for
34: Solve \( LR(\lambda_t) \) as described in §6.4, and let \( (F,x) \) be the optimal solution.
35: if \( z_{LR}(\lambda)/z_{LR}(\bar{\lambda}) \geq 1 + m_1 \) then \{serious step\}
36:   Reset the counter \( n = 0 \).
37:   Set \( \bar{\lambda} = \lambda \).
38: else \{null step\}
39:   \( n \leftarrow n + 1 \)
40: if \( n \geq N \) then \{too many null steps\} stop end if
41: end if
42: end while

Output: the proposed multiplier vector \( \bar{\lambda} \)
Figure D.1: The line-event graph of NS3600. The station nodes are coloured blue, and the line-event nodes are white. The driving arcs are coloured black, the non-halt arcs green, and the change arcs and the halt arcs are both coloured blue. Finally, the passenger arcs that model the possibilities for passengers to enter and leave trains are coloured red.

<table>
<thead>
<tr>
<th></th>
<th>NS3600</th>
<th>NSNH</th>
<th>NSRandstad</th>
</tr>
</thead>
<tbody>
<tr>
<td># Var. initially</td>
<td>31642</td>
<td>99536</td>
<td>2257130</td>
</tr>
<tr>
<td># Var. after</td>
<td>15348</td>
<td>55790</td>
<td>1055054</td>
</tr>
<tr>
<td># Con. initially</td>
<td>148</td>
<td>212</td>
<td>554</td>
</tr>
<tr>
<td># Con. after</td>
<td>88</td>
<td>162</td>
<td>443</td>
</tr>
<tr>
<td>Size reduction</td>
<td>71%</td>
<td>57%</td>
<td>63%</td>
</tr>
</tbody>
</table>

Table D.1: Statistics for the original STOP formulation, before and after preprocessing.

Figure D.2: The performance at the root node of combinations of the RVA and the MAA methods to close the gap between the trivial lower bound $z_{LR}(0)$, and the best lower bound according to the linear programming relaxation (see §6.6).
Figure D.3: The first 66 nodes in an enumeration tree of NS3600. The labels above the nodes show the branching station and type. Note the repeating branched stations (see §6.6).

<table>
<thead>
<tr>
<th></th>
<th>NS3600</th>
<th>NSNH</th>
<th>NSRandstad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct</td>
<td>93.5% (-0.1%)</td>
<td>91.6% (-0.3%)</td>
<td>80.0% (-1.0%)</td>
</tr>
<tr>
<td>1 Change</td>
<td>6.3% (0.0%)</td>
<td>7.8% (0.7%)</td>
<td>18.3% (0.8%)</td>
</tr>
<tr>
<td>2 Changes</td>
<td>0.2% (0.1%)</td>
<td>0.5% (-0.4%)</td>
<td>1.7% (0.2%)</td>
</tr>
<tr>
<td>Avg. speed (kph)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct</td>
<td>72.0 (3.6%)</td>
<td>62.9 (5.6%)</td>
<td>66.0 (4.5%)</td>
</tr>
<tr>
<td>1 Change</td>
<td>47.2 (-7.3%)</td>
<td>43.0 (-0.6%)</td>
<td>50.7 (4.5%)</td>
</tr>
<tr>
<td>2 Changes</td>
<td>0.0 (0.0%)</td>
<td>39.1 (-4.3%)</td>
<td>46.6 (1.7%)</td>
</tr>
<tr>
<td>Solution</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upgraded</td>
<td>0 (0.0%)</td>
<td>2 (5.6%)</td>
<td>0 (0.0%)</td>
</tr>
<tr>
<td>Downgraded</td>
<td>3 (10.7%)</td>
<td>3 (8.3%)</td>
<td>11 (9.0%)</td>
</tr>
<tr>
<td>Tr. time</td>
<td>5489 (-2.1%)</td>
<td>4139 (-4.4%)</td>
<td>4138 (-3.3%)</td>
</tr>
<tr>
<td>Tr. distance</td>
<td>6274 (0.1%)</td>
<td>4057 (-0.1%)</td>
<td>4100 (0.4%)</td>
</tr>
</tbody>
</table>

Table D.2: Solution statistics for travellers in the proposed solution. The differences with using the current station types are shown between brackets. The distribution of the travellers and their average speeds are shown for the direct travellers, and for those travellers that have to change trains once, etc. The last rows of the table show the percentages of all stations that were upgraded or downgraded relative to the currently operated station types, and the new total travel time and total travelled distance.
Figure D.4: Visualisation of the passenger flows around station Dvd. The width of the bars represent the number of passengers. This station was downgraded in the new type assignment from IC to R.

Figure D.5: The number of open and done (processed) nodes at various depths in the enumeration tree of NIS3600 (Zero technique). There are no remaining open nodes, since this instance was solved to optimality.
Figure D.6: The number of open and done (processed) nodes at various depths in the enumeration tree of **NSNH** (Zero technique). There are no remaining open nodes, since this instance was solved to optimality.

Figure D.7: The number of open and done (processed) nodes at various depths in the enumeration tree of **NSRandstad** (Zero technique).
Bibliography


http://www.ertms.com/.


Index

2-Cover cuts, 62, 69
2C cuts, see 2-Cover cuts

Aardal and Van Hoesel [1], 26
ABACUS [2], 71
Abbink et al. [3], 7, 35
Abrahamsson [4], 37
Affine combination, 22
Affine hull, 22
Agglomerational, see Regional
Ahuja et al. [5], 27, 29, 31
Alfieri et al. [6], 7
Bahiense et al. [7], 119
Basic one-hour pattern, 11
Basic one-hour timetable, 11, 35
Basic platform assignment, 11
Bazaraa et al. [8], 29
Bierlaire [9], 37
Big M constraint, 112, 116
BMVBW [10], 5
BOP, 11
BOT, 11
Bottleneck planning, 35
BPA, 11
Branch-and-bound, 26, 119, 121, 125
Branch-and-bound nodes, 26
Branch-and-cut, 27
Branching based on degradation estimates, 122
Branching cycle, 70, 71
Branching on capacities, 66, 71
Branching on lines, 66, 70
Branching on maximum unavailable flow, 122
Branching on special ordered sets, 65, 70
Branching on variables, 65
Branching on violated constraints, 122
Branching rule, 27, 65, 73, 122, 127
Bussieck [11], 16, 45, 51, 53–55, 58, 59, 63, 69, 71, 74, 76, 100
Bussieck et al. [12], 39, 52
Bussieck et al. [13], 51
Bussieck et al. [14], 4
Capacity-subset constraints, 93
Caprara et al. [15], 5
Caprara et al. [16], 9
Central planning, 11
Change arcs, see Line-event, arcs
Claessens [17], 47
Claessens et al. [18], 16, 39, 46, 47, 51–55, 58, 59, 71, 76, 100
CLP, 51, 53–55, 79, 100
Coefficient reduction, 55
Coefficient strengthening cuts, 68
Combining, 9, 34
Commodity, 80, 83, 110
Complementary slackness, 28, 113, 121
Constraints dominance, 59
Convex combination, 22
Convex hull, 22
Cordeau et al. [19], 5
Corridors, 3
CoS cuts, see Coefficient strengthening cuts
Cost-optimising Line planning Problem, 51
cp, 46
Crew rostering, 9
Crew scheduling, 9
Cross-section data, 35
CS constraints, 93
Cut-and-branch, 27, 51
Cutting plane, 25, 60, 68

Dantzig et al. [20], 25
Dietrich and Escudero [21], 55, 56, 63
Direct traveller, 16, 51, 130, 143
Dominated line, 115
Driving arcs, see Line-event, arcs
Dual bound, 22, 26, 27, 29
Dualisation, 112
Dwells, 46

EC [22], 1
EC [23], 1
Eck van der Sluijs, van et al. [24], 13
ECP, 17, 80, 83
Edge capacity problem, see ECP
Edmonds [25], 26
Edmonds [26], 26
ERTMS [27], 42
Even et al. [28], 31

FC cuts, see Flow cover cuts
Fisher [29], 27, 29
Fisher et al. [30], 29
Fixed block system, 42
Flow cover cuts, 63, 69
Freling [31], 9
Freling et al. [32], 9
Freling et al. [33], 10

Gallo and Di Miele [34], 10
Garey and Johnson [35], 24
Generalised line planning problem, see GLP
Geoffrion [36], 27–29, 114
GLP, 55, 61
Gomory cut, 63
Goossens et al. [37], 79
Goossens et al. [38], 51
Goossens et al. [39], 103
Grötschel et al. [40], 26, 31

Halfspace, 22

Halt arcs, see Line-event, arcs
Halting pattern, 46, 104
Halts, 46, 104
Homogenisation, 4, 41
Hyperplane, 22
Härte [41], 46

IC, see Intercity, 53
Increasing capacity, 3
Indirect delays, 2
Integrality property, 30
Intercity, 42, 46
Interregional, 46
IR, see Interregional
Itai [42], 31

Kroon and Fischetti [43], 9
Kroon et al. [44], 10
Kuhn and Tucker [45], 113

Ladányi et al. [46], 27
Lagrangian dual, 28, 29, 114, 120
Lagrangian multiplier, 28, 29, 119
Lagrangian relaxation, 27, 28
Linderoth and Savelsbergh [47], 27, 65, 70, 73, 122

LinDner [48], 4, 34
Line, 44, 52, 104
dwells, 46
halts, 46
route of a -, 44
type, 46
Line plan, 1, 5, 14, 33, 52, 83, 103, 130
Line-event
arcs, 105
change, 105, 107, 142
driving, 105, 107, 142
halt, 105, 107, 142
non-halt, 105, 107, 142
passenger, 105, 107, 142
graph, 104
nodes, 105

Linear combination, 22
Linear programming relaxation, 24, 25, 27, 65, 67, 68

Local planning, 11
INDEX

London Underground [49], 14, 15

Maintenance planning, 10
Martello and Toth [50], 58
MC cuts, see Multi-cover cuts
MCLP, 100, 130
Minkowski [51], 23
Modal split, 37
Moving block system, 42
Multi-cover cuts, 63, 69
Multiplier adjustment, 29, 119

Nachtigall [52], 34
Nachtigall [53], 34
Nachtigall [54], 5
Nemhauser and Wolsey [55], 21, 26, 29, 113
Network edges, 80
Network graph, 44, 52
Node selection strategy, 27
Non-halt arcs, see Line-event, arcs
Nonlinear formulation, 51
$\mathcal{NP}$-hard, 24, 31, 54, 79

NRC [56], 15
NRC [57], 15
NRC [58], 2
NS, 1
NS [59], 3, 35, 41, 42
NS [60], 35, 133
NSR, 2, 11, 15, 34, 76, 123
NYFER [61], 3, 4, 16, 42

OD matrix, see Origin-destination matrix, 80
Odijk [62], 5
Oltrogge [63], 37, 52, 79
On time, 3
Origin-destination matrix, 5, 35, 52

Padberg and Rao [64], 26
Papadimitriou and Steiglitz [65], 21
Passenger arcs, see Line-event, arcs
Passenger demand, 5
Peeters [66], 4, 5, 11, 33, 34, 42
Peeters and Kroon [67], 8
Performance contract, 15

Pijpers [68], 36
Planning problems, 4
Planning process, 11
Platform assignment, 6
Polyhedron, 22
Polytope, 23
Preprocessing, 54, 55, 67, 71, 114, 125
Prestatiecontract, 15
Primal bound, 22
Primal heuristic, 67, 123, 126
Prins [69], 11
Probing, 61
Probing cuts, 61, 69
Process simplification, 3
Project B&B [70], 2, 3, 16, 41, 42
ProRail, 1
ProRail [71], 1, 15
Punctuality, 2, 3

R, see Regional
Railforum [72], 3, 41, 42
Regional, 42, 46
Reinartz and Fassaert [73], 13
Relaxation, 22, 96
Reliability, 3
Rolling stock, 2–5, 7, 12, 34, 47, 54
Rolling stock planning, 7
Rondje om de Kerk, 3

Safwat and Magnanti [74], 37
Schrijver [75], 21
Schrijver [76], 7, 8
Schrijver [77], 21
Schrijver and Steenbeek [78], 5
Secondary delays, 2
Separation problem, 26, 68, 95
Service constraints, 52, 53
Sherali and Park [79], 37
Shunting, 10, 41
Splitting, 9, 34
Station
  nodes, 105
  type, 46
Station layout, 10
Station type, 104, 128
Strong branching, 70
Subgradient, 29
Subgradient optimisation, 29
System split, 39, 52, 100

Thienel [80], 71
Timetable, 5, 14
Track assignment, 6
Traffic assignment, 37, 104, 130
Type, 46
Type assignment (t), 107
Type edges, 81

V&W [82], 2, 39
V&W [83], 2, 39
V&W [84], 3, 5, 16
Valid inequality, 23, 61, 62, 68, 69
Valid path, 108
Variable dominance, 58
Violated inequality, 23
Volgenant and Jonker [81], 29

Weyl [85], 23
Wijzigingsbladen, 12
Winter and Zimmermann [86], 10
Wren and Fares Gualda [87], 9

Zwaneveld [88], 4, 6, 16
Zwaneveld et al. [89], 6
Samenvatting

Door de toenemende vraag naar mobilititeit, maar zeker ook door de privatisering van de spoorwegen is het onderzoek naar het oplossen van planningsproblemen in de spoorwegwereld in een stroomversnelling geraakt. Dit proefschrift onderzoekt zowel wiskundige modellen, als oplossingsmethoden voor zogenaamde lijnvoeringsproblemen voor spoorwegmaatschappijen.

Een lijnvoering ligt ten grondslag aan vrijwel alle vormen van transport die werken met behulp van lijnen en dienstregelingen. Hierbij kan het zowel gaan om, bijvoorbeeld, vervoer per bus, tram, of trein, als om vliegschema's van luchtvaartmaatschappijen. In alle gevallen beschrijft de lijnvoering een lijst van vervoersverbindingen, oftewel lijnen, tussen paren van locaties, die aangeboden worden met een vaste regelmaat, zoals per uur of per week.

De wiskundige modellen en technieken die besproken worden in dit proefschrift richten zich op het ontwerpen van lijnvoeringen zodanig dat de beschikbare middelen van de vervoerder efficiënt worden gebruikt. Met beschikbare middelen kunnen zowel kosten bedoeld worden, als benodigde hoeveelheden materieel en personeel.

Hoofdstuk 1 van het proefschrift bespreekt oude en nieuwe ontwikkelingen in de wereld van het openbaar vervoer per trein in Nederland. In het bijzonder de verwachte grote toename in het aantal treinreizigers stelt zware eisen aan de organisatie van het vervoer per spoor. Zo zal een significante verhoging van de betrouwbaarheid en de capaciteit van het spoorwegennet gerealiseerd moeten worden. Daarnaast geeft dit hoofdstuk een overzicht van het scala van planningsproblemen, waarvan het lijnvoeringsprobleem een onderdeel vormt. Hier worden problemen besproken als het maken van een dienstregeling, het plannen van personeel, maar ook het ontwerpen van onderhoudsschema's van treinstellen.

Het resterende gedeelte van het proefschrift is opgebouwd uit twee delen. In het eerste deel, met daarin hoofdstuk 2 en hoofdstuk 3, wordt een theoretische en praktische basis gelegd. Allereerst geeft hoofdstuk 2 een overzicht van de wiskundige concepten die veelvuldig gebruikt zullen worden in de latere hoofdstukken. Hoofdstuk 3 beschrijft de aspecten die een rol spelen bij het modelleren van lijnvoeringsproblemen. Hierbij worden onderwerpen besproken als de samenhang tussen lijnvoeringen en dienstregelingen, het schatten van de vervoersvraag door het netwerk, en beperkingen door de infrastructuur bij het ontwerpen van lijnvoeringen. Tevens wordt een aanzet gegeven voor het modelleren van het lijnvoeringsprobleem.

Het tweede deel van het proefschrift bespreekt drie toepassingen van wiskun-
digie modellen voor het oplossen van lijnvoeringsproblemen. In hoofdstuk 4 wordt
begonnen met het beschrijven en analyseren van een algoritme voor het samen-
stellen van lijnvoerings. Hierbij geldt de beperking dat slechts een lijnvoering wordt
gemaakt voor treinen van één treintype, zoals intercity treinen, sneltreinen of stop-
treinen. De bestaande theorie wordt hierbij sterk uitgebreid met onder andere me-
thoden voor het vereenvoudigen en compacter maken van de probleembeschrijving,
een aantal technieken voor het oplossen van de wiskundige modellen. Het hoofd-
stuk wordt afgesloten met een toetsing van deze methoden en technieken aan de hand
van een aantal praktijkvoorbeelden. Hieruit blijkt ten eerste dat de ontwikkelde me-
thoden deze problemen veel compacter maken. Daarnaast zorgt de gecombineerde
aanpak voor goede oplossingen binnen een redelijke tijd en kan voor nagenoeg alle
probleemvoorbeelden de best mogelijke oplossing worden gevonden.

Tegenover de lijnvoeringsproblemen van hoofdstuk 4, die zich beperken tot treinen
van één treintype, staan de problemen die besproken worden in hoofdstuk 5. Dit
hoofdstuk introduceert het lijnvoeringsprobleem voor het geïntegreerd oplossen van
lijnvoeringsproblemen voor meerdere treintypen. Hiervoor worden een drietal wi-
kundige modellen ontwikkeld. Hoewel aangetoond wordt dat deze modellen het
onderliggende probleem equivalent beschrijven, blijkt dat één modellering duidelijk
beter in staat is de bekeken praktijkinstanties op te lossen. Verder komt naar voren
dat bij de geteste voorbeelden de geïntegreerde aanpak uit dit hoofdstuk beduidend
betere resultaten levert dan de gesplitste aanpak uit hoofdstuk 4.

De lijnvoeringsproblemen zoals zij aan bod komen in hoofdstuk 4 en hoofdstuk 5
zijn erop gericht een volledig nieuwe lijnvoering te ontwerpen. Voor iedere lijn in een
dergelijke lijnvoering worden de stations waar deze lijnen stoppen gedicteerd door de
types van de lijnen en de stations in het netwerk. Zo zullen intercity treinen alleen
stoppen op intercity stations, terwijl stoptreinen ieder station aandoen dat op hun
route ligt. De stationstypes worden hierbij als gegeven beschouwd, en liggen dus
vooraf al vast. In tegenstelling hierop introduceert hoofdstuk 6 een wiskundig model
dat de lijnvoering als gegeven beschouwd, maar nu probeert andere stationstypes te
vinden zodanig dat de totale reistijd voor passagiers wordt geminimaliseerd. Neem
bijvoorbeeld een stoptrein station in het huidige netwerk. Een intercity trein zal
niet bij dit station stoppen. Dit is voordelig voor de passagiers die zich in deze
trein bevinden, aangezien er een extra, onnodige, stop zou betekenen dat zij een aantal
minuten extra reistijd zouden hebben. Echter, voor passagiers die vertrekken vanaf
dit station zou een hoger stationstype voordelig zijn. Dit zou er namelijk voor zorgen
dat zij meteen al kunnen instappen in een snellere intercity trein.

Voor het oplossen van het optimalisatie probleem voor het kiezen van stations-
types wordt in hoofdstuk 6 een aantal nieuwe modelleringsconcepten besproken.
Aan de hand hiervan wordt een modelformulering beschreven die gebaseerd is op
een klassiek netwerkstroom model met extra beperkingen en variabelen. Voor deze
formulering wordt een decompositie besproken waarbij het probleem kan worden
opgelost aan de hand van een groot aantal kleinere deelproblemen. Net als in de
voorgaande hoofdstukken wordt dit hoofdstuk afgesloten met een toetsing van de
beschreven ideeën aan de hand van een aantal praktijkinstanties van de Nederlandse
Spoorwegen. Hieruit blijkt dat het oplossen van deze problemen met behulp van
SAMENVATTING

de voorgestelde methoden een duidelijke verbetering brengt. Daarnaast zorgen de gevonden oplossingen ieder voor een substantiële tijdswinst vergeleken met de oorspronkelijke stationstypes.

Curriculum Vitæ
Hierbij geldt de beperking dat alleen standaardlijnvaart gemaakt voor treinen van één trein typen, zoals Intercity treinen, snelstrepen of stopstrepen. De bestaande theorie wordt hierbij sterk uitgebreid met onder andere methoden voor het verwezenlijgen en compactere maken van de probleem-beschrijving en een aantal technieken voor het oplossen van de wiskundige modellen. Het hoofdstuk wordt afgesloten met een discussie van deze methoden en technieken aan de hand van een aantal praktijkvoorbeelden. Hieruit blijkt hoe eerste dat de ontwikkelde methoden deze problemen veel compacter maken. Daarnaast zorgt de geïntegreerde aanpak voor goede overgangen binnen een redelijke tijd en kan voor negen procent alle problemen besproken de best mogelijke oplossing worden gevonden.

Tegenover de lijnvoeringsproblemen van hoofdstuk 4, die zowel betrekking tot treinen van één trein typen, staan de problemen die besproken worden in hoofdstuk 5. Dit hoofdstuk introduceert het lijnvoeringsproblemen voor het samengevoegd oplossen van lijnvoeringsproblemen voor meerdere trein typen. Hierbij worden drie principiële modellen onderscheiden. Hoewel aangetoond wordt dat deze modellen het onderliggende probleem equivalent beschrijven, blijkt dat één modellering uitsluitend betrekking heeft in staat de bekend geclassificeerde oplossingen te benutten. Verder komt naar voor dat bij de meeste voorbeelden de geïntegreerde aanpak uit dit hoofdstuk beïnvloedde betere resultaten levert dan de parallelle aanpak uit hoofdstuk 4.

De lijnvoeringsproblemen zoals zij aan het eind komen in hoofdstuk 4 en hoofdstuk 5 zijn op het gehele verschil tussen de vele trein typen lijnvoering te onderscheiden. Verschillende in een overzichtelijke lijnvoering worden de stations waar deze lijnen stoppen gekozen door de typen van de lijnen en de stations in het netwerk. Zo telling Intercity treinen alleen stoppen op Intercity stations, terwijl stopstrepen ledere station aantreffen op dat op het netwerk liggen. De stations worden hierbij als pagetrap beschouwd, en liggen het vooraf af vast. In begeleiding hierboven introduceert hoofdstuk 6 een wiskundig model dat de lijnvoering als gegeven beschouwt, maar niet probeert andere stations typen te vinden netzels dat de totale reis tijd van passagiers wordt geminimaliseerd. Nog een belangrijke aspect is het stoppen van stations in het netwerk. Een Intercity trein zal niet bij dit station stoppen. Dit is voornamelijk voor de passagiers die zich in deze trein bevinden, aangezien een extra, onnodige, stop zou betekenen dat zij een aantal minuten extra wezentijd zouden hebben. Echter, voor passagiers die vertragen van dit station aan een hoger stations typen voorbij zijn. Dit zou te onacht voor stoppen dat zij neutraal al kunnen instappen in een snellere Intercity trein.

Voor het oplossen van het optimaliserende probleem voor het kiezen van stations typen wordt in hoofdstuk 6 een aantal nieuwe modelformuleringen besproken. Aan de hand hiervan wordt een modelformulering beschreven die genoemd is op een klassiek graphoptimalisatie model met extra beperkingen en variabelen. Door deze formulering wordt een decentrale beslissing waarbij het probleem kan worden opgelost en de hand van een groot aantal kleinere dagopdrachten. Het is in de voorgaande hoofdstukken wordt de hoofdstuk afgesloten met een toestand van de formulering keer om aan de hand van een aantal praktijkvoorbeelden van de Nederlandsche Spoorwegen. Daaruit blijkt dat het oplossen van deze problemen met behulp van
Curriculum Vitæ

Jan-Willem Henricus Mathijs Goossens was born on December 17, 1975 in Stein, the Netherlands. In 1994, he obtained his atheneum degree (secondary school) at the Scholengemeenschap Groenewald in Stein. Later that year he started to study econometrics at the University of Maastricht (then the Rijksuniversiteit Limburg) with Operations Research as the main subject. After a student research project at the logistics department of NS Reizigers, he graduated in 1998 with an MSc thesis on cost minimisation for line planning problems, under supervision of Prof. dr. ir. C.P.M. van Hoesel and Prof. dr. L.G. Kroon.

At the end of 1998 he took up a position as a PhD candidate at the University of Maastricht. This project for four years concerned the development of models and algorithms for the optimisation of railway line plans. This research was partly funded by NS Reizigers.

Sponsored by the European Union’s network for Algorithmic Methods for Optimizing the Railways in Europe (AMORE), he visited the Algorithms and Data Structures group at the Universität Konstanz in Germany from April 2002 until March 2003. He then returned to the University of Maastricht.

From April 2004 until August 2004, he worked as an application development consultant at Paragon Decision Technology in Haarlem.

As of October 2004, he works at the NATO Consultation, Command and Control Agency (NC3A) in The Hague.