Doctoral Thesis

Algorithms for railway traffic management in complex central station areas

Author(s):
Fuchsberger, Martin

Publication Date:
2012

Permanent Link:
https://doi.org/10.3929/ethz-a-007583671

Rights / License:
In Copyright - Non-Commercial Use Permitted

This page was generated automatically upon download from the ETH Zurich Research Collection. For more information please consult the Terms of use.
Algorithms for railway traffic management in complex central station areas

A dissertation submitted to the
ETH ZURICH

for the degree of
DOCTOR OF SCIENCES

presented by
MARTIN FUCHSBERGER

accepted on the recommendation of
Prof. Dr. Hans-Jakob Lüthi, examiner
Prof. Dr. Ulrich Weidmann, co-examiner
Prof. Dr. Peter Widmayer, co-examiner

2012
Acknowledgments

Working at the Institute for Operations Research (IFOR) at ETH Zurich was a privilege where I learned about the mathematics of operations research. More than that I was able to further my knowledge and experience in the more practical application of mathematics specifically in the domain of railway operations in collaboration with Swiss Federal Railways (SBB).

I especially want to thank my advisor, Prof. Hans-Jakob Lüthi for introducing me to operations research and allowing me to write both my master and doctoral thesis on railway operation topics. His perpetual energy and enthusiasm in closing the gap between operations research and industry had motivated all his advisees, especially me. I greatly appreciate his continuous, kind and encouraging support during these past few years. His valuable guidance and ideas, the numerous fruitful discussions and his communication skills were essential for the progress of this thesis working so closely with the SBB.

I am also very grateful to Prof. Ulrich Weidmann. He not only agreed to be referee of my thesis, but his expertise in the area of railway operations was critical to my collaboration with SBB. I appreciate all the stimulating and constructive discussions, as well as suggestions and comments that helped improve this thesis.

Many thanks to Prof. Peter Widmayer for his willingness to referee this thesis, his friendly support and helpful suggestions.

I am very much indebted to PD Dr. Marco Laumanns, who led the collaboration with SBB and supervised my thesis. His constructive criticism in numerous discussions and meetings, helped me refine my approaches and models. His expertise was essential for conducting this project with SBB and I learned many soft skills from him, which are essential for a PhD. I learned a lot about leadership from his way of guiding and teaching students during their master or bachelor thesis and the scientific writing of papers.

I would further like to thank SBB for funding my research and to the many people working at SBB, without whom this collaboration would not have been possible. Most notably, I’d like to thank Dr. Peter Grossenbacher and Médard Fischer which were particularly involved in this collaboration. I owe my gratitude to Martin Kaufmann, Peter Kreuzer, Dr. Matthias Krista and Peter Schaub for the many fruitful technical discussions and helpful hands during problems concerning the implementation or data processing. I’d
also like to thank Dr. Felix Laube, Dr. Raimond Wüst, Heinz Egli, Samuel Roos, and Oskar Stalder which initiated the collaboration between SBB and my former IFOR colleagues. They also shaped the beginning of the collaboration during the start of my own thesis.

This dissertation would not have been the same without the input of my former colleagues who were also collaborating in railway projects. I am deeply indebted to my recruiters Dr. Dan Burkholter, Dr. Thomas Herrmann and Dr. Gabrio Caimi. They introduced me to the topic and I had the great pleasure of collaborating with, and learning from Gabrio Caimi in the initial phase of my thesis.

I am indebted to all my former and current colleagues at the Institute for Operations Research which made my time as a PhD student so enjoyable. A special thank to my office colleagues Kathrin Ballerstein and Sabrina Wiedersheim, my friends Michael Bürgisser and Kaspar Schüpbach and my former colleagues Dr. Vânia Eleutério, Dr. Michael Guarisco and Dr. Rico Zenklusen. They were always there for discussions, new ideas, support, or simply a good laugh.

I am also grateful to Dr. Frédéric Weymann, Dr. Marco Lüthi, Dr. Holger Flier and Dr. Marc Nunkesser for their collaboration in research projects on train dispatching.

This time at ETH was also very enriching due to the various students that I had the opportunity to work with, in particular I like to thank Gioele Balmelli, Arsin Grünig and Jörg Priewasser. Their initial approaches on speed profile computation and dynamic dispatching of trains contributed to the implementation and the good results of this thesis.

I am also grateful to Hilda Fritze-Vomvoris for proofreading the thesis.

Finally, my warmest thanks go to my parents, my siblings, my girlfriend Ylena, my closest friends David Adijashvili, Valérie Haas and Caroline Hofer for their support, encouragement and patience.
Abstract

This thesis addresses the problem of managing dense railway traffic in a complex central station area by providing ongoing decision support to the dispatcher. For this purpose a closed-loop discrete-time control framework is developed, in which a model predictive controller provides decision support in the form of disposition schedules. The model predictive controller assumes that the railway network topology, the timetable, the connections and a forecast of future train movements are given. The controller’s time-critical task is to provide the dispatcher with a conflict-free disposition schedule, which assigns a travel path and a start time to each train movement inside the considered time horizon and additionally, minimizes the delays and broken connections that could occur at the central station.

Many models and algorithms for train dispatching have already been proposed in the literature, but only a few of them with successful application in practice. The few successful applications are either limited to a restricted area of switches, consider a simpler network topology or are based on heuristics that lack a quality assertion. Approaches that are not affected by any of these restrictions, have so far not be applied in daily operations mostly due to their computational complexity. The approach of this thesis for dispatching trains in a central railway station area is sufficiently fast in order to be used in a decision support system and in addition, provides a guarantee of quality.

The train dispatching problem is addressed in this thesis in three consecutive steps:

1. Alternative train paths, alternative departure times and speed profiles are combined and form a set of alternative, so-called blocking time stairways, which model essentially safety corridors for the train movements in the railway network.

2. A mathematical constrained assignment model is formulated as a binary linear program in which constraints preclude safety-related and operational conflicts between alternative blocking time stairways and thus, the choice for assigning a blocking time stairway to each train movement is limited.

3. A bi-objective function is added to the model to measure the quality of an assignment in terms of generated delays and broken connections at the central station.
Finally, a commercial solver based on mathematical optimization techniques computes the best assignment.

The efficiency of this approach in terms of computation time depends on the formulation of the binary linear program. The formulation proposed in this thesis has considerably fewer variables as well as fewer and stronger constraints compared to previous formulations. The associated LP relaxation provides better bounds on objective values due to the stronger constraints, which in turn can be exploited by the solver and result in much shorter computation times.

A major aspect of this thesis was to showcase the viability of this framework in a comprehensive case study, so that industry will take this framework a step further towards its integration in practical operations. In close collaboration with the Swiss federal railways (SBB) the closed-loop discrete-time control system was simulated in a laboratory setting. In the simulation trains were dispatched over the course of a day in the central railway station area Berne, Switzerland. The closed-loop discrete-time control framework was successfully able to dispatch trains in a time interval of one minute. Nevertheless, the laboratory setting precluded a study of the qualitative impact that the closed-loop system would have on a physical railway system. The thesis outlines three additional important steps, which are required before a successful integration of this framework in practice becomes possible:

1. The framework is interfaced with a simulation environment that will be responsible for simulating the train movements in the physical railway system.

2. The dispatcher has to be incorporated in the close-loop system. A good platform for the interaction of the control system with the dispatcher is critical for a successful integration of the presented decision support system.

3. When the platform for the interaction between dispatcher and the control system can be successfully operated over a simulated physical railway system, the simulation can be replaced by the real world railway system.

It is obvious that the SBB strives for improved operational processes and thus supported this thesis. But it is quite remarkable that the SBB also supports the stepwise approach suggested by the thesis towards integrating decision support systems in their operational processes. As a consequence, the SBB started a new project in which the framework will be studied further by interfacing the closed-loop control system with a railway simulation software.


Der Ansatz für die Disposition von Zügen ist in drei aufeinanderfolgende Schritte unterteilt:

1. Alternative Laufwege, alternative Abfahrtszeiten und Geschwindigkeitskennlinien von Zugbewegungen werden zu so genannten Sperrzeitentreppen kombiniert. Diese Sperrzeitentreppen sind Raum-Zeit-Korridore und werden vom Sicherungssystem konzeptionell verwendet, um die kollisionsfreie sichere Bewegung der Züge zu ga-
rantieren.


Die an der Rechenzeit gemessene Effizienz dieses Ansatzes hängt stark von der Formulierung des binären linearen Programms ab. Die von dieser Dissertation vorgeschlagene Formulierung reduziert deutlich die Anzahl der zu betrachtenden Variablen und enthält stark verschärfte Bedingungen im Vergleich zu früheren Formulierungen. In der assoziierten LP Relaxierung führen die verschärften Bedingungen dann zu besseren Zielwertschranken und diese wiederum erlauben es dem mathematischen Löser, deutlich schneller die beste Zuweisung zu ermitteln.


1. Das Rahmenwerk wird ergänzt durch eine Simulationsumgebung, welche das Systemverhalten des Eisenbahnnetzes realistisch abbildet.

2. Der Disponent wird in die Umsetzung einbezogen und interagiert im geschlossenen Regelkreis mit dem Entscheidungsunterstützungssystem. Der Fokus dieses Schrittes ist die Umsetzung einer guten Schnittstelle für die Interaktion zwischen dem Disponenten und dem System.
3. Nachdem die Interaktion zwischen dem Disponenten und dem System wie gewünscht verläuft, wird die Simulationsumgebung durch einen Anschluss an das physikalische System des Eisenbahnnetzes ersetzt.

Es ist offensichtlich, dass die SBB Verbesserungen in betrieblichen Prozessen anstreben und unter anderem deshalb, diese Dissertation unterstützt haben. Die SBB werden die in dieser Dissertation vorgeschlagene schrittweise Umsetzung eines Entscheidungsunterstützungssystem verfolgen. Als Konsequenz wurde erfreulicherweise bereits eine Folgestudie gestartet, welche sich mit dem ersten Umsetzungspunkt, nämlich der Integration einer Simulationsumgebung, beschäftigt.
Cette thèse de doctorat a pour but d’assister le planificateur dans sa tâche de gérer l’intense trafic ferroviaire dans l’environnement complexe de gares centrales. Dans cette optique, on y développe un système qui fournit au planificateur des horaires opérationnels constamment actualisés grâce à une commande prédictive opérant à l’intérieur d’un circuit logique fermé à temps discret. Afin d’élaborer des horaires opérationnels impeccables, le régulateur a besoin des données de l’infrastructure du réseau ferroviaire, les informations concernant les correspondances et une prévision des futurs parcours des trains. Ces horaires opérationnels attribuent une heure de départ et une route adéquate à tout mouvement de train à l’intérieur de l’intervalle de temps considéré, évitant tout conflit entre les différents mouvements de trains. En outre, la qualité de l’horaire opérationnel exige une minimisation de retards et d’interruptions de correspondances par rapport à l’horaire régulier.

La littérature fournit bon nombre de modèles et d’algorithmes pour le dispatching de trains. Peu d’entre eux ont été implémentés avec succès dans la pratique. Ces rares applications se limitent à un nombre d’aguillages restreint, à une topologie de réseau simplifiée, ou se basent sur des méthodes heuristiques qui ne fournissent aucune garantie de qualité. Jusqu’à présent, les approches faisant fi de ces restrictions n’ont pas pu s’appliquer à la pratique étant donnée leur complexité de calcul. L’approche de cette thèse pour la régulation de trains dans des gares centrales fournit dans un délai adéquat des horaires opérationnels capables d’améliorer le système de support à la prise de décision et de plus comprend une garantie de qualité. Le problème du dispatching de trains est abordé dans cette thèse en trois pas consécutifs :

1. Des parcours de trains, des heures de départ et des profils de vitesse alternatifs sont combinés pour former des escaliers de temps de blocage. Ces derniers modélisent essentiellement des corridors temps-espace que le système de sécurité utilise de façon conceptionnelle pour garantir le déplacement sr et sans collision des trains.

2. Un problème mathématique d’attribution est formulé en tant que logiciel binaire linéaire, dont les prémisses incluent des restrictions pour éviter des collisions entre trains. Il s’agit d’attribuer un escalier de temps de blocage adéquat à chaque mou-
vement de train, tout en considérant les restrictions opérationnelles et celles dues aux exigences de sécurité.

3. Le modèle est complété par une fonction à double objectif, qui mesure la qualité de l’attribution par l’analyse des retards engendrés et des correspondances interrompues. Finalement, un solveur commercial basé sur les méthodes d’optimisation calcule la meilleure attribution possible.

L’efficacité de cette approche, mesurée en temps de calcul, dépend largement de la rédaction du logiciel binaire linéaire. La formulation proposée dans cette thèse réduit notablement le nombre de variables à considérer et comprend de fortes restrictions par rapport aux formulations préalables. Grâce à l’usage de la relaxation continue, ces restrictions renforcées autorisent de meilleures bornes d’objectif, qui à leur tour permettent un gain de rapidité du solveur mathématique dans la détermination de la meilleure attribution.

Mis à part l’efficacité algorithmique, la capacité pratique de l’approche poursuivie est d’une importance évidente. Une étude de cas présente dans cette thèse l’application des méthodes algorithmiques et mathématiques développées dans le but de convaincre l’industrie de la viabilité d’un tel système et de pousser à l’intégration d’approches algorithmiques et mathématiques dans l’exploitation des chemins de fer. En étroite collaboration avec les chemins de fer fédéraux (CFF), le circuit logique fermé à temps discret a été simulé sous conditions de laboratoire. Cette simulation a organisé les mouvements des trains durant une journée fictive dans l’environnement central de la gare de Berne (Suisse). Minute après minute, le système s’est montré capable de pourvoir un horaire opérationnel, sans excéder le temps de calcul d’une minute. Cela prouve la faisabilité des calculs. L’avantage qualitatif du système de support à la prise de décision proposé ne peut pas s’analyser par cette étude de cas qui ne tient pas compte du comportement du système. Néanmoins, la thèse esquisse trois étapes importantes à franchir avant l’introduction d’un tel système dans la pratique.

1. La trame de ce système est complétée par un environnement fictif qui simule le mouvement des trains du réseau ferroviaire.

2. Le planificateur participe à l’application et interagit avec la commande prédictive. L’établissement d’une bonne plateforme d’interaction entre le planificateur et le système est essentielle pour le succès d’un tel modèle.

3. Une fois que cette plateforme opère sans problèmes, l’environnement fictif peut être remplacé par la connexion au système physique du réseau ferroviaire.

Il faut remarquer que les CFF aspirent à une amélioration des processus à l’intérieur de leur entreprise, d’où leur soutien pour cette thèse. Dorénavant, ils favoriseront l’introduction graduelle comme prévue ci-dessus d’un tel système de support à la prise de décision. Comme premier pas dans cette direction, les CFF ont mis sur pied un nouveau projet dans le but de tester ce système au moyen d’autres logiciels de simulation utilisant les données du secteur ferroviaire.
## Contents

**Acknowledgments**  
**Abstract**  
**Zusammenfassung**  
**Résumé**

### 1 Introduction

1.1 Motivation  
1.2 Railway traffic management today  
  1.2.1 Dispatchers work station  
  1.2.2 Action space for dispatching  
  1.2.3 Hierarchical control by geographical decomposition  
1.3 Industrial state of the art and development trends  
  1.3.1 Centralising control  
  1.3.2 Automatic conflict detection  
  1.3.3 Adaptive train control and speed profile optimization  
  1.3.4 Towards sophisticated train control systems  
1.4 Focus and contribution of this thesis  
1.5 Outline of the thesis

### 2 Modelling the railway system: topology and train movements

2.1 Railway system characteristics  
  2.1.1 Classification of (German) railway infrastructure  
2.2 Safe train operation and control  
  2.2.1 Separation of train movements  
2.3 Modelling the railway topology  
  2.3.1 Double vertex graph  
  2.3.2 Spurplan graph
## Contents

2.3.3 Fahrnetz .................................................. 24  
2.3.4 Railway topology graph ................................. 25  
2.3.5 Routing graph ............................................ 27  
2.3.6 Demarcation portals ..................................... 27  
2.3.7 Topology models used in this thesis .................... 27  
2.4 Travel paths .................................................. 28  
2.4.1 Definition and notation .................................. 28  
2.4.2 Computing $K$ shortest, loop-less paths ............... 28  
2.5 Train speed profiles ......................................... 29  
2.6 Blocking time stairways ..................................... 31  
2.7 Forecast of train movements ............................... 33  

3 Static train dispatching ........................................ 35  
3.1 Related work .................................................. 35  
3.1.1 Re-ordering train movements at fixed points .......... 36  
3.1.2 Combining re-ordering with re-routing ................. 37  
3.1.3 Other frameworks for train dispatching ................ 38  
3.2 Problem statement ............................................ 38  
3.3 Optimization model .......................................... 40  
3.3.1 Notation .................................................... 40  
3.3.2 Decision and indicator variables ......................... 42  
3.3.3 Unique assignment of blocking time stairways to trains .................................................. 42  
3.3.4 Train separation: Unique (conflict-free) blocking of resources ........................................... 43  
3.3.5 Connection between arriving train $t$ and departing train $t'$ ........................................ 45  
3.3.6 Proper platform usage at a central railway station .................................................. 49  
3.3.7 Bi-level multi-objective formulation ...................... 50  
3.3.8 Summarized static train dispatching (STD) model .................................................. 52  
3.4 Computational experiments .................................... 52  
3.4.1 Key figures of an STD instance .......................... 52  
3.4.2 Computation time .......................................... 53  

4 Towards dynamic train dispatching ......................... 57  
4.1 Closed-loop discrete-time control ......................... 57  
4.2 Model predictive control .................................... 58  
4.3 The dynamic system of train dispatching .................. 59  
4.3.1 Modelling disturbances ................................... 63  
4.3.2 A remedy to infeasibility: Additional alternative blocking time stairways .......................... 64  
4.3.3 Heuristic for generating postponed blocking time stairways ........................................... 65  

5 Test case Berne ................................................. 67  
5.1 System boundaries ............................................ 67  
5.2 Considered trains .............................................. 68  
5.3 Temporal scopes .............................................. 68  
5.4 Computational environment ................................... 69  
5.5 Sampled disturbances ........................................ 70
List of Figures

1.1 Increase in railway traffic in Switzerland ........................................... 4
1.2 Closed dispatching control loop ....................................................... 5
1.3 Workstation in the traffic management centre Zurich, Switzerland .......... 6
1.4 Time-distance graph ................................................................. 7
1.5 Hierarchical system structure of railway traffic management .................. 9
1.6 Merging of traffic control in control centres ......................................... 9
1.7 Geographical disposition areas ..................................................... 10
1.8 A conflict situation and two paradigms resolving it .............................. 11
1.9 RCS tasks and control loop ....................................................... 13
1.10 Closed-loop discrete-time control framework .................................... 14
1.11 Dispatching framework realization: Step 1 ...................................... 15

2.1 German railway network terminology ............................................. 18
2.2 Fixed block signalling ............................................................. 21
2.3 Spurplan graph ........................................................................ 23
2.4 Fahrnetz graph ........................................................................ 25
2.5 Turnout types ........................................................................... 26
2.6 Locating interlocking reference points (MitteAGs) .............................. 26
2.7 Routing graph ............................................................................ 27
2.8 Speed profiles computed with dynamic programming ......................... 30
2.9 Blocking time ........................................................................... 32
2.10 Blocking time stairway ............................................................ 33

3.1 Blocking schema and interval graph .............................................. 44
3.2 Two blocking time stairways lead to a broken connection ..................... 46
3.3 Illustration of the Algorithm 3.1 for computing $T_{p,q}^{out}$ ..................... 47
3.4 STD model: Computation times .................................................. 54
3.5 STD model: Computation times - Average and standard deviation ......... 55

4.1 Closed-loop control .................................................................... 57
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>MPC control scheme</td>
<td>59</td>
</tr>
<tr>
<td>4.3</td>
<td>Temporal scopes in closed-loop discrete-time train dispatching</td>
<td>60</td>
</tr>
<tr>
<td>4.4</td>
<td>MPC control scheme for train dispatching</td>
<td>62</td>
</tr>
<tr>
<td>4.5</td>
<td>MPC state transfer function</td>
<td>63</td>
</tr>
<tr>
<td>5.1</td>
<td>Aggregated topology of the central railway station area Berne, Switzerland</td>
<td>68</td>
</tr>
<tr>
<td>5.2</td>
<td>Delay prediction</td>
<td>70</td>
</tr>
<tr>
<td>5.3</td>
<td>Computation times for the dynamic train dispatching instances</td>
<td>71</td>
</tr>
<tr>
<td>5.4</td>
<td>N. of infeasibilities with 13 travel paths per train</td>
<td>72</td>
</tr>
<tr>
<td>5.5</td>
<td>N. of considered train movements in dynamic train dispatching instances</td>
<td>73</td>
</tr>
<tr>
<td>5.6</td>
<td>N. of infeasibilities with 130 travel paths per train</td>
<td>73</td>
</tr>
<tr>
<td>5.7</td>
<td>Size of dynamic train dispatching instances (N. of variables)</td>
<td>74</td>
</tr>
<tr>
<td>5.8</td>
<td>Size of dynamic train dispatching instances (N. of constraints)</td>
<td>74</td>
</tr>
<tr>
<td>5.9</td>
<td>N. of paths and solution quality</td>
<td>75</td>
</tr>
<tr>
<td>5.10</td>
<td>N. of paths and computation time</td>
<td>76</td>
</tr>
<tr>
<td>5.11</td>
<td>Trade-off between delays and connections</td>
<td>77</td>
</tr>
<tr>
<td>6.1</td>
<td>Dispatching framework realization: Step 2</td>
<td>84</td>
</tr>
<tr>
<td>6.2</td>
<td>Dispatching framework realization: Step 3</td>
<td>85</td>
</tr>
<tr>
<td>A.1</td>
<td>Yen’s Algorithm Phase 1 - Equivalent sub-paths</td>
<td>100</td>
</tr>
<tr>
<td>A.2</td>
<td>Yen’s Algorithm Phase 1 - Different sub-paths</td>
<td>100</td>
</tr>
<tr>
<td>A.3</td>
<td>Yen’s Algorithm Phase 2 - Create candidate path</td>
<td>100</td>
</tr>
<tr>
<td>B.1</td>
<td>Traction forces for different traction control (RE 460 locomotive)</td>
<td>109</td>
</tr>
<tr>
<td>B.2</td>
<td>Adhesion can limit the traction of a (RE 460) locomotive</td>
<td>110</td>
</tr>
<tr>
<td>B.3</td>
<td>Electrical braking force curve of ICE locomotive</td>
<td>111</td>
</tr>
<tr>
<td>B.4</td>
<td>Propulsion resistances of passenger rolling stock</td>
<td>113</td>
</tr>
<tr>
<td>B.5</td>
<td>Propulsion resistances of freight rolling stock</td>
<td>113</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The different vertex types and associated data of a Spurplan-graph.</td>
<td>24</td>
</tr>
<tr>
<td>3.1</td>
<td>Notation for input data of the train dispatching model</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>Notation for pre-processing data of the train dispatching model</td>
<td>41</td>
</tr>
<tr>
<td>3.3</td>
<td>Notation for variables of the train dispatching model</td>
<td>42</td>
</tr>
<tr>
<td>3.4</td>
<td>Key figures of a STD instance</td>
<td>53</td>
</tr>
<tr>
<td>B.1</td>
<td>Notation for modelling train dynamics</td>
<td>106</td>
</tr>
<tr>
<td>C.1</td>
<td>Conflict clique sizes in the RTCG model for the three scenarios</td>
<td>118</td>
</tr>
<tr>
<td>C.2</td>
<td>Problem sizes for the two train dispatching models (CG and RTCG)</td>
<td>118</td>
</tr>
<tr>
<td>C.3</td>
<td>Processing times for the two train dispatching models (CG and RTCG)</td>
<td>119</td>
</tr>
<tr>
<td>C.4</td>
<td>Solution quality and progress for the two train dispatching models (CG and RTCG)</td>
<td>120</td>
</tr>
</tbody>
</table>
# List of Algorithms

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Computing $\Omega_{p,p'}^{v,u}$</td>
<td>48</td>
</tr>
<tr>
<td>A.1 Yen’s algorithm for computing (travel) paths</td>
<td>102</td>
</tr>
<tr>
<td>A.2 Yen’s algorithm: Computing a temporary graph</td>
<td>103</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The increasing demand for transportation services in already densely operated railway networks has fostered recent innovations in railway traffic management. With these innovations, network operators intend to improve on capacity usage, reliability as well as reduce costs. These intentions are shared in our motivation for this thesis, since the efficiency and quality of a railway transportation system is also a matter of public concern. After the motivation we describe how railway traffic is managed today, followed by an overview of the current industrial state of the art and development trends. In the last part of this chapter we propose a framework for railway traffic management which capitalizes on the new opportunities provided by the recent developments in the field. This thesis focuses on a first step towards the realization of this framework, namely the realization of an automated dispatching assistant. For this purpose we apply operations research techniques, i.e. we use advanced analytical modelling methods to help make better decisions.

1.1 Motivation

The increasing demand for public transportation services is a challenge for railway network operators. The Swiss Federal Railway (German: Schweizerische Bundesbahnen, SBB) has recorded a continuous increase in demand for passenger (see Figure 1.1) and freight traffic over the past decade. As a consequence, the SBB is operating at capacity at various places in their network. Although infrastructure and rolling stock investments increase capacity along selected lines and at key hubs, these investments are cost-intensive, subject to environmental concerns and in densely populated areas the expansion of the network is a slow and difficult process (the project 'Stuttgart 21' in Germany is a notorious example Geissler (2011)).

An alternative to capacity augmentation is the improvement of capacity usage. We identify the following improvement fields:

**Infrastructure** - Improved signalling systems reduce train headway times (e.g. compare developments in the European Train Control System (ETCS) Eichenberger (2007), Winter (1998)), thereby trains can follow each other at a higher rate and
thus, the capacity usage is improved. Furthermore, continuous and proper maintenance of rolling stock and infrastructure ensures more reliable operations, which in turn causes less disturbances and improves the capacity usage.

Planning - Similarly (robust) timetabling (Caimi, 2009), crew scheduling (Huisman, 2004) and rolling stock scheduling (Maróti, 2006) heavily influence the quality of capacity usage.

Operations - Monitoring, data provision, rail control systems (Dolder et al., 2009, e.g.) and dispatching support are integral to smooth operations, which results in a better usage of capacity.

This thesis addresses innovation in operations, namely a railway traffic management support system for complex central railway stations. Although research on railway traffic management is quite active (see Section 3.1), a successful implementation in practice is still missing. This is partially due to the novelty of applying computer supported approaches to a domain governed for decades by humans, but even more so, due to the inherent complexity of coordinating various, by themselves difficult aspects:

- adequate level of model detail (i.e. blocking time theory)
- modelling the network topology/layout in terms of the number of alternatives for train routing and the number of potential resource conflicts
- variety of speed profiles
- accurate prediction of the train movements (especially after disturbances or events)
- coordination with neighbouring territories (especially for central railway station areas)
• limited available response time

Nevertheless, recent developments in railway operations (see Section 1.3) can facilitate the development of smooth traffic management systems for a more efficient use of infrastructure capacity. A smooth traffic management system is key to reduce traffic bottlenecks, which often arise in central railway stations. In our proposal for a smooth traffic management system we design a new near real-time decision support system, which manages traffic in bottleneck areas by arrival and departure time adjustments and re-routing of trains as well as partial speed profile coordination.

1.2 Railway traffic management today

Today railway traffic is managed by dispatchers through a closed control loop illustrated in Figure 1.2.

Figure 1.2: A dispatcher controls trains through dispatching decisions based on a forecast of train movements in form of a closed control loop.

In the physical layer, detection systems collect information of train movements. This information is processed by the interlocking and signalling system in order to provide information to the operational information layer about the position and velocity of trains, the track clearance status of (block) sections between signals as well as the status of switches and signals.

The operational information layer maintains a forecast of the train movements. This forecast must be updated based on the dynamic changes taking place in the physical layer and (in this context) static data, such as timetable information, network infrastructure data or characteristics of train dynamics.
Finally, in the control layer the dispatcher observes at his (computerized) workstation the forecast of the train movements and takes dispatching decisions based on his expertise. These disposition decisions need then to be translated into control actions such as switching commands for signals and switches or movement commands for the control of speed of trains. If a disposition decision affects the timetable, the operational timetable (of the operational information layer) has to be updated accordingly.

Subsequently, today’s dispatcher’s work station is explained in detail, followed by a list of actions a dispatcher has at his disposal in order to intervene in the case of a disturbed situation. Finally, we show how several dispatchers, each of them responsible for a designated geographical area, work together in a hierarchical organisation.

1.2.1 Dispatchers work station

Today’s modern dispatcher’s work station is equipped with a group of monitors providing (almost) up to date information of the state of the railway network by means of different graphical user interfaces (GUIs). Figure 1.3 shows a dispatcher in front of such a work station at the traffic management centre in Zurich, Switzerland.

Figure 1.3: The new traffic management centre in Zurich, Switzerland offers many work stations for a crew of dispatchers. Each dispatcher operates with the help of a group of monitors, each of which displays different interactive graphical user interfaces from the Swiss Rail Control System (RCS). Copyright @ SBB (www.sbb.ch)
The dispatcher’s most important GUI displays the so-called *time-distance graph*. In this GUI the dispatcher can select a specific railway line, which results in the display of a forecast of future running trains along that line. Figure 1.4 depicts such a time-distance graph from a dispatcher’s work station at the SBB. For a selectable railway line (on the top, from left to right) the forecast of train speed profiles as well as past (area in grey) train runs are illustrated graphically by time-distance curves. The distance is measured along the x-Axis and the time runs downwards along the y-Axis, while the current time is highlighted by the horizontal white bar. The forecast of the future train runs is displayed by train time-distance curves. These curves are maintained in the operational information layer of the control loop (see Figure 1.2). Based on the (almost) up to date forecast of the future behaviour of the railway network along a railway line under supervision, the dispatcher has the possibility to intervene by issuing dispatching decisions directly through the graphical user interface in order to feed back his dispatching decisions to operations, i.e. the signalling and interlocking system or train drivers. We subsequently describe the action space for such dispatching decisions.

1.2.2 Action space for dispatching

For the solution of conflicts the dispatcher has in principle, the following measures at his disposal:

- Change a train’s *route* or even *travel path*: Each interlocking provides a set of routes along which a train’s movement can be authorized by the safety system. When a
train is passing through a sequence of interlockings and block sections, the chosen routes (and block sections) can be chained together to form the travel path of the train. In order to prevent a conflict, a train can be rerouted inside an interlocking (small scale re-routing) or its complete travel path might be adjusted (large scale re-routing).

- Adjust a train’s speed profile. A speed profile captures the movement progress of a train along a fixed travel path and is commonly visualized in time-distance graphs. Although the admissible speed profiles along a train’s travel path are constrained by physical and safety limits or even by (soft) limits originating from a timetable, the continuous vehicle dynamics admit in principle, infinitely many speed profiles. Today this flexibility is not fully exploited and usually just used to slow down trains in order to harmonise speeds in mixed traffic situations. However, current developments in the railway industry aim to exploit this flexibility further, in order to save energy and as a dispatching instrument (compare Section 1.3.3).

- Change of dwell times including adding or removing a stop along a train’s travel path.

- Change of start time: The start time of a train movement is either the time of a train movement entering the control area or the time where a train movement starts inside the control area. Both can be changed, if the dispatcher’s decision is made sufficiently in advance.

- End a train movement early.

- Drop a train movement completely.

- Turn around a train earlier than planned: When a train moves back and forth along a train line, it might be advisable to turn around the train earlier than planned.

1.2.3 Hierarchical control by geographical decomposition

Worldwide, railway traffic management across national networks is hierarchically structured according to functional purposes (Figure 1.5) and achieved by geographical decomposition of the network at hand. At the operational level dispatchers in traffic management centres supervise the railway network and resolve situations occurring after delays, failures, interruptions or similar unplanned events. Dispatchers’ decisions are forwarded to remote traffic control centres usually by phone. Traffic control centres set the train routes and issue speed limits through signals as a measure of control. Train positions are gathered automatically at the lower level and forwarded upwards to the dispatchers enabling supervision of the trains.

Note that in such a hierarchy, the dispatchers’ possibilities for train control are limited to the possibility of route setting and by means of signals speed can be regulated as well. Furthermore, a dispatching order has to pass through traffic control centres, thereby delaying the time until this order comes into effect. Although direct radio or phone communication with a train driver is occasionally used to prolong the dwell time of a train, accelerate or slow down a train, a continuous control is currently not in place. The next
1.3 Industrial state of the art and development trends

1.3.1 Centralising control

The hierarchy in railway traffic management is changing as a result of two important trends: First, traffic control centres have been merged with traffic management centres to...
speed up the control flow from the tactical control level to the operational level. And second, new communication channels between dispatchers at traffic management centres and train drivers are more able to provide continuous and direct control of trains. Figure 1.6 illustrates the effects of these trends on the future control hierarchy and Figure 1.7 shows the intended geographical decomposition of train control over the Swiss railway network by the year 2014.

![Figure 1.7: By 2014, four traffic management centres in Switzerland (in German "Betriebsleitzentrale, BZ") shall control the railway traffic in their corresponding network areas. Additionally, the "Operation Center Infrastructure" has the lead in network-wide disposition matters. Source: SBB Infrastructure, www.sbb.ch](image-url)

### 1.3.2 Automatic conflict detection

There is also a trend towards the development of automated conflict detection. Until recently, dispatchers have been working with computer generated time-distance graphs to visually identify operational conflicts (i.e. endangered connections, concurrent allocations, knock-on delays, etc.). Automatic conflict detection shall facilitate the dispatchers’ decision making by automatically identifying potential conflicts and bringing them to the attention of the dispatcher. The detection is based on comparing the forecast of train movements (see Section 2.7), conveyed in form of blocking time stairways. A future (and therefore potential) conflict is the predicted, simultaneous allocation of a block section by more than one train movement.
1.3.3 Adaptive train control and speed profile optimization

Currently railway operators are developing *adaptive train control* which allows them to optimize speed-profiles of trains by so-called green wave policies, preventing unnecessary stops of trains at red signals. In the following we briefly describe this concept and show how adaptive control could be used to coordinate the entry of trains into bottleneck areas.

In traditional railway systems a train stops when encountering a red signal, which is activated by an automated signal system exclusively as a safety measure, in anticipation of an occupied track section ahead (Figure 1.8a). One could envisage a better solution to such situations (compare Figure 1.8b) by a more active train control, which would allow traffic management to adapt and optimize the speed profile of a train according to its environment in the railway system. The optimized speed profile first reduces the train speed in anticipation of the occupied track section ahead, then increases the speed appropriately in anticipation of the conflict being resolved. This allows the train to again reach maximum possible speed at the point where the signal had been red but has already gone back to green, thus eliminating wasted acceleration time following the *green wave* (GW) policy: The trains only encounter green signal lights and ultimately the railway network achieves smoother operations and as an additional benefit, reduce energy consumption.

![Figure 1.8: A conflict situation and two paradigms resolving it.](image)

GW policies are seen as particularly effective for smoothing operations in areas of central railway stations. A first computational, quantitative study of the real-time effects of the GW policy bottleneck areas of the Dutch network was done by [Corman et al. (2009)](cite). Corman concluded that the GW policy is assessed to be effective for reducing delays and
energy consumption, but it is only applicable with the help of advanced decision support systems in order to resolve occupation conflicts. He also points out that the efficiency of a GW policy is dependent on sufficient capacity in the stations. Railway operators are currently developing continuous communication channels between dispatching centres and train drivers, allowing for a more active and continuous train control. Computed GW speed policies could be forwarded by such communication channels to the train driver and the train driver will become aware of upcoming red signals lights before he sees them. Even more importantly, with continuous train control it is possible to dictate the time at which a train passes a certain point in the railway network. We suggest exploiting this envisaged possibility by accelerating or decelerating trains as they approach the bottleneck area, in order to prevent knock-on delays involving trains both inside and outside the bottleneck area. Active and continuous train control together with speed profile optimization are therefore essential technologies for the subsequently described approach for train dispatching.

1.3.4 Towards sophisticated train control systems

The increasing difficulties and costs faced in railway operations is often exacerbated for historical reasons by incompatible signalling and interlocking systems, different signal-cabin technologies, differences across national railways and by increased network traffic densities. In response, in 1990 the European Union formed an initiative, the so-called European Rail Traffic Management System (ERTMS), to enhance cross-border interoperability by the creation of European-wide standards for train control systems. In 2003 a cost/benefit analysis was performed for future ERTMS corridors in Europe by the UIC, which describes the large variety of control command systems (more than 20!) existing across Europe at that time.

ERTMS basically partitions into two main components, the European Train Control System (ETCS), a standard for in-cab train control, and GSM-R, the GSM mobile communications standard for railway operations as enabling technology. Besides the harmonisation, ETCS can also provide the technical basis for improved capacity usage [Eichenberger 2007], safety and quality of the European railway network. To satisfy the many different requirements originating from different tracks, usage principles and railway administrations, it was decided to introduce different levels in ETCS called ETCS Level 1 to 3. ETCS Level 1 up to Level 2 still use fixed track blocks, while in ETCS Level 3 trains are driving independent of fixed blocks with the help of cab signalling. ETCS Level 3 will thereby allow for better capacity usage by means of denser train sequences. The implementation, development and roll-out of ETCS are tedious processes for both political and economic reasons.

In Switzerland, besides the developments concerning the ERTMS project, in 2009 large investments were made to integrate and introduce a system-wide rail control system, called (Swiss) Rail Control System (RCS). The SBB infrastructure division started the development of RCS in the year 2005, it was introduced in the year 2009 and since then it has been operational. In Dolder et al., 2009 the authors describe the goals for the development of RCS as follows:

Since 2005, the Infrastructure Division of SBB has been developing RCS
as a standardised, integrative dispatching system for rail traffic on the SBB, BLS (Bern-Lötschberg-Simplon) and SOB (Schweizerische Südostbahn AG) networks. Operations management currently operates with two different software systems (VST Zurich and SURF), which originate from the late 1980s and mid 1990s respectively. RCS will replace the existing rail operations systems and in doing so will play a role in increasing the quality and stability of operational production of what is an extremely dense rail traffic.

Today, RCS is an integral part for the train control loop and executes central tasks in daily operations illustrated in Figure 1.9. Currently, additional efforts are made towards an integrated and automated conflict detection, adaptive train control and speed profile optimization.

1.4 Focus and contribution of this thesis

This thesis aims at exhibiting the synergism of recent and current industrial research and development trends in railway operations tied with the progress made in combinatorial optimization (see Mahjoub, 2011) and its application to railway dispatching. Our ambition is to convince the railway industry that operations research methods are key to improving capacity usage as well as being beneficial in fostering the development of computer supported dispatching systems. For this matter, the thesis suggests a general framework for the integration of a computer supported dispatching assistant into railway operations and provides proof of this concept based on an initial, but comprehensive case study.

Figure 1.10 describes our proposed design of a closed-loop discrete-time control framework for train dispatching. A forecast module computes a forecast of train movements based on the current system state and intended commercial offer. Based on this forecast and intended commercial offer, a (mathematical) rescheduling model computes new disposition schedules (Section 3.3). These disposition schedules can be combined with time-distance graphs from the forecast in order to visualize their effects on the system for
the dispatcher. Once the dispatcher takes a decision, it is forwarded to the infrastructure as well as to the trains and the intended commercial offer is changed accordingly, thus closing the control loop and thereby rescheduling trains in discrete time steps (Chapter 4).

To realize this framework in operations, it is necessary to approach this goal gradually as the mere complexity of the framework makes it impossible to develop it at once. Thus, we suggest developing the framework in four consecutive steps. The first step is the focus and the major contribution of this thesis, whereas the remaining three steps are discussed in Chapter 6.4 as an outlook for future work.

This first step includes the design, implementation and rudimentary testing of a mathematical rescheduling model based on operations research techniques. This step focuses on showcasing the basic viability of applying optimization techniques in railway traffic management by means of a credible case study for the central station area Berne, Switzerland. The study is based on authentic recorded, historic operational data provided by SBB, whereas the physical layer is rudimentary simulated with additional sampled disturbances. Note, that dispatchers did not directly take part in this study but they were essential advisers for building the mathematical rescheduling model.

Figure 1.10: The suggested closed-loop discrete-time control framework for dispatching trains.
1.4 Focus and contribution of this thesis

Figure 1.11: Step one of realising the closed-loop framework: A dispatching model is implemented in the control layer and tested in a limited control loop.

For the case study of this thesis, this first step was implemented through the following three tasks (see Figure 1.11):

I. Prepare recorded historic, operational data (timetable) and topology data

II. Forecast the train movements

III. Iteratively:
   a) Instantiate a (mathematical) model for dispatching in the control layer based on the forecast of train movements
   b) Solve the dispatching model in order to get a new disposition schedule
   c) Superimpose the disposition schedule with sampled disturbances at the physical layer
   d) Simulate the railway system according to the new (disturbed) disposition schedule by following the forecast of train movements
1.5 Outline of the thesis

The thesis is structured as follows: Chapter 2 explains how the railways system can be modelled in an operational information layer. This includes the modelling of the railway infrastructure and the movement of trains on this infrastructure. These models are then used to forecast the train movements, which is the point of departure for static train dispatching in Chapter 3. In this chapter we explain how mathematical optimization models can find conflict-free and qualitative disposition schedules for a specific point in time given the forecast of the train movements. Chapter 4 then continues by introducing a model predictive control approach, which uses the developed optimization models for static train dispatching in a dynamic context, thus allowing their iterative application in a closed-loop discrete-time control framework. In Chapter 5, an extensive case study showcases the viability of the model predictive control approach to the dispatching of railway traffic in the central main station area Berne, Switzerland. The thesis concludes in Chapter 6 with an appraisal of the results and an outlook on the necessary steps for a realization of the closed-loop control framework in practice.
We briefly introduce the reader in the following to the characteristics of railway systems and current operating and control principles based on the "Signalling and Interlocking Compendium" Chapters 1 and 3. For a more extensive and comprehensive description on these topics the interested reader is referred to Pachl (2002, 2011). This chapter then explains railway topology models based on graphs and furthermore, how train movements can be represented on top of these topology graph models. A train movement consists of three components: The travel path describes the sequence of used railway tracks, the speed profile determines the time the train is at a given position along its travel path and the blocking time stairway contains the temporal allocation of infrastructure elements by the signalling system for the train movement. We present a model for each of these components and provide algorithms for these models. These concepts are critical for forming a forecast of train movements (tasks II and IIId in Figure 1.11) and furthermore, the train dispatching model discussed in Chapter 3 is built upon the concept of blocking time stairways.

2.1 Railway system characteristics

Railway systems have two identifying characteristics:

1. The rail tracks guide the movement of trains and thereby the trains’ paths are pre-determined. As a result, trains can only pass each other at particular locations, such as loops.

2. Trains have a relatively high running speed but in contrast to cars the applicable braking effort is quite poor. This poor braking behaviour results from limited adhesion between steel wheels and the steel rail and from large inertia due to the heavy mass of a train.

Consequently railway systems require three safety-related mechanisms:
• **Route locking**: All movable route elements in front of a train have to be locked and kept in their correct positions, until they have been completely passed by that train.

• **Route clearing and safeguarding**: All track sections in front of a train have to be clear and kept clear, until they have been completely passed by that train.

• **Speed control**: A train driver has to trigger a speed change sufficiently in advance, in order to reach a target (permitted) speed in the future.

These three mechanisms have developed over time alongside with technological innovations, but at varying speed and in different forms among national railways. This has led to significant international differences, but despite these differences, the original operating principles of any railway system can be classified into either the British, the German, the North American operating principles or a mixture of them. Our collaboration partners, the SBB, operate trains according to the German operating principles. Henceforth, this brief introduction is restricted to the German railway network infrastructure as opposed to that of the British and North American’s.

### 2.1.1 Classification of (German) railway infrastructure

Figure 2.1 covers the essential technical terms in railway networks and illustrates classified railway tracks, signals and network areas. In this section we explain these terms and classifications, a basic knowledge required for the later discussion of train disposition.

![Diagram of railway network infrastructure](image)

**Figure 2.1**: Partition of the railway network according to German operating principles and terminology with left-sided traffic operation. The figure is based on illustrations in [Pachl, 2011](#).

**Tracks**

We distinguish two types of railway tracks based on the type of train movements which are executed upon them:

1. **Main tracks** are used for regular train movements, e.g. a station track, a loop or a track belonging to an open line.

2. **Secondary tracks** (British: **Sidings**) are not used for regular train movements, but for shunting movements; e.g. yard tracks.
2.1 Railway system characteristics

Signals

Trackside signals can be classified into three categories according to their functionality (Pachl, 2011):

1. A main signal indicates the authorization status for the entrance of trains into the track section beyond that signal.

2. A distant signal announces an approach (or warning) aspect to the train driver: If a main signal is currently not authorizing an entrance into the track section beyond that signal, a warning aspect has to be announced by the distant signal to approaching trains, such that the train can stop in time in front of the red signal.

3. A shunting signal authorizes shunting movements into the track section beyond the shunting signal.

Main signals subdivide further into signal subclasses according to their location:

1. A home signal at the entry of a station area (Bahnhof) authorizes train movements to enter the station area. Opposing home signals limit the station area.

2. An exit signal at the exit of a station area (Bahnhof) authorizes train movements to exit the station area into the next block section beyond the signal.

3. An intermediate interlocking signal inside a station area authorizes train movements within the station area.

4. A block signal authorizes train movements into the block section beyond the corresponding signal of an open line area.

Network areas

The railway network can be partitioned into two types of areas:

1. A station area is defined as the area between two opposing home signals, which contains at least one turnout. Train movements originate, terminate and turn in these areas.

2. An open line is the area between two station areas, usually consisting of parallel tracks.

Additionally the following terms are used to classify parts of the railway network:

- A block section is a track section between two (consecutive) main signals, which is not (completely) inside a station area.

- An interlocking is (according to Pachl, 2002) an arrangement of switches and signals interconnected in a way that each movement follows the other in a proper and safe sequence. At first interlocking of actions was achieved mechanically through the locking frame, then electro-mechanically by relays in the signal box. Now, interlockings are largely computerised using a two in three voting system, diverse hardware and software or protocols (Schmid and Watson, 2011).
20 | Chapter 2: Modelling the railway system: topology and train movements

• A junction is an arrangement of signals and switches which converges or diverges two or more railway tracks.

• A station junction is a junction outside of a station area and part of an open line.

• A crossover is an arrangement of two switches connecting two parallel tracks which allows trains to cross over from one track to other

• A station crossover is a crossover outside of a station area and part of an open line.

Train movements in station junctions and station crossovers are often safeguarded by an interlocking safety system. The following section describes how trains are safely operated in the railway network.

2.2 Safe train operation and control

Railway operations distinguishes two movements in the network which require some form of control:

1. A train movement is a self-propelling railway vehicle (usually a locomotive) and zero or more coupled railway vehicles, which have the authority to move on main tracks. Note that in the following chapters we often refer to train as a short-cut name for train movement.

2. A shunting movement is a movement of one railway vehicle or several coupled railway vehicles without explicit authorization to move on main tracks. They are executed over short distances and at low speed to allow the train driver to drive by sight. If a shunting movement involves moving on main tracks, this movement is not explicitly authorized by main signals but will be safeguarded by shunting signals and an additional shunting aspect at involved main signals.

Subsequently, we discuss the safe separation of train movements from one another.

2.2.1 Separation of train movements

The separation of train movements is classified into three categories according to it’s form of signalling:

Fixed block signalling

Figure 2.2 illustrates the basic layout of a fixed block signalling system: The main signals govern the entry of train movements into fixed block sections. Distant signals provide a warning aspect to the train driver and clearing points serve as train rear detection points, which allow the safety system to clear any occupations of previous block sections and overlaps of the train movements associated with the detected trains. Overlaps are an additional safety measure in case a driver fails to stop a train before a (red) stop signal: Many main signals are equipped with a safeguarding mechanism, which automatically
2.2 Safe train operation and control

Clearing point behind signal 1
Clearing point behind signal 2
Clearing point behind signal 3

Block section
Overlap

Control length of signal 1
Control length of signal 2

Figure 2.2: Basic network layout of a fixed block signalling system and control areas (This figure is based on illustrations in Chapter 3 (Pachl) of the Signalling and Interlocking Compendium).

The benefits for railway operations of a cab signalling system differ greatly depending on the sophistication of the on board cab signalling system. One can distinguish three major categories of cab signalling systems:

1. Cab signalling without supervision of braking curves and fixed block sections
2. Cab signalling with supervision of braking curves and fixed block sections
3. Cab signalling with continuous automatic train protection system

For the first category one can remove (or at least reduce) the signal viewing time in the calculation of the blocking time (compare Figure 2.9), but the approach time remains the same since there is no additional supervision of braking curves to prevent jumping a red signal. In the second category, the additional (on board) supervision of braking curves reduces the approach time. The last category requires a continuous automatic train protection system which includes a continuous surveillance of the train completeness. In fixed block signalling systems the train completeness is examined by (discrete) track-side installations, resulting in a train protection based on fixed blocks. On the other hand, a continuous train protection system safeguards trains by following a moving block principle, which could result in improved infrastructure capacity usage. In moving block protection systems, the block length can be regarded as being zero and thus, the block passing time becomes zero as well. However, Pachl (Chapter 3 in Signalling and Interlocking Compendium) points out, that the potential gain in capacity is often over-estimated, when it becomes actually rather limited on mixed traffic lines.

Non-signalled operation

Non-signalled operation is based on radio or telephone communication between the dispatcher and the train crew. This operation mode is in place at territories without any signalling system (dark territories), where movement authorities have to be issued by a verbal communication channel. In Europe’s central station areas such dark territories do not exist, thus this thesis focuses on fixed block signalling (with the optional help of cab signalling).

2.3 Modelling the railway topology

For the purpose of train dispatching, a railway topology model should provide all necessary data for the computation of travel paths, speed profiles and blocking time stairways. Travel paths are usually computed by path searching algorithms which operate in graphs as a model of the railway topology. Once a travel path has been computed a speed profile and a blocking time stairway along that travel path can be assigned. In the following we look at different approaches to modelling the railway topology by graphs, keeping these computations in mind: Montigel (1992) proposes the so-called double vertex graph where the double vertices implicitly allow the computation of admissible travel paths. An alternative graph model was proposed in Brügner (1995), which is conventionally called Spurplan graph. The Spurplan graph encodes data which is used for speed profile and blocking stairway calculations, but lacks an implicit encoding of the admissible travel paths. In order to determine travel paths the so-called Fahrnetz, a directed graph over an aggregated topology, can be used. In this thesis we use a similar modelling approach, namely we encode the topology data in a directed railway topology graph and the routing information in the directed routing graph which can easily be transformed into a Fahrnetz graph. On one hand the railway topology graph is suitable for the computation of speed profiles and blocking time stairways, while on the other hand the directed Fahrnetz graph allows for fast computation of travel paths. In this section we subsequently
describe these different graph models and additionally, the so-called *demarcation portals*, the border points which confine the main station area and can serve as coordination places with neighbouring dispatching areas.

### 2.3.1 Double vertex graph

Caimi (2009) defines a double vertex graph in his thesis as follows:

**Definition 2.1 (Double vertex graph).** Let $V$ be a finite set of vertices, $E \subseteq V \times V$ a finite set of edges between the vertices where $E$ does not contain any loops or multiple edges. Moreover, let $\circ : V \rightarrow V$ be the joining mapping, which satisfies $\circ(v) \neq v$ and $\circ(\circ(v)) = v$ for all vertices $v \in V$ and where $v^\circ = \circ(v)$. Then a triple $D = (V, E, \circ)$, where $E$ contains no edges of the form $(v, v^\circ)$ is called a double vertex graph.

Additionally, Caimi (2009) suggests two assumptions based on railway infrastructure properties. These assumptions are slightly changed here in order, such that three way turnouts (a special type of railway junction) can be modelled as well:

**Assumption 2.1 (Railway double vertex graph properties).**

**i)** No vertex has more than three outgoing edges (there are no switches with more than three outgoing tracks).

**ii)** There are no loops or edges connecting two joined vertices (this special case does not occur in railway tracks).

Figure 2.5 illustrates the modelling of different types of *turnouts* (i.e. the junctions in railways where lines diverge or converge) by double vertex graphs.

### 2.3.2 Spurplan graph

The Spurplan graph (see Figure 2.3) was first suggested to be used for modelling railway topologies in Brünger (1995). Since 1995, further developments were incorporated and it is now the commonly used railway topology model at the German Railways and serves as a basis in the railway operations research software LUKS of VIA Consulting &
Development GmbH. As there is no recent and detailed public documentation available we complemented and summarized with the support of Dr. Weymann, the descriptions of the Spurplan-graph given in [Jacobs (2003)] and [Weymann (2011b)]:

The infrastructure topology is modelled with the undirected Spurplan-graph. All topology information is stored at the vertices, the edges have no data associated except chainage length and the edges represent in principle the railway tracks. Vertices represent either infrastructure elements along the railway tracks (i.e. signals or switches) or changes in track properties (i.e. grade change or maximum allowed velocity change). Each vertex contains as data an identifier, the location usually provided in the form of a railway line and the chainage number, its type and type dependent data. Table 2.1 lists in the left column the different vertex types and in the right column additional type dependent data. Additional edges between logical elements are introduced with length zero and keep the Spurplan-graph connected. An example are edges between two demarcation portals or edges between vertices of a switch which encode the possible routings over that switch.

| Two-way turnout                  | Three vertices: the switch-head, the main track and the branch track and each of them uses as location the centre of the switch. |
| Crossing or slips                | Four vertices: left-start, left-end, right-start and right-end. The vertices are connected by edges according to transition possibilities. |
| Interlocking border              | One vertex indicating an end of an interlocking area. |
| Demarcation portal               | One vertex indicating an end of the considered dispatching area. |
| Bumper                           | One vertex indicating an end of a track, where a safeguarding bumper is located. |
| km-Jump                          | One vertex indicating a sudden change in a railway line chainage. |
| Signal                           | One vertex indicating a track-side signal and its type (Pre-, Main- or Protection-signal). |
| Train completeness               | One vertex indicating a point where the completeness of a train is checked for the safe release of a route in an interlocking. |
| Time measurement                 | One vertex indicating a measurement point for passing train movements. |
| Stopping point                   | One vertex representing a location for a (potentially) planned stop. Contains additional data concerning the usable length and the train types, which are allowed to stop. |
| Speed limit                      | One vertex containing information on the maximal speed from this point onward or starting from the last passed signal. |
| Grade change                     | One vertex containing data on the change in grade. |

Table 2.1: The different vertex types and associated data of a Spurplan-graph.

2.3.3 Fahrrnetz

The Fahrrnetz graph complements the Spurplan graph by its purpose of faster travel path computations. In the Fahrrnetz graph each vertex represents either a route, a start or end
2.3 Modelling the railway topology

terminal and arcs between the vertices indicate valid transitions for a train movement through the railway network. A route itself can be thought of as an ordered list of vertices in the Spurplan graph. A valid transition from a route A to another route B is only possible if route A ends in the corresponding Spurplan graph at an interlocking border vertex and that vertex is additionally connected by an edge to the interlocking border vertex where route B starts. Finally, a directed path in the Fahrnetz is a travel path composed of a start terminal, a chain of routes through interlockings and an end terminal. An example of a Fahrnetz graph is given in Figure 2.4.

**Definition 2.2 (Fahrnetz graph).** A Fahrnetz graph is a directed, weighted graph $G(V, A, c)$ with $n$ vertices and $m$ arcs, where any vertex $v \in V$ has an associated non-negative cost $c_v \in \mathbb{R}^+$. The costs have to be assigned by experts and a vertex cost $c_v$ represents the inconvenience generated for a train movement when it uses the route associated with vertex $v$ and in case that $v$ is a terminal vertex the cost is set to be zero.

Section 2.4 gives an overview on algorithms on path computation in such graphs and concludes with the algorithm applied in the case study of this thesis.

![Figure 2.4: The Fahrnetz graph is a directed graph over which travel paths are computed, with the advantage of not containing multiple arcs but usually requires significantly more nodes and edges than the travel path graph. This Fahrnetz graph is deduced from the routing graph in Figure 2.7.](image)

2.3.4 Railway topology graph

The railway topology graph is closely related to the Spurplan graph and was used to model the topology in this thesis. The difference between the two graphs are the added orientation in the railway topology graph, thus edges are directed and two adjacent vertices may contain two differently orientated edges between them, if the two vertices can be passed in both directions by a train movement. Additionally each vertex contains a transfer matrix, which specifies the transfer possibilities from incoming arcs to outgoing arcs. In Figure 2.5 railway topology graphs are shown for different types of turnouts and
the transfer possibilities are illustrated in a simplified way by green or red coloured arcs: A train movement can either follow the green or the red arcs, but not a mixture of them. 

Figure 2.5: Connor (2011) classified different types of turnouts (left column). In the middle column the turnouts are modelled as double vertex graphs and in the right column as railway topology graphs.

The railway topology graph contains in principle the same vertex types and data as in the Spurplan graph, but some other data is added in order to model peculiarities of the Swiss railway network. One important peculiarity are logical reference points at each interlocking. At each interlocking these points are located at intersections formed by a cut through the interlocking (see Figure 2.6).

Figure 2.6: MitteAGs are logical reference points formed by cuts through interlocking areas and used for forming travel paths by consecutively joining routes at these points.
In German these points are called 'Mitte Aufnahme Gebäude' or short 'MitteAG' which translates as 'middle of a storage facility'. MitteAGs have been designated by the SBB infrastructure operator over time. Today, these points reference possible start and terminal locations of routes. By joining routes at MitteAGs a routing graph can be formed:

### 2.3.5 Routing graph

Similar to the Fahrnetz graph, the routing graph aggregates the railway topology and implicitly stores the feasible travel paths. The routing graph is a directed multigraph, in which each vertex represents a reference point (at SBB called MitteAG, see Section 2.3.4) and each arc between two vertices represents a route to get from the tail vertex to the head vertex. Figure 2.7 showcases a routing graph. Due to the possibility of multiple arcs between two adjacent vertices the number of travel paths between two MitteAGs (vertices) can grow exponentially in the number of edges, e.g. in the provided example there are already 22 paths to get from MitteAG 1 to the MitteAG 8. In our case study we first transformed the routing graphs provided by SBB into Fahrnetz graphs, which have the advantage of not containing multiple arcs, in order to facilitate the later computation of travel paths.

![Figure 2.7: An example of a routing graph with multiple arcs.](image)

### 2.3.6 Demarcation portals

Section 1.2.3 describes the hierarchical control of railway traffic based on geographical control areas. This network partitioning principle in railway operations and the problem complexity of train dispatching advise to apply a network decomposition in a similar fashion when modelling train dispatching. We suggest using demarcation portals at the borders of the network area under consideration. These demarcation portals are conceived to be logical coordination points between different control areas, where the time and the velocity at which train movements cross from one control area to an adjacent area shall be coordinated. This thesis focuses on one type of control area, namely central station areas. Thus, for our case study the demarcation portals were located after discussions with dispatchers at designated boundary points of the central station area.

### 2.3.7 Topology models used in this thesis

Among the previously described approaches for modelling the railway topology for the case study of this thesis the following were used:
1. The *railway topology graph* was used for the computation of speed profiles and blocking time stairways.

2. The *routing graph* is the topology model used by the SBB to encode travel paths and the topology data of the case study was provided to us in the form of this model.

3. In the *fahrnetz graph* the travel paths were computed. The fahrnetz graph can be derived from the routing graph by a simple transformation (see Figure 2.4).

### 2.4 Travel paths

This section defines the concept of a travel path, introduces some basic notation and discusses the problem of computing the $K$ shortest, loop-less paths between two vertices in a Fahrnetz graph.

#### 2.4.1 Definition and notation

Section 2.3 described different approaches for modelling the railway topology. In the following definition of a travel path we assume that the aggregated topology is modelled by a Fahrnetz graph:

**Definition 2.3 (Travel path).** A travel path $p$ from $i \in \mathcal{V}$ to $j \in \mathcal{V}$ in a Fahrnetz graph $G(\mathcal{V}, \mathcal{A}, c)$ is a sequence of the form $p = (i = v_1, v_2, \ldots, j = v_{l(p)})$, where $(v_k, v_{k+1}) \in \mathcal{A}$ for any $k \in \{1, \ldots, l(p) - 1\}$. Additionally, travel paths shall be loop-less, meaning that no vertex shall be visited more than once. $l(p)$ denotes the length of $p$, that is, its number of vertices and the distance, or total cost, of $p$ is defined by $c(p) = \sum_{v \in p} c_v$.

For the subsequent problem statement and the discussion of Yen’s algorithm (see Appendix A) for computing the $K$ shortest, loop-less paths some notation is required:

- $\mathcal{P}_{ij}$ ($\overline{\mathcal{P}}_{ij}$) denotes the set of (loop-less) paths from $i$ to $j$ in $G(\mathcal{V}, \mathcal{A}, c)$.
- $\text{sub}_p(x, y)$ denominates the sub-path from vertex $x$ to vertex $y$ of a path $p$.
- Two paths are said to be equivalent, written as $p \equiv q$, if the sequence of nodes is exactly the same.
- The concatenation of two paths $p \in \mathcal{P}_{ii}$ and $q \in \mathcal{P}_{ij}$, written as $p \diamond q$, is the path from $i$ to $j$ formed by path $p$ followed by $q$.

#### 2.4.2 Computing $K$ shortest, loop-less paths

In the context of computing reasonable paths in a railway network, one is not interested in computing all conceivable paths between two destinations but rather looking for computing the $K$ most reasonable paths, meaning the $K$ shortest, loop-less paths. We slightly adapt the definition provided in Pascoal (2006) for the $K$ shortest, loop-less path problem to the context of travel paths in Fahrnetz graphs:
2.5 Train speed profiles

Definition 2.4 (K shortest, loop-less path problem). Given a Fahrnetz graph $(V, A, c)$, a desired number of paths $K \in \mathbb{N}$, a start and end terminal vertex $s, t \in V$, the $K$ shortest loop-less path problem consists of computing $K$ loop-less paths $p_1, \ldots, p_K$ from $s$ to $t$ in $G(V, A, c)$, by non-decreasing order of total cost, that is, such that $c(p_1) \leq \ldots \leq c(p_K)$ and $c(p_K) \leq c(p)$, for any $p \in \mathcal{P}_{st} \setminus \{p_1, \ldots, p_K\}$. In case that $|\mathcal{P}_{st}| < K$, it is sufficient to compute $\mathcal{P}_{st}$.

Efficient algorithms proposed in the literature for solving the $K$ shortest, loop-less path problems are so-called deviation algorithms and can be grouped into two classes: Algorithms based on the paper (Yen, 1971) for general (and thus also directed) graphs and the approach in Katho et al. (1982) for undirected graphs. The basic idea of such deviation algorithms is to maintain a list $A$ which stores the so far computed shortest $k-1$ loop-less paths and additionally, serves for computing a candidate list $B$ of new paths by deviating from the already computed paths $A^1, \ldots, A^{k-1}$ in $A$. Once new candidates are computed by the deviation procedure, the best among them is selected and added to $A$. The algorithms presented in the comparison paper of Pascoal (2006) differ by the kind of deviation method used and by the algorithms used for computing these deviations.

Although many implementations of Yen’s algorithm can be done, we implemented Yen’s original version of the algorithm described in Appendix A and we refer to Pascoal (2006) for a comparison of even more efficient implementations (such as Martins and Pascoal, 2003; Perko, 1986). The complexity analysis of Yen’s original algorithm (included in Appendix A) shows that the time complexity is $O(K(n(m \log n) + \log(K)))$ and the space complexity is $O(n^2 + Kn)$ for a (sparse and connected) Fahrnetz graph with $n$ vertices and $m$ edges. Pascoal (2006) provides extensive computational results for different graph classes, graph sizes and different choices of $K$.

The Fahrnetz-graph of the central railway station area Berne, which we consider in our case study (see Chapter 5) contains $n \approx 4'000$ vertices and $m \approx 480'000$ arcs. The weights of the vertices were provided by experts. To compute $K = 100$ shortest paths between two vertices requires approximately one second. Although this might be considered fast enough for the purpose of dispatching trains in an online closed-loop control framework, we propose to compute travel paths of planned train movements in advance, i.e. during preprocessing. With the help of parallel computing we could compute the 175 shortest (loop-less) travel paths for each of the 1600 planned train movements in less than 10 minutes (the computational environment is described in Section 5.4).

2.5 Train speed profiles

In order to compute a forecast of train movements (recall the framework steps II and IIIId in Figure 1.11), it is necessary to know at which speeds the train movements follow their travel paths. Thus, for getting a forecast so called speed profiles have to be computed for each train movement along their travel paths.

Definition 2.5 (Speed profile). A speed profile of a given train movement is the time-distance curve along a given travel path. Numerical computations of speed profile curves often discretise along the travel path, thus a speed profile is usually represented by a time-distance table.
Chapter 2: Modelling the railway system: topology and train movements

For a given train movement and travel path infinitely many speed profiles are in principle conceivable, but during the computation of a forecast a single speed profile can be selected by considering optimization criteria, such as travel time or energy consumption. For the computation of speed profiles one must first understand and model correctly longitudinal train dynamics. The longitudinal movement of a train is a complex dynamic system since several factors influence the longitudinal dynamics of a train:

1. tractive and braking effort
2. track grade and curvature
3. propulsion resistance (rolling resistance and aerodynamic drag)
4. slack action (relative movement between the wagons)

Furthermore, these factors themselves depend on the characteristics of the rolling stock formation, the bogies, the engines, the brakes, etc. Therefore, railway vehicle dynamics is a broad and complex topic in railway engineering, which we cannot deeply discuss here. The interested reader is referred to Hay (1982) and to Iwnicki (2006) for a deeper understanding of railway vehicle dynamics. However, we formulate in Appendix B the problem of computing speed profiles by an optimal control model and illustrate how this model can be solved. Figure 2.8 shows computed energy-efficient speed profiles for a commuter train composition and for four different target trip times. The upper graph contains the four computed speed-distance curves and the maximum allowed speed curve (in red). In the lower graph the gradient of the track profile is shown.

Figure 2.8: Several energy-efficient speed profiles for different trip times computed by applying dynamic programming (Appendix B) during a supervised master thesis (Balmelli, 2009).
2.6 Blocking time stairways

Section 2.2.1 describes how the railway infrastructure is partitioned into fixed blocks and how train movements over these blocks are safeguarded by the signalling system. This form of safe train separation shall now be described more formally in terms of blocking time stairways. For this purpose we first define fixed blocks (infrastructure resource) and the concept of a blocking time as follows:

**Definition 2.6** (Infrastructure resource or fixed block \( r \)). An infrastructure resource is a fixed block, which is safeguarded by the signalling and interlocking system to prevent its concurrent use by more than one train movement at any point in time. The infrastructure resources in a dispatching area are indexed by \( r \in R \).

**Definition 2.7** (Blocking time). The blocking time is a time interval during which an infrastructure resource is operationally allocated exclusively to exactly one train movement, thus blocking it during that time for other train movements. The blocking time is constituted by several time intervals (illustrated in Figure 2.9), namely the

- **Route forming time**: Time required to clear the (entrance) main signal (lights to green) and in case of junctions or crossings, to set switches into the correct position for the route.

- **Signal viewing time**: Time required by the train driver to view the status of the distant signal (or, in case there is no explicit distant signal, of the previous main signal).

- **Approach time**: Travel time required for a train-head to get from the distant signal to the (entrance) main signal.

- **Block passing time**: Travel time required for a train-head to get from the (entrance) main signal to the (exit) main signal.

- **Clearing time**: Travel time required for a train with the head in front of the (exit) main signal until the last (rear) axle of the train passed the next clearing point.

- **Route releasing time**: Time required to reset the signals and eventual switches of junctions or crossings back to its standard configuration.

Based on Graffagnino (2007) and in discussions with SBB we made the following assumptions for the computation of blocking times in our case study (see Chapter 5):

- **Route forming time**: Due to the lack of available detailed timings, it is suggested to assume an universally valid threshold value of 7 seconds.

- **Signal viewing time**: The train driver is assumed to see and react after at most 5 seconds.

- **Clearing time**: As the location of clearing points have not yet been digitized, clearing points were assumed to be located 500 meters after each signal or switch.
Figure 2.9: A block section is exclusively allocated by the operation of a train movement over this section for a time interval, the *blocking time* (This figure is based on illustrations in Chp. 3 (Pachl) of the [Signal and Interlocking Compendium](#)).

- **Route releasing time**: Similar to the route forming time, we assume 7 seconds for the release of a route and add an additional 12 seconds as a general safety time buffer, resulting in a total of 19 seconds.

During the computation of the forecast the travel path and the speed profile of each train movement is determined and therefore, it is also possible to compute for every consecutive fixed block along a travel path the corresponding blocking time, thereby forming a stairway of consecutive blocking times (see Figure 2.10). The mathematical optimization model of Chapter 3 is built upon conflict-free blocking time stairways:

**Definition 2.8** (Blocking and conflict-free blockings). A *blocking of a critical railway infrastructure resource*, consists of

- the infrastructure resource $r \in R$ and
- the blocking time (interval) $\tau$ during which the critical infrastructure resource $r$ is blocked.

Furthermore, two blockings $(r_1, \tau_1)$ and $(r_2, \tau_2)$ are conflict-free, if one of the following holds:

- the blocked infrastructure resources are different: $r_1 \neq r_2$
- the blocking time intervals do not overlap: $\tau_1 \cap \tau_2 = \emptyset$

The above definition allows us to introduce the conflict-free blocking time stairways:
**Definition 2.9** (Blocking time stairway and conflict-free blocking time stairways). A blocking time stairway $b$ is given by a finite sequence of $|b| > 0$ blockings 

$$b = \{(r_1, \tau_1), (r_1, \tau_1), \ldots, (r_{|b|}, \tau_{|b|})\}$$ 

Furthermore, two blocking time stairways are conflict-free, if all of their pairs of blockings are conflict-free.

Figure 2.10: A blocking time stairway arises from consecutive blocking times $(\tau_1, \ldots, \tau_{|b|})$ over fixed blocks $(r_1, \ldots, r_{|b|})$.

### 2.7 Forecast of train movements

In the previous sections we described how for each train movement we can compute several travel paths, speed profiles and blocking time stairways. These computations contribute to the generation of a forecast of train movements. For dispatching trains in a central railway station area, the forecast shall be limited to the train movements $T$ of that central railway station area:

**Assumption 2.2** (Central railway station area and train movements $T$). A central railway station area is represented by a Fahrrnetz-graph, a railway topology graph and an (index) set $P$ for the platforms at the central station. Additionally, $V_p \subset V$ specifies which routes in the Fahrrnetz-graph start or end at a platform $p \in P$ of the central station. The forecast is generated only for the train movements $T$ which have travel paths starting or ending at a platform $p \in P$ and additionally, take place in the near future (referred to as dispatching time horizon in Chapter 4).
When all the train movements $T$ for a forecast are known, their travel paths, speed profiles and blocking time stairways have to be computed. This result in the forecast:

**Definition 2.10 (Forecast).** A forecast $(T, B)$ predicts for each train movement $t \in T$ inside a given dispatching time horizon and a given central railway station area, a set of possible realizations in form of alternative blocking time stairways $B_t \subseteq B$. The alternative blocking time stairways arise from alternative travel paths, which the train movements can follow in the future.

It is the task of static train dispatching (see Chapter 3) to decide, based on a forecast, which alternative blocking time stairway each train movement should follow.
Chapter 3

Static train dispatching

In chapter 2 we described how a forecast of the train movements can be modelled in terms of alternative blocking time stairways. In the model predictive control framework (recall Chapter 1, Figure 1.11) we assume that such a (in this chapter assumed to be static) forecast is provided by the operational layer. Such a forecast is the point of departure for this chapter: Given a forecast, the central task of static train dispatching is to assign conflict-free blocking time stairways to train movements. Here we suggest a model, which approaches this task in the form of a constrained assignment problem: A solution to the problem assigns a (unique) blocking time stairway to each train movement, such that the assigned blocking time stairways are (pairwise) conflict-free and other operational constraints such as connections are fulfilled.

In this chapter, at the start an overview on related work to train dispatching is given, followed by the problem statement for static train dispatching. A mathematical optimization model for static train dispatching is then introduced and computational results show the model’s viability for practical application. Chapter 4 will then extend the subsequently developed model for static train dispatching, by including the dynamic context of the control loop framework. This chapter is an extension of what we published in the journal *Transportation Science* (Caimi et al., 2010) and most of this chapter has been published in the journal *Computers and Operations Research* (Caimi et al., 2012).

3.1 Related work

Subsequently we describe different computer-supported approaches for conflict-free train dispatching which have been made over the recent years. We describe first approaches, which focus on finding optimal train sequences at specific points of a dispatching area. The second part then summarizes approaches, which additionally consider re-routing of trains as a possible dispatching action and finally, we consider research on complete frameworks for train dispatching.
3.1.1 Re-ordering train movements at fixed points

The following approaches look for optimal orders of train movements at fixed points in a dispatching area. Note that the travel paths of the train movements are usually assumed to be fixed in these approaches. These approaches are also not concerned with exploiting the flexibility found in train speed profiles, because they assume a limited control of train movements by means of signals.

Sahin (1999) models a single track line as a set of stations with infinity capacity. Occurring conflicts at sections between two train movements and two stations are solved iteratively. The conflict resolution decides which train movement has to wait after evaluating the expected delay and a rough estimation of knock-on conflicts.

Ping et al. (2001) consider a double track line and the optimal departure sequences of train movements at the stations which allow trains to overtake each other. In a genetic algorithm the list of departure sequences is then used as a genetic representation of the solution domain and within, the sum of delays serves as a fitness function.

Similar to Ping et al., Takagi et al. (2006) apply a genetic algorithm for deciding on the order of trains but limit their considerations to a single junction. The order of the arriving trains at the junction is assumed to be fixed and the algorithm only decides on the order of departing trains at the junction. Ho et al. (1997) study the same problem, but use a dynamic programming approach and compare them with approaches based on genetic algorithms, tabu search and simulated annealing in Ho and Yeung (2001).

Wegele (2005) extends the considered dispatching area and studies the optimization of train orders at several crossovers in a part of a railway network. The approach determines first at each crossover the locally optimal train order, then coordinates the local optimal orders by means of a branch-and-bound procedure into a global feasible solution. Finally a genetic algorithm intends to improve upon this initial solution. Similarly Chou et al. (2009) decomposes the dispatching area into smaller parts but uses an agent-based approach for the coordination of train movements. In this approach the order of the train movements which enter a part is provided by the surrounding agents (and thereby fixed), but an agent is allowed to change the order of trains which are leaving the part under his supervision. The coordination is done iteratively, where first all agents determine a local solution which is then provided to their neighbouring agents. In the subsequent iteration, each agent determines again a local solution based on the new entry sequences of train movements. Once no more changes in entry sequences occur between two iterations, the iterative process stops and a global feasible solution is found.

Jacobs (2003) approaches the train ordering problem differently with a priority based greedy approach. After the train movements have been sorted according to their priority, they are iteratively assigned a travel path and blocking time stairway according to their priority order. Occurring conflicts are solved in a forward fashion by a locally greedy method.

Note that we omit describing research, which focuses on delay management, that is in a way related to train re-ordering. This research is usually concerned with passenger flows, but in turn prescinds from a detailed model of the train movements and railway topology.
3.1.2 Combining re-ordering with re-routing

Törnquist and Persson (2005) formulate a MIP in which train movements are modelled as a fixed sequence of events (thus travel path changes are not possible). Each event represents the occupation of a fixed block by the corresponding train movement. Binary decision variables in the MIP determine the order of train movements over fixed blocks, which are used by more than one train. The authors try to solve the MIP in two optimization layers. In the master layer a tabu search or simulated annealing algorithm assigns values to the binary decisions variables. Once the binary decision variables are fixed the slave layer is left with solving an LP. In Törnquist and Persson (2007) the authors extend the approach by modelling a platform choice for each train movement, which can be seen as a rudimentary form of re-routing. As a consequence of adding the platform choice, the extended MIP formulation was too big to be solved efficiently.

Flamini (2005); D’Ariano (2008); Corman (2010) identify the train dispatching problem as a job shop scheduling problem with no-wait and blocking constraints (see Mascis and Pacciarelli 2002). These problems can be modelled by so-called alternative graph models and are usually solved with disjunctive programming, which fixes the trains’ ordering. In a post-processing step the speed profiles have to be coordinated in order to obey the trains’ ordering. Although in alternative graphs the routing of the trains are fixed, local re-routing can be included through neighbourhood search, as explained in D’Ariano et al. (2008). This approach has the advantage of allowing variable speed profiles, but lacks the possibility of efficiently considering many alternative routes in dense areas. The approach was implemented in a (laboratory) real-time traffic management system called Railway traffic Optimization by Means of Alternative graphs (ROMA). Flamini and Pacciarelli (2007) applied this approach to the management of a metro rail terminus. The solution methods were gradually improved (D’Ariano et al. 2007; Corman et al. 2010a) to speed up the computation times and achieve a viable approach for on-line train disposition. Furthermore, an initial attempt to broaden the geographical scope of this approach was made by Corman et al. (2010c) by coordinating two geographically neighbouring, complex dispatching areas in the Netherlands. Each dispatching area is controlled by a single dispatcher with the support of a local ROMA system. A coordination level is introduced in order to manage the interaction among the two local ROMA systems.

Rodriguez (2007a) consider the re-routing of trains more explicitly in a constraint programming approach, in which a travel path of a train movement is modelled as a sequence of resources, i.e. fixed blocks. The conceivable travel paths of each train movement are modelled by a constraint, which is violated by an infeasible sequence of resources. Additional continuous variables determine the entry and exit times of the train movements at their corresponding resources. Synchronization constraints guarantee that the exit time at a resource is the equal to the entry time at the next resource along the travel path of a train movement. Capacity constraints guarantee a conflict-free occupation of the infrastructure resources. The authors solve the problem with constraint propagation and the branch-and-bound method, in which the travel paths are determined first, followed by an optimization over the train orders, the entry and finally the exit times. In Rodriguez (2007b) the author modified their approach, in order to improve the computational running time for dispatching areas which include single line tracks. In Rodriguez (2008) modify the search criteria in the branch-and-bound search tree.
Weymann (2011b) used a similar approach based on mixed integer programming. Decision variables determine the ordering of the trains and continuous variables provide variable speed profiles. A characteristic difference to the alternative graph approach is that additional decision variables determine the travel path for each train. As a travel path is modelled by a sequence of routes, the decision variables fix this sequence. Infeasible travel paths (or sequences of routes) are prevented by additional constraints. Computational results look promising for rescheduling trains along corridors. A potential drawback when applied to complex central railway station areas could be the growing amount of required ordering variables (Weymann, 2011a).

3.1.3 Other frameworks for train dispatching

Cui (2010) proposes a framework which is somewhat similar to the closed-loop control framework introduced in Chapter 1. The framework forecasts train movements from the current operational situation by a detailed synchronous simulation. The author suggests applying optimization routine based on the forecasts, which shall decide on the order of the train movements. The suggested routine uses adapted FIFO-principles and a mechanism for preventing deadlocks, but was not yet implemented.

Mazzarello and Ottaviani (2007) introduce an architecture for a traffic management system in large railway networks which was developed under two subsequent EU projects, called COMBINE and COMBINE 2. The architecture consists of two core modules, the conflict detection and resolution which is based on the alternative graph model (disjunctive programming) and a speed profile generator. The architecture was successfully tested in a laboratory setting for the Dutch railway bottleneck Schipol and the timetable of 2007, thereby promoting its further development.

Mannino and Mascis (2009) developed (to the best of our knowledge) the first real-time automated control system for operating trains in metro stations. Similar to ROMA the approach is based on disjunctive programming, which exploits the fact that metro stations are of relatively small size, and thus the resulting disjunctive MILP contain few binary variables. The authors’ approach manages to identify a suitable routing and establishes an optimum schedule of the performed operations. The solution routine is based on the branch-and-bound method which optimizes a convex cost function and relies on an effective lower bound. The system was put into operation in the Milan metro in July 2007.

Recent research considered the integration of the above described and rather academic approaches into already existing processes in railway operations. Lüthi (2009) and Kuckelberg (2011) analyse how computer supported train disposition concepts could be integrated into daily operations of railway.

3.2 Problem statement

Chapter 2 described how future train movements can be predicted in terms of alternative blocking time stairways by the operational layer. The control layer of the closed-loop discrete-time control framework (see Figure 1.10) has to analyse this forecast in order
to find conflict-free disposition schedules. We formulate this task conceptually in the following problem statement:

**Problem 3.1 (Static train dispatching problem).** Given a (static, one-time) forecast \((T, B)\) of train movements in a central railway station area, a timetable and connections, the static train dispatching problem consists of finding a conflict-free disposition schedule for each train movement occurring in the forecast. Furthermore, the disposition schedule has to respect operational constraints (e.g. proper platform usage) and should be optimal with regard to quality criteria, e.g. punctuality or connectivity.

Several terms in this problem statement require clarification and additional assumptions:

**Definition 3.1 (Timetable).** If intended (i.e. published) by the commercial offer, the timetable specifies arrival or departure times \((\hat{\alpha}_t, \hat{\delta}_t)\) for train movements at the central station.

**Definition 3.2 (Connections \((C, W)\)).** Some connections \(C \subseteq T \times T\) between train movements at the central station may be part of the commercial offer. Each maintained connection has its own specific benefit. It is assumed that these benefits are indicated by positive values in the connection benefit matrix \(W \in \mathbb{R}^{\left|T\right| \times \left|T\right|}_+\).

**Definition 3.3 (Conflict-free disposition schedule).** Given a forecast \((T, B)\), a conflict-free disposition schedule uniquely assigns to each train movement \(t \in T\) a blocking time stairway \(b \in B_t\), such that every pair of assigned blocking stairways is conflict-free.

**Assumption 3.1 (Operational constraints).** There are many operational constraints which could be considered when dispatching trains in a central station area. In our subsequent optimization model we focus on the following important operational constraints, which are related to proper rolling stock and platform usage:

- **Unique platform occupation:** A platform can be occupied by at most one train.

- **Rolling stock share:** If an arrival train movement and a departure train movement share the same rolling stock, they shall use the same platform and the arrival event and departure event have to be in proper order.

**Assumption 3.2.** [Quality criteria] It is often not clear what the quality of dispatching is and even more so, how it could possibly be measured. This is an inherent problem of the many different participants in a socio-technical system, as each participant has its own criteria. Nevertheless, all participants of the railway system can probably agree that punctuality is a key quality criterion. Besides punctuality, we included in our considerations also the connectivity, as bad connectivity can heavily affect the passenger experienced punctuality. Note that we measure these criteria on a per train level and not for each single passenger, as data on passenger level was not available to us.
3.3 Optimization model

We formulate the task of finding a conflict-free disposition schedule for the trains as a con-
strained combinatorial assignment problem. Among (finitely) many alternative blocking
time stairways exactly one shall be assigned to each train. Additionally, this assignment
has to respect safety restrictions and other operational constraints. The constrained com-
binatorial assignment problem is modelled as a binary linear program. A brief overview
on the utilized notation will facilitate the following elaboration of the binary linear pro-
gram.

3.3.1 Notation

In the following notation a *train* is represented by two train movements:

- The arrival train movement is the train movement from the boundary (or depot) to
  the central station.

- The departure train movement is the train movement from the central station to the
  boundary (or depot).

We explain later in section 3.3.3 why this splitting into two train movements is helpful for
the modelling process.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t \in T$</td>
<td>Set of trains $T$ indexed by $t$.</td>
</tr>
<tr>
<td>$p \in P$</td>
<td>Set of platforms $P$ of the central railway station indexed by $p$.</td>
</tr>
<tr>
<td>$A_{t,p}$</td>
<td>Set of arrival blocking time stairways for train $t$ which end at platform $p$.</td>
</tr>
<tr>
<td>$A_{.,p} = \bigcup_{t \in T} A_{t,p}$</td>
<td>Set of arrival blocking time stairways of all trains in $T$ which end at platform $p$.</td>
</tr>
<tr>
<td>$A_t = A_{t,.} = \bigcup_{p \in P} A_{t,p}$</td>
<td>Set of arrival blocking time stairways of train $t$ which end at any platform in $P$.</td>
</tr>
<tr>
<td>$A = \bigcup_{t \in T} \bigcup_{p \in P} A_{t,p}$</td>
<td>Set of arrival blocking time stairways of all trains in $T$ which end at any platform in $P$.</td>
</tr>
<tr>
<td>$D_{t,p}$</td>
<td>Set of departure blocking time stairways for train $t$ which start at platform $p$.</td>
</tr>
<tr>
<td>$D_{.,p} = \bigcup_{t \in T} D_{t,p}$</td>
<td>Set of departure blocking time stairways of all trains in $T$ which start at platform $p$.</td>
</tr>
<tr>
<td>$D_t = D_{t,.} = \bigcup_{p \in P} D_{t,p}$</td>
<td>Set of departure blocking time stairways of train $t$ which start at any platform in $P$.</td>
</tr>
<tr>
<td>$D = \bigcup_{t \in T} \bigcup_{p \in P} D_{t,p}$</td>
<td>Set of departure blocking time stairways of all trains in $T$ which start at any platform in $P$.</td>
</tr>
<tr>
<td>$B_{t,p} = A_{t,p} \cup D_{t,p}$</td>
<td>Arrival and departure blocking time stairways of train $t$ which start or end at platform $p$.</td>
</tr>
<tr>
<td>$B_t = B_{t,.} = A_{t,.} \cup D_{t,.}$</td>
<td>Arrival and departure blocking time stairways of train $t$ which start or end at any platform in $P$.</td>
</tr>
</tbody>
</table>
Table 3.2: Notation for data created during pre-processing.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_p = A_p \cup D_p$</td>
<td>Arrival and departure blocking time stairways of any train in $T$ which start or end at platform $p$.</td>
</tr>
<tr>
<td>$B = A \cup D$</td>
<td>Arrival and departure blocking time stairways form the set of all blocking time stairways $B$.</td>
</tr>
<tr>
<td>$C \subseteq T \times T$</td>
<td>Intended connections between trains.</td>
</tr>
<tr>
<td>$P_t = { p \in P \mid (A_{t,p} \neq \emptyset) \land (D_{t,p} \neq \emptyset) }$</td>
<td>Set of platforms usable for train $t$.</td>
</tr>
<tr>
<td>$r \in R$</td>
<td>Set of all resources $R$ (i.e. fixed blocks) are indexed by $r$.</td>
</tr>
<tr>
<td>$\alpha : A \rightarrow \mathbb{R}$</td>
<td>Maps an arrival blocking time stairway to the arrival time at the platform.</td>
</tr>
<tr>
<td>$\hat{\alpha}_t \in \mathbb{R}$</td>
<td>Published arrival time of train $t$.</td>
</tr>
<tr>
<td>$\delta : D \rightarrow \mathbb{R}$</td>
<td>Maps a departure blocking time stairway to the departure time at the platform.</td>
</tr>
<tr>
<td>$\hat{\delta}_t \in \mathbb{R}$</td>
<td>Published departure time of train $t$.</td>
</tr>
<tr>
<td>$\gamma : B \rightarrow \mathbb{R}$</td>
<td>Maps a blocking time stairway to the platform event time (either arrival or departure time).</td>
</tr>
<tr>
<td>$\gamma(b) = \begin{cases} \alpha(b) &amp; b \in A, \ \delta(b) &amp; b \in D. \end{cases}$</td>
<td></td>
</tr>
<tr>
<td>$\phi_A : \mathbb{R} \rightarrow \mathbb{R}$</td>
<td>Penalty function for arrival delays.</td>
</tr>
<tr>
<td>$\phi_D : \mathbb{R} \rightarrow \mathbb{R}$</td>
<td>Penalty function for departure delays.</td>
</tr>
<tr>
<td>$f : f_B : B \rightarrow \mathbb{R}$</td>
<td>(Multi-)Objective function used in optimization.</td>
</tr>
<tr>
<td>$f$</td>
<td>Objective function for assigned blocking time stairways contributing to $f$.</td>
</tr>
<tr>
<td>$w_B \in \mathbb{R}$</td>
<td>Coefficient of assigned blocking time stairways objective.</td>
</tr>
<tr>
<td>$f_A : B \rightarrow \mathbb{R}$</td>
<td>Objective function for arrival delay contributing to $f$.</td>
</tr>
<tr>
<td>$w_A \in \mathbb{R}$</td>
<td>Coefficient of arrival delay objective.</td>
</tr>
<tr>
<td>$f_D : B \rightarrow \mathbb{R}$</td>
<td>Objective function for departure delay contributing to $f$.</td>
</tr>
<tr>
<td>$w_D \in \mathbb{R}$</td>
<td>Coefficient of departure delay objective.</td>
</tr>
<tr>
<td>$f_C : B \rightarrow \mathbb{R}$</td>
<td>Objective function for connections contributing to $f$.</td>
</tr>
<tr>
<td>$w_C \in \mathbb{R}$</td>
<td>Coefficient of connection objective.</td>
</tr>
<tr>
<td>$W \in \mathbb{R}^{</td>
<td>T</td>
</tr>
</tbody>
</table>

Table 3.2: Notation for data created during pre-processing.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{r,B} \subseteq \mathbb{R}^2 \times B$</td>
<td>Blocking schema arising from blockings of resource $r$ by blocking time stairways in $B$.</td>
</tr>
<tr>
<td>$Q \in Q_r, Q_r \subseteq P^B$</td>
<td>Set of all maximal conflict cliques $Q_r$ is a subset of the power set of all blocking time stairways $B$. $Q$ is used to iterate over $Q_r$.</td>
</tr>
<tr>
<td>$(U, V) \in \Omega_{p, p'}^{t, t'}$</td>
<td>$\Omega_{p, p'}^{t, t'}$ is the set of all maximal incompatible pairs of sets of blocking time stairways for two train movements $t$ and $t'$ using platforms $p$ and $p'$. $(U, V)$ is used for iterating over $\Omega_{p, p'}^{t, t'}$.</td>
</tr>
</tbody>
</table>
3.3.2 Decision and indicator variables

The task of the optimization model is to find a disposition schedule, thus in principle each train movement is required to be assigned with a blocking time stairway. In mathematical optimization problems this task is usually modelled with decision variables. Here, we introduce for every blocking time stairway \( b \in B \) in the forecast \((T, B)\) a binary (decision) variable \( x_b \in \{0, 1\} \) which specifies whether the associated blocking time stairway is part of a disposition schedule or not. A disposition schedule is thereby represented by an assigned binary vector \( x \). Additional (indicator) variables \( c_{t,t'} \in \{0, 1\} \) indicate for each intended connection (denoted by the set \( C \subseteq T \times T \)), whether a disposition schedule \( x \) maintains that connection from train \( t \) to train \( t' \).

The static train dispatching problem (Problem 3.1) includes both safety-related and operational considerations, which restrict the choice to admissible disposition schedules. These restrictions are subsequently included into the mathematical model in form of either equalities or inequalities.

3.3.3 Unique assignment of blocking time stairways to trains

Every train is assumed to have an arrival and a departure train movement. The arrival train movement of a train \( t \) can be realized by exactly one of the predicted arrival blocking time stairways (the set \( A_t \)), and similarly, the departure train movement can be realized by exactly one of the predicted departure blocking time stairways (the set \( D_t \)). Theses two unique choices for each train are modelled with the following two type of equations:

\[
\sum_{b \in A_t} x_b = 1, \quad \forall t \in T
\]

\[
\sum_{b \in D_t} x_b = 1, \quad \forall t \in T.
\]

We will see later (Section 4.3.2), that it can occur that none of the predicted blocking time stairways respect all the other problem limitations. And thus, the above equations can not be fulfilled, i.e. the mathematical model would become infeasible. In the later context of dynamic train dispatching (Chapter 4) and in order to deal with infeasibility (Section 4.3.2) it is convenient to relax these constraints such that

\[
\sum_{b \in A_t} x_b \leq 1, \quad \forall t \in T, \quad \text{(3.1)}
\]

\[
\sum_{b \in D_t} x_b \leq 1, \quad \forall t \in T. \quad \text{(3.2)}
\]

Note that there are two reasons for the distinction between arrival and departure blocking time stairways:

---

| \( x_b \in \{0, 1\}, \forall b \in B \) | Equal to 1 if blocking time stairway \( b \in B \) is assigned, 0 otherwise. |
| \( c_{t,t'} \in \{0, 1\}, \forall (t, t') \in C \) | Equal to 1 if train \( t \) has a connection to train \( t' \), 0 if connection is broken. |
1. The distinction significantly decreases the necessary amount of binary indicator variables which are required to represent a disposition schedule (i.e. the amount of $x_b$ variables): If only one blocking time stairway would cover both the arrival and departure train movement of each train, the necessary amount of indicator variables would be in $\Theta(|A| \cdot |D|)$, since all combinations of arrival blocking time stairways with departure blocking time stairways would have to be considered explicitly. By splitting the train movement in arrival and departure train movements, only $|A| + |D|$ indicator variables are required, since any combination of arrival and departure blocking time stairway can still be implicitly expressed by an indicator variable for the arrival blocking stairway and an indicator variable for the departure blocking stairway. But this modelling approach leads also to a disadvantage, namely that the combinations of arrivals and departures have to be coordinated (see Eq. 3.5 and Ineq. 3.6), still this disadvantage is small compared with the benefit of the reduced number of the overall required decision variables.

2. With this distinction, relations between trains, such as connections between trains, can fairly easily be included as constraints in the binary linear program (Ineq. 3.4).

### 3.3.4 Train separation: Unique (conflict-free) blocking of resources

This part of the model will guarantee that the train movements are safely separated when following the assigned blocking time stairways. Section 2.6 explains how proper train separation is achieved by preventing the simultaneous blocking of a critical infrastructure block (resource) by more than one train with the help of blocking time stairways. To model corresponding constraints efficiently, we build a so called blocking schema $S_{r,B}$ for each resource $r \in R$ and for the considered blocking time stairways $B$ (in Caimi et al. (2010) we referred to it as allocation schema).

**Definition 3.4 (Blocking schema).** A blocking schema $S_{r,B}$ of a resource $r$ for a set of blocking time stairways $B$ is the family of blocking time intervals which belong to a blocking of resource $r$ and to a blocking time stairway in $B$:

$$S_{r,B} = \bigcup_{b \in B} \bigcup_{1 \leq i \leq |b|} \tau_i.$$  

With the help of a blocking schema it is easy to identify pairs of blockings which are not conflict-free (see Definition 2.8). The conflicts between the pairs of blocking time intervals of a blocking schema induce an interval graph, which has one vertex for each blocking time interval in the blocking schema. Every edge between a pair of vertices in this interval graph corresponds to two intersecting blocking time intervals and represents a conflict between two blockings, which each belong to a blocking time stairway. A blocking schema and the induced interval graph is illustrated in Figure 3.1.
Chapter 3: Static train dispatching

Figure 3.1: A blocking schema (left side) induces an interval graph (right side). The numbers indicate different blocking time intervals which belong to blockings of different blocking time stairways. The overlapping intervals 0, 2, 4, 6 and 7 (indicated by the red crosses) on the left induce a maximal clique on the right (indicated by the red edges), a concept used for strengthening the mathematical formulation.

The simultaneous blocking of a resource by two different blocking time stairways \( b \neq b' \) can be prohibited by the corresponding constraint

\[ x_b + x_{b'} \leq 1, \quad \text{for } b' \text{ in conflict with } b. \]

These type of constraints have been considered before in the setting of train dispatching (Zwaneveld et al., 1996) and also arise in the integer programming formulation of node-packing problems (or stable set problems). In graph theory, a stable set is a set of vertices in a graph, no two of which are adjacent. The maximum stable set problem is one of the most studied problems in combinatorial optimization and is a well-known NP-hard problem (Schrijver, 2003) on general graphs (in this context often referred to as conflict graph). Zwaneveld et al. (1996) proposed to model train dispatching by a formulation close to the standard integer linear programming (ILP) formulation of maximum stable set problems. Unfortunately, this ILP formulation was hard to solve to optimality already for moderate conflict graph sizes because of its weak linear relaxation. This is due to the fact that conflict graphs are usually dense, and each edge requires a separate constraint in the ILP formulation. To overcome the problem, Zwaneveld et al. (2001) proposed a pre-processing for reducing the size of the conflict graphs based on graph-theoretical considerations, which helps to reduce the computation time. Nevertheless, there is currently no model that allows very large instances of such problems to be solved, i.e., when considering a large number of possible blocking time stairways for many trains simultaneously, to optimality quickly (in the order of seconds).

Padberg (1973) showed that the convex hull of integer solutions to a node-packing problem arises from facets of all the maximal cliques in the intersection graph associated

\footnote{Existing algorithms for solving integer linear programs - enumerative procedures such as branch-and-bound as well as cutting plane approaches - exhibit a strong dependency in their performance on the strength of their formulations (for a definition of strength and examples we refer to Bertsimas and Weismantel (2005)). To strengthen an ILP formulation, one is usually interested in facet-defining linear inequalities of the underlying polyhedron.}
with the problem. In intersection graphs, the number of maximal cliques can grow exponentially in the size of the graph (see Tucker [1980] for such an example in circular-arc graphs). However, in static train dispatching the conflict graph which arises from conflicting blocking time stairways is composed of all interval graphs which are induced by the blocking schemas. In an interval graph with \( k \) intervals (here induced by a blocking schema) all maximal cliques can be computed in \( O(k \log(k)) \) steps by first sorting the intervals according to time followed by a sweep line algorithm to build all the maximal cliques (see Caimi et al., 2010, page 218). Figure 3.1 illustrates how a maximal clique arises in the induced interval graph of a blocking schema based on in the blocking schema overlapping blocking time intervals. All the maximal cliques (denoted \( Q_r \) for a resource \( r \)) are then used to formulate the much stronger constraints

\[
\sum_{b \in Q} x_b \leq 1, \quad \forall Q \in Q_r, r \in R.
\]

Furthermore, interval graphs are chordal and perfect graphs (Schrijver, 2003) and the stable set polytopes of perfect graphs are described completely by non-negativity and clique constraints (Chvátal, 1975).

Since the blocking time stairways exhibit a temporal interdependence between interval graphs induced by different blocking schemas, we are ultimately interested in stable sets which are simultaneously valid in all interval graphs. In fact, we look for a description of the stable set polytope in a graph which is composed of a finite number of interval graphs. Although the above formulation based on maximal cliques describes (together with non-negativity) each stable set polytope of every interval graph, it is to the best of our knowledge, still unclear, how strong the formulation (consisting of all the maximal cliques of each interval graph and non-negativity) is, when compared to the stable set polytope of the entire conflict graph. Nevertheless, we summarized in Appendix C some computational results from our paper Caimi et al. (2010, page 223-224) which show the significant computational benefits of the stronger, clique-based formulation for train separation in comparison to the simple pairwise conflict formulation. Furthermore, in Section 3.4 experimental results show that even with the addition of the subsequently described operational constraints the formulation is viable in praxis.

3.3.5 Connection between arriving train \( t \) and departing train \( t' \)

In complex central stations, train operators usually offer connections between some trains, which influence the decisions made in train disposition. We model connections similarly to the blocking time stairways by binary variables. These variables should not be seen as decision variables but rather as indicator variables, as their values are implicitly determined by the assignment of blocking stairways. A binary indicator variable \( c_{t,t'} \) models a connection between an arriving train \( t \) and a departing train \( t' \). \( c_{t,t'} = 1 \) if the connection is maintained by the assignment of the arrival blocking time stairway of the arriving train \( t \) and the departure blocking time stairway of the departing train \( t' \). Constraints ensure that \( c_{t,t'} \) is set to 0 if the connection is broken by a blocking time stairway assignment \( x \). To determine whether a connection is maintained or broken, the difference between the departure and the arrival trains has to be compared with the minimum connection time.
Chapter 3: Static train dispatching

The minimum connection time is usually dependent on what type of trains are used and at which platforms they stop, thus we assume that these minimum connection times are provided by the infrastructure operator in form of values \(m_{t,t'}^{p,p'}\). Consider for example an arrival blocking time stairway \(b \in A_{t,p}\) of train \(t\) stopping at platform \(p\) and a departure blocking time stairway \(b' \in D_{t',p'}\) of train \(t'\) stopping at platform \(p\) where the time between the arrival of train \(t\) and departure of train \(t'\) is too short in order to maintain the connection (Figure 3.2).

Figure 3.2: An example of a broken connection: for an assigned arrival (black) and departure (grey) blocking time stairway the time between the arrival and the departure of two trains is too short to maintain the connection.

We call such two pairs of blocking time stairways incompatible (see Definition 3.5) and the following constraint would ensure that \(c_{t,t'}\) is set to zero if those blocking time stairways are assigned by a disposition schedule \(x\):

\[
c_{t,t'} \leq 2 - x_b - x_{b'}.
\]

**Definition 3.5.** [Incompatible and compatible pair of blocking time stairways] An arrival blocking time stairway \(b \in A_{t,p}\) and a departure blocking time stairway \(b' \in D_{t',p'}\) are incompatible for the trains \(t\) and \(t'\) if

- a connection between train \(t\) and train \(t'\) exists,
- the minimum connection time \(m_{t,t'}^{p,p'} \in \mathbb{R}_+\) which is dependent on the used platforms is given and
- \(m_{t,t'}^{p,p'}\) is strictly larger than the difference between departure time \(\delta(b')\) of train \(t'\) and arrival time \(\alpha(b)\) of train \(t\).

Otherwise a pair of blocking time stairways is defined compatible.

Instead of including a constraint for every occurring incompatible pair of blocking time stairways, we strengthen the formulation of the binary linear problem: The constraints are aggregated for two trains \(t, t'\) over a platform choice \(p, p'\) through pairs of maximal incompatible (see Definition 3.6) sets of blocking time stairways \((U, V) \in \Omega_{p,p'}^{t,t'}\) as

\[
c_{t,t'} \leq 2 - \sum_{b \in U} x_b - \sum_{b' \in V} x_{b'}; \quad U \subseteq A_{t,p}, V \subseteq D_{t',p'}; \quad (U, V) \in \Omega_{p,p'}^{t,t'}; \quad p \in P_t, p' \in P_{t'}.
\]

(3.4)
Definition 3.6. [Maximal incompatible pair of sets of blocking time stairways \((U, V)\) for \((t, p, t', p')\)]
A pair of non-empty sets of blocking time stairways \((U, V)\) with \(U \subseteq A_{t,p}, V \subseteq D_{t',p'}\) is incompatible for \((t, p, t', p')\), if all pairs \(b, b'\) with \(b \in U, b' \in V\) are incompatible. Furthermore, \((U, V)\) is maximal incompatible for \((t, p, t', p')\) if for all \(b^* \in \bigcup_{p \in P_t} A_{t,p} \setminus U\) the pair \((U \cup b^*, V)\) contains at least one compatible pair of blocking time stairways and similarly, for all \(b'^* \in \bigcup_{p' \in P_{t'}} D_{t',p'} \setminus V\) the pair \((U, V \cup b'^*)\) contains at least one compatible pair of blocking time stairways. We denote with \(\Omega_{t,t'}^{l, l'}\) the set of all maximal incompatible pairs of sets of blocking time stairways \((U, V)\).

Notice that

- only constraints based on maximal incompatible pairs have to be included, since they dominate constraints which are based on non maximal incompatible pairs, and
- a connection can be forced to be maintained at all costs, by simply adding an enforcing constraint \(c_{t,t'} = 1\).

Computing \(\Omega_{t,t'}^{l, l'}\)

Algorithm 3.1 computes \(\Omega_{t,t'}^{l, l'}\) for a given \(t, t' \in T\) and \(p, p'\) with the time complexity \(O(k \log k)\) and the space complexity \(O(k)\) where \(k = |A_{t,p}| + |D_{t',p'}|\). The complexity analysis and the proof of correctness for the algorithm are given in Appendix D.

![Diagram](image-url)

Figure 3.3: Illustration on how the Algorithm 3.1 for computing \(\Omega_{p,p'}^{l,l'}\) operates.

The algorithm creates two sorted lists \(U = \{\gamma(u_1), \ldots, \gamma(u_{|A_{t,p}|})\}\) and \(V = \{\gamma(v_1), \ldots, \gamma(v_{|D_{t',p'}|})\}\). \(U\) assembles the arrival times from all the arrival blocking time stairways of train movement \(t\) which end at platform \(p\). \(V\) is assembled in a similar fashion for train movement \(t'\) and platform \(p'\) by considering the departure times. The lists are illustrated by the two arrays in Figure 3.3. Then the algorithm proceeds by creating pointers \(p\) (illustrated as arrows in the figure) from each element in \(V\) to the first incompatible element in \(U\). In the example depicted by the figure, we assume for simplicity a minimum connection time of zero and thus, the pointers basically look for the next bigger
number $\gamma(u(j))$. In case there is no bigger number (i.e. no incompatible blocking time stairway), the pointer is instead pointing to a blank attached to the end of the list $U$. Once the lists and the pointers are created, the algorithm walks over the elements in $V$. Each time the current pointer $p(v_i)$ points to a lower number (or earlier time) than the number appointed by the next consecutive pointer $p(v_{i+1})$ (in the figure these pointers $p(v_i)$ are depicted with bold arrows), the corresponding sets $(U, V)$ are generated based on the current position in $V$ and the current pointer $p(v_i)$ (as suggested by the ellipsoids). Elements in $V$ with pointers pointing to the blank are not considered by the algorithm, since there are no incompatible blocking stairways in $U$ for these elements. Finally, the last element in $V$ has to be considered separately, as the criteria of a change in the next consecutive pointer left can not be applied. Instead, the algorithm forms the sets $(U, V)$ only in the case that the last pointer does not point to the blank.

Algorithm 3.1 Computing $\Omega_{t,t'}^{p,p'}$

**Input:** $t, t', (t, t') \in C, p, A_{t,p, p'}, D_{t', p'}, m_{t,t'}^{l,l'}$

**Output:** $\Omega_{p,p'}^{l,l'}$

1: $\Omega_{p,p'}^{l,l'} = \emptyset$
2: Sort $A_{t,p}$ in a list $U$ according to increasing arrival times at platform $p$
3: Sort $D_{t', p'}$ in a list $V$ according to increasing departure times at platform $p'$
4: $p_{\text{prev}} = 1$
5: for $v = v_1, \ldots, v_{|V|}$ do
   6: Define $p(v) := \begin{cases} \arg\min_{i=p_{\text{prev}}, \ldots, |U|} (m_{t,t'}^{l,l'} > \gamma(v) - \gamma(u_i)), & \text{if existent} \\ |U| + 1, & \text{otherwise} \end{cases}$
   7: $p_{\text{prev}} = p(v)$
8: end for
9: for $i = 1, \ldots, |V| - 1$ do
   10: if $p(v_i) < p(v_{i+1})$ then
      11: $U = \bigcup_{p(v_i) \leq j \leq |U|} u_j$
      12: $V = \bigcup_{1 \leq j \leq i} v_j$
      13: $\Omega_{p,p'}^{l,l'} = \Omega_{p,p'}^{l,l'} \cup (U, V)$
   14: end if
15: end for
16: if $p(v_{|V|}) \neq |U| + 1$ then
   17: $U = \bigcup_{p(v_{|V|}) \leq j \leq |U|} u_j$
   18: $\Omega_{p,p'}^{l,l'} = \Omega_{p,p'}^{l,l'} \cup (U, V)$
19: end if
20: return $\Omega_{p,p'}^{l,l'}$
3.3.6 Proper platform usage at a central railway station

There are three modelling steps necessary to achieve a proper platform usage at a central railway station:

1. A train which arrives at a platform has to use the same platform for departure.
2. The arrival of a train has to occur before its departure (proper chronological order).
3. A platform can be used at most by only one train at any given time.

Each step is modelled using different type of constraints:

1. **Using the same platform for arrival and departure**
   The trains have to arrive and depart at the same platform, that means that the assigned arrival blocking time stairways and the assigned departure blocking time stairways have to use the same platform respectively. The following constraints are added to the formulation for every possible platform $p$:
   \[
   \sum_{b \in A_{t,p}} x_b - \sum_{b' \in D_{t,p}} x_{b'} = 0, \quad \forall p \in P_t, \forall t \in T
   \] (3.5)

   Since the decision variables are binary and with (3.1) and (3.2) there are only two ways that (3.5) can be fulfilled:
   - A train does not arrive and does not depart from a platform $p$. Both sums are zero and therefore the difference between the sums is zero as well.
   - A train (uniquely) arrives and (uniquely) departs from a platform $p$. Both sums are one and therefore the difference between the sums is zero.

2. **Arrival event before departure event**
   Based on the above first step, the train is indeed guaranteed to use the same platform for arrival and departure, but the correct chronological order of the arrival and departure event subject to the minimal technical dwell time is not yet forced. This can be done by enforcing a connection between the train to itself at the same platform based on the connection inequalities (3.4), where in the computation of maximal incompatible sets, the minimal dwell time is used instead of the minimal connection time:
   \[
   0 \leq 1 - \sum_{b \in U} x_b - \sum_{b' \in V} x_{b'}, U \subseteq A_{t,p}, V \subseteq D_{t,p}; \quad (U, V) \in \Omega_{p,p}^{I,I}; \quad \forall p \in P_t; \forall t \in T
   \] (3.6)

3. **Unique platform occupation**
   Even if the correct chronological order of the arrival and departure of a train $t$ at a platform $p$ is now modelled, it could be that another train arrives or departs from platform $p$ while train $t$ is already occupying that platform. To exclude the possibility of this situation, another set of constraints imposes that a platform can be occupied at most by one train at any given point in time. As the possible arrival
times of trains are restricted by the finiteness of the amount of arrival blocking time stairways, it is enough to consider only these arrival time points \( \alpha^* \in \{ \alpha(b) \mid b \in A_{,p} \} \).

In order to count the number of trains currently at a platform we formulate the difference between the number of past and current arrivals and the number of departed trains:

\[
\# \text{trains at platform} \ p \ \text{at time} \ \alpha^* = \sum_{\alpha(b) \leq \alpha^*} x_b - \sum_{\delta(b) < \alpha^*} x_b
\]

To impose an unique platform occupation at each platform \( p \) this difference has to be smaller or equal to 1 for any arrival time point \( \alpha^* \) and for any platform \( p \):

\[
\sum_{\alpha(b) \leq \alpha^*} x_b - \sum_{\delta(b) < \alpha^*} x_b \leq 1, \ \forall \alpha^* \in \{ \alpha(b) \mid b \in A_{,p} \}, \ \forall p \in P
\]

With (3.6) the number of trains at any platform is non-negative:

\[
0 \leq \sum_{\alpha(b) \leq \alpha^*} x_b - \sum_{\delta(b) < \alpha^*} x_b \leq 1, \ \forall \alpha^* \in \{ \alpha(b) \mid b \in A_{,p} \}, \ \forall p \in P \tag{3.7}
\]

The above introduced three type of constraints (3.5, 3.6, 3.7) combined with the previous constraints (3.1, 3.2) guarantee that the trains arrive and depart at the same platform and in the right chronological order (This is proofed in the Appendix E).

3.3.7 Bi-level multi-objective formulation

Following the Assumption 3.2 on quality criteria of the problem statement in Section 3.2 we focus on maximizing passenger satisfaction, measured by punctuality and reliability. For this purpose we consider three criteria which are later aggregated into one objective function \( f \) (Eq. 3.9) as a weighted sum as follows.

1. Schedule all trains (reliability): In principle the described model may not schedule (i.e. assign an arrival and departure blocking time stairway to) every train (Ineq 3.1 and 3.2 may not be binding). But in order for there to be an incentive to schedule all trains, the number of dispatched trains is made an optimization criterion. Each assigned blocking will contribute 1 to this objective criterion:

\[
f_B = \sum_{t \in T} \sum_{b \in B_t} x_b \tag{3.8}
\]

2. Passenger dissatisfaction caused by delays (punctuality): Each dispatched train \( t \) has an assigned arrival time and departure time. These times may deviate from the published arrival time \( \hat{\alpha}_t \) and departure time \( \hat{\delta}_t \) and thereby generate a delay. Customer dissatisfaction originating from a delay is different depending on the amount of delay and whether the delay occurred at an arrival or a departure. We model this
distortion by relating customer dissatisfaction to delay through penalty functions \( \phi_A \) for arrival delay and \( \phi_D \) for departure delay as

\[
f_A \equiv \sum_{t \in T} \sum_{a \in A_t} \phi_A(\alpha(a) - \hat{\alpha}_t) \cdot x_a,
\]

\[
f_D \equiv \sum_{t \in T} \sum_{d \in D_t} \phi_D(\delta(d) - \hat{\delta}_t) \cdot x_d.
\]

The definition of \( \phi_A \) and \( \phi_D \) are subject to discussion and could be determined by customer service experts. We used for our computational study an admittedly simple definition:

\[
\phi_A(z) = \begin{cases} 
\frac{z}{360} & z \leq 0 \\
\frac{-z}{180} & \text{otherwise}
\end{cases}
\]

\[
\phi_D(z) = \begin{cases} 
\infty & z < 0 \\
\frac{-z}{180} & \text{otherwise}
\end{cases}
\]

3. **Passenger satisfaction achieved by maintained connections (reliability):** In order to offer a reliable railway service to the customers, connections between trains should be maintained. The customer satisfaction achieved by maintained connections can vary and we assume they are estimated in a normalised non-negative weight matrix \( W \in \mathbb{R}_+^{|T| \times |T|} \) as part of the input data. The overall passenger satisfaction based on connections is then evaluated through weighted summation:

\[
f_C \equiv \sum_{t,t' \in T, t \neq t'} [W]_{t,t'} \cdot c_{t,t'}
\]

These three criteria are further combined by weighted summation into one hierarchical multi-objective function.

\[
f = w_B \cdot f_B + w_A \cdot f_A + w_D \cdot f_D + w_C \cdot f_C \tag{3.9}
\]

We assume that scheduling all trains is the most important criterion for passenger satisfaction. Thus, the corresponding objective coefficient \( w_B \) should be chosen large enough, such that this criterion dominates the other two objective criteria in any feasible solution. The choice of the coefficient parameters \( w_A, w_D \) and \( w_C \) determines the trade-off between passenger dissatisfaction caused by delays and passenger satisfaction achieved by maintained connections.

**Remark on the weighted sum approach and the choice of weights:** The weighted sum method is a commonly used approach when optimizing over multiple criteria. Furthermore, it is essentially subjective, in that a dispatcher or any expert needs to supply the weights. A choice of weights reflects how important the criteria are valued relative to each other in terms of passenger satisfaction and thus they could serve as model parameters which a dispatcher could use in order to tweak computed disposition schedules.
3.3.8 Summarized static train dispatching (STD) model

Summarizing all constraints and the objective leads to the following binary linear optimization problem, which we will refer to later as the static train dispatching (STD) model.

Maximize \( f = f_t + w_A \cdot f_A + w_D \cdot f_D + w_C \cdot f_C \)  

Subject to:

\[ \sum_{b \in A_t} x_b \leq 1, \quad \forall t \in T \]  
\[ \sum_{b \in D_t} x_b \leq 1, \quad \forall t \in T \]  
\[ \sum_{b \in Q} x_b \leq 1, \quad \forall Q \in Q_r, r \in R \]  
\[ c_{t,t'} + \sum_{b \in U} x_b + \sum_{b' \in V} x_{b'} \leq 2, \quad U \subseteq A_{t,p}, V \subseteq D_{t',p'}; \quad (U,V) \in \Omega_{p,p'}^{t,t'} \]  
\[ \sum_{b \in A_{t,p}} x_b - \sum_{b' \in D_{t,p}} x_{b'} = 0, \quad \forall p \in P_t, \forall t \in T \]  
\[ \sum_{b \in U} x_b + \sum_{b' \in V} x_{b'} \leq 1, \quad U \subseteq A_{t,p}, V \subseteq D_{t',p'}; \quad (U,V) \in \Omega_{p,p'}^{t,t'} \]  
\[ 0 \leq \sum_{\alpha(b) \leq \alpha^* b \in A_{t,p}} x_b - \sum_{\alpha(b) \leq \alpha^* b \in D_{t,p}} x_{b'}, \quad \forall \alpha^* \in \{ \alpha(b) \mid b \in A_{t,p} \}, \forall p \in P \]  
\[ x_b \in \{0, 1\}, \quad \forall b \in B \]  
\[ c_{t,t'} \in \{0, 1\}, \quad \forall (t,t') \in C \]

3.4 Computational experiments

The subsequent results are based on the case study, which Chapter 5 describes in detail and the same computational environment (listed in Section 5.4) was used. In a first part we describe key figures of an STD instance, followed by a discussion of the required computation time to solve STD instances of various problem sizes.

3.4.1 Key figures of an STD instance

The figures in Table 3.4 originate from a rather large STD instance, which is based on the case study Berne (see Chapter 5) and in which all train movements for the first 110 minutes of an operational day are included. In this instance each train movement has to be assigned to one of (in average) \( \approx 110 \) alternative blocking time stairways while at the same time respecting the roughly 11’000 train separation constraints, which are formed over \( \approx 550 \) infrastructure resources. In all of the experimental results we could observe
that the number of constraints concerning the proper use of platforms at the central station ((3.5), (3.6) and (3.7)) is rather small compared to the number of train separation constraints. The zeros in the number of connection constraints and the number of ‘arrival before departure’ constraints is explained by the setting of the case study: The predicted alternative blocking time stairways were based on a recorded historic operational timetable and thus, the predicted alternative blocking time stairways usually would maintain the intended connections and respect the temporal order of arrival and departure. The solver found an optimal solution to this instance in \( \approx 10 \) seconds. Subsequently, the interaction between problem size and computation time is discussed based on experimental results.

| Number of train movements \( |T| \) | 31 |
| Number of infrastructure resources \( |R| \) | 557 |
| Number of connections \( |C| \) | 6 |
| Number of blocking time stairways \( |B| \) | 34'237 |
| Number of unique assignment constraints (3.1) and (3.2) | 31 |
| Number of train separation constraints (3.3) | 11’306 |
| Average number of variables in a train separation constraint | 480 |
| Number of connection constraints (3.4) | 0 |
| Number of ‘same platform for arrival and departure’ constraints (3.5) | 52 |
| Number of ‘arrival before departure’ constraints (3.6) | 0 |
| Number of ‘unique platform occupation’ constraints (3.7) | 309 |

Table 3.4: Key figures of an STD problem instance of average size encountered during the case study.

3.4.2 Computation time

Section 3.3.4 on train separation constraints established that the STD problem is an NP-hard optimization problem through a reduction to node-packing or independent-set problems (see Schrijver (2003)). When solving such problems, super-polynomial time has to be expected, unless and what is unlikely, that the two complexity classes \( \text{P} \) and \( \text{NP} \) are equal. Nevertheless, many instances of NP problems which arise in practice can still be solved in reasonable time. The subsequent results illustrate that quite large instances of the STD optimization problem are solvable in reasonable time.

Figure 3.4 shows for each of approximately 18’500 STD problem instances of varying problem size the necessary computation time in order to solve it to optimality (0% integrality gap). The size of the instances are represented in the number of binary decision variables, that is the number of assignment variables for the blocking time stairways. Clearly, most of the instances with 15’000 or less variables are solved in less than 10 seconds. With bigger instances (up to 20’000 variables) the computation times start to increase, but they are about equal in size and computation time when compared to previous results on the older RTCG model (compare Tables C.2 and C.3 of Appendix C). As opposed to the RTCG model, the STD model is also concerned with the proper usage of platforms as well as the connections between trains. Thus one might suspect that the STD problem instances should be more difficult to solve for the ILP solver, which, judged by
these results, does not seem to be the case.

![ILP computation times](image)

Figure 3.4: The computation times for solving optimally the STD optimization problems using IBM CPLEX 12.3. For comparison three examples of the former RTCG model (see Appendix C) are included as well.

Figure 3.5 clusters the instances of the problem into bins according to the number of variables and for each bin, the average and standard deviation in computation time is computed. One can observe that with growing size of the problem instances not only the average computation time increases, but also the standard deviation. This is to be expected, since it is after all a NP-hard problem. Nevertheless, in our case study we dispatched trains for a whole day and the number of binary variables of the STD instances were mostly between 4000 and 12000 (illustrated in Figure 3.7). The very fast computation times and considerably small standard deviation for this range of problem size, encourages the integration of such an optimization model in the control framework for dynamic train dispatching. This step is discussed in the next chapter.
Average and stdev in computation times for different clustered problem sizes

Figure 3.5: The 18'500 STD problem instances are clustered according to their problem size into bins along the horizontal axis and along the vertical axis the average computation time and standard deviation is given for each bin.
Chapter 4

Towards dynamic train dispatching

Chapter 3 introduced an optimization model that allows the computation of a disposition schedule based on static forecast of train movements. This chapter integrates this model into a closed-loop discrete-time system in order to dispatch trains dynamically. This and the following chapter are an extension of what we published in the journal *Computers and Operations Research* (Caimi et al., 2012). The chapter starts with an introduction to general closed-loop discrete-time systems and describes how model predictive control is applied in our setting as a control concept in the closed-loop discrete-time framework for train dispatching. The last section discusses a safeguard mechanism which can be utilized in the case that the alternative blocking time stairways provided by a forecast are not sufficient for dispatching all train movements.

4.1 Closed-loop discrete-time control

Figure 4.1 depicts the standard layout of a closed-loop control system where the information path follows a closed-loop: The system input affects the system output, which is measured with sensors, compared to a reference and then processed by the controller; the result (a control signal) is used as system input and the control loop is closed.

![Figure 4.1: A closed-loop controller uses feedback to control states or outputs of a dynamical system.](image)
If the closed-loop controller operates in discrete-time steps the dynamic system has the form (Bertsekas, 2007)

\[ x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots, N - 1, \]

where

- \( k \) indexes discrete time
- \( x_k \) is the state of the system and summarizes past information that is relevant for future optimization
- \( u_k \) is the control to be selected at time \( k \)
- \( w_k \) is a random parameter representing disturbance or noise
- \( N \) is the horizon or number of times control is applied
- \( f_k \) is the (system) state transfer function

The system can additionally involve constraints, such as limited control (\( u_k \in U_k \)), restrictions on the feasible states (\( x_k \in X_k \)) or bounds on the disturbance \( w_k \in W \).

The above formulation of a closed-loop discrete-time system is very general and the possible realizations of the controller can vary heavily depending on the characteristics of the dynamic system. Nevertheless, a well-established concept for the control of such complex dynamic systems is Model Predictive Control (MPC). Subsequently MPC and its application as a controller in the closed-loop discrete-time framework for train dispatching is explained.

### 4.2 Model predictive control

The basic concept of MPC was introduced in the 1960s and with its success as an advanced method of process control in process industries since the 1980s, MPC has evolved into a well established control technique. Zeilinger (2011, p. 28) describes MPC accurately as follows:

MPC is an optimal control method, where the control action is obtained by solving a constrained finite horizon optimal control problem for a current state of the plant [or dynamic system] at each sampling time. The sequence of optimal control inputs is computed for a predicted evolution of the system model over a finite horizon. However, only the first element of the control sequence is applied and the state of the system is then measured again at the next sampling time. This so-called receding horizon strategy introduces feedback to the system, thereby allowing for compensation of potential modeling errors or disturbances acting on the system.

Figure 4.2 follows this description and shows the flow-chart of a typical MPC system. The MPC scheme is built around two core components:
4.3 The dynamic system of train dispatching

The above sections explained how the closed-loop discrete-time control system can be based on a MPC scheme. Subsequently the MPC scheme is applied to our setting, namely
the first phase of the closed-loop discrete-time framework (see Figure 1.11) for train dispatching. For this purpose, the state, the control action(s), the disturbances and the transfer function of the underlying dynamic system have to be described. Observing first the changes in time-distance graphs, which would occur in a closed-loop discrete time control system for train dispatching facilitate the later formal description of the dynamic system.

Figure 4.3 illustrates the different temporal scopes, which arise between two consecutive control iterations.

![Diagram of temporal scopes](image)

Figure 4.3: Different temporal scopes determine which and how blocking time stairways are considered at each control iteration.

The left part shows along the vertical axis the temporal scopes in the current control iteration $k = 0$. The controller computes a disposition schedule, which is depicted in the form of the assigned blocking time stairways in the time-distance graph on the left. When moving to the next control iteration, the temporal scopes are shifted by the time between two control iterations, denoted by $\text{dispatching frequency time interval } \Delta$. The already fixed blocking time stairways (depicted in black in the figure) of train movements, which start before the blocking time stairway fixing horizon, have to remain the same and there might be additional blocking time stairways, which are fixed during the next control iteration. The reason for fixing blocking time stairways is based on an assumption in our setting, namely that the controller has no influence on train movements, which have already started. This assumption has to be modified if the controller does actually have such possibilities (e.g. adaptive train control, see Section refsec:adaptiveControlAndSpeedProfileOptimization) at its disposal. The travel
4.3 The dynamic system of train dispatching

paths of train movements that start between the *blocking time stairway fixing horizon* and the *travel path fixing horizon* are fixed in the current control iteration and can not be changed by the controller in the next control iteration (in the figure the corresponding stairways are depicted in red). Although the disposition schedule also assigns blocking time stairways to train movements that start between the *travel path fixing horizon* and the dispatching time horizon, the controller can assign different blocking time stairways in the next control iteration without any restrictions. Finally, additional train movements might have to be considered in the next control iteration. Thus, three parameters have a great impact on the dynamics of the system:

**The dispatching frequency interval** $\Delta \in \mathbb{R}$ determines with which frequency the train dispatching problem shall be solved. The shorter this time interval, the more responsive the control-loop becomes to changes in the railway system. Note, that the choice of $\Delta$ also implicitly defines the *blocking time stairway fixing horizon*. It may become obsolete, if the framework is extended in a future step by a simulation environment of the physical system (see Section 6.4.1). This would be the case if the simulation environment could incorporate dispatching control actions, which influence trains that have already started their train movement.

**The travel path fixing horizon** arises from the safety systems of railway networks: To set a travel path (i.e. set switches and signals along its routes through the railway network) a minimum period of time is required before a train movement can start moving along its travel path. A train movement with a start time which is earlier than the travel path fixing horizon, can no longer change its travel path, and thus its travel path is fixed.

**The dispatching time horizon** $N \cdot \Delta$ is the receding finite time horizon of a control approach based on MPC. The controller limits its analysis of the system to train movements which start before the dispatching time horizon, and thus the dispatching time horizon has a direct influence on the complexity of the control task.

The above observations for our setting lead to the following form of a closed-loop discrete time system:

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \ldots, N - 1,$$

where

- $k$ indexes discrete time or the control iteration, respectively
- $x_k$ consists of all the previously fixed blocking time stairways and the forecast of train movements which start earlier than the dispatching time horizon
- $u_k$ consists of the assigned blocking time stairways at control iteration $k$
- $w_k$ consists of changes in the forecast of train movements, namely the entry time delays of train movements which will enter the dispatching area
- $N$ is the horizon or the number of times control is applied
Chapter 4: Towards dynamic train dispatching

\( f_k \) adds to the next state \( x_{k+1} \) newly fixed blocking time stairways as well as the forecast of train movements, which have to be considered as soon as the dispatching time horizon is shifted.

Additionally, the control space \( U_k \) of control iteration \( k \) is given by the predicted alternative blocking time stairways \( (B_T)_k \), which have to be consistent with previously fixed travel paths.

![Model Predictive Control Scheme applied in the context of dynamic train dispatching.](image)

The two flow-charts (Figures 4.4 and 4.5) illustrate this closed-loop discrete time system: Figure 4.4 describes how the MPC scheme is applied in the context of dynamic train dispatching. In this scheme the two core components are on one side the model of the railway system, which predicts the train movements over a finite time horizon in the form of alternative blocking stairways (forecast \( (T, B) \), see Definition 2.10) and on the other side, the static train dispatching model (STD, see Section 3.3.8) which computes the best disposition schedule.

The flow-chart in Figure 4.5 illustrates how the MPC state is updated by the state transfer function during a MPC iteration. The state transfer function determines the new state based on the current best disposition schedule, the current disturbance and the previous state.
4.3 The dynamic system of train dispatching

As this thesis focuses on the first phase of the closed-loop discrete-time framework (see Figure 1.11), neither the physical railway system nor the dispatcher are available and thus the MPC scheme cannot measure train movements nor can it apply the disposition schedule in form of control actions to the railway system. A few compromises had to be considered for the case study:

- A historical snapshot of train movements of a whole operating day serves as a reference timetable, which is used to build the forecast of train movements.

- Control is modelled implicitly by the state transfer function (by means of fixing blocking time stairways) and the assumption that given a fixed blocking time stairway the train movement follows the associated speed profile exactly.

- Disturbances have to be modelled, as measurements of a physical system are not available.

4.3.1 Modelling disturbances

The missing physical railway system precludes the possibility to measure disturbances in the network. Any computational study executed based on the closed-loop system, but which lacks the consideration of disturbances, impeaches the credibility of any insights gained from computed results. Lacking both a physical system and a simulation environment, our setting can not provide qualitative results on the stability and quality of the controller. Nevertheless, this thesis focuses on showcasing the computational tractability
of the approach and in order to achieve that, the controller still has to be exposed to some disturbances.

Many disturbances can occur in a complex central railway station area, but most of them are difficult to model when no railway simulation software is available. The disturbances considered in the case study are limited to delays, occurring only for train movements which enter the dispatching area, i.e. their entry time is delayed. Without these delays the train movements would enter the dispatching area according to the time of the operational timetable. It is also clear, that the actual delay only manifests itself at exactly the time when the corresponding train enters the dispatching area. Before the actual delay becomes actually known during a MPC iteration, the delay has to be be predicted in each previous MPC iteration. In the case study all the delays are generated in a preprocessing step before even starting the iterative MPC scheme (see Section 5.5 for details).

4.3.2 A remedy to infeasibility: Additional alternative blocking time stairways

Although the static train dispatching model (see Section 3.3.8) is deliberately formulated in such a way that its feasibility region is always non-empty (e.g. \( x_b = 0, \forall b \in B \) is always a solution), practical feasibility of a solution, namely a disposition schedule, is not guaranteed: Even though the dominating objective number of assigned blocking time stairways (3.8) guarantees that an optimal solution will dispatch as many train movements as possible, some train movements might still be left without an assigned blocking time stairway. This can happen when the finite set of considered blocking time stairways or the station capacity is not sufficient, for a conflict-free disposition schedule to be found, which would assign a blocking time stairway to all considered train movements. This phenomena relates to stability considerations in control systems which have a constrained state or control space. That all train movements must have an assigned blocking time stairway can be regarded as constraint, which affects the feasible control space \( U_k \): Controls \( u_k \) which do not assign all blocking time stairways to the train movements are not considered as feasible controls.

The remedy we propose to overcome situations, where the forecast of train movements would not allow for a feasible control, is to include additional blocking time stairways in the forecast. These additional alternative blocking time stairways should be formed according to the dispatcher’s action space (recall Section 1.2.2). For our test case (Chapter 5) we only considered additional alternative travel paths and postponing the start time of already existing blocking time stairways, i.e. a later departure at the central station or a later entry at the portal. In fact, procrastination is a commonly used action taken by the dispatcher when coping with disturbances. After such postponed blocking time stairways have been created, the train dispatching model is recreated and resolved. This procedure is repeated until the optimal solution assigns for each train movement in the dispatching time horizon a blocking time stairway.

Under the assumption that every considered train movement has at least one travel path, this procedure terminates after a finite number of steps. This can be seen easily, when one considers that the controller can postpone blocking time stairways of unassigned train movements beyond the dispatching time horizon. The MPC controller considers only
train movements in the dispatching time horizon, and therefore, the railway infrastructure resources are in principle completely available for scheduling blockings, which occur after the dispatching time horizon. It is important to note that the postponing of blocking time stairways beyond the dispatching time horizon does only resolve the infeasibility for the current MPC iteration. The next MPC iteration is probably dealing again with infeasibility issues, when previously postponed blocking time stairways are now in conflict with train movements, which entered the dispatching time horizon. In railway operations this issue is also reflected by the knock-on delays, which occur when punctual train movements start to be delayed because of already delayed train movements.

The heuristic for generating postponed blocking time stairways, which was applied in the case study is subsequently described.

### 4.3.3 Heuristic for generating postponed blocking time stairways

For the test case we applied an heuristic to generate additional postponed blocking time stairways: A train movement lacking an assigned arrival (departure) blocking time stairway received the possibility to be postponed. For this purpose the train movement’s set of alternative arrival (departure) blocking time stairways was extended by adding postponed arrival (departure) blocking time stairways. For each already considered arrival (departure) blocking time stairway, postponed stairways were added with arrivals (departures) which start +5, +10, . . . , +60 seconds later.
Proof of concept simulations based on an implementation of the described approach have been executed in collaboration with the SBB for the central railway station area of Berne, Switzerland. We explain in this chapter the network layout, the considered temporal scopes and trains of the test case, the computational environment, the simulated closed-loop discrete-time framework and comment on computation results.

5.1 System boundaries

The through station Berne with 13 platforms (excluding narrow-gauge platforms) is a major hub in Switzerland’s railway network and is one of the most challenging train dispatching areas in Switzerland. As a consequence the SBB is investigating possibilities for increasing the capacity of the central station (i.e. in [Weidmann et al. (2009)]). This additionally motivated our choice to choose Berne as central station area for the test case. The considered area (see Figure 5.1) has a diameter of roughly six kilometres and contains approximately 900 switches.

All detailed network data, i.e., signal positions, maximum velocity bounds, switches etc. has been provided by the SBB in the form of a database. Based on this database, we implemented modules to compute available travel paths, speed profiles and finally blocking time stairways for the considered trains. It is important to note that we applied a travel path reduction in form of a $k$-shortest path computation (see Section 2.4 for details) as the topology contains in principle millions of travel paths for each train, many of which would never be used in practice. The travel path reduction operates on a Fahrnetz-graph (see Definition 2.2) which contains $n \approx 4'000$ vertices and $m \approx 480'000$ arcs. If not otherwise stated the average number of considered travel paths for the arrival (or departure) of a train was 175 during our computations. The resulting total number of infrastructure resources ($|R|$) which had to be considered by all these travel paths was approximately 600.
5.2 Considered trains

Similar to the topology, the SBB provided us with historic train records of an operational week in January 2011 accompanied by the commercial timetable and connections from their database. For the test case we picked for a day, all freight and passenger trains which stopped at or passed through the central railway station Bern. As a result roughly 1500 trains arriving in and departing from Berne were part of the simulation. For each train and each of the on average 175 travel paths the optimal control problem for computing speed profiles had to be solved by applying dynamic programming (Appendix B). The recorded passing times of the trains at their entry portal (or at their starting point in case they started their journey inside the area) served as reference start times for considered arriving train movements. For the departure blocking time stairways as a reference we used start times from the commercial timetable if provided, and otherwise, the recorded departure times. In addition to the reference times, additional initial start times (namely +5,+10,+15,+,+55,+60s) were considered for trains with commercial timetable times to allow their rescheduling as a dispatching action. Once the speed profiles have been computed, the blocking time stairways were generated according to the definition of blocking times and the described safety parameters of Section 2.6.

5.3 Temporal scopes

Section 4.3 explained how different temporal scopes arise when trains are dispatched over time by a closed-loop discrete-time framework. Subsequently we describe the defining parameters values we picked for the case study:

Dispatching frequency interval \( \Delta \)

The dispatching frequency interval was chosen to be one minute, as the computation of a disposition schedule in each MPC control iteration (see Figure 4.4) required less than
a minute. However if significantly more alternative blocking time stairways were to be considered in each MPC step, the dispatching frequency interval must be chosen at a higher value.

**Travel path fixing horizon**

We estimated with dispatchers that a travel path should be considered fixed six minutes before the allocating train movement follows that travel path.

**Dispatching time horizon**

In current practice a dispatcher at a central station usually only considers trains predicted to arrive or depart within the next twenty minutes. There are two main reasons for this: first, the farther into the future within the forecast, the larger the gap between predicted timings and their realization is. A performance evaluation of the Swiss *Rail Control System* (RCS) prognosis (see Dolder et al. (2009)) illustrates this effect. Therefore the disposition schedules are short-termed, as planning too far into the future does not provide much additional gain. Second, the farther one plans into the future the more complex the task becomes. Hence, in the test case the dynamic train dispatching is done with a rolling time horizon of twenty minutes.

### 5.4 Computational environment

The following software was used for our computational study:

- Pre-processing and runtime environment: Java 1.5 with additional libraries (Parallel Java (Kaminsky and McOmber, 2011), Java plot 0.4 (Lakhiani 2011), JGraphT 0.7.3 (Barak, 2011) and Apache-commons (Apache, 2011))

- Database management system: Oracle RDBMS 11.2 (Oracle 2011)

- Solver for integer programs: IBM ILOG CPLEX Optimizer Version 12.3 (IBM 2011)

The machine on which the computations were executed has the following hardware characteristics:

- Architecture: x86_64

- 24 Processors: Intel(R) Xeon(R) CPU X5650 at 2.67GHz

- Memory (RAM): 95 GB
5.5 Sampled disturbances

Section 4.3.1 explains how the disturbances are modelled as entry delays and are a part of the MPC framework for the dynamic system described in Section 4.3. The delays are generated in two consecutive steps: First, the actual delays which shall manifest themselves later during the MPC iterations are computed and consecutively, predictions of these delays are computed by adding normal distributions to the actual delays. Figure 5.2 shows some predicted entry delays for several train movements during consecutive iterations of the MPC scheme. The delays within have been sampled based on the subsequently described approach. The actual occurring delays (in the figure these are the delays at

![Figure 5.2: Example of predicted entry time delays \( f_t(K-k) \) for train movements \( t \), for which the blocking time stairway will be fixed after \( K = 14 \) MPC iterations.](image)

MPC iteration \( k = 14 \)) are usually computed based on a modified exponential function (Herrmann (2005))

\[
F(x) = 1 - p \cdot e^{-\lambda x}, \quad x > 0
\]

where \( p \) is the expected percentage of the entering train movements with delays and \( \lambda \) is reciprocally proportional to the expected delay. The actual delay values are then sampled by computing its inverse over uniformly sampled values \( y_t \in [0, 1], \forall t \in T \) as

\[
F^{-1}(y_t) = \frac{-1}{\lambda} \cdot \ln \left( \frac{1 - y_t}{p} \right).
\]

It is reasonable to assume that the forecast of such delays converges over time towards the actual delays. Assuming that the actual delay \( F^{-1}(y_t) \) will manifest itself after \( K \) MPC iterations from now, the current prediction of the delay \( f_t(0) \) can be computed by the following backward computation (in the figure this corresponds to starting at the delays at the right side and computing the predicted delays by moving towards the left side):

\[
f_t(K) = F^{-1}(y_t); \quad f_t(K - k) = f_t(K - k + 1) + \mathcal{N}(\mu, \sigma), \quad K \geq k \geq 1
\]
For the case study the following parameter values were chosen:

- \( p = 7\% \) of trains are delayed
- \( \lambda = \frac{1}{40} \), i.e. the expected average delay of a (delayed) train was 40 seconds
- \( \mu = 0 \) and \( \sigma = 20 \) were chosen as normal distribution parameters for the prediction of delays
- \( K = \frac{\text{dispatching time horizon} - \text{travel path fixing horizon}}{\text{dispatching frequency interval} \Delta} = 14 \)

## 5.6 Analysis of simulation results

### Computation times

The computation times are promising. Figure 5.3 shows the times required for solving the binary linear problems of each MPC iteration to optimality during the simulation. Note that in the case of an observed infeasibility during a single MPC iteration several consecutive, increasingly bigger instances of the static train dispatching problem had to be solved, since additional blocking time stairways had to be considered (see Section 4.3.2 on dealing with infeasibility). The outliers in Figure 5.3 with a computation time above 12 seconds are a result of this phenomena. Nevertheless, even with such infeasibilities an overall computation time above one minute for one MPC iteration was never observed. This is critical, as one minute was chosen for the train dispatching frequency interval. The time required to build the model was usually below a second and at most three seconds. The overall simulation of the operational day took approximately 1.5 hours.

![ILP computation times during simulation](image)

**Figure 5.3:** The time required to solve the binary linear problems (0% gap) was below a minute and on average about 6 seconds.
Infeasibilities

The observed infeasibilities during the simulation indicate that the number of modelled alternative travel paths and arrival (departure) time points are sometimes not enough (recall Section 4.3.2). In these cases, additional time points were added to the model and the resulting problem instance was resolved. In Figure 5.4 the routing possibilities for each train was limited to 13 paths and as a result many infeasibilities occurred which had to be resolved by postponing trains.

![Figure 5.4: The number of occurring infeasibilities during the simulation when considering 13 paths for each train.](image)

If one considers the number of predicted train movements during the simulation which have not yet an assigned (and fixed) blocking time stairway (Figure 5.5), it becomes clear that the train movements follow the Pulse-timetable of Switzerland: Trains arrive and depart at main stations around the complete hour and around the half hour, in a form of a pulse. As a consequence, roughly 20 minutes before a pulse several new train movements have to be considered by a MPC iteration all at once, which in turn increases the chance that a train movement can not be assigned by the optimization routine without considering additional blocking time stairways, thus leading to an infeasibility. Thus the peaks of observed infeasibilities in Figure 5.4 are related to the Pulse-timetable.

We therefore devise a strategy to reduce the occurrence of infeasibilities in the MPC iterations. The strategy is based on providing more scheduling flexibility, by considering additional travel paths for the train movements. The effect of increasing the number of travel paths to on average 130 paths for each train, one can see in Figure 5.6 that far fewer infeasibilities occur and additionally, that they now only occur around traffic peak hours when many trains have to be scheduled. Thus, the number of considered travel paths (or in consequence the number of considered blocking time stairways) as a representative for flexibility is an important system design parameter.
5.6 Analysis of simulation results

Figure 5.5: The number of train movements inside the dispatching horizon which had to be dispatched during the discrete-time simulation.

Figure 5.6: The number of occurring infeasibilities during the simulation when considering 130 paths for each train.

**STD problem instance sizes**

The size of the static train dispatching optimization problems is illustrated by the number of binary variables (Figure 5.7) and the number of constraints (Figure 5.8). The outliers are a consequence of the iterative inclusion of additional blocking time stairways in the train dispatching optimization problem in the case of observed infeasibilities during a MPC iteration. Each additional blocking time stairway results in a new binary assignment variable and may lead to several additional constraints.
Trade-off: Solution quality vs. computation time

Measuring the quality of train dispatching is difficult as the results are based on the simulation environment, prohibiting a comparison with current operational practice. Thus
we decided to highlight the potential of our approach by comparing different simulations in our setting. Namely, we considered on average 13, 14, 15, 16, 17, 20, 30, 40, 50 or 70 routing possibilities and combined each of these routing possibilities with 20 sampled disturbance scenarios (see Section 5.5). In a next step the objectives, namely the number of broken connections and the amount of occurred delays were averaged for each of the routing possibility over the 20 different delay scenarios.

Figure 5.9: The number of considered paths influences solution quality.

Figure 5.9 shows the average overall achieved solution quality for each of the ten different routing possibilities. A point represents the average achieved solution quality of 20 simulations (for each sampled disturbance scenario one simulation) and the point’s label indicates the average number of routing possibilities. The solution quality is measured in terms of number of missed connections along the horizontal axis and delays along the vertical axis. One can observe, that the solution quality significantly improves when considering more travel paths as alternatives during train dispatching. Furthermore, the solution quality does not improve monotonically: The simulation with 30 travel path possibilities resulted in more missed connections compared to a simulation which considered 17 travel path possibilities. This phenomenon is a consequence of the MPC framework, where every MPC iteration only considers the train movements starting before the dispatching time horizon and thus can be seen as a local optimization routine.

Adding more routing possibilities for improving the quality of train dispatching (see Figure 5.9) affects the size of STD instances, since more blocking time stairways and thus more binary variables and constraints are part of the optimization model. Figure 5.10 shows the average total computation time of each of the ten routing possibilities. One observes that, for very few routing possibilities (13, 14, 15) the total computation increases, as few routing possibilities result in more infeasibilities, which in turn have to be dealt
with by including additional blocking time stairways (see Section 4.3.2 for the details). For 16 and more routing possibilities the overall computation times start to increase, as the infeasibilities are no longer the deciding factor but rather the growing size of the STD problem instances. With the increase of routing possibilities the amount of binary variables in each STD problem instance increases. Figure 3.5 already illustrated how the computation time increases for single STD problem instances when increasing the number of binary variables therein. The increase of the overall computation time in Figure 5.10 for 16 and more routing possibilities is a direct consequence of this.

![Figure 5.10: The average total computation time for a simulation remains small even when considering on average 70 paths per train.](image)

### 5.6.1 Objective trade-off: Delay vs. Connections

In order to illustrate the trade-off between the occurrence of delays and broken connections the simulation was executed for different coefficient ratio’s of the objectives, i.e. the ratio between the connection objective coefficient and the arrival (or departure) delay objective coefficient was chosen as

\[
\frac{w_C}{w_A} = \frac{w_C}{w_D} \in \{10, 1, 0.1, 0.05, 0.04, 0.03, 0.025, 0.01, 0.001, 0.0001\}.
\]

Additionally, for each of these ratios the simulation was executed 20 times for 20 different disturbance scenarios. Each disturbance scenario was formed by sampling entry time delays as it is explained in the Sections 4.3.1 and 5.5. Afterwards the objective values achieved by the simulations were averaged in order to dampen possible effects, which single disturbance scenarios could have on the objective trade-off. Figure 5.11 shows the
5.6 Analysis of simulation results

In each planning step of the described closed-loop discrete-time control framework a temporally local problem is solved, and thereby, pareto optimal solutions for the complete operational day are not guaranteed to be found. Nevertheless, the averaged computed solutions can be seen as an approximation to the pareto front. Furthermore, solving an instance of the train dispatching problem with an extended planning time horizon of a complete operational day is computationally not possible. Although computing the Pareto front for the complete day seems intractable, it can be approximated by heuristics in each planning step for the temporally local problems (see Corman et al. [2010b] for such an approach).

Figure 5.11: Trade-off between delays and connections. Each point is labelled with the ratio between the connection objective coefficient and the delay objective coefficient. High ratio values correspond to focusing on keeping connections and small ratio values correspond to focusing on reducing delays.
Conclusions

This concluding chapter first summarizes the milestones of the thesis, followed by an appraisal of the results and the lessons learned. Finally the outlook provides a long term strategy for the integration of the thesis’ framework in practice.

6.1 Achieved milestones

The work presented in this thesis provides essential algorithms and methods for dispatching trains in a complex central railway station area by means of a computer supported, closed-loop discrete-time framework. The thesis has been written in close collaboration with Swiss Federal Railways, Infrastructure Division and the framework has been studied for the central station area Berne, Switzerland. However, the proposed algorithms and models in this framework are sufficiently general for their application to other railways and infrastructure networks.

Subsequently three milestones, all of which are essential parts of this thesis, are highlighted:

6.1.1 Modelling the railway system

The first milestone is described in Chapter[2] which discusses models and algorithms for representing the topology and train movements in a railway system. There are three basic models for representing a train movement:

1. The travel path models the path which a train follows in the railway network.

2. The speed profile models at which velocity a train follows a travel path.

3. The blocking time stairway models the blocking of infrastructure elements by a moving train, in order to ensure the safety of that train against other train movements along its travel path.
Given these models, the thesis suggests algorithms for creating instances of the models, by processing already available data in today’s railway control systems:

- Yen’s algorithm is suitable for computing meaningful travel paths in railway networks.
- A dynamic programming approach, which is suitable for computing speed profiles, was developed in collaboration with Gioele Balmelli during his master thesis.
- Blocking time stairways are computable based on a travel path and a speed profile by following a fairly simple set of rules.

Furthermore, given the above models and algorithms, the chapter discusses their use in the prediction of future train movements: In railway traffic management the dispatching process has to consider past, current and future train movements. For past and current train movements the travel paths and the speed profiles are known but the outcome of future train movements is not yet determined. The thesis suggests to predict the future train movements in the form of a forecast, which is a discrete set of blocking time stairways based on alternative travel paths and different but plausible speed profiles.

Given such a forecast, it is then the task of train dispatching to assign to each train movement a blocking time stairway, such that the assigned blocking time stairways are not conflicting. The second milestone is a solution approach for this train dispatching task:

### 6.1.2 Static train dispatching

The second milestone is described in Chapter 3 which discusses a discrete optimization model for optimally dispatching trains in a complex central station area. At a given single point in time this model considers a (static) forecast of future train movements in form of many alternative blocking time stairways. The forecast contains for each train different conceivable realizations of its future movement by taking into account the freedom of choice among different travel paths or travel speeds.

The resulting train dispatching problem is looking for an one-to-one assignment between trains and blocking time stairways, such that the assigned blocking time stairways are not conflicting and additionally, the assignment is optimal with respect to a previously defined objective. This thesis considers and combines two objectives in form of a linear function which measure the quality of a solution. The two objectives the thesis consider are:

1. Improved punctuality by minimizing delays.
2. Improved reliability by dispatching all trains and maintaining connections.

Together with several operational and safety constraints the objective function are combined into one mathematical optimization problem, the so-called static train dispatching problem.

It is at this point where the thesis bridges the problem of train dispatching and mathematical optimization theory to first solve the problem and in addition, to certify the quality
of computed solutions, i.e. determine a provable statement on the solution quality. The static train dispatching model is a linear and discrete optimization model. Unfortunately, discrete optimization problems are usually very hard to solve in general. However, existing algorithms for solving discrete optimization problems exhibit a strong dependency on the strength of their mathematical formulation. Thus, the thesis improves the mathematical formulation of previous models by paying better attention to the structure of the underlying railway system. A good example is the improved conflict modelling in this thesis: Instead of pairwise modelling the conflicts between blocking time stairways the conflicts are gathered into conflict groups (cliques). As a result of the improved modelling, conducted computational experiments showed a significant improvement in computation times: Instances which previously hours to solve could now be solved in a few seconds.

The significantly reduced computation times encouraged us to integrate this - so far - static approach in a dynamic setting, which is the topic of the next and last milestone of this thesis.

6.1.3 Dynamic train dispatching

The third milestone is covered in two chapters: Chapter 4 describes how the static train dispatching problem is integrated in a closed-loop discrete-time framework in order to dispatch trains over time. Chapter 5 describes a proof of concept by a simulation of the closed-loop discrete-time framework for the complex central railway station area Berne, Switzerland and comments on collected results.

In short the closed-loop discrete-time framework works as follows: The changes in the real world railway system, which are already measured in today’s railway control systems, are processed and the dispatching optimization model is updated accordingly; new solutions to the optimization problem are computed in fixed time intervals and may trigger control actions in form of travel path changes or speed recommendations. These control actions serve as feedback control over the real world railway system and with them the control loop is closed.

During the application of this framework in the test case Berne, Switzerland, two compromises were necessary:

1. Our access to operational data of the railway system was limited to detailed data records of train movements during a week in early 2011 and data on the railway topology. As a consequence the physical layer of the railway system had to be simulated: The measurements of train movements were based on recorded data and sampling of disturbance.

2. Keeping the model consistency when facing the continuous changes in the railway system as well as control actions triggered by dispatching decisions is a challenging task. The thesis proposes a solution approach based on model predictive control in which temporal scopes allow for a distinctive view on past, current and future train movements.

Bringing all the pieces of the thesis together in proof of concept simulations for the central railway station area of Berne, Switzerland (see Chapter 5), concludes the last milestone.
Chapter 6: Conclusions

6.2 Appraisal of the results

Dispatchers are confronted with complex decisions when managing the increasingly dense traffic in highly utilized central railway station areas. This thesis presents a computer-supported dispatching framework to assist train dispatchers in their decision making by providing optimized disposition schedules. The dispatching framework is conceived as a closed-loop discrete-time system, which benefits on one side from the recent and today’s industrial research and development trends in railway operations, and on the other side from progress made in discrete optimization and its application to train dispatching.

The proposed closed-loop discrete-time system involves a model predictive controller, which iteratively solves (static) train dispatching problems that are modelled as mathematical optimization problems. The mathematical optimization problems are in principle constrained assignment problems, for which problem knowledge is exploited in order to solve them in reasonable time. The computational results of the thesis’ case study for the central railway station area Berne confirms that the controller runs in fact sufficiently fast to be viable in practical applications. However, the case study cannot be used for judging the quality of computed disposition schedules, since the setting of this thesis lacks the inclusion of a dispatcher and a complete model for the physical layer. Nevertheless, a lot of effort was made during this thesis to model the train movements and the railway safety system realistically in order to present a case study, which is sufficiently convincing so that industry will take this framework a step further toward its integration in practical operations (see outlook for details).

The framework of this thesis could also serve as a tool for analysing what effects such changes could potentially have, be it in the infrastructure, timetable or in the rolling stock, by evaluating the changes in a simulative, operational setting based on this framework. An example could be an infrastructure operator, who analyses the effects of different options for capacity extensions in an operational setting by simply extending the underlying topology models accordingly.

6.3 Lessons learned

Many lessons are learned during the course of writing a PhD thesis, the following being the most relevant from my personal perspective:

- Train movements in today’s railway systems can be accurately modelled by the concept of blocking time stairways. In fact today’s railway operating and planning systems begin to include algorithms for conflict detection and are based on the concept of blocking time stairways.

- A model based on blocking time stairways which represents the state of the railway traffic in a complex central station at a given time is computable in matter of a second or even less.

- Algorithmic dispatching support is computationally tractable even for complex central railway stations, asking for its integration in today’s railway control systems in
The thesis’ case study prohibits any conclusions on the quality of the approach. In the outlook of this thesis further steps are suggested, which could allow researchers and railway operators to qualitatively assess dispatching decisions.

A net-wide application of the presented train dispatching approach is currently not conceivable. An approach based on decomposition of the railway network with coordination interfaces might be a successful approach.

Bridging the gap between operations research and industry is essential for industrial progress and knowledge transfer. By writing a thesis which helps bridging the two areas, one receives the opportunity to foster knowledge and understanding for both areas.

6.4 Outlook: Integrating the framework in practice

The introduction (Section 1.4) proposes a closed-loop discrete-time framework (recall Figure 1.10) for train dispatching in a central station area. This thesis suggest to develop such a framework in four steps. Since the first step is already covered by this thesis, the remaining thee steps are as follows:

6.4.1 Interfacing to a simulation environment

The second step is to interface the control layer with a simulated infrastructure layer and simulated operational data. For this purpose, it is necessary to interface a railway simulation software, preferably one that is accepted for its accuracy by railway operators, with the operational layer and the control layer in two directions (see Figure 6.1):

I. In the upward direction on the left, the (simulated) operational data is now gathered based upon inputs from the simulated physical system and the forecast has to be made based on the simulated data.

II. In the downward direction on the right, a computed disposition schedule (a solution of the mathematical model) has to be translated into operational commands, such that it can be interpreted by the simulation software of the physical layer. Note that, similarly to the first step, the dispatcher is not yet taking an active part in this second step.

A trusted simulation allows conclusions to be drawn on the quality of the models and concepts which have been developed in the control layer. Qualitative insights and comparisons of approaches for the control of trains can be gained. Thus, leading to improvements in such approaches and hopefully sufficient trust in the industry means that such approaches can be realized and be beneficial to railway operators.
6.4.2 Incorporating the dispatcher

In the third step the interface between a computerized dispatching assistant and the dispatcher shall be investigated and developed (see Figure 6.2). In this very important step the main topic is to investigate and learn about the interaction between the human dispatcher, and the computer; a dispatching assistant driven by mathematical models.

Interfacing the framework to the real world

Only in the forth and last step will the transition from a simulated infrastructure and operational layer toward the real world infrastructure and operational data be done.
Figure 6.2: Step three of realizing the closed-loop framework: The interaction between the dispatcher and the dispatching assistant is investigated in a learning environment, as the physical layer is still based upon a simulation.
approach time  Travel time required for a train-head to get from the distant signal to the (entrance) main signal.

bahnhof  German term for station area.

block passing time  Travel time required for a train-head to get from the (entrance) main signal to the (exit) main signal.

block section  Track section between two (consecutive) main signals which is not (completely) inside home signal limits.

block signal  Trackside main signal that governs train movements in open lines.

blocking time  Time interval during which a track section is operationally allocated exclusively to a train movement, thus blocking it during that time for other train movements.

blocking time stairway A sequence of pairs consisting each of a blocking time and a track section, indicating the exclusive operational allocation of each track section during the associated blocking time to a train movement.

clearing point  Train rear detection point which allows the safety system to clear any occupations of previous block sections and overlaps of the train movement associated with the detected train.

clearing time  Travel time required for a train with the head in front of the (exit) main signal until the last (rear) axle of the train passed the next clearing point.

crossover  Arrangement of two switches connecting two parallel tracks which allows trains to cross over from one track to other.

dark territory  Infrastructure tracks which are not safeguarded by a signalling system.
distant signal  Trackside signal for announcing an approach (or warning) aspect towards a red main signal which is still in braking distance.

double vertex graph  An undirected graph representation of the railway topology where double vertices implicitly allow the computation of admissible routes.

exit signal  Trackside main signal at an exit of a station area (Bahnhof) that authorizes train movements to exit the station area into the next block section beyond the signal.

fahrrnetz  A directed graph representation for computing train paths in associated Spurplan graphs.

fixed block signalling  Safety system based on fixed installed track-sided signals governing the train movements in the corresponding fixed block sections.

home signal  Trackside main signal at an entry of a station area (Bahnhof) that authorizes train movements to enter the station area; opposing home signals limit the station area.

interlocking  An arrangement of switches and signals interconnected in a way that each movement follows the other in a proper and safe sequence.

interlocking signal  see intermediate interlocking signal.

intermediate interlocking signal  Trackside main signal that governs train movements inside interlockings.

junction  Location in the network where tracks converge or diverge by an arrangement of signals and switches.

loop  Main track for passing and overtaking trains.

main signal  Trackside signal that indicates the authorization status for the entrance of trains into the track section beyond that signal. In areas with small distances between main signals the main signals may also function as a distant signal.

main track  Track used for regular train movements, e.g. a station track, a loop or a track belonging to an open line.

mitteAG  Logical point which references a possible start or end locations of a route.

open line  Tracks which are outside of home signal limits of station areas.

overlap  Breaking distance after a main signal providing an additional safety measure in case a train would jump the red main signal.
route A technically protected path for a safe train or shunting movement through an interlocking.

route forming time Time required to clear the (entrance) main signal (lights to green) and in case of junctions or crossings, to set switches into the correct position for the route.

route releasing time Time required to reset the signals and eventual switches of junctions or crossings back to its standard configuration.

secondary track Track not used for regular train movements, but for shunting movements; e.g. yard tracks.

shunt board Trackside signal to designate a limit of a shunting area; shunting movements beyond the shunt board are not allowed without exceptional permission by an operator.

shunting movement Any movement with rail vehicles without explicit authorization for movements on main tracks, executed at low speed, over short distances and usually over secondary tracks.

shunting signal Trackside signal which authorizes shunting movements and protects train routes against shunting movements.

diding see secondary track.

signal cabin A local interlocking station that controls points (e.g. switches) and signals of the respective interlocking.

signal viewing time Time required by the train driver to view the status of the distant signal (or, in case there is no explicit distant signal, of the previous main signal).

speed profile The time-distance curve of a train movement along its path, often visualised in a graph.

spurplan graph An undirected graph representation of the railway topology, mainly used in Germany for the computation of speed profiles and blocking time stairways.

station area An arrangement of tracks and signals limited by opposing home signals with at least one turnout. Train movements originate, terminate and turn in these areas.

station crossover A crossover outside (!) of a station area and part of an open line.

station junction A junction outside (!) of a station area and part of an open line.

station track Main track inside a station area.

train movement A locomotive coupled to zero or more vehicles with authority to move on a main track.
travel path  A sequence of routes through a sequence of interlockings forms the complete travel path of the train.

warning signal  See distant signal.
Acronyms

CG  Conflict Graph.

DP  Dynamic Programming.

ERTMS  European Rail Traffic Management System.

ETCS  European Train Control System.

GUI  Graphical User Interface.

ILP  Integer Linear Program.

NP  Non-deterministic Polynomial time.

NP-hard  Non-deterministic Polynomial-time hard.

RCS  Rail Control System (developed by SBB).

RTCG  Resource Tree Conflict Graph.

SBB  Schweizerische Bundesbahnen.

SFR  Swiss Federal Railways.

STD  Static Train Dispatching.

UIC  International Union of Railways.


Yen’s algorithm for computing $K$ shortest, loop-less paths

Yen’s Algorithm [A.1] suggests to generate the list of shortest paths $A$ iteratively by deviating in each iteration step from paths computed in previous iteration steps. In the following, let $A^k = \langle v_1 = s, \ldots, v_l(p_k) = t \rangle$ for $k = 1, \ldots, K$ denote the $k$-th shortest loop-less path, furthermore let $A$ be the list containing the computed shortest paths and let $B$ a candidate list for additional shortest paths. For simplicity, we assume that there exist at least $K$ loop-less paths between the given start terminal $s$ and end terminal $t$ in the given Fahrnetz graph $G(V, A, c)$. Additionally, note that the costs are all non-negative and thus, the problem of negative cycles can not occur in shortest path computations.

The algorithm starts in Line 1 by computing all shortest paths and list them in the candidate list $B$ by application of Dijkstra’s algorithm ([Dijkstra, 1959]). In case there are already at least $K$ shortest paths, $K$ arbitrary paths are selected from $B$ and returned as the result (line 2 and 3). However, generally there are less than $K$ shortest paths, thus one arbitrary path $A^1$ is selected and transferred from $B$ to $A$ (line 5 to 7). Then, after these initialization steps, the algorithm iteratively creates and adds the missing $2, \ldots, K$ next shortest paths to $A$ (in the loop from Line 9 to 37). In each iterative step, the algorithm computes for all but the last vertex of the latest already accepted path $A^{k-1}$ new candidate paths, which deviate from all the accepted paths in $A$, thus the problem of negative cycles can not occur in shortest path computations.

The algorithm starts in Line 1 by computing all shortest paths and list them in the candidate list $B$ by application of Dijkstra’s algorithm ([Dijkstra, 1959]). In case there are already at least $K$ shortest paths, $K$ arbitrary paths are selected from $B$ and returned as the result (line 2 and 3). However, generally there are less than $K$ shortest paths, thus one arbitrary path $A^1$ is selected and transferred from $B$ to $A$ (line 5 to 7). Then, after these initialization steps, the algorithm iteratively creates and adds the missing $2, \ldots, K$ next shortest paths to $A$ (in the loop from Line 9 to 37). In each iterative step, the algorithm computes for all but the last vertex of the latest already accepted path $A^{k-1}$ new candidate paths, which deviate from all the accepted paths in $A$, thus the problem of negative cycles can not occur in shortest path computations.

The algorithm starts in Line 1 by computing all shortest paths and list them in the candidate list $B$ by application of Dijkstra’s algorithm ([Dijkstra, 1959]). In case there are already at least $K$ shortest paths, $K$ arbitrary paths are selected from $B$ and returned as the result (line 2 and 3). However, generally there are less than $K$ shortest paths, thus one arbitrary path $A^1$ is selected and transferred from $B$ to $A$ (line 5 to 7). Then, after these initialization steps, the algorithm iteratively creates and adds the missing $2, \ldots, K$ next shortest paths to $A$ (in the loop from Line 9 to 37). In each iterative step, the algorithm computes for all but the last vertex of the latest already accepted path $A^{k-1}$ new candidate paths, which deviate from all the accepted paths in $A$, thus the problem of negative cycles can not occur in shortest path computations.

In phase 1 the algorithm starts with the $i$-th vertex $v^i$ of $A^{k-1}$ as the deviation vertex. It is the vertex from which a new candidate path shall deviate from all the already computed paths in $A$. Algorithm [A.2] then prepares a temporary graph $\hat{G}$ for computing a new deviating candidate path by a shortest path computation. The new deviating path shall have the same sub-path up to vertex $v^i$ as $A^{k-1}$ but then will deviate from all other paths in $A$. The temporary graph $\hat{G}$ is created from the original graph $G$ by changing costs at outgoing arcs of vertex $v^i$ and by removing the first $i - 1$ vertices and their adjacent arcs of $A^{k-1}$. Cost changes are made for the at $v^i$ outgoing arc of $A^1$, if the sub-path of $A^j$ up to $v^i$ is equivalent to the sub-path of $A^{k-1}$. The costs are then increased to infinity in the new graph in order to force a deviation from the path $A^j$ (compare line 3 to 5 in
Appendix A: Yen’s algorithm for computing K shortest, loop-less paths

Algorithm [A.2] and Figure [A.1]. If the sub-path of a $A^j$ already differs, then $A^j$ has no impact on the search for a deviating path. Thus, no cost changes are needed in this case (see Figure [A.2]).

Figure A.1: The last path $A^{k-1}$ in $A$ and another previous path $A^j$ in $A$ have the equivalent sub-path up to the vertex $v^1$, from which the algorithm intends to deviate from. The weight of the arc going from $v^1$ to $v^3$ is changed to $+\infty$ (line 3 to 5 in Algorithm [A.2]), such that the search for a deviating path based on a shortest path computation between the vertices $v^1$ and $t$ will not follow the path $A^j$.

Figure A.2: The last path $A^{k-1}$ in $A$ and another previous path $A^j$ in $A$ have different sub-paths up to the vertex $v^1$, from which the algorithm intends to deviate from. Thus, $A^j$ has no impact on the search for a deviating path and no cost changes are made for the new temporary graph.

In phase II, a shortest, loop-less path $\hat{p}$ from vertex $v^1$ to the terminal vertex $t$ is computed in the temporary graph $\hat{G}$ (compare Line 15 and Figure [A.3]). If there are several such paths, the algorithm arbitrarily picks one. This shortest, loop-less path computation between two vertices can also be done by Dijkstra’s algorithm.

Figure A.3: Once the temporary graph $\hat{G}$ has been prepared by adjusting costs (see Figure [A.1]) and the nodes and arcs of the sub-path of $A^{k-1}$ up to vertex $v^1$ have been removed, a shortest path computation between $v^1$ and $t$ is executed on the temporary graph. This computed path (highlighted red box) yields, when appended to the root sub-path leading to $v^1$, a new candidate path $b^*$ in the original graph. Additionally, $b^*$ deviates from all other paths in $A$.

In phase III the computed path $\hat{p}$ is, if valid (line 14), appended to the root sub-path of $A^{k-1}$ starting at $s$ and ending at $v^1$, to form a new candidate path, denoted by $b^*$ =
sub\(A_{k-1} (s, v^1) \diamond \hat{p}\) (line 15). The new candidate is then added to the candidate list \(B\), if \(B\) has not yet enough candidates (line 17). If \(B\) has enough candidates, \(b^*\) may replace the worst candidate in \(B\) in case \(b^*\) is better than this candidate (line 19 and 20). Finally, the cost values are reset and the next candidate path can be computed by deviating at the next vertex of \(A^{k-1}\).

After all candidates have been computed and represented in the list \(B\), three cases have to be distinguished:

1. \(B = \emptyset\): No other loop-less paths exist between \(s\) and \(t\) and thus, the current set \(A\) is returned (line 35).

2. The number of paths \(z\) with minimum cost in \(B\) and the paths in \(A\) are at least as many as \(K\): \(A\) and \(K - |A|\) arbitrarily selected, minimum cost paths of \(B\) are returned as a sorted list (line 28 and 29).

3. Otherwise, one minimum cost path from \(B\) is chosen and transferred to \(A\) (line 31 and 32). The algorithm then continues with the next iteration.

### A.0.3 Complexity bounds of Yen’s algorithm

An algorithm’s efficiency is measured by the number of operations (time complexity) and in the number of memory addresses (space complexity) required for its execution.

The overall time complexity is \(K\) times the time complexity of each of the (at most) \(K\) iterations of the algorithm. In each iteration, the workload heavily depends on the length of the path \(A^{k-1}\) and in the worst case, \(n - 1\) vertices have to be considered for the computation of new path candidates. A new shortest, loop-less path between two vertices \(v^1\) and \(t\) can be computed based on Dijkstra’s Algorithm, which runs in \(O(n^2)\). Since Fahnnetz graphs are usually sparse, Dijkstra’s algorithm can be implemented more efficiently using Fibonacci-heaps \[\text{Fredman and Tarjan, 1987}\] as data structure. With the help of Fibonacci heaps the time complexity reduces to \(O((m + n) \log n)\), which is, if the graph is additionally connected, dominated by \(O(m \log n)\). The operations required for the preparation of the temporary graph are negligible compared to the computation of a shortest path. The maintenance of the candidate list \(B\) requires the selection of the best candidate, doable in \(O(1)\), and the insertion of a new candidate, which can be done in \(O(\log(K))\) since at most \(K\) candidates are stored in \(B\). Thus, the overall time complexity is \(O(K(n^3 + n + \log(K)))\), or simply \(O(K(n^3 + \log(K)))\) and \(O(K(n(m \log n) + \log(K)))\) for connected, sparse graphs. From this analysis it becomes clear, that two factors, the method used for solving the shortest path problems and the length of the shortest, loop-less paths, have a strong influence on the time complexity. Both factors are exploited in slightly more efficient algorithms studied in \[\text{Pascoal, 2006}\].

The space complexity is formed by the two lists \(A\) and \(B\), as well as the arc cost values \(c_{ij}\) (which simultaneously represent the complete Fahnnetz graph). This results in a worst case space complexity of \(O(n^2 + Kn)\).
Appendix A: Yen’s algorithm for computing $K$ shortest, loop-less paths

Algorithm A.1 Yen’s algorithm for computing $K$ shortest, loop-less paths

**Input:** $K \in \mathbb{N}$, $G(V, A, c : A \to \mathbb{R}^+)$, start vertex $s \in V$, terminal vertex $t \in V$  

**Output:** Sorted list of $K$ shortest, loop-less paths from $s$ to $t$ or, in case there are less than $K$ loop-less paths from $s$ to $t$, return those as a sorted list.

1. Determine all shortest, loop-less paths in a list $B$ 
2. if $|B| \geq K$ then 
   3. return List of $K$ arbitrary chosen paths from $B$
3. else 
   4. Chose an arbitrary path $A^1$ from $B$
   5. $A \leftarrow \{A^1\}$
   6. $B \leftarrow B \setminus \{A^1\}$
   7. end if 
   8. for $k = 2, \ldots, K$ do 
      9. for $i = 1, \ldots, l(A^{k-1}) - 1$ do 
         10. Let $v^i$ be the $i$-th vertex in the path $A^{k-1}$
         11. Compute temporary graph $\tilde{G}$ with Algorithm A.2
         12. Compute a shortest, loop-less path $\tilde{p}$ from $v^i$ to $t$ in $\tilde{G}$.
         13. if $\tilde{p}$ exists (and with finite total cost) then 
            14. Generate a new candidate path $b^*$ by concatenation: $b^* = \text{sub}_{A^{k-1}}(s, v^i) \circ \tilde{p}$
            15. if $|B| < K - k + 1$ then 
               16. $B \leftarrow B \cup b^*$.
            17. else if $c(b^*) < \max_{b \in B} c(b)$ then 
               18. $b^* = \arg\max_{b \in B} c(b)$
               19. $B \leftarrow B \setminus \{b^*\}$ 
            20. end if 
            21. end if 
         22. end if 
      23. Reset the cost values: $\hat{c} = c$
   24. end for 
25. if $B \neq \emptyset$ then 
   26. Let $z$ be the number of paths in $B$ that have minimum cost 
   27. if $|A| + z \geq K$ then 
      28. Choose arbitrary $K - |A|$ paths in $B$ of minimum cost and add them to $A$.
      29. return $A$
   30. else 
      31. Choose an arbitrary path $b^*$ in $B$ of minimum cost.
      32. $A = A \cup \{b^*\}$, $B = B \setminus \{b^*\}$
      33. end if 
   34. else 
   35. Return $A$, since there exist no other paths between $s$ and $t$.
   36. end if 
37. end for
Algorithm A.2 Yen’s algorithm: Phase I - computing a temporary graph.

Input: $\mathcal{G}, A, k, i$

Output: $\hat{\mathcal{G}}$

1: for $j = 1, \ldots, k - 1$ do
2:   Let $v^2$ be the $i$-th vertex in the path $A^j$
3:   if $\text{sub}_{A^{k-1}}(s, v_1) \equiv \text{sub}_{A^j}(s, v_2)$ then
4:     Let $v^3$ be the $i + 1$-th vertex in $A^j$
5:     Set the cost to get from vertex $v^2$ to $v^3$ temporarily to $\infty$: $\hat{c}(v_2, v_3) \leftarrow \infty$
6:   end if
7: end for
8: Create $\hat{\mathcal{G}}(\hat{V}, \hat{A}, \hat{c})$ by removing the first $i - 1$ vertices of $A^{k-1}$ (and their adjacent arcs) from the graph $\mathcal{G}$ and by replacing the cost values with $\hat{c}$
9: return $\hat{\mathcal{G}}$
Appendix B describes first the applied simplifications and assumptions used during the computation of train speed profiles, followed by a mathematical optimal control model and concludes with a solution method based on dynamic programming for solving the model.

B.1 Simplifications and assumptions

- **Only longitudinal dynamics**
  In textbooks (e.g., Garg and Dukkipati [1984]; Iwnicki [2006]) train dynamics are simplified by assuming that no vertical and lateral movement of rail wagons takes place. To the best of our knowledge this is also common practice in today’s train simulation software. We follow this simplification and just consider longitudinal train dynamics.

- **Train as point mass**
  We represent the train mass as a single point in order to facilitate computations. Similar to Chang and Jong (2005) we model the train as a single point for vehicle dynamics and extend it to a single line model to respect the speed limits correctly. In railway engineering it is also common to simplify the train mass by a three mass model in order to better account for slack action, the relative movement between the wagons (see Iwnicki [2006]).

- **Slack action ignored**
  For simplicity we did not consider relative movement between wagons in our model.

- **Starting resistance ignored**
  Starting (static) resistances are usually higher than the (dynamic) propulsion resistance and should be included to model the start of a train movement correctly. As
Appendix B: Speed profile computation

empirical estimates for starting resistances were not available, we did not include them in our model.

- **Tunnel resistance ignored**
  Aerodynamic drag can change significantly when a train enters and exists a tunnel and we omitted these considerations for simplicity in our model.

- **Assume good weather conditions**
  Weather conditions significantly influence adhesion between rail and wheels. We assume good weather conditions which is reflected in our limited choice of adhesion coefficients.

- **Simplistic models for dynamic and mechanical braking forces**

- **Fixed percentage of dynamic braking is regenerative**
  When traction motors are used as generators for dynamic braking the electric energy dissipates as heat by passing the current through resistor banks (*rheostatic* braking) or by sending the current back to the supply (*regenerative* braking). We assume that a fixed percentage of the electric energy gained by dynamic braking is regenerative.

- **Thermal derating of tractive effort ignored**
  When trains drive with full tractive effort for a longer period of time, the maximum achievable traction effort decreases. This is a consequence of decreased motor torque when the transformer resistance increases over time, because of the heating of transformer windings. We ignore this effect in the modelling of traction effort.

### B.2 Mathematical model

Before we introduce an optimization model for the computation of speed profiles, Table B.1 shall first familiarize the reader with some notation. Subsequently, the complete model is presented followed by a discussion of the model components: The model objective, Newton’s law of motion, tractive force and adhesion, braking forces, resistances, speed limits, passenger comfort, freight safety and boundary conditions.

#### Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t )</td>
<td>time [s]</td>
</tr>
<tr>
<td>( x(t) )</td>
<td>position of the train at time ( t ) along the track [m]</td>
</tr>
<tr>
<td>( v(t) )</td>
<td>speed of the train at time ( t ) [m/s] (state)</td>
</tr>
<tr>
<td>( \beta(x) )</td>
<td>grade angle at position ( x ) measured in radians [-]</td>
</tr>
<tr>
<td>( \gamma(x) )</td>
<td>curvature radius at position ( x ) [m]</td>
</tr>
<tr>
<td>( F_L(t) )</td>
<td>achieved traction force of the locomotive at time ( t ) [N]</td>
</tr>
<tr>
<td>( F_T(t) )</td>
<td>applied traction force at time ( t ) [N]</td>
</tr>
<tr>
<td>( F_D(t) )</td>
<td>applied dynamic braking force at time ( t ) [N]</td>
</tr>
</tbody>
</table>
### Model overview

\[
\begin{align*}
\text{Min} & \quad w_{\text{time}} \cdot \int_{t_s}^{t_e} dt + w_{\text{energy}} \cdot \int_{t_s}^{t_e} (F_T(t) \cdot v(t) - k_R \cdot F_D(t) \cdot v(t)) \, dt \quad \text{(B.1a)} \\
\text{Subject to:} & \\
\rho \cdot M \frac{dv}{dt} = F_L(t) - F_D(t) - F_M(t) - R(v, x, \gamma) & \quad \text{(B.1b)} \\
F_L = \min(F_T(t), F_A) & \quad \text{(B.1c)} \\
F_T(t) = \begin{cases} 
    u_T(t) \cdot F_{\text{max}} - k_T \cdot v(t), & \text{if } (u_T(t) \cdot F_{\text{max}} - k_T \cdot v(t)) \cdot v(t) < u_T(t) \cdot P_{\text{max}} \\
    u_T^2(t) \cdot P_{\text{max}} v(t), & \text{otherwise}
\end{cases} & \quad \text{(B.1d)} \\
F_A = k_A \cdot m_{\text{Loco}} \cdot g & \quad \text{(B.1e)} \\
F_M = u_M \cdot F_{\text{max}}^M & \quad \text{(B.1f)}
\end{align*}
\]
Appendix B: Speed profile computation

\( R_P(v) = A + Bv + Cv^2 \) \hspace{1cm} (B.1g)
\( R_G(x) = m \cdot g \cdot \sin(\beta(x)) \) \hspace{1cm} (B.1h)
\( R_C(x) = \frac{6.116}{\gamma(x) \cdot m} \) \hspace{1cm} (B.1i)
\( 0 \leq v(t) \leq V(x(t)) \) \hspace{1cm} (B.1j)
\( a \leq \frac{F_T(t) - F_D(t) - F_M(t) - R(v, x, \gamma)}{\rho \cdot M} \leq \bar{a} \) \hspace{1cm} (B.1k)
\( x(t_s) = x_s \) \hspace{1cm} (B.1l)
\( v(t_s) = v_s \) \hspace{1cm} (B.1m)
\( x(t_e) = x_e \) \hspace{1cm} (B.1n)
\( v(t_e) = v_e \) \hspace{1cm} (B.1o)

Planning objective: Minimize energy consumption and travel time \( (\text{B.1a}) \)

In practice speed profiles which minimize travel times are not considered as they give rise to several deficiencies (see Chang and Jong, 2005):

- Inefficiency due to wasted energy;
- Missing travel time reserves easily can cause train delay;
- Higher erosion of materials;
- Frequent changes between acceleration and deceleration causes discomfort to passengers.

As a consequence speed profiles are computed with respect to travel time and energy consumption. In our mathematical model we respect both by the commonly used weighted sum approach:

\[
 w_{\text{time}} \cdot \int_{t_s}^{t_e} dt + w_{\text{energy}} \cdot \int_{t_s}^{t_e} \left( \frac{\text{Tractive effort}}{F_T(t) \cdot v(t)} - \frac{\text{Recuperation energy}}{k_R \cdot F_D(t) \cdot v(t)} \right) dt \rightarrow \min
\]

In Chang and Jong (2005) this multi-objective function is called \textit{Proper Operation Time}.

Newton’s second law of motion \( (\text{B.1b}) \)

According to Newton’s second law of motion the acceleration of a body is parallel and directly proportional to the net force and inversely proportional to the constant-mass. For trains there are several forces contributing to the net force: Traction, braking and the overall resistance. For a running train the constant-mass consists of rotating mass and static mass and is usually computed by the static mass times the mass factor \((M \cdot \rho)\). The
B.2 Mathematical model

A rotating mass originates from stored kinetic energy in rotating axles and wheels (Andrews, 1986).

\[ F_L(t) - F_D(t) - F_M(t) - R(v, x, \gamma) = \rho \cdot M \frac{dv}{dt} \]

### Traction force (B.1d)

Typically the traction force of a locomotive is limited for low speeds by current and at higher speeds by power. We model the tractive effort at time \( t \) similar to Iwnicki (2006):

\[
F_T(t) = \begin{cases} 
    u_T(t) \cdot F_{max} - k_T \cdot v(t), & \text{if } (u_T(t) \cdot F_{max} - k_T \cdot v(t)) \cdot v(t) < u_T^2(t) \cdot P_{max}, \\
    \frac{u_T^2(t) \cdot P_{max}}{v(t)}, & \text{otherwise.}
\end{cases}
\]

where \( F_{max} \) is the maximum locomotive traction force [N], \( P_{max} \) the maximum locomotive horsepower [W], \( k_T \) the torque reduction \( \frac{N \cdot s}{m} \), and \( u_T(t) \in [0, 1] \) the applied traction control, where 0 corresponds to no tractive effort and +1 to full tractive effort. Figure B.1 illustrates the dependency of the traction force on speed and the applied traction control.

![Figure B.1: Traction forces for different traction control of a Swiss RE460 locomotive (\( P_{max} = 6'100 \text{ kW}, F_{max} = 300 \text{ kN}, k_T = 0 \)).](image)

### Adhesion and traction force (B.1e), (B.1f)

The adhesion between rail and wheels limits the applicable traction of a locomotive, since high traction combined with low adhesion could cause the wheels to start to slide. The critical force when wheel slipping would start to occur is equal to the adhesion coefficient of the wheels multiplied by the weight on these wheels:

\[ F_A = k_A \cdot m_{Loco} \cdot g \]
where \( k_A \) is the adhesion coefficient. Thus, the traction a locomotive finally achieves is given by

\[
F_L(t) = \min(F_T(t), F_A).
\]

The range for adhesion in railways starts from 0.1 under very bad weather conditions up to around 0.45 under good weather conditions. For our computations we consider an adhesion coefficient of 0.35. Figure B.2 illustrates the influence of adhesion on the applicable traction at different speeds.

Dynamic braking force

In most modern electric locomotives traction motors can act as generators transforming the kinetic energy into current, thereby exerting a wheel-independent braking force which is not limited by adhesion. The dynamic braking force works best for medium speeds, but at low speeds the magnetic field generated by the motors becomes rather weak and the dynamic braking ineffective, while at high speeds limitations arise from electric equipment (commutator, maximum voltage, current density). In the past decade a great deal of improvement has been made in the design of traction motors to achieve a more stable braking force and additionally, allow dynamic braking for lower velocity. In modern train compositions electrical brakes are used for regular service braking, while pneumatic or mechanical brakes are applied to bring a train to a complete stop. Wendel (2003) describes in detail how conventional electrical brakes can be modelled, but many parameters for these models are often not known or not available to the public. Thus we decided to model the electrical braking force using the few available characteristic maximum, electrical braking force curves of some locomotives. Figure B.3 exemplifies a maximum electrical braking force curve: At low velocities the maximum electrical braking force is limited by the force which can be generated by the braking cylinders. At higher velocities
the brake heats up, and the maximum electrical braking force is additionally limited to prevent overheating.

![Figure B.3: A maximum electrical braking force curve of a German ICE 1 locomotive. Source Wende (2003, page 239)](image)

**Energy recuperation**

Increasing global ecological awareness and railway operators financial awareness towards energy consumption motivates the use of dynamic braking for energy recuperation (so-called *regenerative braking*). In regenerative braking the generated current is sent back to the power supply, while in the so-called *rheostatic braking* the produced electrical energy is dissipated as heat by a bank of on-board rheostats (i.e. variable resistances). We assume in our computations that a constant factor \( k_R \in [0, 1] \) of the energy generated by the applied electrical braking force can be recuperated.

**Mechanical braking force (B.1f)**

A mechanical brake is either a braking pad brake or a wheel disc brake, but most modern passenger trains are operated today with wheel disc brakes. In order to compute the mechanical braking force applied by a braking pad or wheel disc brake, one has to model first the force of the producer of the braking force. This force is generated by a braking cylinder and is dependent on the applied mechanical braking control \( u_M \). The generated force is then carried over as either pad force to the braking pads or as disc force to the wheel discs. Finally, the mechanical braking force of a braking pad (or wheel disc) can be computed from the pad force (or disc force) by taking friction loss into account. Wende (2003) models the mechanical braking force of a vehicle \( F_M^V \) with wheel disc brakes roughly as follows:

\[
F_M^V = \mu_{Disc} F_{Disc}^V \frac{r_{brake}}{r_{wheel}}
\]
where $\mu_{\text{Disc}}$ is the friction coefficient of the wheel discs, $F_{\text{Disc}}^V$ the complete disc force, $r_{\text{brake}}$ the radius of the friction area of one braking disc and $r_{\text{wheel}}$ the radius of a running wheel. Modelling and determining the friction coefficients and the disc forces is a very tedious task (we refer the reader to Wende, 2003, Chapter 5) and thus we decided to model the applied mechanical braking force with the following allowedly simplistic model:

$$F_M = u_M \cdot F_{M}^{\text{Max}}$$

where $F_{M}^{\text{Max}}$ is the for each train separately estimated maximal mechanical braking force.

**Resistances** (B.1g), (B.1h), (B.1i)

From the various forms of resistances which influence the movement of a train we consider the following ones in our computations:

- **Propulsion resistance** $R_P(v)$ is the sum of rolling resistance and aerodynamic drag. Rolling friction occurs at axle bearings and at the contact zones between the wheels and the rails and is proportional to speed, while aerodynamic drag is proportional to the square of the speed. Differing shapes, material and designs of rolling stock as well as differing track conditions due to curvature, tunnels and weather conditions ultimately lead to empirical formulas for estimating the propulsion resistance of a train. Almost every national Railway company has its own empirical formula but they conventionally take the form

$$R_P(v) = A + Bv + Cv^2.$$  

In practice, the coefficients of such empirical formulas are determined by correlating measured values (e.g. friction coefficients) in order to get adequate estimates of the propulsion resistance (Radosavljevic, 2006). Cole assembled in Iwnicki (2006) different illustrative empirical formulas for propulsion resistances which we depicted in Figure B.4 for passenger rolling stock and in Figure B.5 for freight rolling stock.

- **Grade resistance** is due to the gravitational force between the earth and the train. When we consider a train as a point mass it can be computed as

$$R_G(x) = m \cdot g \cdot \sin(\beta(x)),$$

where $g$ is the earth acceleration and $\beta(x)$ is the grade angle at position $x$. Obviously, the grade resistance is negative for downhill movements and positive for uphill movements. The grade angle is usually not known at every position of the train route, and instead averaged values for grade angles along the tracks are used in computations.

- **Track curvature resistance** occurs in curved railway tracks. It is foremost proportional to the curvature of the track, but additionally depends on rolling stock, track
B.2 Mathematical model

Figure B.4: Propulsion resistances of passenger rolling stock

Figure B.5: Propulsion resistances of freight rolling stock
profile, rail lubrication, gauge, speed, etc.. According to Hay (1982) the commonly applied empirical formula for estimating the curvature resistance is

\[ R_C(x) = \frac{6.116}{\gamma(x) \cdot m} \]

where \( \gamma(x) \) is the curve radius in meters at position \( x \). Similar to grade angles, the curvature radii are usually only known as averaged values along the tracks.

**Speed limits (B.1j)**

Speed limits along the train route guarantee a safe operation of the train. These speed limits depend on track curvature, downgrade, signals, rolling stock composition of the train, construction work, etc. For our mathematical model we assume that the lowest (maximum) speed limit at every position is given by the function \( V(x(t)) \), thus

\[ 0 \leq v(t) \leq V(x(t)) \]

**Passenger comfort and freight safety (B.1k)**

To guarantee a certain level of comfort to the passengers and in case of freight train a safety level for transported goods, railway operators additionally regulate the maximum allowed acceleration \( \bar{a} \) and deceleration \( a \) of some of their trains, thus restricting the allowed control:

\[ a \leq \frac{F_T(t) - F_D(t) - F_M(t) - R(v, x, \gamma)}{\rho \cdot M} \leq \bar{a} \]

**Boundary conditions (B.1l), (B.1m), (B.1n), (B.1o)**

We want to compute a speed profile for a train which originates at a given time \( t_s \), a given location \( x_s \) and a given speed \( v_s \). The train shall additionally arrive at a given time \( t_e \), a given destination \( x_e \) and a given specific speed \( v_e \). We assume a train route has already been fixed before and thus it is enough to consider the one-dimensional distance for locating the train. The starting boundary conditions are

\[ x(t_s) = x_s, \]
\[ v(t_s) = v_s. \]

Similarly, the end boundary conditions are

\[ x(t_e) = x_e, \]
\[ v(t_e) = v_e. \]
B.3 Solution approach: Dynamic Programming

Dynamic programming (DP) has shown in past works (Sidelnikov (1965), Erofeyev (1967) and Franke et al. (2002)) to be an easily implementable and robust approach for solving energy efficient train control problems. In this section we provide some fundamental theoretic background on dynamic systems and how they can be solved using dynamic programming. This brief introduction summarizes the sections 1.1-1.3 of Bertsekas (2007). Consider a dynamic system given by

\[ x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \ldots, n-1, \]

additive cost: \[ c_k(x_k, u_k), \]

total cost: \[ c_n(x_n) + \sum_{k=0}^{n-1} c_k(x_k, u_k), \]

policy: \[ \pi = \{\mu_0, \mu_1, \ldots, \mu_{n-1}\} \]

where \( \mu_k \) maps \( x_k \) into control \( u_k = \mu_k(x_k) \); the DP-algorithm is based on a simple idea, the principle of optimality.

**Principle of optimality**

Let \( \pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{n-1}^*\} \) be an optimal policy for the basic problem, and assume that when using \( \pi^* \), a given state \( x_i \) occurs at position \( i \). Consider the sub-problem whereby we are at \( x_i \) at position \( i \) and wish to minimize the “cost-to-go” from position \( i \) to position \( n \)

\[ c_n(x_n) + \sum_{k=i}^{n-1} c_k(x_k, \mu_k(x_k)). \]  

Then the truncated policy \( \{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{n-1}^*\} \) is optimal for this sub-problem.

Now the DP-algorithm for the basic problem can be stated; for more details and demonstrations, refer to Bertsekas (2007).

**DP - algorithm**

For every initial state \( x_0 \), the optimal cost \( J^*(x_0) \) of the basic problem is equal to \( J_0(x_0) \), given by the last step of the following algorithm, which proceeds backward in position from period \( n-1 \) to the initial position:

\[ J_n(x_n) = c_n(x_n), \]

\[ J_k(x_k) = \min_{u_k} \{c_k(x_k, u_k) + J_{k+1}(f_k(x_k, u_k))\}. \]

Furthermore, if \( u_k^* = \mu_k^*(x_k) \) minimizes the right side of eq. (B.4) for each \( x_k \) and \( k \), then the policy \( \pi^* = \{\mu_0^*, \ldots, \mu_{n-1}^*\} \) is optimal.

---

1This section is based on the supervised master thesis Balmelli (2009)
Results regarding train separation constraints

In this section, we highlight some of the computational results regarding the train separation constraints which we published in Caimi et al. (2010). In that work we compared in several case studies for central station areas of Switzerland different model formulations for the static train dispatching problem. In the summarized results two formulations are distinguished:

- A simple pairwise formulation of conflicts between blocking time stairways. It is referred as the conflict graph (CG) formulation and follows the approach of Zwan- eveld et al. (1996).

- A clique based formulation for conflicts between several blocking time stairways occurring at resources. It is referred to as the resource tree conflict graph (RTCG) model, and follows the same train separation principle of the STD model presented in Section 3.3.4.

The comparison of these two models provides insight in the different model sizes and most importantly, on the model performances in finding optimal or near optimal solutions to problem instances. Specifically, the experimental comparison of the CG model and the RTCG model demonstrates the benefit of bundling the conflicts into cliques. We refer the reader to the paper Caimi et al. (2010) for a detailed description of the models, different scenarios and the test environment.

Clique sizes

Table C.1 displays the conflict-clique sizes of the RTCG model for the considered scenarios. We can observe that they are on average quite large, particularly for dense scenarios. The maximum can also be really huge, meaning that for each scenario there is at least one fixed block (or resource) and one point in time where many train movements could allocate it, making it de facto a bottleneck in the railway network. Thus we can expect
Appendix C: Results regarding train separation constraints

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Average</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>11</td>
<td>2</td>
<td>127</td>
</tr>
<tr>
<td>Medium</td>
<td>75</td>
<td>2</td>
<td>851</td>
</tr>
<tr>
<td>Big</td>
<td>65</td>
<td>2</td>
<td>1170</td>
</tr>
</tbody>
</table>

Table C.1: Conflict clique sizes in the RTCG model for the three scenarios.

From this table a significant speed-up in solving the corresponding optimization models should be expected.

Model sizes

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Model</th>
<th>Before Preprocessing</th>
<th>After Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Binary</td>
<td>Conflict</td>
</tr>
<tr>
<td>small</td>
<td>RTCG</td>
<td>23’477</td>
<td>1’999</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>661</td>
<td>27’897</td>
</tr>
<tr>
<td>medium</td>
<td>RTCG</td>
<td>319’142</td>
<td>4’401</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>6’820</td>
<td>6’822’627</td>
</tr>
<tr>
<td>large</td>
<td>RTCG</td>
<td>535’264</td>
<td>9’773</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>18’191</td>
<td>10’568’742</td>
</tr>
</tbody>
</table>

Table C.2: Statistics of the two different ILP formulations of three scenarios, before and after the (standard) preprocessing of CPLEX.

Table C.2 gives information on the size of the resulting integer linear program for the two scenarios. We observe that the number of variables and the number of constraints strongly depend on the type of model. Since the RTCG model is based on a tree formulation as opposed to the CG model, more variables are required. This is in fact an issue, which is resolved by the STD model formulation proposed in this thesis. The differences in the amount of constraints is explained by the different context in each model. In the CG, a constraint is built between two conflicting blocking time stairways and between blocking time stairways which belong to the same train movement. In the RTCG, concurrent resource allocations are captured by the maximal conflict cliques. For small scenarios, like Lucerne 2006, the RTCG models uses already considerably less constraints. For medium size scenarios like Bern East, we can observe that the number of constraints in the RTCG is much smaller than for the CG model. This is due to the fact that for each conflict group we have a linear number of constraints in the RTCG model but a quadratic number in the CG models, due to the pairwise consideration of conflicts. For large scenarios like Bern 2008, RTCG has by far a smaller number of constraints. We can say that the denser the scenario is, the larger is the benefit of the RTCG approach.

Table C.2 also illustrates the effects of the models on the size of the ILP, before and after the preprocessing step of the ILP solver. In the preprocessing phase, the solver tries...
to eliminate redundant information in the problem description and, if possible, to tighten the formulation using logic implications, e.g., using equality constraints to eliminate variables. We can observe that the RTCG model has the fewest number of constraints, with beneficial effects on the memory requirement. This difference in the amount of constraints needed by the models grows with the density of the ILP formulation. Moreover, we can analyse the effects of the preprocessing on the problem size for the different scenarios. One can see that for the small scenario the preprocessing is very effective for both models, leading to very small ILPs to solve. For both Bern scenarios, the preprocessing for the CG model is very weak, due to the fact that the model lacks in structure. The RTCG is already very effective in describing the conflicts so that the effect of preprocessing remains limited. The number of variables, however, is reduced considerably, probably by resolving many flow constraints of the ILP formulation.

### Processing times for solving the optimization models

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Model</th>
<th>Model Creation</th>
<th>ILP Creation</th>
<th>Pre-processing</th>
<th>Root-Relaxation</th>
<th>Branch &amp; Cut</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>RTCG</td>
<td>&lt;&lt;1</td>
<td>&lt;&lt;1</td>
<td>&lt;&lt;1</td>
<td>&lt;&lt;1</td>
<td>&lt;&lt;1</td>
<td>&lt;&lt;1</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>&lt;&lt;1</td>
<td>&lt;&lt;1</td>
<td>&lt;&lt;1</td>
<td>&lt;&lt;1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>RTCG</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>&lt;&lt;1</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>258</td>
<td>5</td>
<td>187</td>
<td>1003</td>
<td>4’973</td>
<td>6’426</td>
</tr>
<tr>
<td>large</td>
<td>RTCG</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>&lt;&lt;1</td>
<td>8</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>432</td>
<td>8</td>
<td>133</td>
<td>1’783</td>
<td>3’743</td>
<td>6’099</td>
</tr>
</tbody>
</table>

Table C.3: Processing times (in seconds) for the various computation steps of the two train dispatching models (CG and RTCG).

Table C.3 lists CPU times for the different computation steps from model creation up to computing the solution for the primary objective of maximizing the number of scheduled trains. For small scenarios, there are basically no noticeable differences, as both models are very fast in all steps. For larger scenarios, one can observe that the RTCG model has enormously beneficial effects on both model creation and solution compared to the CG approach. This table illustrates well the impact of the constraints’ strength in the RTCG model: the linear relaxation is much tighter with strong effects on the branch-and-cut procedure applied by the solver, and therefore, ultimately, on the computation time. The CPU time as a function of the number of binary variables will probably increase drastically also for the RTCG model at some point due to the complexity of the scheduling problems in main station areas, which was proven to be NP-hard in very similar settings [Kroon et al. (1997)]. Fortunately, this point seems to be further away from the problem sizes that we need to solve for realistic instances of large and dense scenarios and the reduction in the number of variables by the STD model might help keeping the computation times small. Nevertheless, the fast computation times of these results were a critical motivator for starting this thesis and for considering the dispatching of trains with the help of mathematical models in an on-line setting.
Appendix C: Results regarding train separation constraints

Solution quality

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Model</th>
<th>Integality gap [%]</th>
<th>Branch-and-bound nodes</th>
<th>Cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>trains</td>
<td>before</td>
<td>after preprocessing</td>
</tr>
<tr>
<td>small</td>
<td>RTCG</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>12</td>
<td>47.8</td>
<td>0</td>
</tr>
<tr>
<td>medium</td>
<td>RTCG</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>15</td>
<td>48.4</td>
<td>7.3</td>
</tr>
<tr>
<td>large</td>
<td>RTCG</td>
<td>67</td>
<td>3.1</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>CG</td>
<td>67</td>
<td>31.9</td>
<td>8.5</td>
</tr>
</tbody>
</table>

Table C.4: Solution quality and steps of the MIP solver. The symbol + after the number in the B&B columns denotes a solution generated by an heuristic process.

Table C.4 shows the quality of the generated solution for the three considered scenarios. For the small scenario, both models yield an optimal solution (0% integrality gap) almost instantaneously, as the relaxed solution with preprocessing had already the optimal value with a directly found integer solution. For the medium scenario, only the linear relaxation of the RTCG model already delivered the optimal objective value, so that the solver was able to find an integer optimal solution very quickly using heuristics only. For the CG model, some branch-and-bound nodes as well as many cuts were necessary to overcome the integrality gap of around 7% obtained after the preprocessing, leading to quite long computation times. For the large scenario, the linear relaxation of both models generated an integrality gap. However, the gap for the RTCG was much smaller and the solver was still able to find the optimal solution during the heuristic pre-solve phase. This procedure failed for the CG, where branch-and-bound nodes and cuts were necessary to find the optimal solution, with correspondingly longer CPU times.

It is also instructive to investigate the difference between the solution of the pure linear relaxation, i.e., when all preprocessing functionality of the solver is switched off, and the optimal integer solution, which gives a better insight into the strengths of the different model structures. For the small and medium size scenarios, the linear relaxation of the CG model is quite poor. The RTCG model was able to generate directly an integer solution, demonstrating the very strong structure of the RTCG model, i.e, its clique constraints. For the large scenario, both models generate an integrality gap even after the preprocessing phase of the solver. But also here, the RTCG model yields a much better linear relaxation compared to the other model.
Appendix D

Algorithm 3.1: Complexity analysis and correctness

Complexity analysis

Theorem D.1 (Time complexity of Algorithm 3.1). Algorithm 3.1 for computing $\Omega^{t,t'}_{p,p'}$ can be implemented with a time complexity $O(k \log(k))$, where $k = |A_{t,p}| + |D_{t',p'}|$. 

Proof. It is well known (see Knuth [1998]) that the time complexity for sorting a list with $n$ items is $O(n \log(n))$. Our algorithm sorts in lines 2 and 3 the two lists $A_{t,p}$ and $D_{t',p'}$, both of which contain at most $k$ items. The time complexity of these two steps is therefore $O(k \log(k))$. The next phase of the algorithm consists of the creation of the pointers $p(v)$ (lines 4 to 8). The for-loop creates $|V| = |D_{t',p'}| < k$ pointers with the help of the pointer $p_{prev}$. The iterative $\text{argmin}$ operations (Line 6) and the update of $p_{prev}$ in Line 7 correspond to walking through the list $U$, which has a time complexity $O(|U|)$. Thus the creation of the pointers has an overall time complexity of $O(|V| + |U|) = O(k)$. The last part of the algorithm iterates over the list $V$ (lines 9 to 19) and provided that it is possible (lines 10 and 16, both conditions require $O(1)$ for their evaluation), determines in each iteration two maximal incompatible sets $(U,V)$ with the help of the pointer $p(v_i)$ and the current iteration counter $i$. Instead of storing in such an iteration the subsets $U \subseteq U$ and $V \subseteq V$ in form of physical copies into the set $\Omega^{t,t'}_{p,p'}$ (lines 11,12,13,17 and 18), it is in fact enough to store $i$ and $p(v_i)$ with a time complexity of $O(1)$. Thus, this last part executes at most $k$ iterations with a workload of $O(1)$ and thus, has a time complexity of $O(k)$. As $O(k) < O(k \log(k))$ the overall time complexity of the algorithm is therefore $O(k \log(k))$. 

Theorem D.2 (Space complexity of Algorithm 3.1). Algorithm 3.1 for computing $\Omega^{t,t'}_{p,p'}$ can be implemented with a space complexity $O(k)$, where $k = |A_{t,p}| + |D_{t',p'}|$. 


Proof. The algorithm creates two sorted lists $U$ and $V$, as well as the pointer $p(v)$ which require $O(k)$ space. An efficient implementation of the algorithm does not create physical copies $(U,V)$ from subsets of $U$ or $V$ in the last part of the algorithm (lines 9 to 19), but rather stores in each iteration the iteration counter $i$ and the current pointer $p(v_i)$, which again requires $O(k)$ space. The overall space complexity is therefore $O(k)$. □

Proof of correctness

Proof. In order to show the correctness of the algorithm, we show first that the algorithm lists all maximal incompatible pairs (completeness) and then show, that all listed pairs are maximal incompatible (validity). Both parts of the proof are done by contradiction:

Completeness: Assume there is a maximal incompatible pair $(U,V)$ which was missed by the algorithm. Without loss of generality we can assume that both $U$ and $V$ are sorted subsets of $U$ or $V$, respectively. Let $\hat{u}$ be the blocking time stairway with the earliest arrival time in $U$ and $\hat{v}$ the blocking time stairway with the latest departure time in $V$. Then, the pointer $p(\hat{v})$ created by the algorithm must necessarily point to the index of $\hat{u}$: As the pointers are only created for incompatible pairs, a lower value for $p(\hat{v})$ than this index would violate the maximality condition of the two sets $U$ and $V$. A higher value for $p(\hat{v})$ than this index on the other hand, would imply that $\hat{v}$ and $\hat{p}$ are compatible, also leading to a contradiction. The algorithm can only have missed this set if either the next pointer points again to the index of $\hat{u}$, or if $p(\hat{v})$ points to the blank $|U|+1$. The former is not possible, as it would violate the maximality condition of $(U,V)$ and the latter is not possible, since $|U|+1$ is not a valid index of $U$. And thus, the assumption is contradicted.

Validity: Assume the algorithm lists a pair $(U,V)$ in iteration $i$ which is not maximal incompatible, i.e. there is either a $u^* \in U$ and $v^* \in V$ which are compatible or the pair $(U,V)$ is not maximal. Let us consider the first case: Let $\hat{u}$ denote the blocking time stairway in $U$, which is indexed by the pointer $p(v_i)$ belonging to iteration $i$. Furthermore, let $\hat{v}$ denote the blocking time stairway $v_i \in V$, which is indexed by the iteration counter $i$. We know that $\hat{u}$ is the blocking time stairway with the earliest arrival time among all blocking time stairways in $U$ due to the construction of $U$ by the algorithm (lines 11 and 17). Similarly, $\hat{v}$ is the blocking time stairway with the latest departure time among all blocking time stairways in $V$. We also know that $\hat{u}$ and $\hat{v}$ are incompatible, based on the way we constructed the pointer $p(v_i)$ (Line 6) and on the two conditions for building a new pair $(U,V)$ (lines 10 and 16). If $u^*$ and $v^*$ are compatible blocking time stairways, the following holds for their time difference $\gamma(v^*) - \gamma(u^*) \leq m_{pl,pr}^{i.t} < \gamma(\hat{v}) - \gamma(\hat{u})$. Since $m_{pl,pr}^{i.t} \in \mathbb{R}_+$ this relationship is only possible, if either $\gamma(v^*) > \gamma(\hat{v})$ or $\gamma(u^*) < \gamma(\hat{u})$. The options either contradict with the definition of $\hat{v}$ or with the definition of $\hat{u}$. Thus, $U$ and $V$ have to be incompatible and contradicting the assumption.

In the second case, we have to assume that there is a $u^* \in U$ with $u^* \notin U$ (or a $v^* \in V$ with $v^* \notin V$) that could be added to $U$ (or $V$) while keeping the incompatibility of the two sets. Let us define $\hat{u}$ and $\hat{v}$ as in the first case. In the way of how $U$ and $V$ are constructed by the algorithm, such a candidate $u^*$ (or $v^*$) would necessarily be compatible with $\hat{v}$ (or $\hat{u}$), thereby contradicting the assumption. □
Appendix E

Consistent platform usage

The following two theorems guarantee that the trains arrive and depart at the same platform and in the correct chronological order:

**Theorem E.1 (Departure directly after Arrival).** Assume the Inequalities (3.3.3), (3.3.3), (3.6) and the Equalities (3.5) hold for an assignment $\pi$ of blocking stairways. If a train $\hat{t}$ arrives at platform $\hat{p}$ with the assigned arrival blocking-stairway $\hat{b} \in A_{\hat{t},\hat{p}}$, that is $\pi_{\hat{b}} = 1$, and with arrival time $\hat{\alpha} = \alpha(\hat{b})$ then

- $b' = \text{argmin}\{\gamma(b) \ | b \in B_{\hat{t},\hat{p}}, \pi_b = 1, \gamma(b) > \hat{\alpha}\}$ exists and is unique.
- Furthermore, $b' \in D_{\hat{t},\hat{p}}$ that is the next assigned blocking-stairway $b'$ in chronological order which required platform $\hat{p}$ after the arrival of train $\hat{t}$ at time $\hat{\alpha}$ is the departure blocking-stairway of the same train $\hat{t}$.

**Proof.** As the departure has to take place at the same platform (Eq. (3.5)) there exists $b^0 \in D_{\hat{t},\hat{p}}$ with $\pi_{b^0} = 1$.

As the departure has to happen after the arrival (Ineq. (3.6)) $b^0$ has to be compatible with $\hat{b}$, i.e. $\delta(b^0) > \hat{\alpha}$ holds. So we know, that the departure happens after the arrival of the train.

What is left to show, is that no other platform event can happen between the arrival and departure. This can be done by contradiction:

Assume that there exists another assigned blocking stairway $b^1 \neq b^0$ with a platform event time in between the arrival and departure time of train $\hat{t}$: Assume $\exists b^1 \in B_{\hat{t},\hat{p}}$ with $\pi_{b^1} = 1, \hat{\alpha} < \gamma(b^1) \leq \delta(b^0)$. We have to consider two cases in order to show that such a $b^1$ can not exist:

- Case $b^1 \in A_{\hat{t},\hat{p}}$:
  By the Eq. (3.5) there exists $b^2 \in D_{\hat{t},\hat{p}}$ with $\pi_{b^2} = 1$ and by the Ineq. (3.6) it holds that $\alpha(b^1) < \delta(b^2)$. Let $\Phi := A_{\hat{t},\hat{p}} \cup A_{\hat{p},\hat{t}}$ and $\Pi := D_{\hat{t},\hat{p}} \cup D_{\hat{p},\hat{t}}$. Let us consider
Appendix E: Consistent platform usage

the number of trains at platform $\hat{p}$ at time $\alpha(b^1)$:

$$\sum_{\alpha(b) \leq \alpha(b^1)} \pi_b - \sum_{\delta(b) < \alpha(b^1)} \pi_b + \sum_{\alpha(b) \leq \alpha(b^1)} \pi_b - \sum_{\delta(b) < \alpha(b^1)} \pi_b \geq 2$$

From Ineq. (3.3.3) it holds that $\sum_{\alpha(b) \leq \alpha(b^1)} \pi_b = 2$ since both arrivals of train $\hat{t}$ and train $t^0$ happen before $\alpha(b^1)$. Furthermore $\sum_{\delta(b) < \alpha(b^1)} \pi_b = 0$ since Ineq. (3.3.3) holds and both departures of these trains happen after $\alpha(b^1)$. The right part (denoted by the right under-brace) sums up how many of the remaining trains apart from the trains $\hat{t}$ and $t^0$ are at the platform at time $\alpha(b^1)$. Again by using the Eq. (3.5) and the Ineq. (3.3.3) this difference is at least 0. Therefore the assumption of the existence of such an arrival blocking stairway $b^1$ violates the Inequalities (3.7), which is in contradiction to the assumptions of the theorem.

- Case $b^1 \in D_{t,\hat{p}}$: By Ineq. (3.5) and Ineq. (3.6) $\exists b^2 \in A_{t,\hat{p}}$ with $\pi_{b^2} = 1$ and $\alpha(b^2) < \delta(b^1)$. Let $\alpha^* = \max\{\hat{\alpha}, \alpha(b^2)\}$. Then by the same arguments as in the above case the number of trains at platform $\hat{p}$ at time $\alpha^*$ is at least two:

$$\sum_{\alpha(b) \leq \alpha^*} \pi_b - \sum_{\delta(b) < \alpha^*} \pi_b \geq 2$$

Therefore the assumption of the existence of such a departure blocking stairway $b^1$ violates the Inequalities (3.7), which is in contradiction to the assumptions of the theorem.

As there is no blocking stairway with platform event time in between the arrival and departure of train $\hat{t}$, it holds that the blocking stairway which we are looking for ($b' = \arg\min\{\gamma(b)\} b \in B_{\hat{p}}, \pi_b = 1, \gamma(b) > \hat{\alpha}\}$, is in fact $b^0$. This shows the existence and the uniqueness of $b'$, as well as that $b'$ is the departure blocking stairway of train $\hat{t}$: $b' \in D_{t,\hat{p}}$.

Theorem E.2 (Arrival directly before Departure). Assume the Inequalities (3.3.3) (3.3.3) (3.6) (3.7) and the Equalities (3.5) hold for an assignment $\pi$ of blocking stairways. If a train $t$ departs at platform $\hat{p}$ with the assigned departure blocking-stairway $\hat{b} \in D_{t,\hat{p}}$, that is $\pi_{\hat{b}} = 1$, and with departure time $\hat{\delta} = \delta(\hat{b})$ then

- $b' = \arg\max\{\gamma(b)\} b \in B_{\hat{p}}, \pi_b = 1, \gamma(b) < \hat{\delta}\}$ exists and is unique.

- Furthermore, $b' \in A_{t,\hat{p}}$, that is the last assigned blocking-stairway $b'$ in chronological order which required platform $\hat{p}$ before the departure of train $\hat{t}$ at time $\hat{\delta}$ is the arrival blocking-stairway of the same train $\hat{t}$.

The proof is similar to the proof of Theorem E.1
Personal Data
Name: Martin Fuchsberger
Date of birth: December 29, 1981
Place of birth: Lucerne, Switzerland
Nationality: Swiss
Marital status: single

Education
2009–2012 Doctoral student at the Institute for Operations Research, D-Mathematics, ETH Zurich
Dissertation: ”Algorithms for railway traffic management in central railway station areas” under the supervision of Prof. Dr. Hans-Jakob Lüthi and in collaboration with Swiss Federal Railways
Graduated as Dr. sc. ETH
2001–2007 Master in computer science at ETH Zurich
1994–2001 High School in Lucerne, Switzerland

Experience
2007–2011 Scientific assistant in research and teaching at the Institute for Operations Research, ETH Zurich
2005–2007 Software Engineer at Visual Cognition Research Group, University Zurich