

DISS. ETH NO. 22548

**ALGORITHMIC DECISION SUPPORT
FOR THE CONSTRUCTION OF
PERIODIC RAILWAY TIMETABLES**

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES OF ETH ZURICH
(DR. SC. ETH ZURICH)

presented by

SABRINA HERRIGEL-WIEDERSHEIM

MSc in Mathematics, ETH Zurich

born on 31.08.1985

citizen of Zurich

accepted on the recommendation of

Prof. Dr. Ulrich Alois Weidmann, examiner
Prof. Dr. rer. nat. habil. Karl Nachtigall, co-examiner
Prof. em. Dr. Hans-Jakob Lüthi, co-examiner
PD. Dr. Marco Laumanns, co-examiner

2015

Acknowledgments

As a combination of operations research, railway systems and computer science the suggested topic for this PhD thesis could not have fitted to my interests better. I would like to thank Prof. Ulrich Weidmann and Prof. Hans-Jakob Lüthi for the opportunity to write this thesis at their institutes. During my first 1.5 years at the Institute for Operations Research, I was able to extend my knowledge about operations research, precise argumentation and writing. In the following years, I was introduced to the applied world of railway planning and operations at the Institute for Transport Planning and Systems. I enjoyed the projects and collaborations close to practice a lot.

A special thank goes to Prof. Dr. Ulrich Weidmann, as the main advisor of this thesis, who brought a lot of practical experience and industrial connections into the work. I greatly appreciated his continuous, kind and encouraging support during the past few years. Although he was contributing in many different projects and PhD theses, he always had time for discussions and advises. Besides railway systems, he thought me a lot about efficient working and collaborating with practice.

Leading my thesis the first 1.5 years at the Institute for Operations Research, Prof. em. Dr. Hans-Jakob Lüthi played another main role for the successful progress of my thesis. I am very grateful to Prof. em. Dr. Hans-Jakob Lüthi for willing to employ me, to referee this thesis and for his patient advise and fruitful discussions.

I am also very grateful to Prof. Dr. rer. nat. habil. Karl Nachtigall from TU Dresden willing to supervise this thesis, as the external referee. I greatly appreciated the visit at his research group and all interesting discussions concerning PESP and its application in practice. At this place, I would also like to thank Dr. Jens Opitz, Peter Grossmann and further members of his group for all exchanges and the presentation of the software TAKT.

As a continuance supervisor, third co-supervisor and as a main impulse for the progress of this thesis, I am very much indebted to PD. Dr. Marco Laumanns. Several times he motivated me to publications and contributions at conferences and offered me a lot of support for writing papers, presentations and the whole thesis. Although he was involved in a lot of different projects, he had always time for fruitful discussions, listening to my doubts and giving me encouraging support. His contribution for the successful outcome of the work was essential. At this point, I further like to thank Dr. Jacint Szabo, who works at the same research group as PD. Dr. Marco Laumanns. He also contributed a lot, especially to Chapter 6.

An applied PhD thesis is not really applied without connections to industry. At this place I would like to thank the Swiss Federal Railways (SBB) and especially Dr. Helga

Labermeier, Médard Fischer and Dr. Peter Grossenbacher for all fruitful discussions and the supplied data. I really enjoyed this insight into the more practical life. In addition to SBB, I would also like to thank TraffIT Solutions, especially Dr. Dan Burkolter and Burkhard Franke for offering me to use the software OnTime to evaluate my models and timetables.

I would further like to thank Prof. Leo Kroon, Dick Middelkoop and Dr. Birgit Heydenreich organizing a visit at ProRail, including a lot of very fruitful and interesting discussions on the automation of timetabling, their software DONS and all mathematical challenges in connection to PESP.

I want to thank all my former colleagues of both institutes also collaborating to similar railway projects, especially to Dr. Martin Fuchsberger, Dr. Kaspar Schüpbach, Dr. Gabrio Caimi and Dr. Dan Burkolter for their introduction and a lot of additional support. I'm also grateful to Prof. Dr. Rico Zenklusen, Dr. David Adjashvili and Dr. Christian Wagner for their ideas and discussions to Chapter 5. Furthermore, I want to thank to Betty Fahrni and Dr. Christian Eichenberger for proofreading my thesis.

I want to address an additional thank to all my present and former colleagues at both research institutes and further international colleagues I had the opportunity to get to know at conferences during the last years. You all made the last years a very pleasant and interesting time. A special thank goes at this point to my office colleagues Dr. Martin Fuchsberger, Dr. Olga Fink and Christian Marti. They were always there for discussions, new ideas, support, or simply a good laugh.

Ein ganz herzlicher Dank geht an alle meine engsten Freunde und die Familie, insbesondere meine Eltern, mein Mann Roger und mein Sohn Levin, die während allen Jahren immer hinter mir standen, mich unterstützten und für mich da waren.

Abstract

This thesis addresses the problem of solving large railway timetabling problems using algorithmic methods. With the increasing demand for better and more frequent services, for higher capacity utilization and for improved reliability, railway timetabling problems become more and more complex. Algorithmic decision support is one promising way to cope with this increasing complexity, and the continuous progress of operations research methods offers a significant potential for the railway industry.

The approach of this thesis concentrates on algorithms for the construction of periodic railway timetables during long- and mid-term planning. As an input, functional requirements and restrictions of the infrastructure have to be described at a model granularity suitable for the given planning stage. The algorithmic approach then creates a periodic timetable with the same model granularity and optimizes the timetable according to a given objective. Compared to a manual timetable construction, the algorithmic approach allows to compare different timetable scenarios in a shorter time. Possible modifications of the infrastructure or the service intention and their influence on the overall timetable can therefore be tested and evaluated more efficiently and also in more detail.

The algorithms studied in this thesis are based on the so called Periodic Event Scheduling Problem (PESP), a mathematical model which proved to be suitable to automate the construction of periodic railway timetables already several times in research and practice. To solve the model there exist different mathematical methods. This thesis summarizes them and shows advantages of a method based on Mixed Integer Linear Programming (MILP). Compared to other solution methods it allows a direct optimization and with this several further advantages concerning a more efficient automation and a better solution quality. However there exists no practical experience of the method for larger timetabling problems and there even can be found the conjecture that the solution method is not suitable for large scale problems.

The algorithms developed in this thesis are based on the so-called Periodic Event Scheduling Problem (PESP), a mathematical model which has become the standard approach for algorithmic construction of periodic railway timetables and has been used successfully already several times in research and practice. Different mathematical methods have been proposed to solve timetabling problems formulated as a PESP. This thesis summarizes the different solution approaches and shows advantages of a method based on Mixed Integer Linear Programming (MILP). Compared to other solution methods, the MILP approach allows a direct optimization, and with this several further advantages concerning a more efficient automation and a better solution quality. However, there exists no practical experience of the method for larger timetabling problems and it has

been an open question so far whether and to what extent it is suitable for large-scale problems.

In this thesis this gap is filled by providing the missing experience of applying the MILP solution approach for a set of increasingly large timetabling problems for parts of the Swiss railway network. To profit from the method's advantages also for large-scale problems, an adaptation of the solution method is introduced. It allows to reduce computational complexity considerably, but still ensures good solution quality. To this end, the thesis provides three main contributions:

- Implementation of the defined algorithms including an automated model construction out of real data provided by Swiss Federal Railways (SBB).
- Extension of the models until a limit of computation time for the used algorithms is reached.
- Development of new methods for the acceleration of the given algorithms to solve and optimize also larger models.

To ensure operational feasibility for the given model granularity and a certain degree of timetable quality for our models, the resulting timetables are evaluated by the software OnTime. This way it can be ensured that the level of detail of our models is rich enough to represent realistic timetable scenarios. However, the models in this thesis are not constructed to study and evaluate a concrete timetable scenario but rather to represent different model sizes to study the computational performance of the proposed algorithms.

After an evaluation of different parameters for the algorithm and the used MILP solver, the best parameter setting is used to determine a computational limit of the given solution strategy for larger model sizes. Although using strong computer servers and state-of-the-art commercial MILP solvers, this limit can be reached for our largest models.

To accelerate the algorithms, two decomposition methods are proposed and studied. The first one is motivated from optimization theory. The PESP-graph corresponding to a PESP model is split into different subgraphs and used to define independent MILPs. Two iterative methods are introduced to coordinate the subproblems in order to find feasible global solutions of sufficient quality. The decomposition approach is successfully applied to a smaller test model splitting it into two subproblems. However, the generalization of the method for a larger number of subproblems or larger model sizes leads to several difficulties.

As a remedy to these shortcomings, a second decomposition idea is developed, motivated from manual planning practice, where timetables are often constructed sequentially, train by train. Traditional driving routes, grown over the history of railway operations, are often kept constant over the years and changed at the most by a few minutes to include new offers. From a mathematical point of view, such an approach is a so-called greedy method and can lead to arbitrarily bad solutions or even to infeasibility. Therefore, algorithms as mentioned above, use synchronous methods instead, thus planning and optimizing all

train lines simultaneously in one step. Their disadvantages are the high computational complexity. The second decomposition method defines a new approach using a compromise between a pure sequential planning approach and a complete synchronous approach. With this idea, the computation time can be reduced considerably and solution quality still can be kept on a high level sufficient for practical applications.

Zusammenfassung

Diese Dissertation befasst sich mit dem Lösen von grossen Eisenbahnfahrplanungsproblemen mit Hilfe von algorithmischen Methoden. Mit der zunehmenden Nachfrage nach besseren und häufigeren Angeboten, nach höheren Kapazitätsausnutzungen und einer besseren Zuverlässigkeit werden Fahrplanungsprobleme immer komplexer. Dabei spielen algorithmische Unterstützungen eine sehr erfolgversprechende Rolle zur Bewältigung dieser stetig wachsenden Herausforderungen. Gerade auch die schnelle Entwicklung vieler Methoden des Operations Research birgt ein bedeutendes Potential für die Eisenbahnindustrie.

Der Ansatz dieser Dissertation konzentriert sich auf Algorithmen zur Konstruktion von vertakteten Eisenbahnfahrplänen während der lang- und mittelfristigen Planung. Als Eingabe müssen funktionale Anforderungen und Infrastrukturrestriktionen beschrieben werden, deren Modellgranularität zum entsprechenden Planungsstand passt. Darauf berechnen und optimieren die Algorithmen einen periodischen Fahrplan mit derselben Modellgranularität, basierend auf einer gegebenen Zielfunktion. Im Vergleich zu der immer noch hauptsächlich verbreiteten manuellen Fahrplankonstruktion ermöglicht der Ansatz ein Vergleichen verschiedener Fahrplanszenarien in einer kürzeren Zeit. Das heisst mit den entsprechenden Algorithmen kann man auch Infrastrukturanpassungen oder Angebotserweiterungen und ihre netzweiten Einflüsse auf den Fahrplan effizienter und somit auch detaillierter untersuchen.

Die Algorithmen, die in dieser Dissertation entwickelt werden, basieren auf dem sogenannten "Periodic Event Scheduling Problem" (PESP). Dieses mathematische Modell wurde zum Standardverfahren zur Konstruktion von Taktfahrplänen und hat sich bereits mehrmals in der Praxis und Forschung bewährt. Zur Lösung von Fahrplanungsproblemen über PESP existieren unterschiedliche mathematische Methoden. Diese Arbeit gibt einen Überblick über mehrere solche Methoden und zeigt Vorteile für die Benutzung einer Methode basierend auf der gemischt ganzzahligen linearen Programmierung (MILP) auf. Im Vergleich zu anderen Lösungsmethoden erlaubt sie eine direkte Optimierung und damit mehrere weitere Vorteile betreffend einer effizienteren Automatisierung und höheren Lösungsqualitäten. Auf der anderen Seite existieren für diesen Lösungsansatz keine praktischen Erfahrungen für grosse Modelle und es blieb sogar die offene Frage ob und bis zu welchem Ausmass grosse Probleme gelöst werden können.

In dieser Arbeit wird diese fehlende Erfahrung durch die Anwendung der MILP-Lösungsmethode auf kontinuierlich grösser werdende Fahrplanungsprobleme des schweizerischen Eisenbahnnetzwerkes aufgeholt. Um jedoch trotzdem von den Vorteilen der Lösungsmethode auch für grosse Modelle zu profitieren werden Anpassungen der

Lösungsmethode eingeführt. Diese erlauben eine deutliche Reduktion der Rechenkomplexität, während gute Lösungsqualitäten trotzdem noch gesichert bleiben. Die Dissertation hat die folgenden drei Schwerpunkte:

- Implementierung der definierten Algorithmen unter Miteinbezug einer automatisierten Modellkonstruktion basierend auf Daten der Schweizerischen Bundesbahn (SBB).
- Vergrößerung der Modelle bis zur Erreichung einer Schranke der rechnerischen Umsetzbarkeit für die verwendeten Algorithmen.
- Entwicklung neuer Methoden für die Beschleunigung der gewählten Algorithmen, um auch grosse Modelle zu lösen und zu optimieren.

Um die betriebliche Durchführbarkeit für die gewählte Modellgranularität und einen gewissen Grad der Fahrplanqualität zu sichern, werden die erstellten Fahrpläne mit der Software OnTime evaluiert. Auf diesem Weg können wir sicher stellen, dass die Rechenkomplexität im Vergleich zu realen Fahrplanszenarien für unsere Modelle repräsentativ bleibt. Die Modelle dieser Arbeit werden jedoch nicht zur Untersuchung und zur Evaluation konkreter Fahrplanungszenarien konstruiert. Sie werden lediglich zu verschiedenen Grössen fixiert, um die Rechengeschwindigkeiten der eingeführten Algorithmen zu untersuchen.

Nach einer Evaluation verschiedener Parameter für die Algorithmen und den benutzten MILP-Solver, wird die beste Parameterwahl verwendet, um die Grenzen der Rechenbarkeit der Lösungsmethode für grosse Modelle zu bestimmen. Trotz Computer-Systemen mit hoher Rechenleistung und einem der zur Zeit besten MILP-Solver konnte diese Grenze für unsere grössten Modelle erreicht werden.

Um die Algorithmen für diese Modelle zu beschleunigen werden zwei Dekompositionsmethoden vorgeschlagen und studiert. Eine erste ist von der Optimierungstheorie abgeleitet. Der zu einem PESP-Modell gehörige PESP-Graph wird in zwei Subgraphen geteilt, welche zur Definition unabhängiger gemischt ganzzahliger linearer Programme genutzt werden. Über zwei eingeführte iterative Methoden werden die Subprobleme anschliessend koordiniert, um eine zulässige oder gar optimale globale Lösung zu finden. Die Dekompositionsmethoden werden auf ein kleineres Testmodell, welches in zwei Subprobleme geteilt wird, erfolgreich angewendet. Die Verallgemeinerung der Methode für eine höhere Anzahl Subprobleme und grössere Modelle führt jedoch zu mehreren Schwierigkeiten, die im Rahmen dieser Arbeit nicht gelöst werden können.

Ein Weg aus dieser Schwierigkeit bringt die Idee der zweiten Dekompositionsmethode. Diese ist von der manuellen Planungspraxis, bei der Fahrpläne mehr oder weniger strikt sequentiell, Zug um Zug, geplant werden, abgeleitet. Historisch gewachsene Fahrlagen werden oft jahrelang beibehalten und höchstens um wenige Minuten verschoben, um neue Angebote mit einzuplanen. Mathematisch gesehen ist ein solches Vorgehen ein sogenanntes Greedy-Verfahren, das zu jedem Zeitpunkt die jeweils beste Möglichkeit wählt. Wendet man ein solches Verfahren auf ein so genanntes NP-schweres Optimierungsproblem,

wie das die betrachtete Fahrplanungsaufgabe ist, an, so kann das zu beliebig schlechten oder ganz unzulässigen Lösungen führen. Deshalb verwenden moderne Algorithmen, wie oben angesprochen, synchrone Lösungsmethoden, welche alle Zuglinien in einem Schritt betrachten und dessen Fahrlagen alle miteinander planen und optimieren. Der Nachteil ist aber ihr hoher Rechenaufwand. Die zweite Dekompositionsmethode definiert einen neuen Ansatz, der einen Kompromiss zwischen einer reinen sequentiellen Planung und einer komplett synchronen Methode verwendet. Mit dieser Idee kann die Rechenkomplexität beträchtlich reduziert werden, währenddessen die Lösungsqualität trotzdem auf einem hohen und für die Praxis genügenden Niveau gehalten werden kann.

Contents

Acknowledgments	i
Abstract	iii
Zusammenfassung	vii
1 Introduction	3
1.1 Motivation	3
1.2 Focus and contribution of this thesis	4
1.3 Outline of the thesis	6
2 Embedding of the algorithms in the railway planning process	9
2.1 Introduction	9
2.2 The railway timetabling process	10
2.2.1 Liberalization of the European railway market	10
2.2.2 Stakeholders involved in the timetabling process	11
2.2.3 Main steps of the timetabling process	12
2.3 Basics of timetabling	15
2.3.1 Characteristics of the infrastructure	15
2.3.2 Functional requirements	16
2.3.3 Degrees of systematization	18
2.4 Computer supported timetable construction	21
2.4.1 Visualization of timetables	22
2.4.2 Macroscopic timetable evaluation	25
2.4.3 Support for automatic timetable construction	26
2.5 Embedding of the algorithms for automatic timetable construction	30
2.5.1 Mid-Term planning	30
2.5.2 Further use cases in short-term and long-term planning	32
2.5.3 Integration into planning software	32
3 Modelling cyclic railway timetables	35
3.1 Introduction	35
3.2 The periodic event scheduling problem (PESP)	35
3.2.1 General idea	35
3.2.2 Types of Constraints	36
3.2.3 Visualization of a PESP graph	41
3.2.4 Model granularity	43
3.3 Optimization Criteria	44
3.3.1 Minimization of passenger travel time	44

3.3.2	Minimization of operational costs	44
3.3.3	Maximizing timetable stability and flexibility	45
3.3.4	Advantages and limits for optimization	45
3.4	Implementation and verification for this thesis	46
3.4.1	Data sources	46
3.4.2	Chosen model granularity	47
3.4.3	Event and constraint definition	48
3.4.4	Choice of objective function	51
3.4.5	Definition and description of test instances	52
3.4.6	Model evaluation with OnTime	52
4	Solving the periodic event scheduling problem	59
4.1	Introduction	59
4.2	Algorithms solving the PESP decision problem	60
4.2.1	A constraint programming method	60
4.2.2	A polynomial reduction from PESP to SAT	61
4.3	Optimizing the PESP using a MILP solver	63
4.3.1	The classical MILP	63
4.3.2	The cyclic MILP	63
4.3.3	Cyclic MILP with non-collision cycles	66
4.4	Choice of solution approach for this thesis	67
4.4.1	Solver and its parameter settings	68
4.4.2	Performance of the cyclic and classical MILP	70
4.4.3	Comparison of different cycle bases	72
5	Decomposition methods	77
5.1	Introduction	77
5.2	Decomposition in optimization theory	77
5.3	Geographical decomposition of the PESP	79
5.3.1	Definition of the decomposition	80
5.3.2	Cuts through track sections	84
5.3.3	Heuristics for the coordination of the master and two subproblems	90
5.4	Computational results	94
5.4.1	Hyperplane heuristic	96
5.4.2	Heuristic space search	97
5.4.3	Expected computation time to find a solution of a certain quality .	98
5.4.4	Decomposition method to find a starting solution	99
5.4.5	Summary of the results	101
6	Sequential decomposition	103
6.1	Introduction	103
6.2	Decomposition in the manual planning process	103
6.3	Sequential decomposition for the PESP	104
6.3.1	Algorithmic approach	105
6.3.2	Mathematical properties of the decomposition	107

6.4	Implementation and computational results	111
6.4.1	Choice of train line partitions	112
6.4.2	Variation of fixation constraints	116
6.4.3	Influence of the fixation margin on computation time and quality .	118
6.4.4	Experience with Infeasibility	127
6.4.5	Comparison to the global solution method	129
7	Synthesis	133
7.1	Summary of results	133
7.2	Conclusion, discussion	134
7.3	Further research	135
7.4	Recommendations for application	136
	Glossary	137
	Bibliography	141
	CV	147

List of Figures

1.1	<i>Increase of transport offers and demand for the Swiss railway network</i>	3
2.1	<i>Overview on the iterative planning process in public transport</i>	9
2.2	<i>Main stakeholders of the railway timetabling process</i>	11
2.3	<i>Overview of railway timetabling process</i>	13
2.4	<i>Line headways out of blocking time stairways</i>	16
2.5	<i>Example of a line map</i>	18
2.6	<i>Typical distribution of passengers over a day</i>	19
2.7	<i>Principle of the integrated-fixed interval timetable [Lüthi, 2009]</i>	20
2.8	<i>Planned IFIT hub network in Switzerland for 2030</i>	21
2.9	<i>Example of a graphical timetable</i>	22
2.10	<i>Example of a commercial timetable</i>	23
2.11	<i>Example of a netgraph</i>	24
2.12	<i>Example of a track occupation diagram</i>	24
2.13	<i>Input and output of OnTime [trafIT solutions]</i>	26
2.14	<i>A glimpse at the model construction of DONS. [Middelkoop, 2010]</i>	28
2.15	<i>Glimpse at the software TAKT [Nachtigall, 2014]</i>	29
2.16	<i>Overview of the input and output of the considered algorithms</i>	31
3.1	<i>Use of multiple PESP constraints to model the union of time intervals</i>	39
3.2	<i>Small example of a PESP graph illustrating its typical structure</i>	41
3.3	<i>Illustrative example of a PESP graph showing different types of constraints</i>	42
3.4	<i>Illustration to the granularity of a PESP model</i>	43
3.5	<i>Visualization to the definition of periodic trains out of a annual timetable</i>	48
3.6	<i>Geographical regions of all test instances</i>	53
3.7	<i>Macroscopic infrastructure of the German-SpeakingCH model</i>	54
3.8	<i>Train lines modelled in the German-SpeakingCH model</i>	55
3.9	<i>Progression of OnTime key values to different time reserves</i>	56
3.10	<i>$(\text{minTrainTimeReserve}, \text{minTripTimeReserve}) = (0,0)$</i>	57
3.11	<i>$(\text{minTrainTimeReserve}, \text{minTripTimeReserve}) = (13,7)$</i>	57
3.12	<i>$(\text{minTrainTimeReserve}, \text{minTripTimeReserve}) = (13,9)$</i>	57
3.13	<i>$(\text{minTrainTimeReserve}, \text{minTripTimeReserve}) = (13,13)$</i>	57
4.1	<i>Illustration to the reduction from PESP to SAT</i>	62
4.2	<i>Collisions and overtaking to prevent for variable trip times</i>	66
4.3	<i>Two types of non-collision cycles</i>	67
4.4	<i>Optimization progress for the cyclic and classical MILP</i>	71
4.5	<i>Distribution of cycle widths for four different cycle bases</i>	74

5.1	<i>Illustration of the cut-graph</i>	81
5.2	<i>Desired block structure for the extended cyclic MILP</i>	81
5.3	<i>Contradicting cycle in H_G</i>	82
5.4	<i>Example of a trip cut with five trains running in the same direction</i>	85
5.5	<i>H-Cycle</i>	85
5.6	<i>Trip cycles of two trains running in the same / opposite direction</i>	86
5.7	<i>Possible orders of the occurrences of three events</i>	87
5.8	<i>Possible assignments for all tension variables in the cut-graph</i>	89
5.9	<i>Line map of the test instance</i>	95
5.10	<i>Infrastructure with three geographical cuts</i>	96
5.11	<i>First solution without enumeration to start global optimization</i>	99
5.12	<i>First solution to start global optimization</i>	100
5.13	<i>Best of five to start global optimization</i>	101
6.1	<i>Flow chart of the sequential decomposition algorithm</i>	107
6.2	<i>Cycle of C_B in A_{i-1}</i>	109
6.3	<i>Example for the blockstructure of the sequential decomposition heuristic</i>	111
6.4	<i>Different number of groups for the sequential decomposition heuristic</i>	115
6.5	<i>Different fixation heuristics for the sequential decomposition</i>	117
6.6	<i>Sequential decomposition for the Thun-Basel model</i>	120
6.7	<i>Computation time over different iterations for the sequential decomposition</i>	121
6.8	<i>Group fusions for the sequential decomposition</i>	123
6.9	<i>Sequential decomposition for the EasternCH model</i>	125
6.10	<i>Sequential decomposition for the German-SpeakingCH model</i>	126
6.11	<i>Global solution method compared to the sequential decomposition</i>	130

List of Tables

2.1	<i>Train types and their commercial function used in Switzerland</i>	17
3.1	<i>Used weights for the objective function</i>	51
3.2	<i>Overview and key figures for introduced PESP models</i>	52
4.1	<i>Influence of CPLEX parameter settings for a small test case</i>	69
4.2	<i>Influence of CPLEX parameter settings for a medium size test case</i>	69
4.3	<i>Comparison - cyclic and the classic MILP</i>	70
4.4	<i>Computational performance for four different cycle bases</i>	73
4.5	<i>Further comparisons for the better two cycle bases</i>	75
5.1	<i>Influence of train separation constraints</i>	88
5.2	<i>Exponential growth of possible assignments for variables</i>	90
5.3	<i>Results of the hyperplane heuristic for all three cuts</i>	97
5.4	<i>Results of the heuristic space search for all three cuts</i>	97
5.5	<i>Expected computation times to find a solution of a certain quality</i>	98
6.1	<i>Typical interval sizes of PESP constraints in practice</i>	110
6.2	<i>Percentage of binding constraints for different group partitions</i>	114
6.3	<i>Overview on parameter settings for the sequential decomposition</i>	118
6.4	<i>Computation times for infeasible test instances</i>	127
6.5	<i>Computation time over different iterations for infeasible instances</i>	128
6.6	<i>Overview of the computation results to the EasternCH model</i>	131

List of Algorithms

1	<i>Heuristic space search to find feasible global solutions</i>	91
2	<i>Hyperplane Heuristic to find feasible global solutions</i>	92
3	<i>LookForGlobalSolution(hyperPlane)</i>	93
4	<i>Sequential decomposition heuristic for PESP</i>	106

1 Introduction

1.1 Motivation

With the ongoing globalization, the strategic importance of transportation facilities and services for society and economy continuously increases. For example for the Swiss railway network the yearly total number of train kilometers grew by nearly 27% between 2002 and 2012 up to 188 million kilometers [UTP, 2014]. In 2012 passengers travelled over 23 billion passenger kilometers (Figure 1.1). Thereby, especially the amount of long-distance traffic increased. It currently covers two thirds of all passenger kilometers. Considering the passenger and freight trains traffic together, the Swiss Federal Railways (SBB) network has the world's highest frequency of train services. 148 trains per kilometer of track circulate every day. The demand has never been as high as today and is supposed to increase further over the next decades.

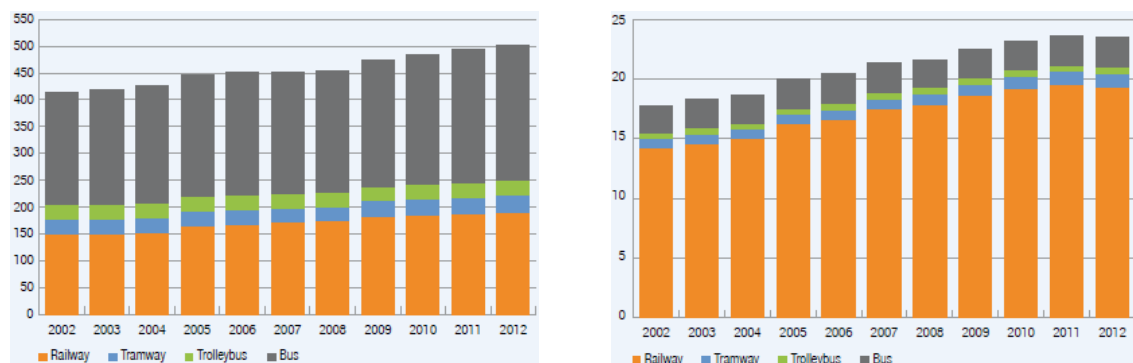


Figure 1.1: *Total supply in millions of journey kilometers for the Swiss public transport network and development of the demand in billions of passenger kilometers [UTP, 2014]*

Thus requirements on capacity and reliability will increase continuously, although they already reach a clear limit for several sections of the railway network today. Infrastructure measures, such as the extensions of tracks, junctions and stations underlie economical and ecological limitations. They are very time consuming, expensive and furthermore often difficult to realize in densely populated areas, as they often appear in Switzerland. Thus infrastructure extensions have to be elaborated very carefully.

Beside infrastructure extensions, also measures on rolling stock and the safety system play an important role for the increase of capacity. New vehicles allowing faster accelerations and higher average velocities can help to deal with capacity bottlenecks.

Furthermore, the advancement of techniques in train operation, allowing for more precise train control and therefore smaller train headway times, contributes essentially to the target of capacity increase.

Several ongoing projects in Europe facilitate these trends. So for example the GSM-R and ETCS as parts of the European Train Management System (ERTMS) improve the safety system and the precision of train control. And the increasing competition with the liberalization of the European railway market subsequently forces railway companies to move along with the currently newest technologies for rolling stock.

In addition to adaptations in train operation, infrastructure and rolling stock, the creation of timetables optimizing connections and frequencies for customers plays an important role to use the maximum of available capacity without losing reliability. In practice, timetable construction is still dominated by manual methods. Computational support is mainly used for the management of data, the visualization of timetables and for subsequent simulations in order to evaluate the constructed timetables. However methods already exist for the automated construction of timetables. These methods are based on algorithms generating timetables out of infrastructure restrictions and functional requirements and use searching and optimization strategies from operation research. Therefore, such techniques from operation research and their ongoing advancements are an essential potential for decision support to the timetabling in railway industry.

This thesis will study and advance such methods for the construction and optimization of periodic railway timetables.

1.2 Focus and contribution of this thesis

The research to this thesis was motivated out of open research questions from the PhD of Burkolter [Burkolter, 2005], Herrmann [Herrmann, 2005] and Caimi [Caimi, 2009] at the same research institutes of ETH Zurich. Burkolter and Herrmann concentrated on the evaluation of capacity and train routings in main station areas, Caimi developed a two level approach to automate the construction and optimization of conflict-free microscopic railway timetables for an entire network in his thesis “Algorithmic decision support for train scheduling in a large and highly utilized railway network” [Caimi, 2009]. This two level approach is based on the idea of the construction of a macroscopic timetable for the entire network optimizing flexibility for the subsequently second level, the fixation of a microscopic timetable partitioned into different regions. For the microscopic level already essential progress in computational performance could be reached in his thesis and the adaption closer to practice also could be ensured by the subsequent PhD of Fuchsberger [Fuchsberger, 2012], but both points were left open in the case of his defined macroscopic approach.

This macroscopic approach is based on the Periodic Event Scheduling Problem, a model which was studied and already used several times in research and practice. Famous

dissertations as [Odijk, 1997, Lindner, 2000, Peeters, 2003, Liebchen, 2006, Opitz, 2009] were finished during the last two decades, some of them in close collaboration with railway companies. Out of these interdisciplinary collaborations first timetabling software could be realized. The oldest and most famous one is DONS (Design of Network Schedules) used in the Netherlands since 2004 [Kroon et al., 2009]. A younger project is the software TAKT used from DB (Deutsche Bahn) in Germany, which was developed at TU Dresden out of the dissertation of Opitz [Opitz, 2009]. Both software products are based on algorithms creating an arbitrary feasible timetable in a first step, followed by a postoptimization to improve the constructed timetable according to a chosen objective function. In case of the DONS, train sequences are already fixed for this postoptimization step, and therefore do not allow much flexibility and potential for optimization. The software TAKT uses more sophisticated procedures for the postoptimization, as for example Modulo Simplex Calculations described in [Nachtigall and Opitz, 2008]. The development of better optimization procedures is still an ongoing research project at TU Dresden. Compared to methods combining a postoptimization step with combinatorial algorithms looking for an arbitrary timetable in a first step, research and famous projects to algorithms directly creating and optimizing timetables in one step also exist. An essential contribution to these algorithms, using mixed integer linear programs (MIP) combined with a MIP solver, was brought by Liebchen and Peters [Liebchen, 2006, Peeters, 2003]. They also could start close collaborations with practice, which lead to a complete reconstruction and optimization of Berlin's underground timetable in 2006 [Liebchen, 2008]. Clear advantages of using direct optimization instead of postoptimization could be underlined several times. But it is often stated that direct optimization based on MIP over current formulations only is applicable for smaller problem instances [Opitz, 2009, Nachtigall, 1998].

Thus this research concentrates on the evaluation and advancements of those algorithms using MIP formulations allowing creation and optimization of periodic railway timetables in one step. They are also part of the macroscopic algorithms defined in [Caimi, 2009]. Our advancements concentrate on the following three points:

- Implementation of the defined algorithms including an automated model construction out of real data provided by Swiss Federal Railways (SBB).
- Extension of the models until a limit of computation time for the used algorithms is reached.
- Development of new methods for the acceleration of the given algorithms to solve and optimize also larger models.

Furthermore, the Periodic Event Scheduling Problem is explained in detail and close to practice, with the motivation of advancing the adaption of such algorithms in practice. Several opportunities for timetable optimization are discussed and the different solution approaches to solve and optimize the models are compared. To ensure conflict-freedom of the created timetables on a mesoscopic planning level, the algorithms are connected to the timetable evaluation software OnTime. This software furthermore provides feedback on timetable stability and the average delay propagation.

To accelerate the given algorithms, two methods are elaborated, both based on the idea of decomposition. A first method is motivated from a mathematical point of view and uses ideas from known decomposition methods in optimization theory. Timetable models are split into several subproblems over geographical cuts and solved iteratively until a global feasible or even optimal solution is found. The method is tested for a special type of geographical cut and two subproblems. With this idea the detection of first feasible global solutions can be accelerated, but an acceleration of the optimization is not reached. Furthermore, the generalization of the method for more than two subproblems and larger problem instances remains unsolved.

A second method, motivated from planning practice, simplifies these two critical points of the first method. It is based on the idea of a partition of all train lines into different groups, which are then scheduled sequentially, while fixing previously scheduled groups of train lines up to a certain time margin. This sequential approach is extended by backiterations ensuring to detect a feasible timetable if one exists. Extensive evaluations to different group partitions and time fixation margins show clear benefits of the methods to accelerate the optimization of timetables also for large problem instances.

1.3 Outline of the thesis

The thesis is partitioned in the following seven chapters:

Chapter 2 introduces basic knowledge of the railway timetabling in practice and describes how the considered algorithms are embedded in the planning process. Section 2.2 summarizes the main steps of the timetabling process in practice, as well as all stakeholders involved in this process. The stakeholders are influenced by the currently ongoing liberalization of the European railway market, which is briefly introduced in a first subsection. Section 2.3 subsequently introduces basic terms of timetabling. They include characteristics of the infrastructure and functional requirements, already with a focus on the considered algorithms of this thesis, and give an overview of different degrees of timetable systematizations. Section 2.4 describes an industrial state of the art for computational support, which concentrates on the visualization of timetables and a macroscopic timetable evaluation. Furthermore, first software products supporting the automated construction of periodic timetables are introduced. Finally, Section 2.5, provides an embedding of the considered algorithms in the planning process, as well as a connection and differences to already existing software supporting timetable construction.

Chapter 3 introduces the periodic event scheduling problem as the model used to construct periodic timetables in this thesis. It starts with the general idea of the model and a detailed description of different constraint types out of literature is given in Section 3.2. This section furthermore discusses different possibilities of choosing the model granularity and introduces the visualization of the models as a graph. Section 3.3 gives an overview of different optimization criteria to optimize the modelled timetable. It includes the minimization of passenger travel time, the minimization of operational costs, the maximization of timetable stability and flexibility and discusses advantages and

limits for optimization. Section 3.4 then explains how the introduced model is used to define our timetabling problems out of data provided by Swiss Federal Railways (SBB). It describes the provided data, the chosen model granularity, all events and constraints defined for the model, as well as the used objective function. Furthermore, seven test models for computational studies in later parts of this thesis are chosen and fixed. The last two subsections of Section 3.4 give an insight into different characteristics of these test models and discuss the evaluation of the models through a timetable evaluation software.

Chapter 4 reviews known solution and optimization methods for the PESP. Section 4.2 gives an introduction into algorithms solving the PESP decision problem, which are already a part of existing software supporting timetable construction. Subsequently, Section 4.3 introduces the algorithms which are also used in this thesis to optimize the PESP using a MILP solver. Section 4.4 justifies the choice of the solution method for this thesis and discusses and evaluates different parameter setting optimizing the computational performance for the solution method. Furthermore computational results for the fixed test instances are shown, illustrating the computational operability of these algorithms.

Chapter 5 introduces the geographical decomposition method. It starts with a review of known decomposition methods in optimization theory described in Section 5.2. Subsequently, Section 5.3 introduces the theoretical framework of the decomposition method, discusses a certain type of geographical cut in more detail and introduces two heuristic algorithms solving the corresponding decomposition problem with two subproblems iteratively. Section 5.4 shows and discusses computational results of both algorithms, it evaluates the expected computation time to find a solution of a certain quality and describes the idea of using the algorithms to find faster starting solutions for the global optimization.

Chapter 6 introduces the second decomposition method. Section 6.2 describes common decomposition methods used in the manual planning process. Subsequently, Section 6.3 introduces the sequential decomposition heuristic, its algorithmic approach and a mathematical reflection. Section 6.4 shows the computational results to the sequential decomposition heuristic. It introduces different partitions of train lines, variations of time fixations strategies and the influence of the time fixation margin on computation time and quality. Furthermore, it discusses the experience with infeasible timetabling problems made by using the decomposition method and a comparison to the original global solution method fixed in Chapter 4.

Chapter 7 summarizes and discusses all results of the thesis. It describes further research, as well as recommendations for applications.

2 Embedding of the algorithms in the railway planning process

2.1 Introduction

The timetable is the core of railway operation. It schedules all arrival and departure events of every train in a network. From a commercial point of view, as well as for operational reasons, fixing a timetable is necessary. Starting from a certain number of passengers with a given number of vehicles the demand of a traffic system has to be bundled. And to ensure operational safety, especially for a traffic system with long braking distances, train slots have to be coordinated.

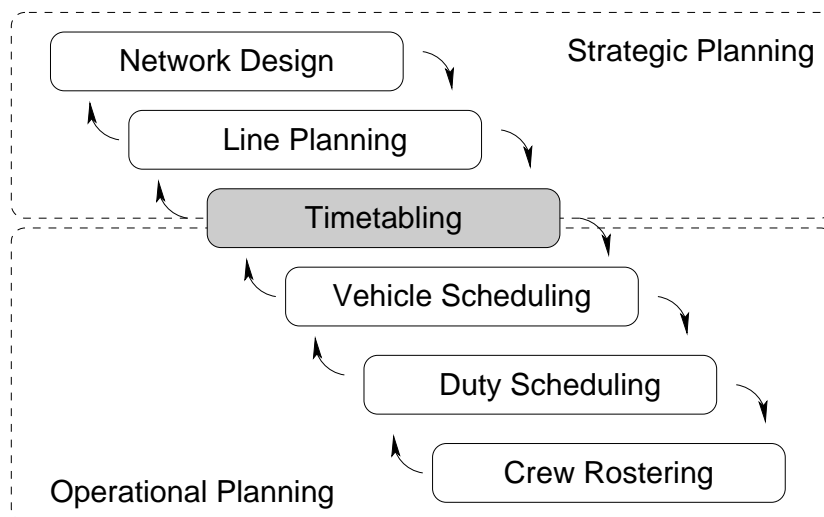


Figure 2.1: *Overview on the iterative planning process in public transport [Weidmann, 2011b], [Liebchen, 2006]*

As Figure 2.1 indicates, the creation of a timetable is an essential interface between the strategic and operational planning of a railway system. In Section 2.2, further details to different planning stages, involved stakeholders and corresponding regulations are explained. It underlines the high complexity and the huge expenditure of the time needed to plan a railway system. Subsequently, in Section 2.3 basic terms of timetabling are introduced already with a focus on the timetabling problem and planning stage considered in this thesis. Section 2.4 gives an overview to existing software supporting the construction of railway timetables. On the basis of the previous three sections, Section 2.5 then discusses the embedding of our research and the developed algorithmic approaches into the timetabling process and its connection to computational support.

2.2 The railway timetabling process

A long time horizon and the coordination of several stakeholders with different goals enlarge the complexity for railway timetabling. In this section we start with the description of the liberalization of the European railway market, which influences the role of stakeholders and the whole railway timetabling process, in Switzerland as well. This actual political situation is used as a basic framework for the introduction of the main stakeholders participating in the timetabling process in Section 2.2.2. Subsequently we give an overview of the most important steps in the timetabling process in Section 2.2.3.

2.2.1 Liberalization of the European railway market

Until the 1990s all European railway networks were state owned. As the first country Sweden started the liberalization of its railways in 1988 [Pham, 2013]. Positive experience from Sweden, especially improving the efficiency of the overall railway network, together with the aim of forwarding a harmonized, integrated European railway network, motivated the European Union Council to reform the heavily subsidized national railway networks. In 1991, with Directive 91/440, a first main directive forcing a vertical separation of the infrastructure management and service operation was fixed. It should allow free access to the rail infrastructure for all train operating companies and therefore promote competition in railway industry. Subsequently, further legislation to deregulate the rail market and three main reform packages were introduced in 2001, 2005 and 2007. A fourth railway reform package is in negotiation at the European parliament. Until today (2014), rail freight transport and the international passenger transport is liberalized. The market opening for domestic passenger rail transport is part of the planned forth reform package. A more efficient and competitive railway market is also a prerequisite for achieving targets of emissions reduction and modal shift. Parallel to the market regulations, the harmonization of infrastructures and the safety system also play an important role. Famous running projects such as the European Rail Traffic Management System (ERTMS) [European Commission and Transport, 2006], with the European Train Control System (ETCS) as a major component, are important contributions to dealing with the obstacles of technical incompatibilities in different subsystems of the railway network.

This process of the union and liberalization of European railway systems also influenced politics and advancements of the railway market in Switzerland. In the last two decades, Switzerland's railway was formally separated from governmental administration and competition was enabled on the available infrastructure [BAV, 2014]. Furthermore financial aspects were clarified. So for example with the revision of the railway laws in 1996 the principle of ordering services in regional public transport from political authorities was introduced. This also ensures a minimal offer of public transport for sparsely populated regions on an organizational level. In 1999, a first reform package was introduced to separate infrastructure management from train operating companies. With this step the three major railway companies SBB, BLS and SOB were each separated in an infrastructure management and a train operating sector. But train path allocations still took place by the same companies. To improve competition and to follow European politics, a separated entity called "Trasse Schweiz AG" was founded to allocate train paths from

2006 onwards. Further adaptations in legislation and technical harmonization are currently going on in Switzerland's politics. The progress of the introduction of ETCS and the ongoing connection to the European high speed train system already are essential steps in this direction.

2.2.2 Stakeholders involved in the timetabling process

In this subsection, we describe the four main stakeholders which are involved in the timetabling process, their interactions and different interests. Customers from passenger and freight rail transport are a first group of stakeholders, using the product of the rail market and involving rail market by their travelling behaviour and demand. A second group of stakeholders are rail operating companies offering services to customers. In the given framework of the current political situation in Switzerland, as described in the last subsection, two more stakeholders play an important role. Public authorities have to ensure a public service all over the country about direct service requests for a basic regional transport offer. Infrastructure managers have to coordinate train path requests from train operating companies providing free access on railway infrastructure for all companies. Figure 2.2 summarizes the configuration of the four stakeholders and their decisions directly involving the timetabling process.

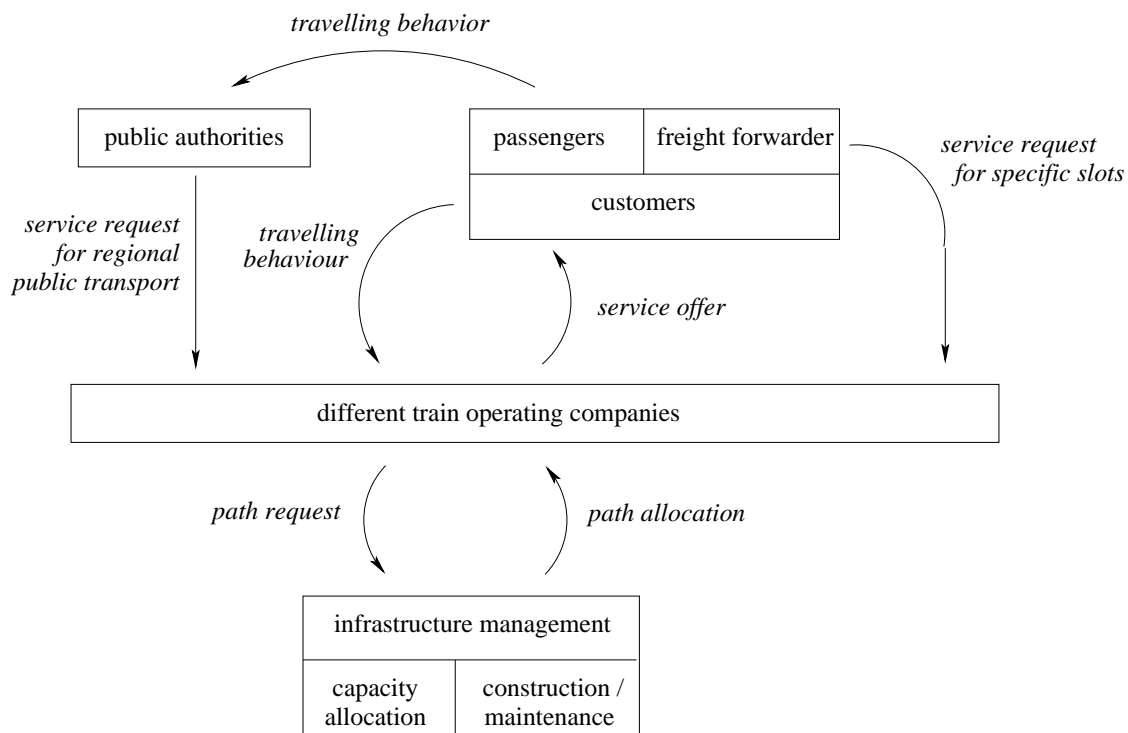


Figure 2.2: The four main stakeholders of the railway timetabling process and their interactions connected to the timetable construction

In Figure 2.2 we can describe two loops. In a first loop, customers, public authorities and train operating companies are involved to define a certain service intended to be offered.

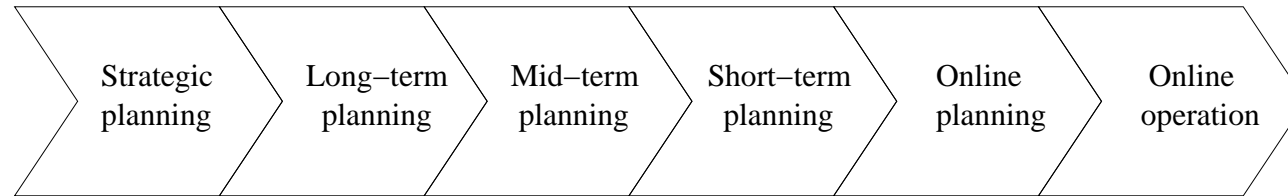
We therefore call it the *service intention loop*. Customers' travelling behaviour and demands influence decisions on planned service intentions for train operating companies. They want to optimize their service to cover and increase as much demand as possible minimizing their operational costs at the same time. In addition to these influences, public authorities subsidize and order services for regional trains of a long-term perspective, evaluating travelling behaviour of passengers and defining a minimal service necessary to provide public transport network all over the country. Their goal is the optimization of the overall inter-modal transportation system. Their requests have to be integrated in the yearly passenger timetables. Further service requests come from freight forwarders, which are in comparison short-term and are often integrated very individually in daily timetables by train operating companies. Closing the service intention cycle, the current timetable influences travelling behaviour of customers. They are interested in having short travel times, frequent and flexible offers and low travelling costs.

A second loop being part of Figure 2.2 involves all train operating companies and the infrastructure manager. It describes an iterative process coordinating all path requests and allocating paths without favouring train operating companies. The interests of infrastructure managers concentrate on the optimization of capacity use and operational stability. It is easy to recognize that more train operating companies will be involved in this *path allocation cycle* with an advancing liberalization of the railway market. Thus the whole timetabling process will get more and more complex. Therefore, efficient timetabling becomes even more important.

2.2.3 Main steps of the timetabling process

Due to the long time horizon and the inherent complexity of the coordination of a lot of different requirements, railway planning process is normally divided into several planning steps. Figure 2.3, partly adopted from [Bickel et al., 2010], illustrates this chronologically separated planning steps and different duties of the stakeholders. Public Authorities are mainly involved in the early planning steps, requiring certain regional passenger services in the concept and service planning step. Customers are more involved after the publication of the annual timetable and during operation. In comparison to these two stakeholders contributing at the border of the railway planning process, infrastructure managers and train operating companies are mainly responsible during the planning process. As Figure 2.3, indicates they have different duties running in parallel over the whole time horizon. Their duties are explained in further detail in the second part of this subsection.

Contrary to the impression Figure 2.3 could give, none of these duties are independent of each other, neither along the time horizon nor between the two stakeholders. Therefore, iterations and combinations between different duties are expected to improve



Time horizon	> 10 years	5–10 years	years–months	months–days	hours	minutes–seconds
Duties						
Infrastructure manager	strategic network development	infrastructure planning	capacity and works planning	capacity allocation	traffic management	traffic control
Train operating companies	concept planning	service planning	timetable planning	timetable production	rolling stock and stuff dispatching	train operation
Objectives	economic efficiency		capacity		reliability	

Figure 2.3: Overview of the main steps of the railway timetabling process distributed on different common time horizons, objectives and duties of the infrastructure management and train operating companies

the result. Nevertheless, in most practical problem settings the complexity of each planning step alone is already so high that combinations of two planning steps only rarely exist. As an example the inclusion of certain aspects of timetable production, such as minimizing the need of rolling stock, into timetable planning sometimes appears [Cadarso and Marn, 2012].

The main reasons for the long time horizon of the railway planning process are the long economic life and construction times of the railways infrastructure and rolling stock. So for example, starting from about five years before a timetable gets into operation, no more larger infrastructure adjustments are possible. Therefore, infrastructure adjustments are planned and performed five to ten years before the infrastructure planning phase. They result from earlier developments considering a larger entire network and strategical decisions in the strategic network development. During a mid-term planning phase infrastructure management starts to plan capacity use corresponding to consecutively incoming path requirements of train operating companies. In the meantime, they already have to take into account construction works when planning. Months or days before a timetable gets into operation the infrastructure management fixes their train path allocations and therefore also the yearly and daily railway timetable. The timetable gets into operation and the infrastructure management concentrates on traffic managements and control, which for example includes train dispatching and trouble-shooting in case of disruptions.

In the meantime train operating companies work on the development of their service and its operation. Since ordering rolling stock also underlies given time limits and early collaboration with infrastructure managers could influence their decisions on necessary infrastructure extensions, the planning process for train operating companies can be distributed over quite the same time horizon. They start with the concept and service planning. Analyzing customers' travelling behaviour they optimize their offer. So for example data is collected describing time, origin and destination of passengers' journeys to define train lines, their frequency and important connections between the lines. With these functional requirements and the given infrastructure restrictions, timetable planning can start. The closer the planning gets to operation, the accuracy and level of detail of the timetable increases. In a first step the timetable is fixed only roughly in a granularity of minutes and for larger track sections, whereas at the end, train movements are described very precisely in time and location also inside main station areas. In the manual timetabling process as well as for algorithms, this timetable planning step is often separated in several steps working with different levels of details. Corresponding to this accuracy, these different planning levels are often referred as a so called *macroscopic* and *microscopic* planning level [Caimi, 2009], [Borndörfer et al., 2010]. Sometimes even a third intermediate level is introduced, called *mesoscopic* level [De Fabris et al., 2014]. With the timetable production rolling stock circulation, crew scheduling and crew rostering also has to be integrated. Sets of trips are combined which can be operated by the same vehicles. Duties are scheduled and distributed over all participants of the crew. All of these three planning steps define complex optimization problem and are studied individually in the literature.

2.3 Basics of timetabling

In this section, we introduce basic terminology of railway operation to describe all requirements on a timetable as used in this thesis. We partition them into restrictions given by the characteristics of the infrastructure, including the whole safety system, and functional requirements, mainly representing the interests of customers. Furthermore, we give an overview of the different degrees of timetable systematizations and their advantages and disadvantages in the subsequent Section 2.3.3.

2.3.1 Characteristics of the infrastructure

On a high level of abstraction, a railway network can be modelled using *nodes* and *links*. Nodes are operation points of a network as for example *junctions*, allowing a train to change from one link to another. Sometimes, and especially in this thesis, nodes can also be referred to extended operating points containing whole station areas in which overtaking, crossing and direction reversal of trains are possible. A link between two nodes can consist of several parallel running tracks. The sequence of tracks a train chooses through the railway network is called *itinerary*. For this thesis two itineraries using different tracks of the same link are supposed to be independent from each other for the entire link. This means trains passing the same link, but using different itineraries, do not conflict with each other. If they did, the conflicting infrastructure segment would have to be added as a node.

For a given itinerary and rolling stock technical minimal running times for each link can be calculated. To improve recovery from small delays and to optimize speed profiles for energy consumption and signalling a running time supplement of some percentages (in Switzerland in average 7%) is added.

The safety system in most European countries, as well as in Switzerland, is based on the fixed block signalling system [Pachl, 2008]. Parts of the infrastructure are defined as blocks and can be used only by one train at the same time. Signals are used to inform train drivers about clearance of blocks. So called *blocking times* are used for timetabling to describe the amount of time a train occupies a block. This blocking time includes the time starting from the route fixation before a train enters a block up to the time the train passed the whole block completely and its end reaches a so called clearing point. Out of the blocking time it is possible to determine a minimum time distance two consecutive driving trains have to hold for a certain line, the so called *minimum line headway*.

Figure 2.4 illustrates the determination of the minimum line headway out of all blocking times of two trains for one line. The union of blocking time squares which are passed by a train in a time-distance-diagram is called *blocking time stairway*. In a time-distance-diagram, approaching two trains until the first pair of blocking time square touches each other, determines the minimum line headway. If two trains run with a larger time distance as the minimum time headway, the additional time distance is called *buffer time*. Adding

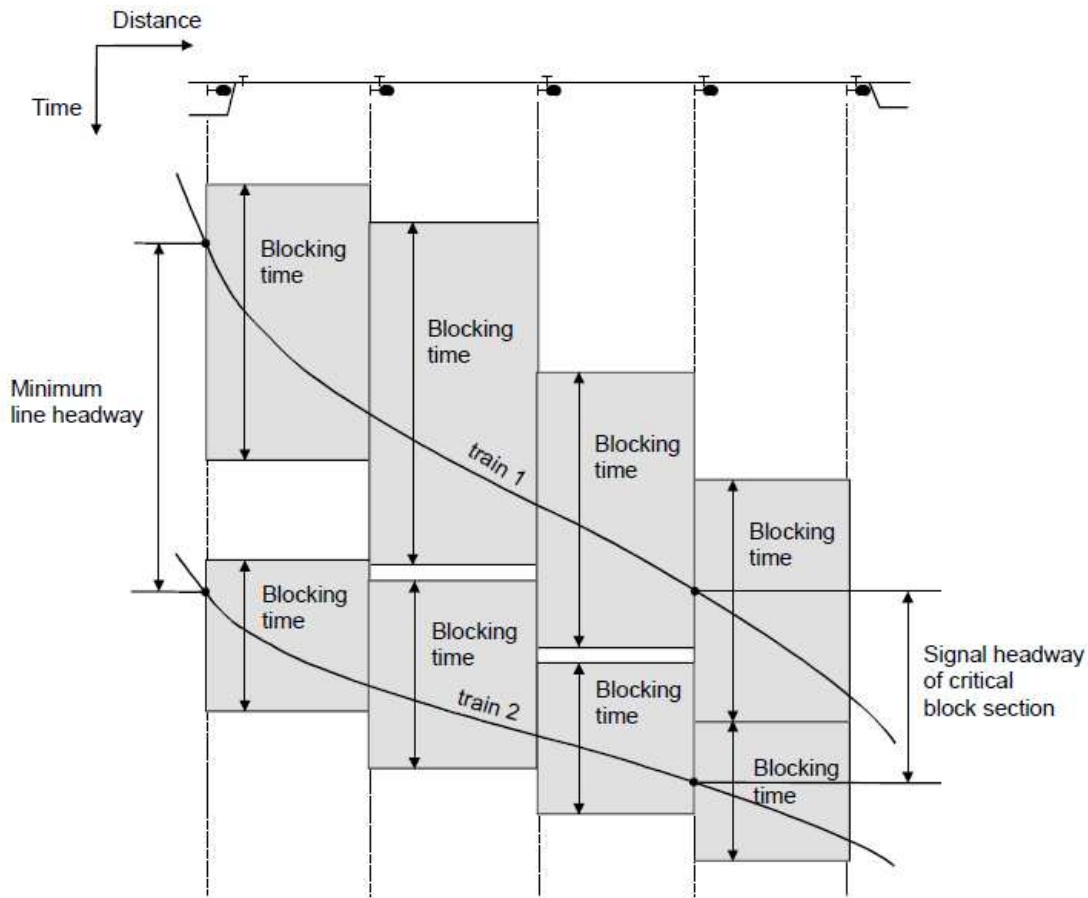


Figure 2.4: From blocking time stairways to minimum line headways
[Pachl, 2008]

buffer time to a timetable allows to reduce delay propagation between trains. In combination with running time supplements, it therefore plays an important role in timetable stability, as further discussed in Section 3.3.

2.3.2 Functional requirements

Operating points in the railway network where trains can stop to board and alight passengers are called *stations* in this thesis. Depending on its commercial function, not every train has to stop at every station on its itinerary. If a train stops at a station to alight and board passengers, we will call this a *commercial stop*.

To improve easy comprehensibility of a timetable to its customers trains are classified to different *train types* corresponding to their commercial function. As an example Table 2.1 gives an overview on different train types used in Switzerland's passenger railway together with their commercial function. For further considerations we denote the two main categories of *fast trains* and *regional trains* as indicated in Table 2.1. Depending on

the choice of train type, different rolling stock is also used having specific characteristics on capacity, comfort and driving behaviour.

train type	acronym	commercial function	main category
High-Speed train	TGV	international connections between larger cities	fast trains
Inter City Express	ICE		
Railjet	RJ		
EuroCity	EC		
EuroNight	EN		
CityNightLine	CNL		
InterCity	IC	national connections between cities and regions	
InterCity tilting train	ICN		
InterRegio	IR		
Regio Express	RE	traffic collection inside a region, can be accelerated for certain sections	regional trains
S-Train	S		
Regio	R		

Table 2.1: *Train types and their commercial function used in Switzerland*

Beside the vehicle and its related commercial function in the network, a train's concrete stopping pattern is of particular interest for a customer. To simplify the description of commercial stops offered by an individual train, *train lines* are defined representing trains serving the same set of stations regularly over a day. Therefore it is further assumed that all trains belonging to the same train line have the same travel time between its stations and the same dwell times at their commercial stops. And normally trains are also running in both directions for each train line. Train lines are often visualized in so called *line maps*, showing the customer all direct connections on a geographical map. Figure 2.5, for example, shows a line map of the Rhetian Railway in Grisons, a Canton of Switzerland, of the year 2014.

Trains of one train line are often scheduled with a regular time distance called *frequency*. This separation in time intends to cover the demand distributed over a day and also simplifies memorability of a timetable for passengers, as discussed in more detail in Section 2.3.3. To further optimize the offer for passengers, trains belonging to different train lines, but sharing a similar offer on a common section, are also separated in time. In this thesis we denote such a timetable constraint *train separation*.

If there is no direct connection for a trip destination with higher demand, railway operators can fix a *connection* between two lines at a certain station. Such a request will then be already part in timetable construction. The goal is to fix the corresponding arrival and departure of the two trains in an adequate time distance for the given station. A time distance has to be long enough to allow passengers time to change from one train to the other, but it should not exceed an upper threshold of time to keep the overall travelling

time of passengers as short as possible. Required connections between train lines are also part in railway operation. If the first train arrives with a small delay, the second train often has to wait, to keep the promised connection.

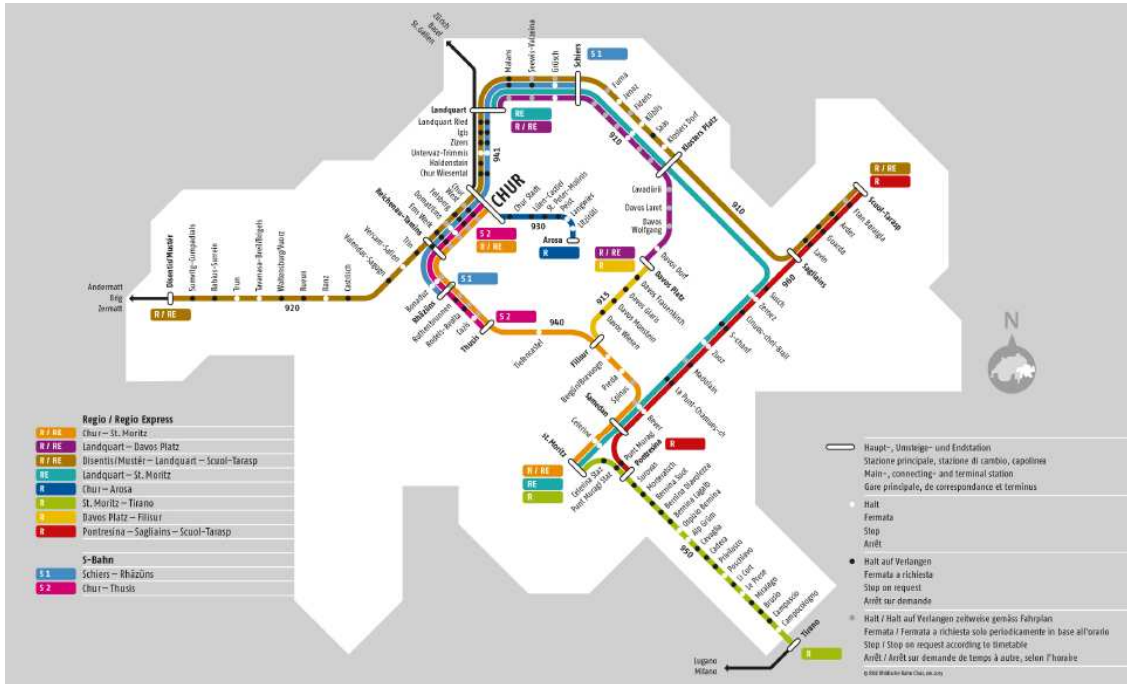


Figure 2.5: Line Map 2014 of the Rhetian Railway. Source: [RhB, 2014]

2.3.3 Degrees of systematization

For a timetable there exists different degrees of systematizations. In this section we give an overview on them.

Periodic timetable

A first systematization of a timetable is periodicity, which is very common in several countries of Europe. All trains belonging to the same line are running equally distributed over a day with a common period. Often this period is one hour.

Definition 1. A timetable with a certain time period is called *periodic timetable*.

A periodic timetable has several advantages for customers and therefore enlarges the comfort for travelling more flexibly and spontaneously. Also the complexity for timetable construction can be reduced, introducing periodic timetables. However, at the same time, productivity of railway operation decreases. The demand in passenger railway is not distributed equally over a day. Figure 2.6 shows the distribution of demand for Switzerland’s passenger trains for an average weekday. Thus the effectivity of the railway system decreases with a periodic timetable.

The demand in the morning and evening peak hours (7-8 a.m. and 5-6 p.m.) is a multiple of the demand in late evening hours. Having a strict periodic timetable with the same rolling stock over a whole day therefore leads to trains which are overcrowded in rush hours and quite empty in the evening. To reduce this negative property of periodic timetables they are often adapted for these special hours. During peak hours some additional trains run especially to meet the demands of commuters and during off-peak hours the train offer is reduced. Such slightly adapted periodic timetables in the literature are sometimes called *partial periodic timetables*, [Caimi et al., 2009b].

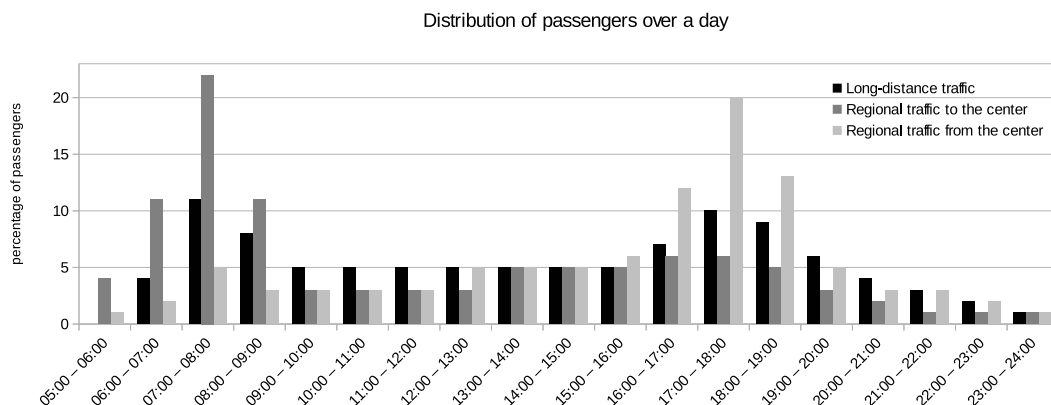


Figure 2.6: Typical distribution of passengers over a weekday in Switzerland's railway system [Weidmann, 2011a]

Symmetric periodic timetable

In addition to periodicity symmetry is often used in practice. In this case each train has equivalent trip and dwell times in both directions. And trains of different train lines, running according to a periodic timetable, meet twice in one period or even more if the frequency of these train lines is larger than one. The two meeting minutes, with a time distance of half a period, fix a so called *symmetry axis*.

Definition 2. A periodic timetable is called *symmetric periodic timetable* if all trains have a common symmetry axis.

The introduction of a symmetric periodic timetable further simplifies memorability for passengers. If they know the departure of a train line in one direction they can also easily deduce the departure in the other direction. Furthermore all connections automatically are offered in both directions. From an operational point of view a symmetric timetable requires identical trip and dwell times of a train in both directions. This can lead to difficulties especially in a network with large differences in altitude. Also obstacles on the side of the infrastructure, as for example a long single track line, can hinder the rolling out of a complete symmetric timetable. Therefore, this symmetry constraint in practice often appears in a slightly relaxed form, allowing deviations of a few minutes. The impact of symmetry constraints in timetable construction, especially in connection with timetable optimization, are discussed in [Liebchen, 2004].

Integrated fixed interval timetable

Nevertheless for a symmetric periodic timetable even a higher degree of systematization exists: The so called integrated fixed interval (IFIT) timetable. It further improves connections between all train lines of the entire network. If the connection from a first train to a second train is optimized in an arbitrary symmetric periodic timetable, this automatically optimizes a connection in the backward direction, too. However, in most cases it worsens the connection from the second to the first train in the other direction. In fact an optimization of all four connections between two train lines only works if all trains stop at the same time at the same station and wait until passengers change from one train to the other. A station where trains are planned to stop at the same time and to wait for all connections is called *IFIT hub*.

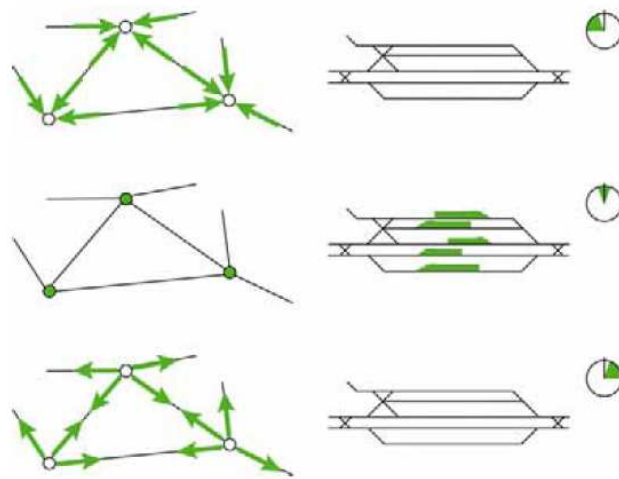


Figure 2.7: Principle of the integrated-fixed interval timetable [Lüthi, 2009]

Definition 3. A symmetric periodic timetable is denoted as an *integrated fixed interval timetable*, if in addition to the periodicity and symmetry constraints a network of IFIT hubs is defined, where all trains provide connections in all directions to the first symmetry minute.

The typical consequences for train movements in an IFIT hub are illustrated in Figure 2.7. With an IFIT timetable passengers get a connection between every train line stopping at an IFIT hub. On the other hand the realization of such a timetable leads to a lot of challenges in operation and infrastructure. For example, the number of platforms and the whole infrastructure around an IFIT hub has to be extended to cope with the higher amount of capacity needed around the arrival and departure of all trains. Travel times between IFIT hubs have to be an integer multiple of the period. Furthermore, the common departure of all train lines leads to small buffer times around IFIT hubs, challenging delay management and very high fluctuations in energy consumption for a whole network. Even for passengers this type of timetable can bring disadvantages such as longer dwell times in hubs, if they do not have to change the train, and fixed travel times for decades without profiting from faster rolling stocks.

Nevertheless the principle of integrated-fixed interval timetables is generally regarded as superior and applied in Swiss rail network [BAV, 2011]. As a main advantage the simplification of long term infrastructure planning and the protection of freight lines are stated. Furthermore, the strategy of running trains as fast as necessary instead of running trains as fast as possible is often pointed out. Figure 2.8 shows the planned hub network of Switzerland for the year 2030.

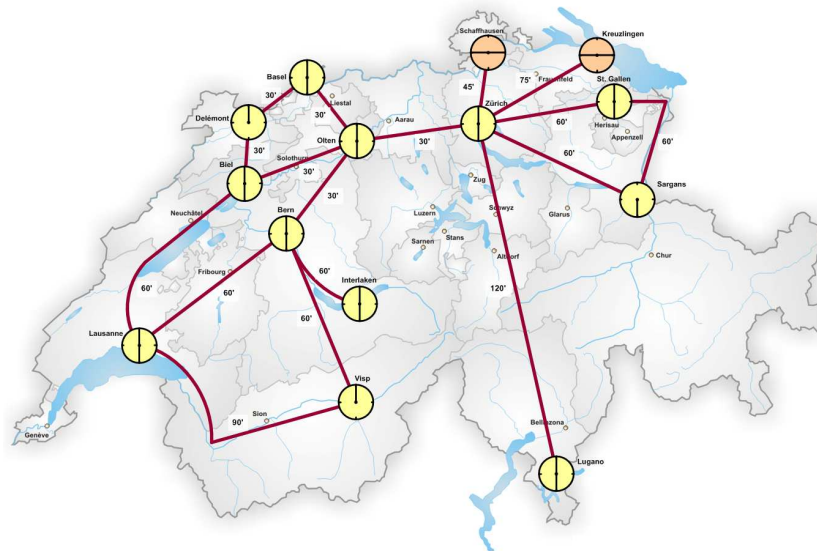


Figure 2.8: *Planned IFIT hub network in Switzerland for 2030, ZEB (Zukünftige Entwicklung der Bahninfrastruktur, SBB)*

2.4 Computer supported timetable construction

With the development of computers computational support for timetabling progressed as well. Thus today railway planning and operation without IT is no longer imaginable. A huge set of different software tools already exists to support different stakeholders in various duties. Often software is developed very specifically for every task and user. Thus the transfer of data from one planning stage to the next and from one company to another often causes time consuming work to define adequate interfaces. Nevertheless those transfers and a fast handling with sets of data in general are one of the first big advantages the computer brought. Also several preprocessing computations for timetabling, as the calculation of running times and headway times, could be shortened considerably. Further, important progress was achieved by the introduction of graphical user interfaces in the eighties of the last century, which allowed the visualization of timetables. Section 2.4.1 introduces different types of timetable visualizations. With the increase of computational performance in the last two decades the introduction of timetable evaluations over simulations has become more and more common. Section 2.4.2 gives a short presentation of a specific evaluation software used in this thesis. For the core part of the timetable construction and optimization, manual methods still dominate due to the high computational complexity. Software packages supporting timetable construction, as

Viriato [SMA, 2014], NeTS [Netcetera, 2014] and TPS [HaCon, 2014] support the management with all necessary data and offer visualizations for timetable construction, but they do not automate timetable construction. Nevertheless, first software developments in this direction already exist. Section 2.4.3 gives a short insight into the two software tools DONS and TAKT supporting the construction of periodic timetables.

2.4.1 Visualization of timetables

In this section, four types of timetable visualizations are introduced which play an important role in planning practice.

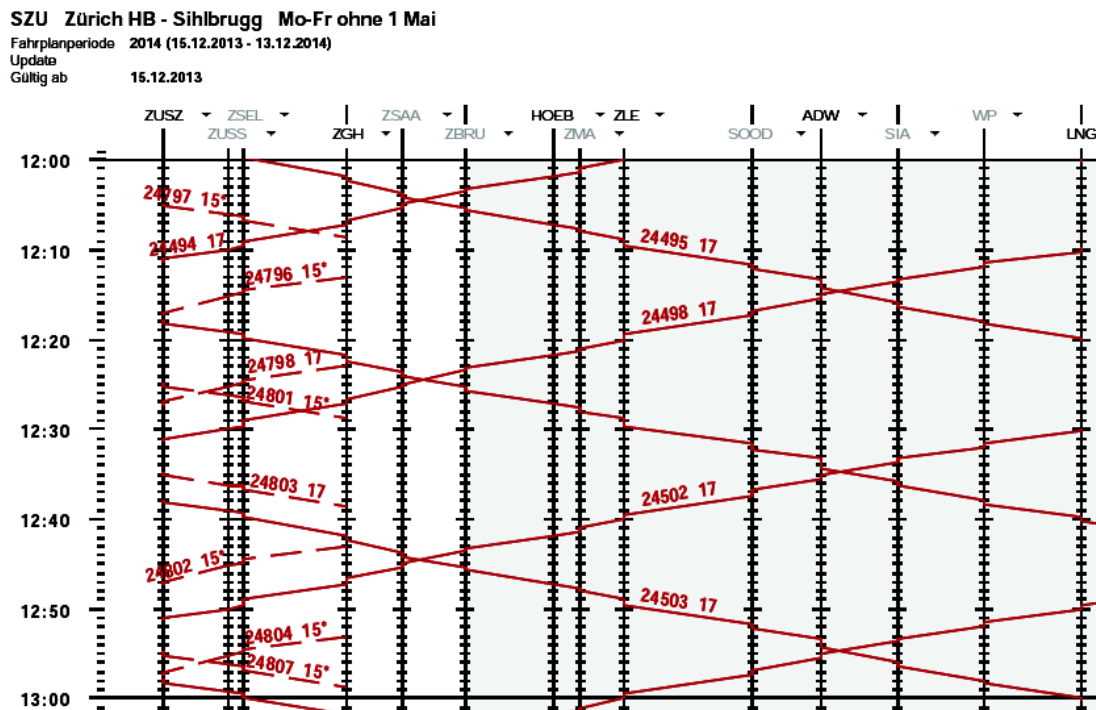



Figure 2.9: Example of a graphical timetable for the Sihltal-Zurich-Uetliberg train of the year 2014, between the two operation points Zurich main station SZU and Langnau-Gattikon for one hour between noon and 13:00 pm [fahrplanfelder.ch].

As a main instrument for train dispatching and timetable planning on lines the so called *graphical timetable* is used. It visualizes the location of each train for a given infrastructure section as a function in time. If the speed profile of a train is already known on a microscopic level the location can be visualized as a continuous function in time. On the other hand, if there are only departure and arrival times known for a macroscopic network of locations in the railway system, they are connected over straight lines in the time-distance diagram, as indicated in Figure 2.9. Out of such a diagram dwell times at stations can also be read out as the length of vertical lines at stations and conflicts on

lines can be easily recognized as crossings of two lines corresponding to trains using the same infrastructure elements.

841 Frauenfeld-Wil 

→


Zürich HB 750				11 523	11 525				607	637
Zürich Flughafen → 750						547			618	648
Winterthur 840	o			11 552	11 550	601			633	703
Winterthur 840			521						635	705
Frauenfeld	o		536						646	716
Weinfelden 840		440	529						629	659
Frauenfeld	o	456	540						640	710
		515	515						515	515
		7052	7058			7060	7062	7064	7068	7070 7072
Frauenfeld		A 519	X 549							
Frauenfeld Marktplatz	x	521	551						649	719
Lüdem	x	523	553						651	721
Murkart	x	525	555						623	723
Weberei Matzingen	x	527	557						625	725
Matzingen	x	529	559						627	727
									629	729
Jakobstal	x	531	601						631	731
Wiesengrund	x	532	602						632	732
Wängi	x	534	604						634	734
Rosental	x	537	607		11 622	634	11 643		704	11 722 734
Münchwilen Pflegeheim	x	539	609		A 624	637	11 645		707	11 724 737
Münchwilen TG	x	541	611		A 626	639	A 647		709	A 726 739
Wil 	o	A 548	X 618		A 628	641	A 649		711	A 728 741
Wil		602	622	625	647	654	702		A 722	725 747 754
St. Gallen 850	o	625	A 647	653	712	715	725		A 747	753 812 815
Wil		602	632				702		732	802
Wattwil	o	625	631	655	X 704		725	731	755	804
Nesslau-Neu St. Johann 853	o		647	X 723			747		823	825 831 847
Wil		610			639	11 652	706	712		739
Winterthur 850	o	626			656	718	723	742		756
Winterthur 850		628			658	720	725	755		758
Zürich Flughafen → 750	o	641			711		737	808		811
Zürich HB 750	o	653			723	11 738	751	821		823

Figure 2.10: Example of a commercial timetable of Switzerland’s annual timetable 2014 for the line Frauenfeld-Wil served by a regional narrow-gauge railway [fahrplanfelder.ch].

Another visualization of a timetable is the description of all relevant arrival and departure times of different trains in a table, often used for the publication of a timetable for customers and therefore called *commercial timetable*. Additional information to the rolling stock, important connections and further offers to customers are often added to a commercial timetable as indicated in Figure 2.10, whereby a compromise between the amount of information and readability for an average customer has to be chosen. In this regard, new web applications brought important improvements allowing to concentrate on information for a certain journey with respect to a specific time slot and date.

To improve readability and a fast overview of a periodic timetable for an entire network the so called *netgraph* is often used. It is a combination of a line map and a commercial timetable visualizing all larger stations as small boxes together with all arrival and departure times for the lines passing these stations. Train lines are represented as in a line map. Colours and structures are used to distinguish between different train types and train frequencies. Smaller stations are added as dots on the lines. Figure 2.11 shows a small section of a netgraph together with its legend.

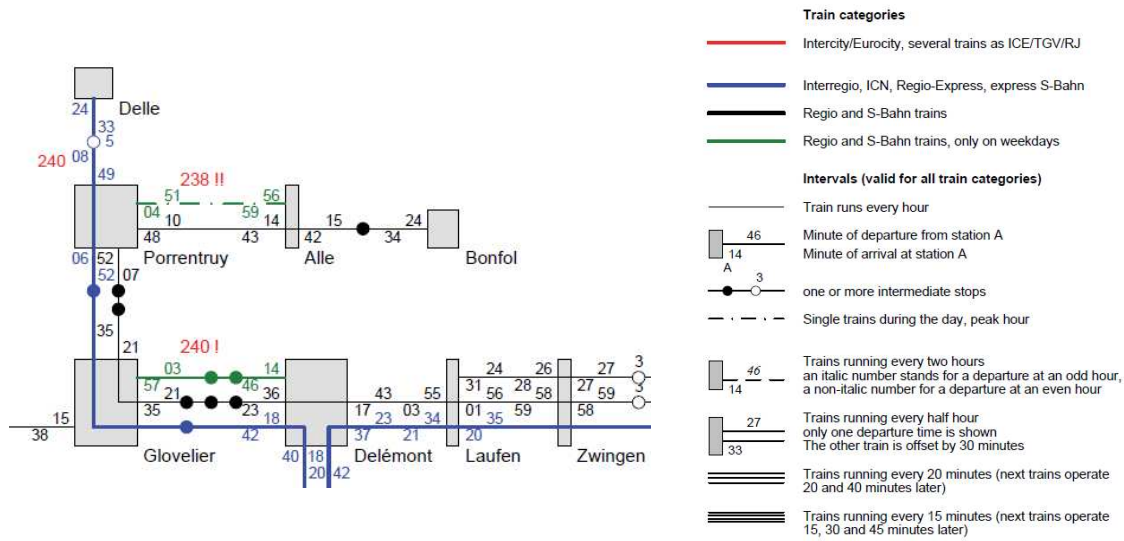


Figure 2.11: Example of a netgraph for a small section around the two cities Delémont and Porrentruy in the French speaking part of Switzerland together with its legend [SMA].

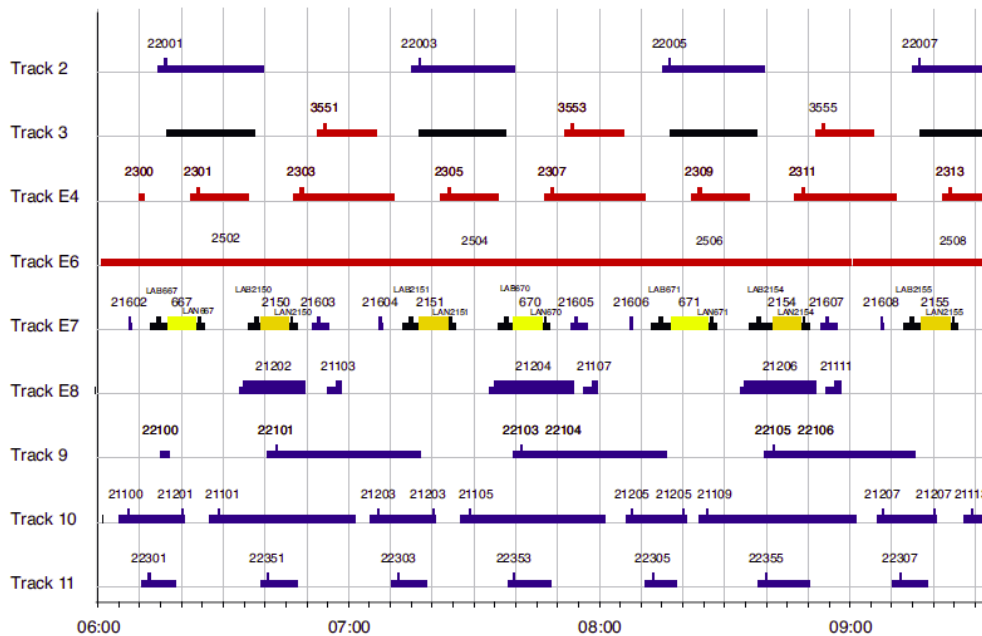


Figure 2.12: Example of a track occupation diagram for main station Lucerne in central Switzerland during three hours in the early morning starting at 06:00 am [OpenTrack].

As a last timetable visualization, we would like to briefly mention the so called *track occupation diagram* in this section. It plays an important role in timetable planning and operation inside larger station areas and visualizes time slots a train uses a platform of a station, as indicated in Figure 2.12.

2.4.2 Macroscopic timetable evaluation

For many years simulation software evaluating the stability of railway timetables mainly worked on a very detailed granularity of data and, because of its computational complexity, only for smaller regions. Examples are software as RailSys [Siefer and Radtke, 2005], OpenTrack [Hürlimann, 2002] and LUKS [Janecek et al., 2010]. They all use deterministic variables to model disruptions by scattering in individual primary delays and conduct a larger number of simulation runs to pool their deterministic results down-stream (“Monte Carlo Simulation”). In this way they get statistically safe pronouncements of the behaviour of different timetable variants [Bücker and Seybold, 2012].

Similar approaches were introduced using macroscopic data to enlarge the model area (Fasta [Noordeen, 1995], SIMONE [Middelkoop and Bouwman, 2001]). The consideration on this aggregated level of detail has furthermore the advantage that it can also be used in earlier planning stages evaluating strategic decisions. To accelerate computation times of the simulation Bueker, in his dissertation [Bueker, 2010], directly uses random variables for delays and introduces suitable distribution functions. This idea allows to enlarge problem sizes and to evaluate timetables for large and complex networks. Together with further elaborations [Bücker and Seybold, 2012], it constitutes the core part of a new evaluation software called OnTime [Franke et al., 2013], available since summer 2011.

The data necessary to run an evaluation with OnTime is very flexible and therefore also suitable for macroscopic timetables in an early planning stage. As a minimal input, an itinerary as a sequence of operation points together with a time of arrival and departure or passing is necessary for each train line. In addition, running-time and dwell-time supplements have to be known, as well as primary delays for each train activity (entrance, stop, ready to start, departure) together with their probability of occurrence. In the ideal case, knowledge of the tracks used on open lines and inside stations is also present. Furthermore, one is able to add inter-train dependencies like turn-arounds and connections, as well as infrastructure information to route exclusions, minimum headway times and train priorities to the model.

Out of the input information a so called activity graph is defined to formalize the occurrence of primary delays and their propagation. Each node of this graph represents a train activity for each train line and station and the four types of activity: entry signal reached (A^*), arrived (A), ready to start (D^*), departure (D). Between those nodes, links are defined to model trips, stops, interlinkings between lines and route conflicts. Cumulative distribution functions, elaborated and described in further detail in [Bueker, 2010], are used to model primary delays. They allow an adequate compromise between modelling delays as precisely as possible and having short computation times. Subsequently

conditional and unconditional convolutions together with “excess beyond” operations, to avoid trains getting ahead of schedule, are used to propagate delays along the train activity graph. During this process further input influences this propagation: Running and dwell time supplements, for example, reduce delays along running and stopping activities, buffer times reduce secondary delays between different trains. Furthermore, maximum waiting times and minimum headway times have to be taken into account in case of connections and conflicts.

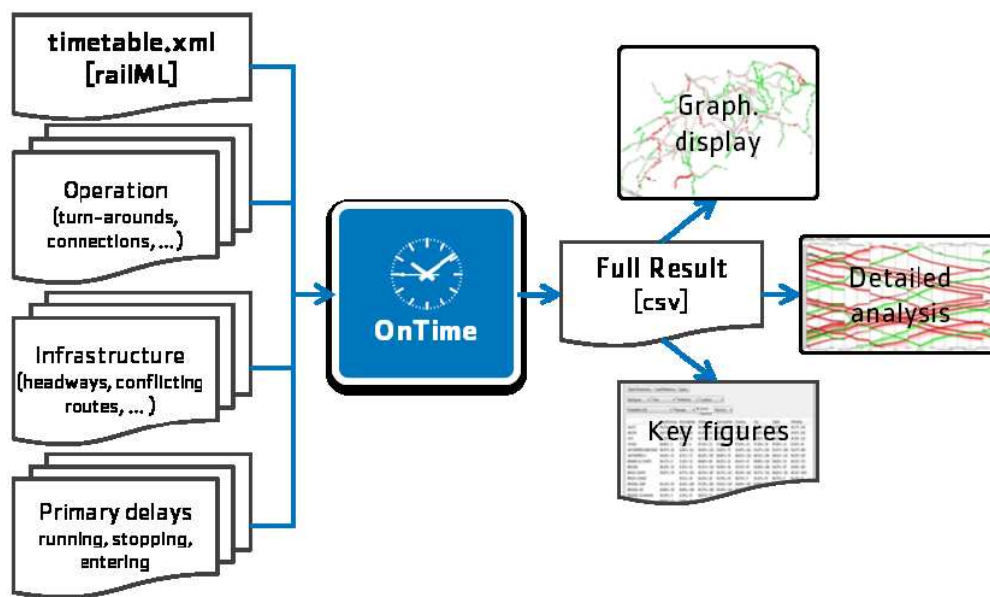


Figure 2.13: *Input and output of OnTime [trafiT solutions]*

OnTime offers different output possibilities to summarize and visualize the evaluation, as illustrated in Figure 2.13. A geographical map using a predefined color scale shows critical regions representing the average delay for all arrival events at every station. A timetable graph allows to illustrate individual delays for every train over time slots around the original scheduled train. Furthermore, a table offers to display different stochastic measures (expected delay, punctuality at different levels and quantiles for arrival and departure times) for individual trains, operations points or time windows. Providing these different types of quality measures, the software strongly simplifies comparisons of different timetables. OnTime is already in use at the Swiss Federal Railway (SBB) and Infrabel to study different timetable variants in practice. Short computation times, also for national problem sizes, allow to evaluate timetables iteratively.

2.4.3 Support for automatic timetable construction

In this section, we briefly present two already existing software tools using models and algorithms similar to the one studied in this thesis. The first one, called *Design of Network Schedules (DONS)* is used by the Netherlands infrastructure management ProRail, as well as by the principal railway operator NS. The second one, called *TAKT* is used by the

German railway company (Deutsche Bahn AG). For both software tools we describe some milestones of their development, their main functionalities and give a short comparison of their mathematical methods to model and solve timetabling problems, further discussed in Section 4.2.

DONS

Starting from 1990, new potentials to apply methods from operation research in the planning process won the attention of Netherlands railway company NS. They motivated the NS to launch several research collaborations with universities and to implement methods as parts of the DONS project starting from 1992. The main goal was to provide long-term planners with a tool to generate basic one hourly patterns for a periodic timetable in less time than it would take to construct them by hand [Kroon, 2008].

A close collaboration between planners and OR experts together with the existence of a central database containing timetable and rolling stock schedules contributed to the success of DONS. It could be developed efficiently and won the confidence of planners in a short time. First timetabling studies began at the end of 1996 [Odijk et al., 1997]. Shortly thereafter it was planned to use the new techniques to construct a completely new timetable from scratch for the whole railway system of the Netherlands for about 5500 daily trains. This huge modification of the current timetable, with its roots in 1970, was seen as the only action able to realize their long-term goal of running more trains on the entire railway system and improving punctuality at the same time [Kroon et al., 2009].

The decision of using DONS for the timetable construction made it possible to add further goals as a simplification of timetable memorability and also an improvement of connections especially to the neighboring states Germany and Belgium. In December 2006 the new Dutch timetable came into operation. It could facilitate the growth of passenger and freight transport and improve the robustness of the timetable, leading to a reduction of operational costs. First extrapolations after the introduction ensured an additional profit of 40 million Euros increasing to 70 million Euros annually together with a modal shift forwarding the reduction of emissions [Kroon et al., 2009].

The DONS software consists of three core parts [Peeters, 2003]. The first part contains a graphical user interface which allows planners to specify a timetable instance and to visualize outputs. A glimpse at this user interface gives Figure 2.15. In addition, a database is attached saving all information to various problem specifications entered by planners together with a detailed description of the infrastructure. Then, a second part, called CADANS, models and solves a macroscopic periodic timetabling problem based on ideas elaborated in [Schrijver and Steenbeck, 1994]. This basic hourly pattern for a macroscopic periodic timetable serves as an input for the third part of DONS, called STATIONS. In this third part, a microscopic timetable routing trains through station areas based on arrival and departure times of the second part is determined, using mathematical techniques elaborated in [Zwaneveld and Kroon, 1995].

Furthermore, the DONS software provides interfaces to a software allowing simulation of a macroscopic entire timetable for a whole railway system, called *Simulation Model for Networks (SIMONE)*, [Middelkoop and Bouwman, 2001] and two further tools supporting rolling stock and crew scheduling (ROSA, [Fioole et al., 2006] and TURNI, [Kroon and Fischetti, 2001]).

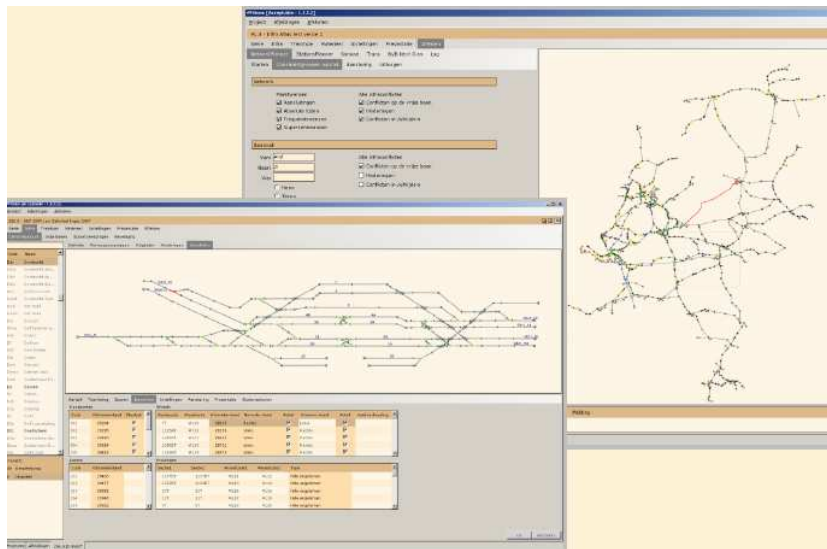


Figure 2.14: A glimpse at the model construction of DONS. [Middelkoop, 2010]

CADANS, as the second core part of DONS, contains models and algorithms similar to the one studied in this thesis. It is also based on the Periodic Event Scheduling Problem, which we will introduce in Chapter 3. However, it uses a different approach to solve the models, which for example requires fixed trip times between nodes of the model and only allows a restricted optimization of timetables. The solution method is based on Constraint Programming and is further explained in Section 4.2.1. After finding a first feasible solution, if one exists, a postoptimization using the commercial LP solver CPLEX is used while fixing the main structure of the timetable (sequence of trains).

TAKT

With the dissertation of Opitz on the automatic construction and optimization of periodic timetables [Opitz, 2009], the group of traffic flow science at TU Dresden started a collaboration with the German infrastructure management (DB Netz AG). This combination of deep mathematical knowledge, also from previous research of the chair holder Nachtigall [Nachtigall, 1998], and experience from practice lead to the development of the software TAKT. The software TAKT accelerates and facilitates the construction of a representative set of periodic timetable scenarios to improve the evaluation of different infrastructure measurements. It was already successfully tested for several regions in Germany, as well

as for different train types, such as local and long-distance trains and freight trains. With this experience and ongoing research projects, the development is still in progress today (2014).

As the CADANS solver of the DONS project and the approach of this thesis TAKT is also based on the PESP. But in contrast to the other two approaches, the routing through station areas already takes place in advance. In fact, all train itineraries are supposed to be fixed on a microscopic level before starting the construction of the periodic timetable. For this itinerary fixation the group of traffic flow science developed an own independent tool. As in CADANS, trip times are supposed to be fixed and a feasible timetable is constructed in a first step before starting to optimize it in a second step. To deal with larger problem instances a new solution method was developed based on a reduction to the SAT problem, further described in Section 4.2.2. This new solution method also allows the modelling of a symmetric periodic timetable which is not possible using the constraint programming approach used in CADANS. In case of infeasibility the algorithms are allowed to relax dwell and connection times for passengers. To keep this relaxation as small as possible the overall sum of all dwell and connection times is minimized in a postoptimization step afterwards. For this optimization, originally based on a modulo network simplex method [Nachtigall and Opitz, 2008], the development of new approaches still is part of current research at the chair.

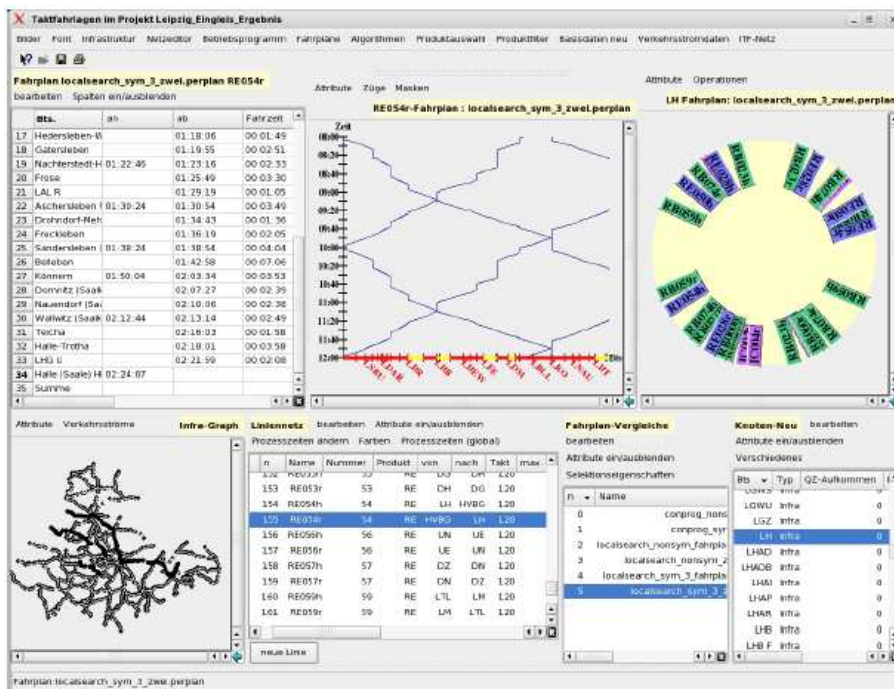


Figure 2.15: Glimpse at the software TAKT [Nachtigall, 2014]

2.5 Embedding of the algorithms for automatic timetable construction

The algorithms discussed in this thesis concentrate on the automated construction of periodic railway timetables and mainly support stakeholders during the mid-term planning phase of the planning process. In this section, we further describe the embedding of the algorithms in the planning process and their value for the different use cases and planning horizons considered. Subsection 2.5.1 describes the common application of the algorithms as an integration in the mid-term planning phase. Subsequently, further use cases in the short-term and long-term planning phase are discussed in Subsection 2.5.2, followed by the integration of the algorithms into already existing planning software in Subsection 2.5.3.

2.5.1 Mid-Term planning

During the mid-term planning, timetable creation starts to become more concrete. The infrastructure together with block sections necessary for the safety system are fixed and train operating companies determine their specific service intentions. Thus the main question in this planning phase is whether all desired service intentions can be realized with the given infrastructure settings and how a resulting timetable could be optimized. With the increasing capacity of the railway networks and the ongoing liberalization of the railway market, these questions become more and more complex to answer. Faster and preciser decisions are necessary. The studied algorithms help to improve considerably the timetable creation.

Figure 2.16 shows an overview of the input and output of the algorithms. Depending on how detailed train itineraries are already fixed at the given time, a corresponding network of nodes and links has to be defined. For each link, the used track of every train has to be known. Nodes can be used to aggregate small parts of the infrastructure and therefore to leave open detailed train itineraries as for example in main station areas. For each track of every link, train and track specific line headways have to be known. In addition, headway times at stations and junctions can be added for specific pairs of trains using common or overlapping infrastructure elements. Furthermore, minimal trip and dwell times have to be known for every train, link and node.

From a functional point of view, all train lines and their routes through the defined macroscopic network of nodes and links have to be fixed. Furthermore, adequate upper bounds for possible trip and dwell times have to be defined. To improve the offer for passengers, connections and train separations can be added. For each connection a station, an arriving train and a departing train has to be chosen, as well as a lower and upper bound for a desired transfer time has to be fixed. Using an objective function minimizing total passenger travel time, it is possible to choose the upper bound for all dwell, trip and connection times more generously, so that the risk of overdetermining the modelled system can be reduced and the algorithms still find a best timetable concerning

the average journey times. In case of the train separation constraints, two or more trains have to be chosen. They should be separated by a certain amount of time. Using a corresponding lower and upper bound for the separation constraints, it is possible to allow a certain flexibility of some minutes, or one can fix a very strong separation using equal lower and upper bounds. From a functional point of view, also symmetry constraints (strong symmetry, or again with some flexibility) can be added or even an integrated fixed interval timetable can be modelled. In general, it is possible to fix arrival and departure times of chosen trains in advance to a given time margin. Depending on the stakeholder and their competence working on a timetabling problem, some train lines can be fixed in small time margins or even completely. Further details on the modelling of a timetable scenario can be read in Chapter 3.

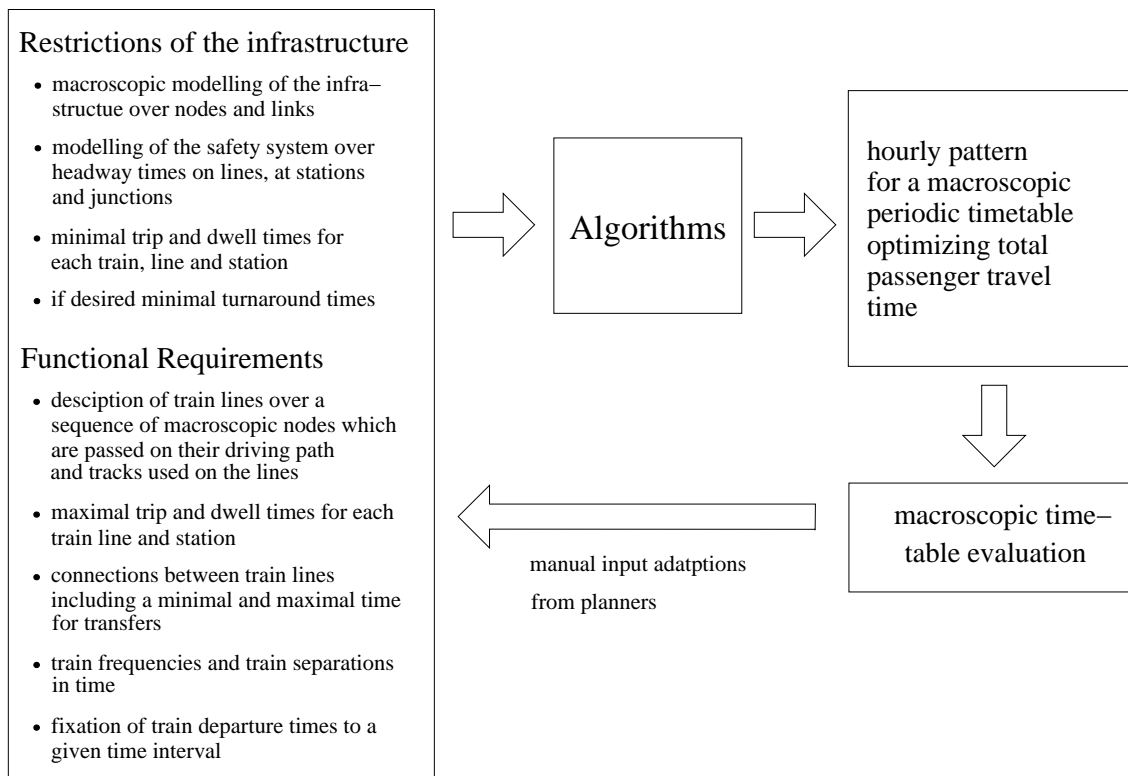


Figure 2.16: Overview of the input and output of the considered algorithms

As soon as the timetable scenario, and with this the input, is fixed, the algorithms compute a corresponding hourly pattern for a macroscopic periodic timetable optimizing total passenger travel time or another objective function, further discussed in Section 3.3. A subsequent macroscopic timetable evaluation, as for example over a software like On-Time (see Section 2.4.2), summarizes different parameters describing timetable quality. Comparing the average delay of trains at different stations and examining the average forecasted delay and their propagations of single trains, it is possible to find the most critical parts of the timetable concerning timetable stability and delay propagation. Specific input adaptations, as for example a selective adaptation of running time supplements

and buffer times together with an iterative use of the algorithms, can be used to subsequently improve timetable quality. A similar iterative adaption can also be used to adapt functional requirements. So for example prioritizations for connection and train separation constraints could be varied, or even different set of train lines and different timetable scenarios in general could be compared.

2.5.2 Further use cases in short-term and long-term planning

In addition to the mid-term planning phase, the studied algorithms can also be used starting in the long-term planning phase and even up to the short-term planning phase. So, for example, in the long-term planning phase infrastructure adaptations play a more important role. With the given algorithms it is possible to evaluate infrastructure measurements and their gain of capacity. Changes of infrastructures for a single corridor can have consequences far outside the corridor. Using an adequate size of the model, the algorithms can bring an essential improvement to such evaluations.

As soon as an hourly pattern for a periodic timetable is fixed after the mid-term planning phase, this period can be rolled out over an entire day and adapted for peak and off-peak hours. Subsequently vehicle and crew scheduling usually follow. An adequate refining of the itineraries after the mid-term planning phase and first fixations of rolling stock circulations allows the use of the algorithms even for the short-term planning. The objective function can easily be adapted to minimize the number of vehicles. The granularity of the infrastructure model and with it the level of detail for the description of all itineraries thereby plays an important role and should be chosen carefully corresponding to the current planning stage.

2.5.3 Integration into planning software

Currently used planning software to support the manual construction of timetables could directly be connected to the algorithms to deliver all technical data needed for the input and to visualize the resulting timetables. Over an additional user interface the control of the algorithms could be simplified, illustrating restrictions on the infrastructure and functional requirements graphically in extended topology and line maps.

The introduction of such algorithms makes it possible to greatly accelerate the construction of periodic timetables. Timetable planners are released from technical work and can concentrate on the comparison and development of different timetable scenarios in a much shorter time than before. Together with the modelling of an adequate objective function and timetable evaluation software, the application of such algorithms can improve timetable quality considerably, as already shown in [Kroon et al., 2009] and [Liebchen, 2008].

The two software tools introduced in Section 2.4.3, both support the construction of periodic timetables as well based on the same model introduced in Chapter 3, but use different solution strategies to construct and optimize a timetable. The difference of the

algorithms is further discussed in Chapter 4 and is mainly given through the fact that the two other methods do not include a direct optimization of the timetables and do not allow flexible speed profiles.

3 Modelling cyclic railway timetables

3.1 Introduction

To model periodic railway timetables the so called *Periodic Event Scheduling Problem* (PESP) gained attention in literature in the last two decades. It was originally introduced by Serafini and Ukovich [Serafini and Ukovich, 1989] and further studied in several PhD works, research papers and practical applications to which we will refer further on. Compared to alternative methods as the introduction of additional periodic constraints in non-periodic timetabling models [Wong et al., 2008] and a quadratic semi-assignment problem formulation, several times applied in the nineties of the last century [Klemmt and Stemme, 1988], [Daduna and Vo, 1993], the PESP seems to be the most appropriate model to describe periodic railway timetabling problems [Liebchen, 2006] and is therefore also used for this thesis.

Section 3.2 introduces the PESP, its general idea known from literature and all notations as used further on in this thesis. Subsequently, Section 3.3 discusses the extension of the model to an optimization problem over different objective functions. Section 3.4 gives further information on the fixation of different test models for this thesis and a description of its underlying data.

3.2 The periodic event scheduling problem (PESP)

In this section the periodic event scheduling problem is introduced. A first subsection is dedicated to the general idea of the model and the definition of the PESP decision problem. Subsequently different types of PESP constraints are explained in Subsection 3.2.2 and a visualization of the PESP model over a PESP graph is introduced in a next subsection. The section on the PESP model is rounded off with a discussion on different choices of model granularities.

3.2.1 General idea

The core part of the PESP is formed by a set of periodic events V , which is scheduled. In the case of a railway timetable, the set of events corresponds to all arrival and departure events of every train on its itinerary through a predefined network of locations (nodes). Depending on the chosen model granularity, this network can consist only of a set of important stations or it could include even operation points outside of commercial

functions down to a granularity of switches. A more detailed discussion on the choice of model granularity is given in Section 3.2.4.

Between every pair of event $v_i, v_j \in V$, $v_i \neq v_j$ a constraint $a = (v_i, v_j)$ can be defined describing a minimum $l(a)$ and maximum $u(a)$ amount of time which has to pass between the two events. A feasible schedule then is a function $\pi : V \rightarrow [0, T)$ which assigns a time $\pi_i \in [0, T)$ to every event $v_i \in V$ satisfying all constraints. The constant T denotes the main period of the periodic timetable, which in practice often is 60 or 120 minutes. A constraint $a = (v_i, v_j)$ is satisfied if

$$l(a) \leq (\pi_j - \pi_i) \pmod T \leq u(a).$$

The PESP is the decision problem of whether there exists a feasible schedule for a given set of events and constraints.

Definition 4. Let $G = (V, A)$ be a directed graph, $T \in \mathbb{N}$ a constant and $l, u : A \rightarrow \mathbb{R}$ two functions with $l(a) \leq u(a), \forall a \in A$. Then we define $I = (G, l, u, T)$ to be an instance of the *PESP decision problem* of whether there exists a feasible schedule $\pi : V \rightarrow [0, T)$ with $\pi(v_i) := \pi_i, \forall v_i \in V$ satisfying $l(a) \leq (\pi_j - \pi_i) \pmod T \leq u(a), \forall a \in A$.

With the help of constraints $a \in A$ we are able to model various types of different functional requirements and infrastructure restrictions a desired periodic timetable π has to satisfy. Section 3.2.2 introduces different types of constraints used in this thesis. The directed graph $G = (V, A)$ we will call *PESP graph* and its visualization is further discussed in Section 3.2.3.

3.2.2 Types of Constraints

In this subsection, we describe different types of constraints which are common to be used to model a periodic timetabling problem over a PESP instance [Liebchen and Möhring, 2007]. We distinguish three main categories of constraints: modelling of train movements, safety system and functional requirements.

Modelling of train movements

For every given train line with a frequency of exactly the main period T , one train in each direction is included in the model. If a train line has a higher frequency than the main period, a corresponding number of copies of both trains are used. For each train a sequence of events is defined corresponding to its itinerary through a modelled network of nodes. Each sequence starts with the departure event at the train's first node, followed by an arrival and departure event of every consecutive node on its itinerary and ends with the arrival event of its last node.

trip constraints Between each consecutive departure and arrival event of a train, a trip constraint is defined. The lower bound for this constraint depends on the corresponding rolling stock and its minimum driving time (technical driving time) for the given path

section. To the minimum driving time running time supplements are normally added to guarantee a certain degree of robustness. The sum of the minimum driving time and the running time supplement then leads to the lower bound for the trip constraint. In most PESP models found in literature the upper bound for the trip constraints equals the lower bound, modelling a fixed driving behaviour for the given path section. This also simplifies the safety system as discussed in the next category of constraints. Using a more sophisticated safety system (see also Section 4.3.3) it is possible to use upper bounds for trip constraints which are larger than the lower bounds. They then allow a certain flexibility in the driving behaviour and the distribution of additional running time supplements over a sequence of path sections. With the addition of an adequate objective function to the whole model, as discussed in further details in Section 3.3, trains will not be slowed down unnecessarily, especially for densely occupied path sections. Thus the upper bound for a trip constraint in this case can be rounded up generously from a modelling point of view. Taking into account solution time for algorithms, too large intervals still have negative impacts on computation times.

dwelt constraints Similar to the trip constraints further constraints are added between each consecutive arrival and departure event to model a minimum and maximum dwell time a train is allowed to stop at a node of its itinerary. If a node, for example, represents a station with a commercial stop, the minimum dwell time then includes a minimum time required for all passengers to board the train and alight from it plus additional time needed to do all duties from train operation necessary to stop and start a train again. An upper bound of time which is still reasonable for passengers to stay in a train at this station leads to the corresponding upper bound for this dwell time. On the other hand, if a node represents an operation point without commercial function and a train is not thought to stop at this node, the minimum and maximum dwell time both are set to be zero. If a node without commercial stop can be used to stop a train, for example for crossings and overtakings, this has to be taken into account in a corresponding larger maximum dwell time. A minimum dwell time of zero minutes in this case can still allow a train to pass the node, if the overtaking and crossing takes place at another node. Using adequate constraints for the safety system the model ensures large enough stopping times in case of an overtaking or crossing. And unnecessary dwell times, if a train passes the node, can be avoided using an adequate objective function.

turnaround constraints If rolling stock circulation is already fixed, turnaround constraints can be used to ensure a minimum time needed from train operation at the last station of a train line to prepare the train for its next trip. It connects the last arrival time of a train at its terminal station with the departure time of the corresponding train running at its first station. Using a large enough turnaround time in combination with an objective function minimizing all turnaround times we can ensure to not overdetermine the system and having good turnaround times at the same time.

overall constraints With an overall constraint we denote a constraint connecting a departure event of a train with an arrival event of the same train several nodes later on its

itinerary. They can be used to fix a minimum and maximum overall trip time and therefore to force a certain amount of total running time supplement for a longer path section.

frequency constraints If a link has a higher frequency than the main period, several trains are included into the model having the same trip, dwell and overall constraints. To ensure a proper frequency of the corresponding train line, in this thesis frequency constraints are added between every pair of commercial departure events corresponding to the same station. They have equal lower and upper bounds both set to the main period T divided by the frequency.

Modelling of the safety system

To ensure restrictions from the safety system, different types of headway constraints are used.

headway constraints on the line If two trains running in the same direction use the same track section between two nodes on their itinerary, a headway constraint is added connecting the departure event of the first train with the departure event of the second train at the starting node of this track section. If both trains have a fixed driving behaviour (equal lower and upper trip bound), it suffices to require a large enough time distance between the two trains at the beginning of the track section. Thus the minimum headway time in this case is the line headway we get by pushing the second blocking-time-stairway after the first one as close as possible to the first one. And the maximum headway time is the main period T minus the line headway we get when running the first train as close as possible after the second train. If the two trains have flexible trip times, the safety system gets more sophisticated. A solution idea to this problem is discussed in Section 4.3.3. For trains using the same track section in opposite directions, a headway constraint between the arrival event of the first train at one end node of the section and the departure event of the second train at the same node can be introduced. Similar to the first case, the lower and upper bound are defined over approaching the corresponding blocking time stairways. Also in this case an adequate adaption for flexible running times might be needed as discussed in Section 4.3.3.

headway constraints at stations If there are conflicts between certain train arrival and departures in a station, they can be avoided by using adequate headway constraints between the corresponding arrival and departure events of these trains. As in case of the headway constraints on the line, both sequences have to be taken into consideration to determine the lower and upper bound of the constraint. If the constraint is directed from the first to the second train, the lower bound is given by the smallest time distance the second train is allowed to enter or leave the station after the first one. And the upper bound is the difference between the main period T and smallest time distance the first train is allowed to enter or leave the station directly after the second one.

headway constraints at junctions If two trains have to pass a junction with a certain time distance, this requirement can be modelled with a headway constraint between

the departure or arrival events of these two trains. Since in the case of a junction the departure and the arrival event both get assigned the same time (the dwell time at a junction is zero), it does not matter whether the arrival or departure events are used. The minimum and maximum bounds of the constraint are defined as in the case of the headway constraints at stations.

Modelling of functional requirements

In addition to train movements and the safety system, further constraints can be added to a PESP to model functional requirements, such as adequate connection constraints, a passenger-friendly distribution of trains offering similar services and further time constraints fixing already known departure events of some trains for example for trains of a higher planning hierarchy.

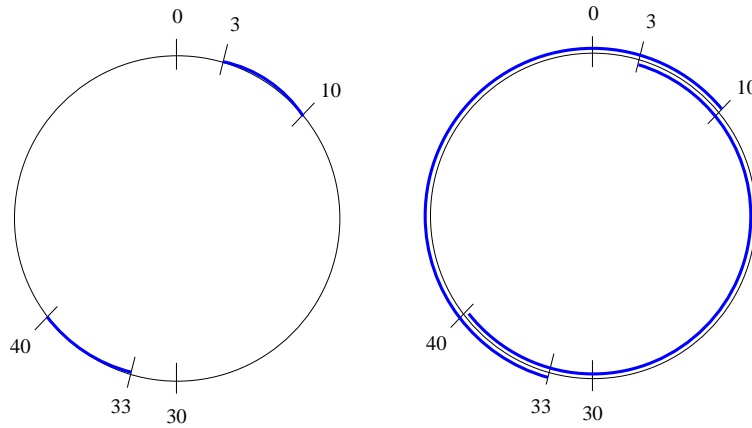


Figure 3.1: An example illustrating the union of time intervals for PESP constraints using multiple constraints. The union of the two time intervals $[3, 10]_{60} \cup [33, 40]_{60}$ (left) can be modelled over the two overlapping intervals $[3, 40]_{60} \cap [33, 70]_{60}$ (right).

connection constraints A connection constraint connects the arrival event of a train in a station with the departure event of another train in the same station. Its lower bound is fixed corresponding to a minimum time needed by passengers to change from the first to the second train. Its upper bound, similar to the turnaround constraints, can be rounded up very generously if an adequate objective function, ensuring the use of a larger connection time only if necessary, is used. If the connection of a train leads to a train line with a higher frequency than one train per main period the introduction of multiple connection constraints can help to leave the decision open to which train of the train line the first train is connected. As Figure 3.1 indicates, multiple constraints can be used to model the union of different time intervals. Thus, if we want to offer a connection with lower bound c_l and upper bound c_u from a first train to a group of two trains of a second train line with frequency two, we can introduce two connection constraints between the arrival of the first train at this station and the departure of one train of the second line at this station. One constraint a_1 with $l(a_1) = c_l$ and $u(a_1) = c_u + \frac{T}{2}$ and a second constraint a_2 with

$l(a_2) = c_l + \frac{T}{2}$ and $u(a_2) = c_u + T$. If the frequency is larger than two, the idea can be adapted to a set of a larger number of intervals using a corresponding larger number of constraints. And if we want to offer a connection between two train lines both having a higher frequency, it suffices to consider one train of the first train line.

time dependency constraints If two trains belong to different train lines but share a similar offer for a certain part of their route, the introduction of time dependency constraints can help to separate these trains to improve the overall offer for passengers. A time dependency constraint is introduced between the departure event of the first train at the first station of the common itinerary and the departure event of the second train at the same station. If the two trains should be separated exactly by half of the period, both, the lower and the upper bound of the constraint, is set to $\frac{T}{2}$. Sometimes this constraint can be relaxed arbitrarily with an amount of c_{flex} minutes choosing a lower bound of $\frac{T}{2} - c_{flex}$ and an upper bound $\frac{T}{2} + c_{flex}$. Especially if the two trains do not share exactly the same driving behaviour, the introduction of some flexibility makes sense. If the trip and dwell constraints of both trains allow a very strong separation and we want to ensure for it, we can repeat time dependency constraints for every common node, otherwise it suffices to fix them for larger commercial stations on the common itinerary.

time window constraints If we want to fix the departure of a train at a certain node to a given time or to a certain time interval, we can introduce time window constraints. They require the introduction of an additional event, called *zero time event*. It is a new event without any connection to a train line. Let us denote this event by v_0 . In contrast to the remaining events, we fix the scheduled time for v_0 already in the beginning to $\pi_0 = 0$. Introducing a constraint starting in v_0 and reaching to a certain departure event of any train line at a given station, allows fixing the departure of this train at the corresponding station. The lower and upper bound of the constraint equals the lower and upper bound of the desired time window for the train departure.

symmetry constraints Depending on the solution method used to solve the PESP, it is possible to require a symmetric periodic timetable or even an IFIT. This can be modelled for example over the introduction of so called symmetry constraints, starting at the zero time event v_0 and leading to all time events of a train. Their lower and upper bounds are set to 0 and T and therefore do not restrict the solution of the model. But they are used to ensure the symmetry property of a symmetric periodic timetable. In case of a strictly symmetric timetable the time distance between event v_0 and the two departure events of a train in both directions at a station has to be equal. Such a restriction for example can be easily included in the cyclic MILP (see Section 4.3.2) over the addition of a constraint ensuring that tension variables of the corresponding constraints have to take exactly the same value (strictly symmetric timetable) or the same value with a small amount of flexibility (relaxed symmetric timetable).

3.2.3 Visualization of a PESP graph

A PESP model can be visualized as a directed, weighted graph called *PESP graph*. In this subsection we introduce the PESP graph and illustrate its visualization over two small examples.

Let $G = (V, A)$ be the directed graph of a PESP instance $I = (G, l, u, T)$. Then we define the constraint bounds $l(a)$ and $u(a)$ for each constraint $a \in A$ as weights for the constraint and add them as closed intervals $[l(a), u(a)]_T$ describing the allowed time modulo the main time period T , which has to pass between the two events corresponding to the two end vertices.

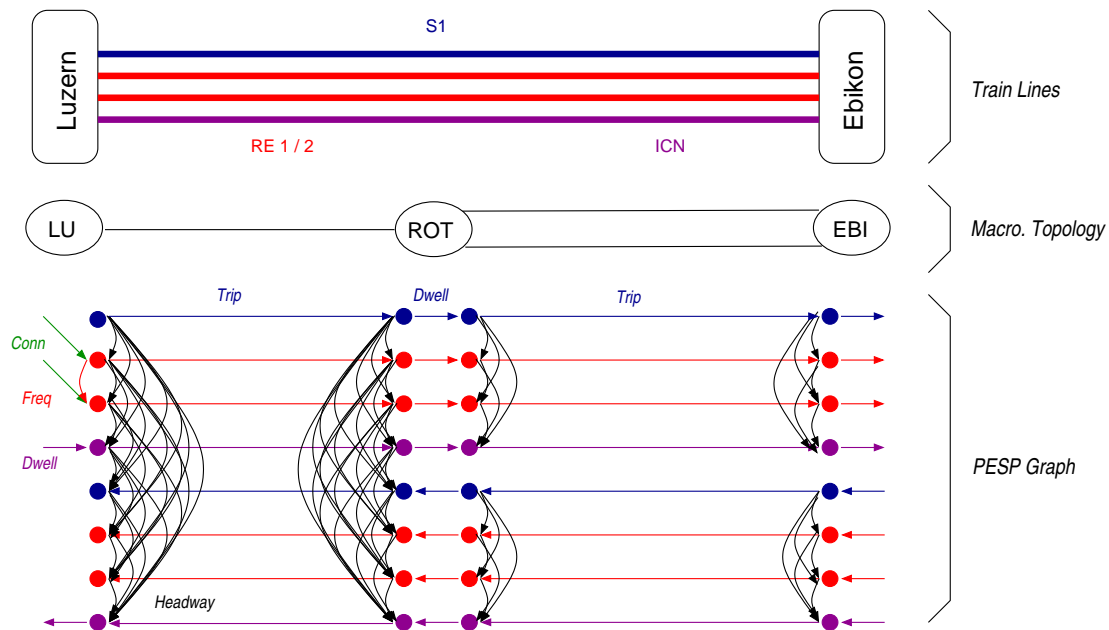


Figure 3.2: Small example of a PESP graph illustrating its typical structure

The illustrative example of Figure 3.3 shows a simple PESP graph for two train lines t_1 and t_2 each connecting three stations. The two train lines both serve the two stations A and B and then have a third stop C, respectively D, separated from each other. Between A and B there are two tracks and trains are separated by their direction. Therefore there are headway constraints between trains of train line t_1 and t_2 running in the same direction. For each train line and direction we have a departure event in its first station and an arrival event in its last station, as well as both types of events in each inner station. For each vertex in the PESP graph of Figure 3.3 a short description to the corresponding event is added, describing its type (dep for departure, arr for arrival), its train line (first digit of t_{xx}) and direction (second digit of t_{xx} , 0: first direction, 1: back direction), as well as the corresponding station. Departure and arrival events of the same train at consecutive stations on its line are connected over trip constraints and arrival and departure events of the same station and train are connected over dwell constraints. Thus for example (v_1, v_2) is a trip constraint with a lower bound of 4 and an upper bound of 5 minutes and (v_2, v_3) is a dwell constraint with bounds 1 and 2. Constraint (v_3, v_6) is a headway

constraint modelling a line headway of one minute independent of the sequence of the two trains. All constraints adjacent to v_0 , the separated zero time event, are time window constraints fixing a departure event corresponding the given interval bounds. The two constraints (v_{14}, v_3) and (v_6, v_{11}) model a connection offered for passengers travelling between station C and D. As illustrated in the case of v_1 and v_9 there could be also more than one constraint between two vertices. Between v_1 and v_9 there are two constraints. A first one with bounds $[2, 58]_{60}$ is representing a headway constraint and the second one with bound $[25, 35]_{60}$ a time dependency constraint separating the two lines t_1 and t_2 on the common line A-B.

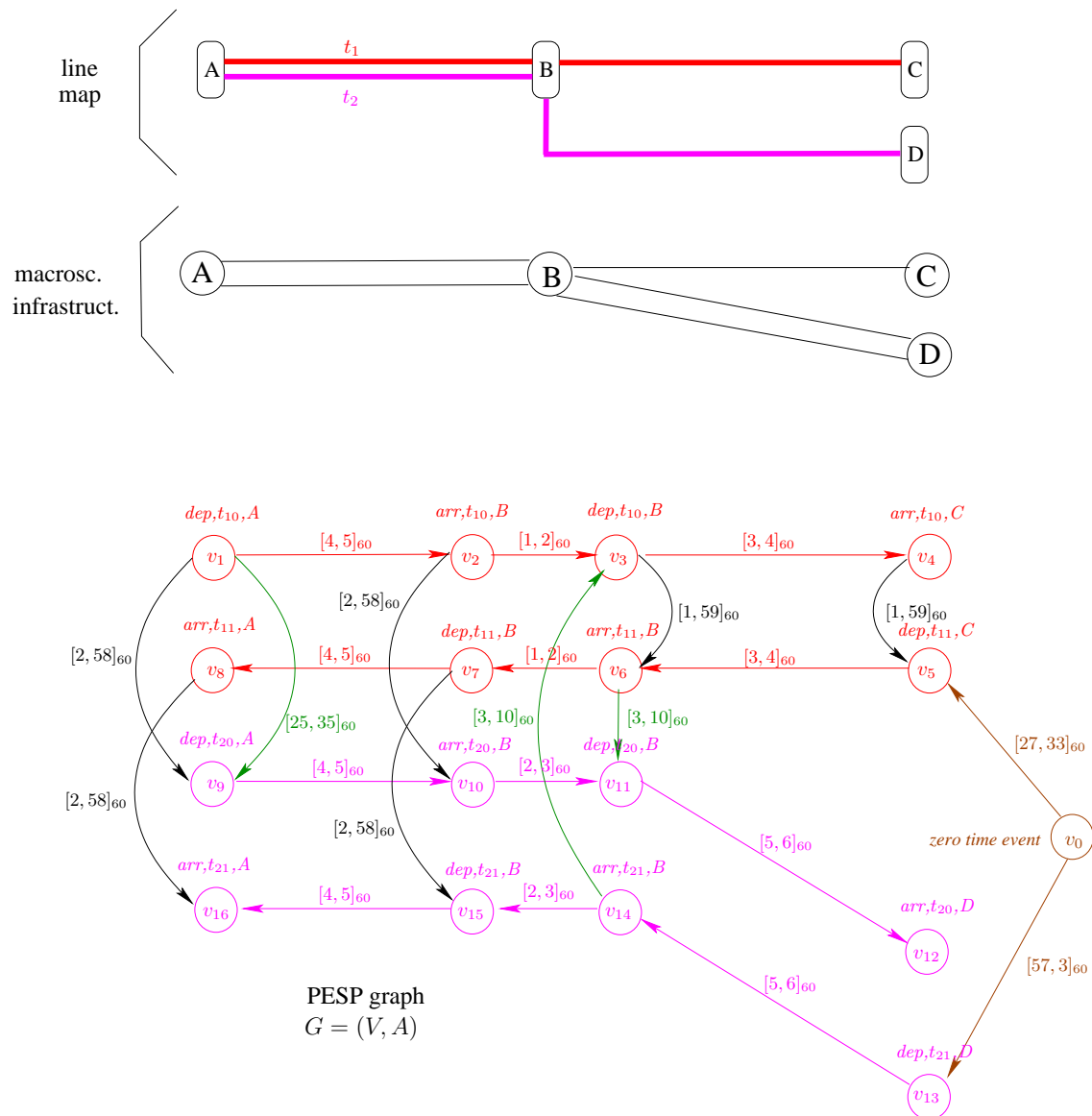


Figure 3.3: Illustrative example of a PESP graph with two lines, four stations and some examples of different constraint types

As soon as the number of trains using the same track grows the typical structure of a PESP graph modelling a dense railway network becomes apparent. Since normally the decision on the sequence of every train is not fixed in advance, a huge set of headway constraints connects all departure and arrival events of these trains at the end nodes of the common track as illustrated in Figure 3.2. The number of headway constraints therefore clearly dominates all types of constraints of an average PESP model, also in this thesis (see Section 3.4).

3.2.4 Model granularity

In this subsection we discuss the choice of model granularity in further detail. The model granularity is determined through the set of nodes, which is used to describe the itineraries of all trains. From now on we call this set of nodes the *macroscopic network* for the PESP model. Driving paths of trains in a PESP model are described over a sequence of macroscopic nodes and the choice of a track between two consecutive nodes. Thus the more nodes are used in this network the more precisely we can describe itineraries and model the safety system. On the other hand, computational complexity grows with enlarging the number of nodes and a strong fixation of itineraries up to platforms in larger station areas could be too restrictive to find good timetables in a first step. Figure 3.4 shows a small example for two different model granularities around a larger station (Lucerne, LZ) in central Switzerland.

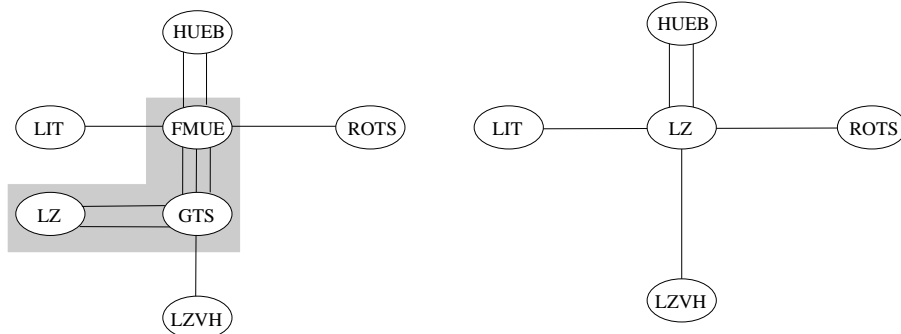


Figure 3.4: *Two different model granularities around main station Lucerne in central Switzerland.*

On the left hand side, two operational points (Fluhmuehle FMUE, Guetsch GTS) close to main station Lucerne are modelled separately in the macroscopic network, whereby on the right hand side these operational points are assumed to be part of the station area of Lucerne. In the more precise model on the left hand side, itineraries for all trains have to be known on the two links FMUE-GTS and GTS-LZ before constructing the timetable, whereby in the other case these itineraries are part of the macroscopic node LZ and therefore are not yet considered. Depending on the planning stage, the fixation of these itineraries in a first step could be difficult and disadvantageous excluding too many, and probably also better, solutions. Therefore, a successive adaption of the model granularity during timetable construction should be used. To reduce computational complexity, while refining the model granularity, we can start to fix driving behaviour over

fixed trip times and we can restrict departure and arrival times successively over time slots. As soon as we use fixed trip times and we know about detailed itineraries, we can even model a microscopic safety system, using headway constraints at stations requiring a large enough time distance for the departure event of two trains using conflicting infrastructure elements on their itinerary up to the next macroscopic node. The importance of using an appropriate modelling granularity at each stage of the timetable construction is also underlined in practice as for example in [Bickel et al., 2010].

3.3 Optimization Criteria

The classical PESP, as described in the last section, is a decision problem on the question whether there exists a set of event times satisfying all constraints and the fixation of an arbitrary timetable, in the positive case. In this section, we briefly discuss the addition of an objective function to the PESP, different optimization criteria and the advantages as well as limits of using objective functions.

Definition 5. Let $I = (G, l, u, T)$ be an instance of the PESP decision problem, Π the set of feasible schedules for I and $f : \Pi \rightarrow \mathbb{R}$ an objective function. Then the problem of finding a feasible schedule $\pi \in \Pi$ optimizing the objective function f is called *PESP optimization problem*.

3.3.1 Minimization of passenger travel time

A very commonly used objective function is the minimization of travel times for passengers normally taking into account dwell and connection times at stations, as well as trip times, if they are not fixed in advance. This type of objective function, with fixed trip times, is also used in [Nachtigall and Opitz, 2008] and [Nachtigall and Voget, 1996]. It represents the main interests of customers by providing short overall journeys. To further improve this goal, the demand for a specific connection can be included as a weight in the objective function. This means the minimization of connection times for frequently asked transfers are prioritized in the optimization. For this inclusion of travellers behaviour origin-destination-matrices (OD-matrices) are used. Unfortunately, they are often difficult to put into practice and furthermore are not completely independent of the current timetable. With continuous digitalization of railway ticketing at least the first problem could be solved in future.

3.3.2 Minimization of operational costs

Having a short look at railways commonly used planning process in Section 2.2.3 the main impact of a timetable on operational costs are given through the number of train compositions necessary to operate a timetable. Therefore, the minimization of rolling stock by minimizing turnaround times was already studied by different researchers, as for example [Lindner, 2000], [Liebchen and Möhring, 2007]. This type of objective function makes sense as soon as rolling stock circulation is known. Otherwise, if for example situations exist where two different train lines could share the same rolling stock, it would be disadvantageous to optimize turnaround times for each train line separately.

3.3.3 Maximizing timetable stability and flexibility

A third main category of objective functions, is given through optimization criteria maximizing timetable stability and flexibility. In combination with the construction of periodic timetables they are studied for example in [Kroon et al., 2006], [Liebchen and Stiller, 2008] and [Caimi et al., 2009a]. All three approaches are based on a sophisticated allocation of running time and buffer time supplements. Since such an allocation also leads to longer travel times, higher operational costs for train operators and less efficient capacity use for infrastructure managers, the assignment has to be done carefully. Furthermore the influence of later train dispatching in case of delays is difficult to include into such a model. Thus the application of timetable evaluations and simulations still play an important role to ensure timetable stability.

3.3.4 Advantages and limits for optimization

We have already mentioned two difficulties of objective functions, namely conflicting objectives and data which is difficult to get independently of the timetable itself. However, the introduction of an objective function for timetable construction belongs to one of the main added-value guaranteed by computer-generated timetable construction. Computer-supported timetable construction over an objective function allows the finding of a best timetable taking into account the given optimization criteria among all feasible timetables, whereby in manual timetabling the construction of one feasible timetable is normally the main focus.

The introduction of an objective function furthermore can help to improve the use of automated timetable construction itself. So for example it allows to relax constraints generously in case of infeasibility. The upper bound of every connection constraint a_{conn} can be enlarged to $u(a_{conn}) := l(a_{conn}) + T$, whereby the use of an objective function minimizing total passenger travel time still ensures to have best possible connection times. The same can be done with all constraints modelling functional requirements.

Since timetable stability and rolling stock circulation cause early be reflected in an objective function, timetable evaluations, simulations and back iterations will play an important role after a successful introduction of automated timetable construction and optimization. Furthermore, an adequate compromise between optimization quality and necessary computation time is further important for a successful introduction of the PESP optimization problem in practice.

3.4 Implementation and verification for this thesis

For the realization of our research goals, as described in Section 1.2, the construction of several adequate PESP models in different sizes is necessary. In this section, we describe in detail how the PESP models are constructed from given input data. After the description of data provided by SBB, which is underlying our models, in Section 3.4.1, the chosen model granularity (Section 3.4.2), the definition of all included constraint types (Section 3.4.3) and the used objective function are introduced. Section 3.4.5 gives an overview of all test instances and some specific properties. And Section 3.4.6 describes evaluations of our timetables to ensure conflict-freedom and an adequate average time reserve with the help of the macroscopic timetable evaluation software OnTime, introduced in Section 2.4.2.

3.4.1 Data sources

For the definition of our PESP models, SBB could provide us with four data sets, which are described further in the next paragraphs.

Trains A first data set, exported out of their timetabling software NeTS [Netcetera, 2014], includes all passenger trains, which were running on Switzerland's standard gauge infrastructure at an average weekday of the year 2011. For each of these trains, an internal train number together with a sequence of operation points, passed by the train on its scheduled journey is given. Furthermore, technical driving times between these operation points and minimum dwell times together with a minimum time for train dispatching at the operation points are provided. All used tracks on the line as well as commercial arrival and departure times of the timetable 2011 are included.

Line Headways To model the safety system, SBB provided us with a list exported from Viriato ([SMA, 2014]), which contains train and line-specific headway times. For each consecutive pair of operation points modelled in the infrastructure of Viriato, we have four values of minimum line headways, depending on the train types of two consecutive trains. These train types are collected in the two main groups of fast and regional trains, as introduced in Table 2.1.

Headway at Junctions Exported from OnTime, we furthermore have a list of junctions modelled as operation points in NeTS together with a headway time that two critical trains have to hold.

Connections For the inclusion of connections, we have got an internal list used by planners from SBB with connections between specific trains at larger stations distributed over a day. For each connection furthermore a priority level, estimated by planners, is given describing the importance of having optimal transfer times for this connection.

3.4.2 Chosen model granularity

With the given data from SBB, a maximum degree of model precision is fixed. It corresponds directly to the model granularity used to start the construction of an annual timetable also in practice. The model granularity given by the software NeTS from now on is called *mesoscopic*. For the definition of our test instances we slightly aggregate this model granularity to a *macroscopic* model granularity as described in Construction 1.

Construction 1. An operation point of the software NeTS is integrated in the *macroscopic* infrastructure of our model if

- a train starts or ends at this operation point,
- a connection constraint is defined at the corresponding operation point,
- the operation point is included in the list of junctions,
- trains are allowed to cross and overtake at this operation point,
- the number of tracks changes at this operation point,
- a headway time changes at this operation point, or
- the operation point is necessary to uniquely describe all itineraries on links.

Operation points where crossings and overtakings can take place are manually read out of graphical timetables [im Auftrag des BAV, 2011] and infrastructure maps [Spo, 2014] to the corresponding given timetable. With the described aggregation in Construction 1, the number of included operation points can be reduced by 40 to 50% for all considered regions allowing to minimize the number of events and constraints necessary to model the corresponding PESP. If there is one or several operation points left out in the macroscopic model, driving and dwell times are added up to a total driving time in between the two modelled operation points next to the aggregated ones.

Under the assumption that trains run with a constant average velocity over all macroscopic links, Construction 1 furthermore ensures that there will not be any conflict concerning the safety system or limitations for itineraries resulting from the aggregation. This can be shown with the following argumentation: Let A be a mesoscopic node which is aggregated in the macroscopic network and t_1, t_2 two arbitrary trains passing A on their itinerary. Out of Construction 1 we know that t_1 and t_2 share common tracks before and after A or they do not meet each other on the mesoscopic network around A . If they use the same tracks on their itinerary before and after A , we know that they do not cross or overtake at A and they have the same line headway between the two closest macroscopic nodes B and C around A . Thus ensuring the given line headway of both trains at B and C together with the guarantee that no overtaking and crossing can take place at A (see Section 4.3.3) and a constant average velocity on the macroscopic link $B - C$, the two trains do not have any conflict on the mesoscopic infrastructure level as well. Consecutively we can aggregate all mesoscopic operation points not included in the list of Construction 1. With this idea, conflict-freedom on the macroscopic level is carried out by Construction 1 over to the mesoscopic level.

3.4.3 Event and constraint definition

The set of events is determined by the set of trains and the given model granularity. Since the PESP models a basic pattern for one period of a periodic timetable, we have to select an adequate subset of trains out of the provided list of trains running distributed over an entire weekday. We only consider periodic trains and only as many trains as run in one period. Since Switzerland's periodic core part of the annual timetable is based on an hourly pattern, we use one hour as the main period of our PESP model. For the selection of trains, we build groups of trains which share the same sequence of commercial stops together with the same departure minute at their first stop in the commercial timetable. As soon as a group has at least ten trains, we assume that they run periodically and we take one train of this group as a reference train for our model. All trains belonging to a group with less than 6 trains are assumed to run irregularly and therefore are not integrated in our models. Figure 3.5 illustrates the number of trains which belong to a group of a specific size.

After this categorization of irregularly and periodically running trains, 8% of all trains running in the given timetable are left. Their groups are compared to already included groups. The number of common commercial stops and their departure minutes are compared. If a group shares more than 50% of their commercial stops with similar departure minutes (± 5 min) and the total number of trains in both groups together is smaller than 20, the group is supposed to be a part of a regularly running train and is not integrated in the model. If two such groups are found, which are supposed to belong to the same periodic train, the group with the longer itinerary is chosen for the model.

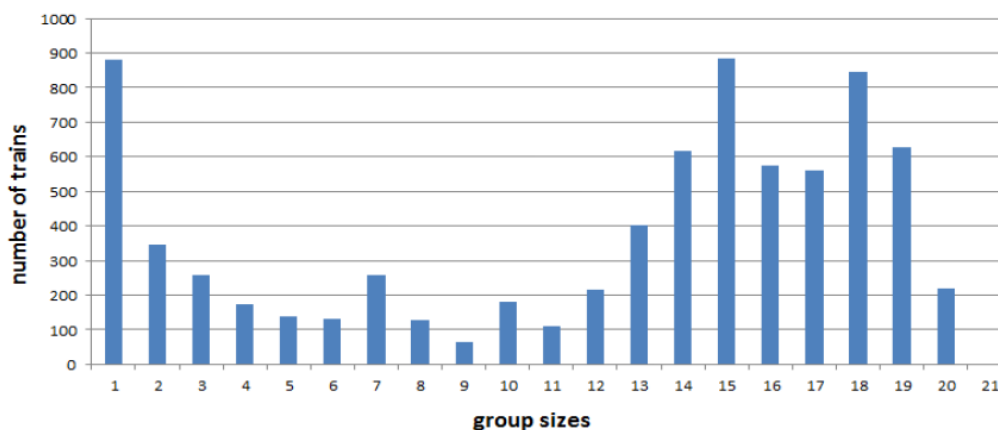


Figure 3.5: *Distribution of passenger trains into groups of identical offers running on a weekday on Switzerland's normal gauge infrastructure in the year 2011*

Having the set of reference trains for the model they are cut down to the corresponding region which is considered for a certain test instance. The region is defined over a set of macroscopic operation points. For each reference train, which passes at least two operation points of this region, a set of arrival and departure events is defined

corresponding to its entire itinerary through this region. All regions are chosen in such a way that no train enters and leaves this region twice.

The following paragraphs describe the introduced constraints:

Trip A trip constraint is defined for each pair of consecutive departure and arrival events of a train. The lower bound is the sum of all technical driving times, minimum dwell and train dispatching times of a train between the corresponding operation points plus a small trip time supplement defined over a parameter *minTripTimeReserve*. This time supplement normally takes a value of 5 – 10% of the minimum trip time. The exact choice for this parameter is further evaluated in Section 3.4.6. The upper trip time is chosen to be 1.5 times the lower trip time.

Dwell Dwell constraints, connecting each consecutive arrival and departure event of a train in a modelled operation point, do have a minimum bound defined as the sum of the minimum dwell and train preparation time. The upper value of a dwell constraint is 2.5 times the lower bound, if it is not defined as an exception. Exceptions are manually defined for some larger stations, where regional trains can stop for a longer time. In this case the upper dwell time is set to 12 minutes. Further exceptional cases are given through operation points where train crossings are allowed, but no commercial stop is planned. The upper bound is then enlarged from 0 to 7 minutes.

Frequency Comparing the sequence of commercial stops and the first departure minute of every train, trains and their back direction explicitly could be determined. If a train has a strict frequency of 30 or 15 minutes our algorithms find this regularity and automatically define frequency constraints modelling the same requirement for the PESP. They connect the departure events of two consecutive running, identical trains at the same station. Their lower and upper bounds are both set to 30 or 15 minutes. Predetermining the sequence of four trains, in case of a strictly periodic group of four trains with a frequency of 15 minutes, the system should not be overdetermined since the four trains are not further differentiated even for connection constraints.

Overall Overall constraints are used to require a minimum time reserve over a whole train run. They connect the first departure event of a train with its last arrival event. The lower bound is set to the sum of all minimum trip and dwell times plus a small time supplement (5 – 13%) defined over a parameter *minTrainTimeReserve*, further evaluated in Section 3.4.6. The upper bound is set to the lower bound plus 10 minutes.

Line headway Unfortunately, the network of operation points defined in NeTS and Viriato are not fully identical. In NeTS as well as in Viriato, there can be some additional operations points, which complicate a direct use of the given line headway times

from Viriato for our model. But over a definition of network components having identical headway times in Viriato and an adequate assignment of operation points from NeTS to these components a unique fixation of line headways for the entire network could be reached. For each pair of two trains using the same itinerary for a certain link two line headway constraints are defined connecting the departure and arrival events at both operation points corresponding to the end points of the link. Both headway constraints are directed from the train with the lower train number to the train with the larger train number to ensure identical directions of the headway constraints essential for the introduction of a safety model further explained in Section ???. This additional safety model is necessary to prevent collisions and overtaking when using flexible running times. Since the introduction of this additional safety model strongly depends on the chosen PESP solution method, further explanations are postponed to Chapter 4. The lower bound of the headway constraint is given by the minimum line headway time the first train, with the lower train number has to hold if it runs directly behind the second. The upper bound then is T minus the minimum line headway time the second train has to hold running directly behind the first.

Junction headway If there are overlaps of two infrastructure elements used by two trains at a junction, a corresponding additional headway is defined to ensure a minimum time distance of the two trains. Since headway times at junctions are independent of the trains and their sequence in the given data, the lower and upper headway times for each constraint at a junction are equal. The lower bound is the minimum headway time and the upper bound is T minus the minimum headway time.

Connection Since connections are given to specific trains of the given daily timetable, they have to be translated to the periodic trains chosen for the model in a first step. A connection constraint is therefore included into the model, if both trains of the connection belong to a group of trains which is included in the model and the corresponding station is part of the considered geographical region. The minimum bound of the connection time is given by an official list of minimum transfer times out of the annual timetable. They are specific for every station. We have fixed the upper bound to 35 minutes, as long as the problem instance was feasible. In case of the largest instance, defined in Section 3.4.5, we enlarged the upper bound globally to 63 minutes. For groups of trains running more than once an hour, multiple edges are used to model connection constraints avoiding unnecessary fixations of train sequences in advance.

Time dependency If two trains share a similar commercial offer, they are separated in time. To allow a certain degree of flexibility time dependency constraints are defined between the departure events of every third common commercial stop and a time tolerance of 5 minutes is added. This means for example in the case of the separation of two trains, that a lower bound of 25 and an upper bound of 35 minutes is used. If we want to separate four trains without fixing the sequence of these trains we introduce four constraints such that every departure event of the four trains is connected to two other departure events. In this case the lower bound is set to 10 and the upper bound to 50 minutes ensuring a train frequency of approximately 15 minutes with a maximal

deviation of 5 minutes. A basic set of trains sharing similar commercial offers is read out of the commercial timetable by the algorithms and some further trains are added manually in an iterative way, studying the results over graphical timetables.

Since we model a macroscopic timetable at an early planning stage, station headways, symmetry, time windows and turnaround constraints are not included in our model. This is the case. At this stage we do not want to fix itineraries inside a station and we furthermore do not have first fixations on rolling stock circulations. We do not fix first departure and arrival events simulating a sequential planning from practice and we decided to leave out symmetry constraints. For commercial reasons symmetry constraints would be forced in most practical applications. The decision to leave out them was motivated from computational side, since the introduction of symmetry constraints accelerates computation times ([Liebchen, 2004]) and we wanted to test the more difficult situation of not having symmetry constraints.

3.4.4 Choice of objective function

As an objective function we concentrate on the minimization of passenger travel time. We minimize a weighted sum of time assigned to every trip, dwell and connection constraint. For the different constraints we use weights, as indicated in Table 3.1. Unfortunately, we do not have data on precise passenger flows, but nevertheless with the given different priority levels for all connection constraints we can differentiate between important and less important connections. Let $x : A \rightarrow [0, T)$ be a function assigning the time $x(a) = \pi_j - \pi_i \bmod T$ to every edge $a = (v_i, v_j) \in A$, then we can describe our objective function as:

$$\min \sum_{a \in A_{Trip} \cup A_{Dwell}} x(a) + \sum_{a \in A_{Conn1}} 0.5x(a) + \sum_{a \in A_{Conn2}} 0.25x(a) + \sum_{a \in A_{Conn3}} 0.1x(a),$$

where A_{Trip} , A_{Dwell} and A_{Conn_i} determine the set of all trip constraints, dwell constraints, respectively all connection constraint of priority level i .

constraint type	weight
trip constraint	1
dwell constraint	1
connection constraint of priority 1	0.5
connection constraint of priority 2	0.25
connection constraint of priority 3	0.1

Table 3.1: *Used weights for the objective function*

Since we do not include rolling stock circulation we neither are interested minimizing rolling stock at this time. And since an introduction of sophisticated methods to optimize timetable stability would break the scope of this thesis, we use the two parameters on a minimum time reserve per train trip and train run for which we determine an adequate fixation by evaluating test instances with OnTime in Section 3.4.6.

3.4.5 Definition and description of test instances

For the research of this thesis seven test instances of different sizes are fixed. In Table 3.2 they are ordered by their size, given the number of PESP constraints, together with further key figures. Trains running in one period in each direction are counted separately and the number of operation points contains all operation points of our macroscopic model. About 67% of the constraints are headway constraints indicating a dense rail operation for several regions already for the group of periodic passenger trains.

No	Name	Events	Constraints	Trains	Op. Points	headway constr.
1	Eastern CH small	809	2310	63	74	1388
2	Lucerne	767	4598	61	37	3222
3	Thun	1289	5864	73	69	3980
4	Basel	1327	5953	96	58	3990
5	Thun-Basel	2275	9594	127	119	6354
6	Eastern CH	3017	12465	142	186	8368
7	German-speaking CH	7459	31847	294	407	21432

Table 3.2: Overview and key figures for introduced PESP models

Figure 3.6 shows the geographical region of every PESP instance and Figure 3.7 and Figure 3.8 visualize the macroscopic infrastructure and a line map for the German-speaking Switzerland instance.

3.4.6 Model evaluation with OnTime

To test our modelled safety system and to evaluate the influence of the two defined parameters `minTripTimeReserve` and `minTrainTimeReserve` (Section 3.4.3) on the average delay propagation, we introduce a connection of our algorithms to the software OnTime over a simplified RailML [Huerlimann et al., 2004] output. We roll out our basic hourly pattern for a periodic timetable and repeat it over six hours. Furthermore, we break down the model granularity from our macroscopic level back to the mesoscopic level including an arrival and departure time for every operation point of NeTS. Running time and dwell time supplements in this case are distributed equally over the whole macroscopic section and their corresponding trip and dwell times.

Besides a comfortable visualization of our timetables, OnTime can also offer several types of feedbacks to our models. In a first step, we concentrate on operational feasibility. Since OnTime can directly export a list of conflicts with its modelled mesoscopic safety system, we can easily check our macroscopic safety system. Especially with the model differences between operation points of Viriato and NeTS, the reduction to only two train categories for our line headways and the assumption of a constant average velocity for aggregation of several operation points in our macroscopic model granularity, the operational feasibility of our models in OnTime is not obvious. But after the inclusion of headway constraints at junctions we could remove all larger conflicts and reached a

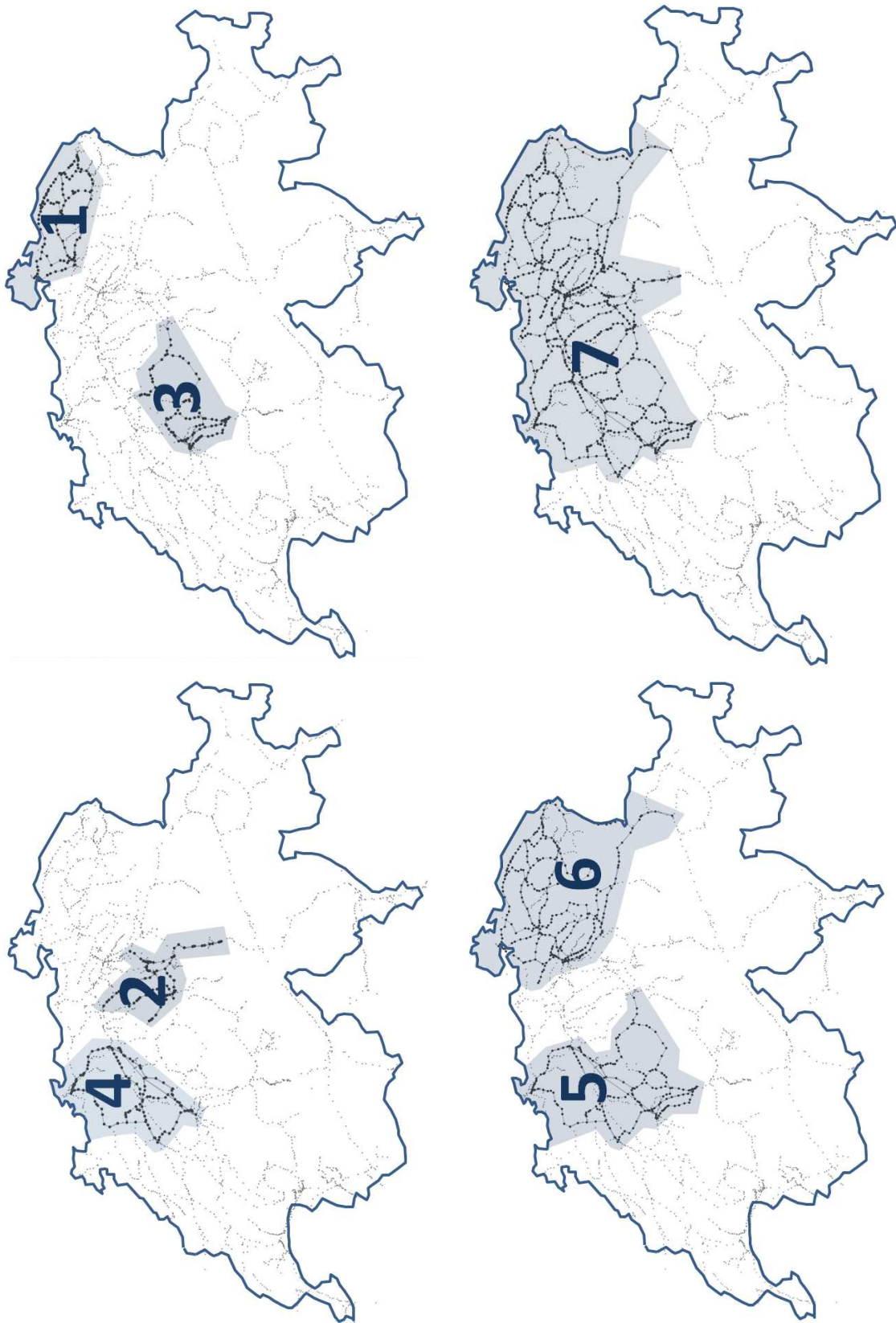


Figure 3.6: *Geographical regions of all test instances*

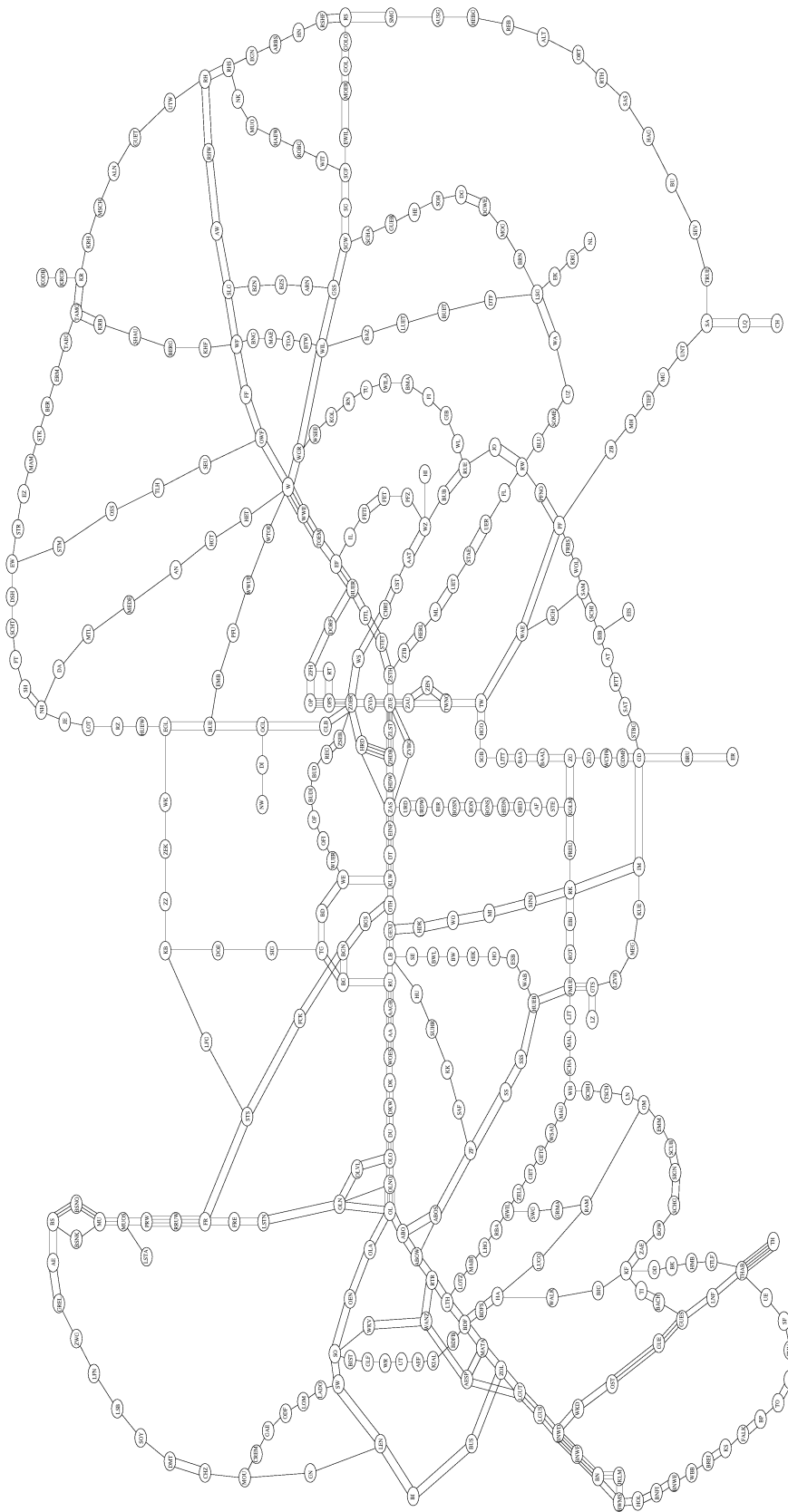


Figure 3.7: *Macroscopic infrastructure of the German-SpeakingCH model*

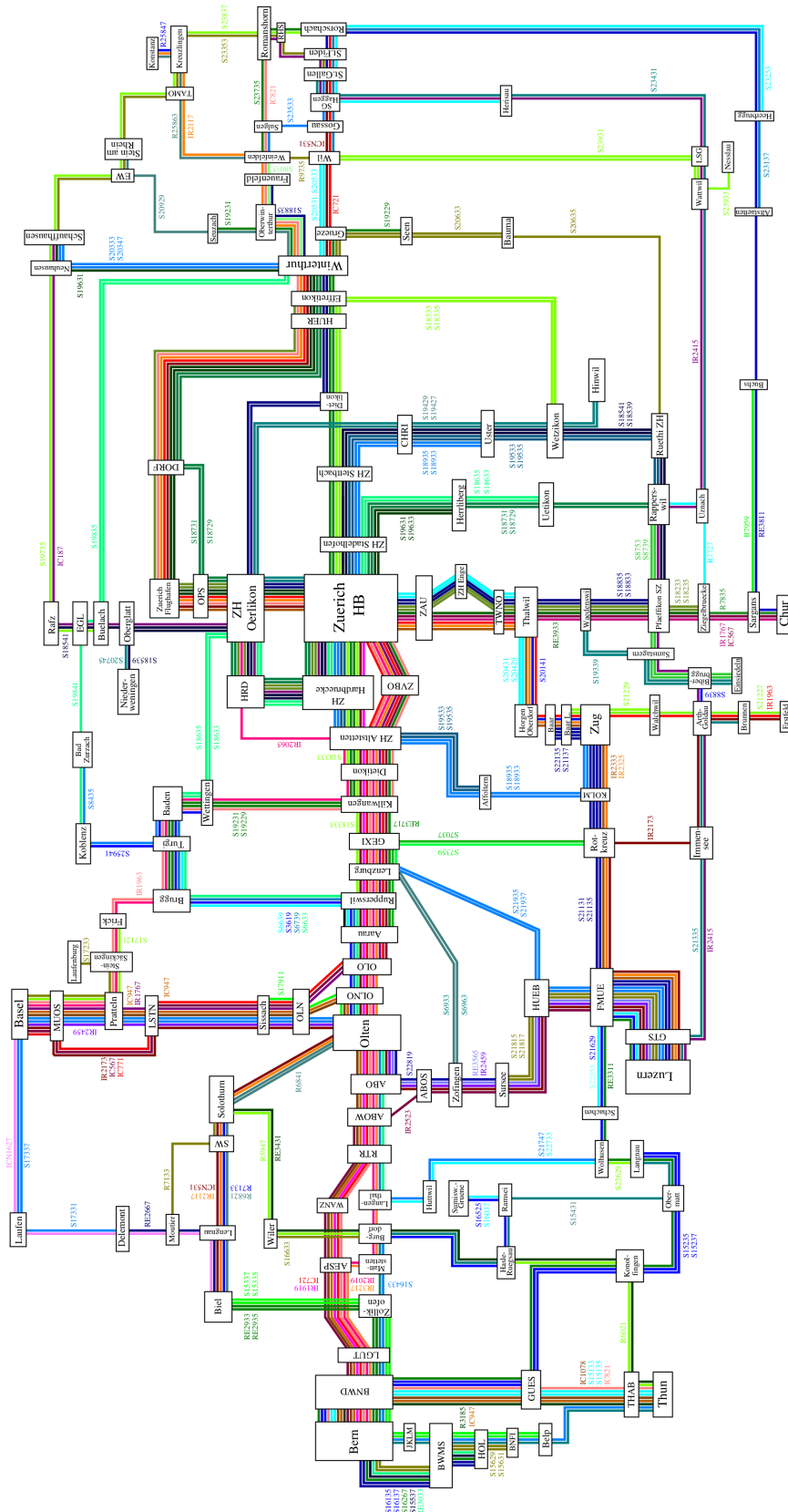


Figure 3.8: Train lines modelled in the German-SpeakingCH model

conflict-free mesoscopic safety system confirmed by a software generally accepted in planning practice.

To improve stability of our timetables we evaluate the influence of the two time reserve parameters `minTripTimeReserve` and `minTrainTimeReserve`, introduced in Section 3.4.3, to require a certain degree of running time supplements for every trip constraint and over a whole train run. Different choices for both parameters are tested and compared for our first two PESP instances. Figure 3.9 gives an overview on the progression of different key figures to the parameter settings for Eastern Switzerland Small. The objective function for this computation is minimized up to a tolerance of 3%. For the second instance the result looks similar. Because of data privacy declaration given by SBB we are not allowed to publish exact values to these key figures. But the progress of curves together with its relative differences already suffice for our evaluation.

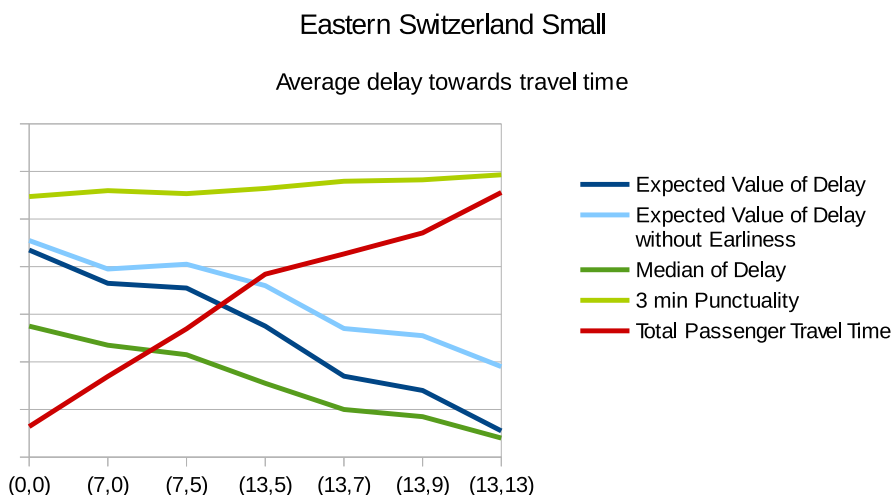


Figure 3.9: Progression of OnTime key values for the average delay over all arrival events compared to the total weighted passenger travel time for different choices of $(\text{minTrainTimeReserve}, \text{minTripTimeReserve})$ [anonymised graphic]

The diagrams shows key values for different percentage values for the tuple $(\text{minTrainTimeReserve}, \text{minTripTimeReserve})$. The key figure Expected Value represents the expected value of delay in average for every arrival event. Since a train is also allowed to arrive earlier as scheduled, but it is not allowed to depart earlier, earliness should not be counted to compensate delays officially. Thus in a second expected value earliness is not counted as negative delay anymore. Beside the two expected values also a median value on all train arrival events is given and the so called 3 minutes punctuality. Since for SBB a train is counted as delayed as soon as it arrives and departs with a delay of more than 3 minutes, the percentage of trains arriving with a punctuality of 3 minutes is essential for SBB. In comparison with these different key figures describing the amount of average delay to each parameter setting, the total weighted passenger travel time of

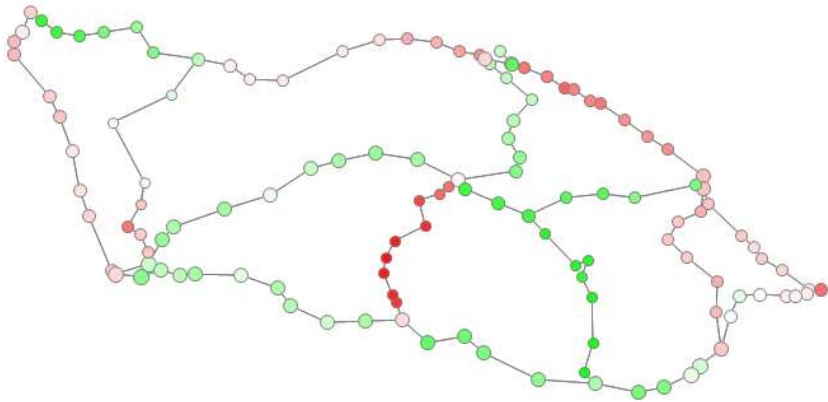


Figure 3.10: $(\text{minTrainTimeReserve}, \text{minTripTimeReserve}) = (0, 0)$

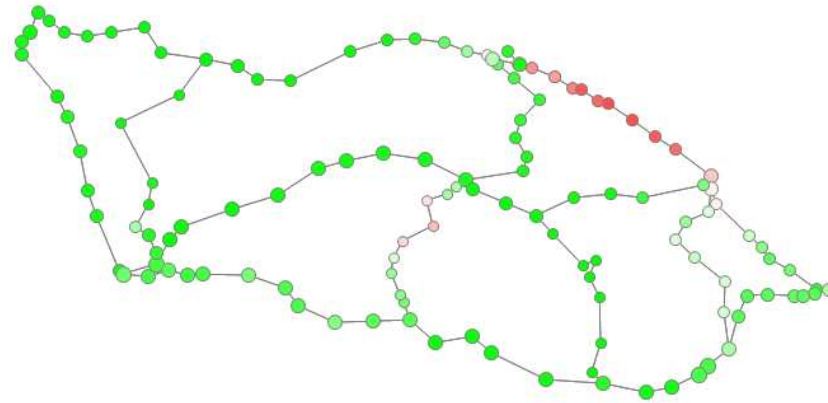


Figure 3.11: $(\text{minTrainTimeReserve}, \text{minTripTimeReserve}) = (13, 7)$

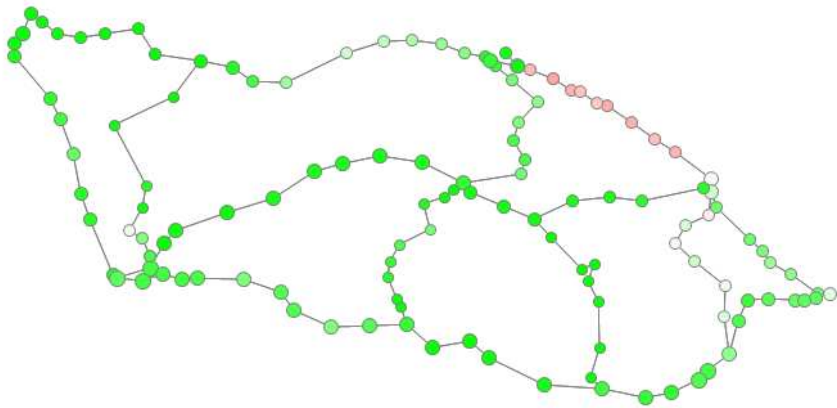


Figure 3.12: $(\text{minTrainTimeReserve}, \text{minTripTimeReserve}) = (13, 9)$

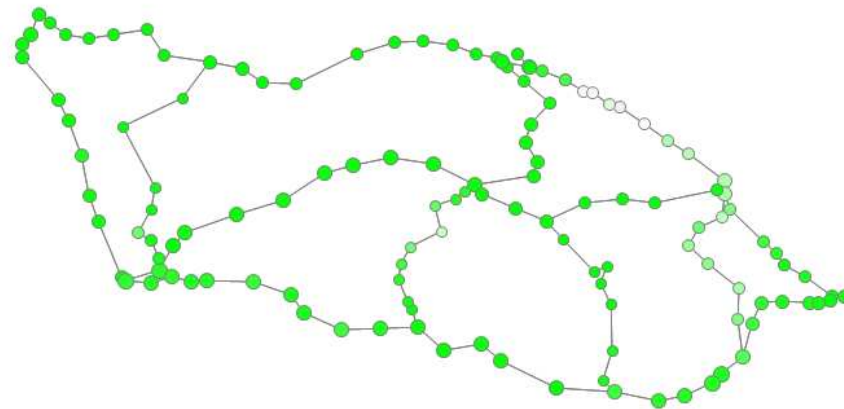


Figure 3.13: $(\text{minTrainTimeReserve}, \text{minTripTimeReserve}) = (13, 13)$

the objective value is added. While all values of delay are reduced by the addition of time reserve, the total passenger times grow. The total difference of passenger travel time for both models is 400 minutes. The expected value varies about 60 seconds for both models and the expected value without earliness about 50 seconds. The lowest and highest value of the 3 minutes punctuality differs by about 10% and the median by 50 seconds.

OnTime also allows to visualize the distribution of delay over the entire network. For each modelled operation point the average delay of all train arrival events is given and visualized using a color scale. As soon as an operation point appears in green, the average delay at this location is only marginal. In case of a red point, the expected amount of delay could complicate operation considerably. In Figure 3.10 to 3.13 this visualization is given for four different parameter settings for the Eastern Switzerland Small instance. In case of no additional forced time supplements we can clearly recognize several deep red operation points. Compared to this worst case the three highest choices of time reserve, all requiring a minimum time supplement of at least 13% for the entire train run and at least 7, 9 or 13% for every train trip, look much better. For a minimum trip time reserve of 13% all operation points appear green. Also for a trip time reserve of 9% the timetable looks stable. Only in smaller stations some delay is expected. In larger stations, where connections play a more important role, all trains should arrive punctually without extreme cases of disturbances.

Besides the timetable also computational performance is influenced by the parameter setting. In case of the smaller instance the solution time varied between 5 and 10 seconds. But for the larger Lucerne instance, the solution time could grow from 5 minutes up to one hour. The computational performance has a tendency to improve when enlarging time reserves. All in all, the evaluation shows a clear profit for timetable stability, as well as for computational performance to include enough time reserve. We therefore fix our two model parameters `minTrainTimeReserve` and `minTripTimeReserve` to 13 and 9 percent.

4 Solving the periodic event scheduling problem

4.1 Introduction

Solving the periodic event scheduling problem as a decision problem, which means finding an arbitrary timetable satisfying all constraints of the PESP model, is already a very hard problem to solve by algorithms. In the theory of mathematical complexity, it belongs to the class of NP -complete decision problems. This result was shown by several researchers studying the PESP over different reductions from other known NP -complete decision problems, as for example the Hamiltonian cycle problem [Serafini and Ukovich, 1989], the k -colourability problem [Odijk, 1997], the linear ordering problem [Liebchen and Peeters, 2002], as well as the SAT problem [Grossmann et al., 2012]. Extending this decision problem with an objective function to a PESP optimization problem therefore leads to a hard (NP -hard) problem in optimization.

Nevertheless many researchers studied solution methods to solve and optimize the PESP in the last two decades. Directly starting with the introduction of the PESP, Serafini and Ukovich introduced a first method to solve the PESP decision problem. Their iterative method subsequently enlarges the set of constraints which are taken into account for the timetable and uses a sophisticated backtracking, if a next constraint cannot be added without changing the timetable considerably. This original method was corrected by Nachtigall and further adapted by Shrijver and Steenback, as well as by Lindner. The work of Lindner [Lindner, 2000] gives a good overview of these early developments. Parallel to these advancements of Serafini and Ukovichs backtracking algorithms, Odijk introduced a constraint generation algorithm [Odijk, 1997] and Voorhoeve a first algorithm using constraint propagation [Voorhoeve, 1993] to solve the same problem. At this time both of them could not reach satisfying computational performance.

The idea of constraint propagation also serves as an important concept in preprocessing to reduce interval widths of PESP constraints, accelerating computational performance before using a standard solution method. In the second part of his work, Lindner extends the PESP decision problem with further variables for operational decisions and an objective function minimizing operational costs. To solve the optimization problem he introduces a decomposition method. Also Nachtigall and Voget extend the PESP decision problem with an objective function at an early time [Nachtigall and Voget, 1996]. They minimize total passenger travel time and use a combination of a heuristic approach and genetic algorithms to solve their optimization problem. The idea of first finding a feasible timetable as fast as possible and then optimizing this timetable in a second step with the

help of local search heuristics is also used by Opitz [Nachtigall and Opitz, 2008] and further elaborated by Goerigk [Goerigk and Schöbel, 2013].

After this starting phase of the development of algorithms to solve the PESP decision and different optimization problems, first applications also gained attention in practice. In this chapter, we give a short description of different algorithms being part of these applications. In Section 4.2 we summarize two basic concepts to solve the PESP decision problem, both part of famous planning software used in the Netherlands, respectively in Germany. In Section 4.3 we give an overview of different mixed integer linear programs to solve the PESP as an optimization problem, including the one used by Liebchen to optimize the entire underground timetable of Berlin in the year 2005 [Liebchen, 2008]. In Section 4.4 we decide on an adequate solution method for this thesis.

4.2 Algorithms solving the PESP decision problem

In this section we describe a constraint programming method to solve the PESP decision problem which is the basic solution method of CADANS, the PESP solver implemented in DONS. In a second subsection, we present another solution approach, based on a reduction to the SAT problem, which was published recently and showed to be more efficient using a state of the art SAT solver. It is already used for practical applications being part of the software TAKT.

4.2.1 A constraint programming method

Based on the early solution method introduced by Voorhove [Voorhoeve, 1993], Schrijver and Steenback developed an improved constraint programming method to solve the PESP feasibility problem [Schrijver and Steenbeck, 1994]. This method constitutes the core part of the known Dutch timetable planning software DONS (Designer Of Network Schedules), used in practice since 2003. In December 2006, the first Dutch timetable constructed with the help of DONS was published and successfully put into practice [Kroon et al., 2009]. It was an introduction of a completely new timetable with an estimated profit of 40 million Euro annually.

The constraint programming method directly works on the PESP graph. Every event time has the set of possible departure respectively arrival times $\{0, d, 2d, \dots, T - d\}$ as its domain, depending on the total period length T and discretisation granularity d . If there are slot constraints and therefore a zero event, the set of possible time events for this vertex can be fixed to $\{0\}$, otherwise this can be done for one arbitrary event to break the cyclic symmetry. Starting from this fixed event, constraints are propagated along the PESP graph, consecutively reducing sets of possible event times.

After reaching a certain criterion, depending on absolute computation time or the ratio of event time reductions per propagation step, the method starts to build a decision tree by partitioning one set of time events in two parts and continuing the propagation process for both parts. As long as there remain unfixed event times, these propagation and

partition steps are repeated alternatingly. If the domain of an event becomes empty, the corresponding branch of the decision tree can be truncated. The algorithm stops as soon as every event time domain has only one value left for each decision variable and each constraint is satisfied (in which case a feasible timetable is found) or there is an empty domain for every branch (the timetabling problem is infeasible).

Two main parameters influence the computation time and the result (if there are several feasible timetables): the breaking criterion of the propagation process and the method chosen to control the partitioning method. Several observations exist for the selection of these two parameters (see [Opitz, 2009] or [Schrijver and Steenbeek, 1994]), but a publication with clear guidelines for these parameters is unknown to the author.

4.2.2 A polynomial reduction from PESP to SAT

For a long time, solution methods using constraint propagation were claimed to be the fastest to solve the PESP decision problem [Opitz, 2009], [Nachtigall, 1998]. In the year 2012, Grossmann et al. introduced a new solution approach based on a polynomial reduction from the PESP to the satisfiability problem (SAT) which empirically was shown to be faster using a state-of-the-art SAT solver [Grossmann et al., 2012]. The new solution method today is part of the planning software TAKT, which was started to be developed out of the dissertation of Opitz at TU Dresden [Opitz, 2009], and still is in progress as a collaboration with Deutsche Bahn, see also Section 2.4.3.

The Boolean satisfiability problem is a basic decision problem with central importance in theoretical computer science. It consists of a formula with boolean variables and three operators (conjunction, disjunction and negation) for which a variable assignment satisfying the given formula is asked. If such an assignment exists the formula is called satisfiable. The decision problem of whether a SAT formula is satisfiable is the first problem which was shown to be *NP*-complete and is therefore often used to prove complexity statements for new decision problems. Furthermore, several modern SAT solvers already exist, which often can handle problems with millions of constraints and hundreds of thousands of variables [Ohrimenko et al., 2007].

For the translation of a PESP to a SAT problem, boolean variables $q_{i,k}$ are introduced for every event. It is defined to be true if $\pi_i \leq k$ and false if $\pi_i \geq k + 1$ for a scheduled event time π_i and an integer time $k \in [-1, T]$. With the help of these variables it is possible to require that every event time π_i has to lie in the interval $[0, T - 1]$ and satisfies the ordering relation: $\pi_i \leq k \Leftrightarrow \pi_i \leq k + 1$. With the introduction of a function

$$enc : v \mapsto (\neg q_{v,-1} \wedge q_{v,T-1}) \bigwedge_{k \in [0, T-1]} (\neg q_{v,k-1} \vee q_{v,k})$$

mapping each vertex $v \in V$ to a SAT clause $enc(v)$, the requirement is encoded as a SAT condition. Similarly every PESP constraint $a \in A$ between two vertices v_i and v_j can be encoded with the help of the boolean variables $q_{i,k}, q_{j,k}$ describing all pairs (π_i, π_j) which are feasible. This set of pairs is also called feasible region of the constraint a and

is denoted by S_a . The research of Grossmann shows advantages in computation time describing these feasible regions indirectly by the infeasible region P_a of every PESP constraint $a \in A$. To minimize further the set of SAT clauses, this set of infeasible pairs (π_i, π_j) is described using rectangles of a maximum diameter completely contained in the infeasible region of the considered constraint as illustrated in Figure 4.1. So for example the exclusion of the rectangle $R = ([2, 5] \times [3, 6])$ can be described by the following SAT clause:

$$\begin{aligned} \text{enc_rec}(R) &:= \neg((\pi_i \leq 5) \wedge (\pi_i \geq 2) \wedge (\pi_j \leq 6) \wedge (\pi_j \geq 3)) \\ &= \neg(q_{i,5} \wedge \neg q_{i,1} \wedge q_{j,6} \wedge \neg q_{j,2}) \\ &= [\neg q_{i,5}, q_{i,1}, \neg q_{j,6}, q_{j,2}]. \end{aligned}$$

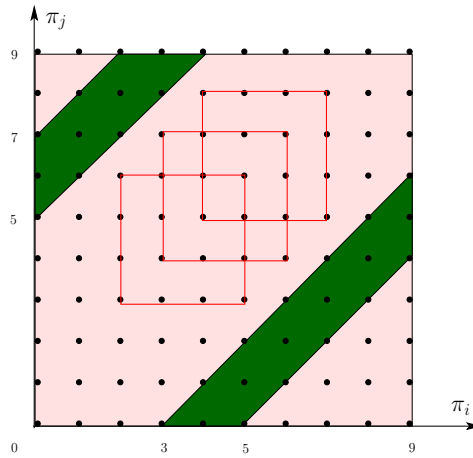


Figure 4.1: Illustration of the feasible and infeasible region of a PESP constraint $[5, 7]_{10}$ with rectangles completely contained in the infeasible region

To cover the whole infeasible region P_a of a PESP constraint $a \in A$ the following function ζ is used:

$$\begin{aligned} \zeta([l(a), u(a)]_T) &= \{H \times G \subset \mathbb{Z}^2 \mid |H| = \delta_x(l(a), u(a)), \\ &|G| = \delta_y(l(a), u(a)), \\ &(H \times G) \cap S_a = \emptyset\} \end{aligned}$$

with $\delta(l(a), u(a)) = u(a) - l(a) - 1$, $\delta_y(l(a), u(a)) = \lfloor \frac{\delta(l(a), u(a))}{2} \rfloor$ and $\delta_x(l(a), u(a)) = \lceil \frac{\delta(l(a), u(a))}{2} \rceil - 1$ for $l(a) < u(a)$. In his research, Grossmann could show that P_a is a subset of the union of all rectangles contained in $\zeta([l(a), u(a)]_T)$ and S_a does not intersect this set of rectangles. Therefore the set of constraints of a PESP instance I can be encoded with

$$\bigwedge_{a \in A} \bigwedge_{R \in \zeta([l(a), u(a)]_T)} \text{enc_rec}(R)$$

which contains approximately T SAT clauses.

For this PESP solution method the domain for all event times is assumed to be $\{0, 1, \dots, T-1\}$.

4.3 Optimizing the PESP using a MILP solver

In this section, we introduce two different MILP formulations for the PESP optimization problem together with a third formulation as an extended formulation to them. The first MILP formulation, presented in Section 4.3.1, is a straightforward translation of the PESP decision problem to an equation system together with an objective function. For a long time, it was the only one which was taken into consideration and therefore is also called the classical MILP. In Section 4.3.2, a more sophisticated MILP formulation based on the choice of an integral cycle basis chosen in the PESP graph is introduced. Section 4.3.3 discusses the extension of this formulation with so called non-collision cycles, which allow the use of flexible trip times.

4.3.1 The classical MILP

The classical formulation of PESP as a mixed integer linear program is the most simple and the first introduced formulation. It directly follows from the definition of the PESP model and uses all time events π_i as continuous decision variables. To translate the constraints $l_a \leq (\pi_i - \pi_j) \bmod T \leq u_a, \forall a \in A$ to valid constraints for a MILP, integer variables $p_a, \forall a \in A$ are introduced to model the modulo operator as shown in Proposition 1.

Proposition 1. [Serafini and Ukovich, 1989] *Let I be an instance of PESP. A vector π is a solution for I if and only if for every constraint $a = (v_i, v_j) \in A$ there exists a unique integer $p_a \in \mathbb{Z}$ such that $l_a \leq \pi_i - \pi_j + p_a T \leq u_a$.*

If $u_a - l_a < T$ the integer variables p_a only take values in the set $\{0, 1\}$ and are therefore binary. They have value 1 if and only if between the time events π_i and π_j a new period starts (e.g. the first event takes place to minute 58 and the second to minute 3 in case of a main period $T = 60$ minutes). The variable p_a therefore is sometimes also referred as *period jump variable*.

Let f_{obj} be an arbitrary linear objective function on π . Then the described MILP can be stated as

$$\begin{array}{ll}
 \text{Minimize} & f_{obj}(\pi) \\
 \text{subject to} & l_a \leq \pi_i - \pi_j + p_a T \leq u_a \quad \forall a \in A \\
 & 0 \leq \pi_i < T \quad \forall v_i \in V \\
 & p_a \in \mathbb{Z} \quad \forall a \in A.
 \end{array}$$

This MILP, its polyhedral structure and possible cutting planes, have been studied in earlier research several times. For interested readers we refer to [Liebchen, 2006] and [Lindner, 2000] who give a good overview of such results.

4.3.2 The cyclic MILP

A more sophisticated MILP formulation that followed from deeper experience with cuts based on cycles in the PESP graph was introduced by Peeters and Kroon

[Peeters and Kroon, 2001]. This so called cyclic formulation turned out to be more efficient for several test cases [Liebchen et al., 2008].

Instead of directly considering all time events π_i as decision variables, time differences between every pair of event times connected over a PESP constraint are used. These new variables $x_a := \pi_i - \pi_j \pmod T$ for every PESP constraint $a = (i, j) \in A$ are called *periodic tensions* in analogy to electrical networks. To satisfy every PESP constraint, each tension variable x_a has to lie between its corresponding edge bounds ($l_a \leq x_a \leq u_a$ with $0 \leq l_a \leq u_a < T$). In addition to these continuous tension variables, integer variables are necessary to require the periodicity for every cycle. Similar to an electric circuit, the directed sum over all tension variables along a cycle in the PESP graph, modulo the total period time T , has to be zero. Thus, for every cycle C in the PESP graph the constraint

$$\sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a = Tq_C \quad (\text{cycle constraint}),$$

has to be satisfied, where C^+ and C^- denote the set of all positively oriented and negatively oriented edges along cycle C , and q_C is an integer variable called *cycle periodicity variable* for cycle C .

Liebchen and Peeters could show that it suffices to require the cycle constraints for an integral cycle basis [Liebchen and Peeters, 2009].

Definition 6. Let $\mathcal{C} = \{C_1, \dots, C_\nu\}$ be a set of oriented cycles in a directed, connected graph $G = (V, A)$, where $\nu := |A| - |V| + 1$ is the cyclomatic number of G . Let C be an arbitrary oriented cycle in G with incident vector γ_C and consider the linear combination $\gamma_C = \lambda_1 \gamma_{C_1} + \dots + \lambda_\nu \gamma_{C_\nu}$. If for all cycles C and for all $i = 1, \dots, \nu$, we have $\lambda_i \in \mathbb{Z}$, then \mathcal{C} is an *integral cycle basis*.

Theorem 1. [Liebchen and Peeters, 2009] Let $G = (V, A)$ be a directed graph, $T > 3$ a positive integer constant and $x \in \mathbb{Q}^A$. Then it is equivalent that

1. x is a periodic tension, and
2. there exists an integral cycle basis $\mathcal{C}_B = \{C_1, \dots, C_\nu\}$ of G , such that along every oriented cycle $C \in \mathcal{C}_B$ there holds $\gamma_C^\top x = \sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a = Tq_C$, for some $q_i \in \mathbb{Z}$, and
3. for every integral cycle basis $\mathcal{C}_B = \{C_1, \dots, C_\nu\}$ of G and along every oriented cycle $C \in \mathcal{C}_B$ there holds $\sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a = Tq_C$, for some $q_i \in \mathbb{Z}$.

With this result, the MILP can be reformulated as the so-called *cyclic MILP*. This was already shown for strictly fundamental cycle bases (a special case of integral cycle basis which is constructed out of a spanning tree in the graph) by Nachtigall [Nachtigall, 1993]. Compared to the classic formulation the number of integer variables can be reduced from $|A|$ to $|A| - |V| + 1$.

$$\begin{aligned}
 \text{Minimize} \quad & f_{obj}(x) & (4.1) \\
 \text{subject to} \quad & l_a \leq x_a \leq u_a & \forall a \in A \\
 & \sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a = Tq_C & \forall C \in \mathcal{C}_B \\
 & x_a \in \mathbb{R}^+ & \forall a \in A \\
 & q_C \in \mathbb{Z} & \forall C \in \mathcal{C}_B.
 \end{aligned}$$

In this formulation the choice of an integral cycle basis is not fixed yet. As Theorem 1 shows, the model works independently of this choice and, therefore, the resulting timetable is not influenced by the integral cycle basis. But computational studies, as in [Liebchen et al., 2008] and also in Section 4.4, show, strong dependencies between this choice and the performance of commercial solvers to solve the MILP formulation. The following theorem of Odijk gives reason for this observation:

Theorem 2. [Odijk, 1996] *Let $I = (G, l, u, T)$ be an instance of PESP, \mathcal{C}_B an integral cycle basis of G , and consider the cyclic MILP formulation (4.1). Then, for a basic cycle $C \in \mathcal{C}_B$, the following inequalities are valid*

$$a_C := \left\lceil \frac{1}{T} \left(\sum_{a \in C^+} l_a - \sum_{a \in C^-} u_a \right) \right\rceil \leq q_C \leq \left\lfloor \frac{1}{T} \left(\sum_{a \in C^+} u_a - \sum_{a \in C^-} l_a \right) \right\rfloor =: b_C$$

The closer the lower and upper bound a_C and b_C for each integer variable q_C are, the less possible integer assignments exist for q_C . This can considerably reduce the number of branches which have to be considered in a branch and bound algorithm, as implemented in commercial MILP solvers. The number of integer values for q_C bounded by a_C and b_C is called the *cycle width* of cycle C .

Definition 7. Let $I = (G, l, u, T)$ be an instance of PESP and a_C, b_C the integer bounds for a cycle $C \in G$ as defined in Theorem 2. Then we denote the number of possible integer assignments for q_C by $w_C := b_C - a_C + 1$ and refer to it as the *cycle width*.

Determining an integer cycle basis \mathcal{C}_B minimizing the cycle width for every cycle $C \in \mathcal{C}_B$ is unfortunately a difficult optimization problem of still unknown complexity studied in [Liebchen and Peeters, 2009]. Integer cycle bases defined as strictly fundamental cycle bases out of a spanning tree which is minimum according to the edge weights $u_a - l_a$ turned out to be advantageous for computational performance [Liebchen et al., 2008]. They lead to small cycle widths and they can be determined algorithmically in a short (polynomial) time.

The inequalities, introduced by Odijk, can also be used for an arbitrary cycle C in G , binding the linear combination $\lambda_1 q_1 + \dots + \lambda_\nu q_\nu$ with $\lambda_i \in \mathbb{Z}$ as introduced in Definition 6. Adding such additional inequalities to the MILP formulation (4.1) can further improve the performance of branch and bound algorithms, as for example shown in [Liebchen and Swarat, 2008].

4.3.3 Cyclic MILP with non-collision cycles

In this subsection, we extend the cyclic MILP formulation (4.1) with so called non-collision cycles introduced in [Caimi, 2009] in order to work with variable trip times in the PESP model. Especially in a very heterogeneous and dense railway network the idea of slowing down a fast train for a short section can help to bundle trains and therefore to overcome bottlenecks in capacity. Furthermore, a reduced velocity for an arbitrary train can improve the stability of a timetable, as also shown in Section 3.4.

In a standard PESP model, as mentioned in Chapter 3.2, trip times of every train are supposed to be fixed. If we allowed variable trip times over trip constraints a_i with bounds $[l_{a_i}, u_{a_i}]$, $u_{a_i} > l_{a_i}$, we could risk overtaking and collisions as illustrated in Figure 4.2.

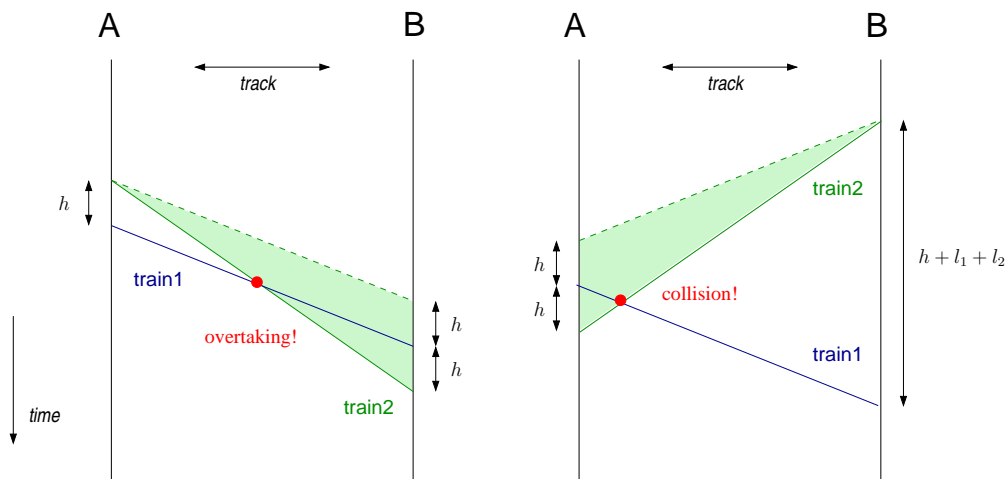


Figure 4.2: Possible collision and overtaking if the safety system uses headway constraints assuming fixed trip times l_1, l_2 and train two could slow down its trip time by twice the headway time h on this track

In fact, such a situation can appear as soon as the interval size of the trip constraint for one train reaches twice the minimal headway time between this and another train. Preventing such a situation by enlarging headway times corresponding to the slowest allowed trip time of every train, could solve this problem, but would reduce capacity of the whole network unnecessarily and therefore even could lead to infeasible timetabling problems. Thus, better approaches to include variable trip times were studied in [Lindner, 2000], [Peeters, 2003] and [Liebchen and Möhring, 2007]. Some of them are based on the idea of partitioning an initial section in smaller ones to also separate the corresponding trip constraints in several constraints with smaller intervals. This is done until no trip interval exceeds the critical threshold of twice the smallest headway time. However the approach introduces additional PESP constraints, therefore we will work with so called non-collision cycles for this thesis. They do not directly fit in the original PESP model and therefore cannot be used for every PESP solution method. But in case of the cyclic MILP formulation they can be added in an elegant way, as shown further on.

The idea of non-collision cycles is based on an observation already made in [Peeters, 2003]. For each pair of trains running on a common track, we assume to have two headway constraints, one at each endpoint of the track, pointing in the same direction, as indicated in Figure 4.3.

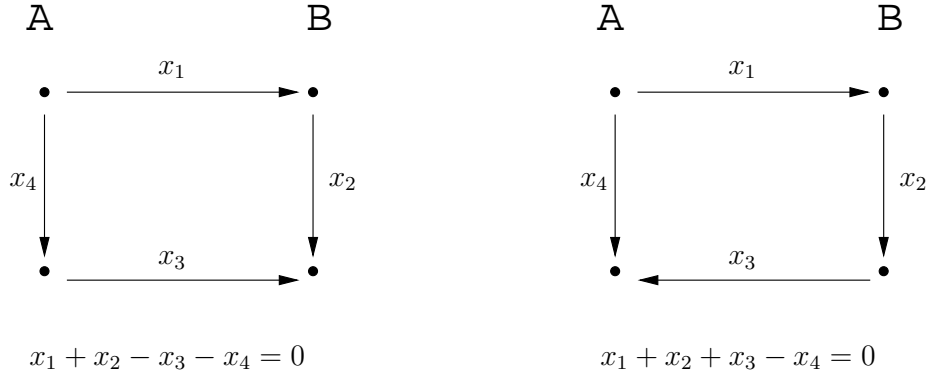


Figure 4.3: *Non-collision cycle for a pair of trains running on the same track in the same (left cycle) and opposite (right cycle) direction.*

Let $C = (x_1, x_2, x_3, x_4)$ be such a cycle containing exactly the described four constraints and $q_C = \frac{1}{T}(\sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a)$. Then q_C only can take two values, 0 or 1 and $q_C = 0$ if and only if the sequence of the two trains does not change on this track. Therefore, cycles C_i with the restriction $q_{C_i} = 0$ can be used to prevent collisions and overtaking on tracks [Caimi, 2009]. To simplify notation, from now on we will call such a cycle *non-collision cycle* and we denote the set of all non-collision cycles in G by \mathcal{C}_N . The constraint $q_{C_i} = 0$ for $C_i \in \mathcal{C}_N$ can easily be added to the cyclic MILP formulation and leads to an extended cyclic MILP formulation already used in [Caimi, 2009].

$$\begin{aligned}
 &\text{Minimize} && f_{obj}(x) \\
 &\text{subject to} && l_a \leq x_a \leq u_a && \forall a \in A \\
 & && \sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a = Tq_C && \forall C \in \mathcal{C}_B \\
 & && \sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a = 0 && \forall C \in \mathcal{C}_N \\
 & && x_a \in \mathbb{R}^+ && \forall a \in A \\
 & && q_C \in \mathbb{Z} && \forall C \in \mathcal{C}_B.
 \end{aligned}$$

4.4 Choice of solution approach for this thesis

Since research on PESP solution methods already reached deeper advances and successful applications in practice for methods solving the PESP decision problem, we want to concentrate on solution methods for the PESP optimization problem. As already discussed in Section 3.3, the introduction of an objective function has clear advantages. Furthermore, we are interested in allowing flexible train trip times with all benefits

we mentioned in Section 4.3.3. Flexible trip times also play an important role in the application of the so-called flexible PESP, introduced in the previous thesis at the same institute by Gabrio Caimi [Caimi, 2009], which can simplify the iteration between macroscopic timetabling and microscopic scheduling in station areas. These preliminary fixations lead us to the use of optimization methods based on mixed-integer linear programming techniques.

In this section, we describe and fix our solution method used for the test instances of this thesis. A first section is dedicated to the used commercial MILP solver and its parameter settings. Subsequently, we compare computational performance for the classical and the cyclic MILP in Section 4.4.2. For the cyclic MILP we further discuss the influence of different cycle bases in Section 4.4.3. All computations in this chapter are performed using a computer server with 2x2 Intel X5650 CPUs with six cores each.

4.4.1 Solver and its parameter settings

As a solver we use IBM ILOG CPLEX Optimizer 12.5 [IBM, 2014] owned and distributed by IBM. CPLEX is able to solve different types of optimization problems including MILP. The MILP solution strategy is based on a branch and cut method. Before starting branch and cut, CPLEX does a preprocessing step. It tries to reduce the problem size and looks for tighter formulations. During the branch and cut procedure heuristic decisions are also used to provide a faster detection of first feasible solutions and to accelerate optimization. The consecutive introduction of new sophisticated solution techniques over the last two decades made CPLEX to one of the strongest MILP solvers of today [Mittelmann, 2014]. Furthermore, CPLEX allows parallelization and offers different parameters to influence the solution strategy. A very useful parameter, especially for practical applications, allows fixing a relative tolerance on the gap between the best integer objective and the objective of the best remaining node. Thus as soon as CPLEX has found a solution for which it can ensure that it varies by at most the chosen tolerance from the optimal solution, it finishes computation.

In this section we discuss the choice of two further parameters which could be relevant for the performance of our solution strategy. They were already studied by C. Liebchen in [Liebchen et al., 2008] for an earlier version of CPLEX. A first one, called *VarSel*, sets the rule for selecting the branching variable at the node which has been selected for branching. Its default choice allows CPLEX to select the best rule based on the problem and its progress. But especially for large and difficult MILP it can be effective to set this parameter to a strong branching strategy. In this case a number of subproblems are partially solved with tentative branches to see which branch is the most promising. A second parameter, we want to discuss, is called *MipEmph* and controls the trade-off between speed, feasibility, optimality and moving bounds in MILP. Its default setting works towards a rapid proof of an optimal solution, but balances that with effort towards finding high quality feasible solutions early in the optimization. Instead of this balanced choice, the parameter can be set to a feasibility strategy, concentrating on generating

more feasible solutions as it optimizes or to an optimality strategy, where less effort is spent to find feasible solutions early.

To have an impression of the influence of the two parameters, we test all combinations of parameter fixations for our small model EasternSwitzerlandSmall and one of our medium size models (Lucerne). For these computations, we use the cyclic MILP formulation with the train and constraint ordered cycle basis, further explained in Section 4.4.3 and we set the relative MIP gap tolerance to 3%. For all computations an upper threshold of one hour computation time is used. Tables 4.1 and 4.2 show the computation time (CT), the reached MIPGap tolerance (MIPGap) and the corresponding objective value (Obj.Val.) in minutes for all combinations of the two parameters settings.

(VarSel,MipEmph)	CT [s]	MIPGap [%]	Obj.Val. [min]
(default, default)	4.92	0.21	2079.13
(strong branching, default)	18.13	2.23	2122.18
(default, feasibility)	4.10	1.48	2103.99
(strong branching, feasibility)	15.76	1.76	2111.97
(default, optimality)	6.17	0.35	2082.10
(strong branching, optimality)	28.56	0.2	2078.93

Table 4.1: *Influence of two CPLEX parameter settings for EasternSwitzerlandSmall*

(VarSel,MipEmph)	CT [s]	MIPGap [%]	Obj.Val. [min]
(default, default)	291.28	1.54	1629.42
(strong branching, default)	> 3600	-	-
(default, feasibility)	571.81	2.77	1649.14
(strong branching, feasibility)	899.64	2.69	1648.90
(default, optimality)	1116.8	1.66	1631.34
(strong branching, optimality)	2472.42	2.57	1646.95

Table 4.2: *Influence of two CPLEX parameter settings for Lucerne*

Both results support the promises of the parameters default setting. And it seems that for the new version of CPLEX the originally most promising parameter setting (strong branching & feasibility) stated in [Liebchen et al., 2008] does not lead to the best solution progress anymore. To further intensify this conjecture we compared this parameter setting with the default setting also for the larger Thun-Basel model. We found a first feasible solution after 7 hours for the default setting, whereas we did not have any solution for the strong branching and feasibility choice after more than 24 hours of computation time. The only parameter setting which could compete with the default setting is using the feasibility strategy for the MipEmph parameter with the default choice for the VarSel parameter. However, in all tests we did, the computation time to find a first feasible solution only

varied slightly to the one of the default setting. Furthermore we are also interested to find good solutions early. Therefore, both parameters, VarSel and MipEmph, are set to their default for further computations in this thesis.

4.4.2 Performance of the cyclic and classical MILP

In this section we want to compare the computational performance of the classical and the cyclic MILP formulation for our test instances. Since we model flexible trip times, we use the cyclic MILP formulation with non-collision cycles. These non-collision cycles can easily be translated into the classical MILP formulation by requiring the directed sum of all integer period jump variables along a non-collision cycle to be 0. As in the last section, we use the train and constraint ordered cycle basis for the cyclic MILP formulation.

Model	Cyclic MILP				Classic MILP			
	CT [s]	Obj.Val.	MIPgap	Sol.	CT [s]	Obj.Val.	MIPgap	Sol.
Eastern CH	4.46	2079.19	0.21	1	2	2456.21	24.41	1
Small	19.67 ⁽¹⁾	2076.82	0	18	5250 ⁽³⁾	2079.29	9.00	277
Lucerne	325	1629.42	1.54	1	600	1746.77	18.56	1
	4900 ⁽²⁾	1614.45	0.46	299	6500 ⁽³⁾	1621.51	11.09	495
Thun	220.93	2145.89	2.77	1	170	2242.10	17.91	1
	16407 ⁽²⁾	2113.55	1.10	267	5700 ⁽³⁾	2132.49	12.79	320
Basel	130	2987.60	1.56	1	30	3224.23	18.46	1
	3596 ⁽¹⁾	2959.38	0	195	8800 ⁽³⁾	3006.36	11.58	752
Thun-Basel	25740	4819.98	2.65	1	5000	5099.27	18.02	1
	69303 ⁽²⁾	4819.98	2.65	1	31000 ⁽³⁾	4755.16	12.08	375

Table 4.3: Comparison of the computational performance of the cyclic and classical MILP formulations

Table 4.3 gives an overview of the computational results. The relative MIP gap tolerance again is set to 3%. But as soon as the MIP gap is reached, the computation is restarted without relative MIP gap tolerance in order to observe further computational performance. For each computation, the first found feasible solution together with its objective value and its relative MIP gap is given. Furthermore, the final values after the computation are given together with the total number of feasible solutions found until then, whereby a computation is stopped as soon as optimality is reached (1), CPLEX reaches a memory problem (2) or the computation is stopped manually (3), because of a progress in optimization, which is too slow compared to the other MILP formulation. The reason of the stop is indicated with a small superscript in the second computation time. For the two largest models, Eastern CH and German-speaking CH no solution could be found after 24 hours of computation time in both cases.

For all instances except the last one, the cyclic MILP formulation performs considerably better. Figure 4.4 further illustrates the progress of optimization for the first four models.

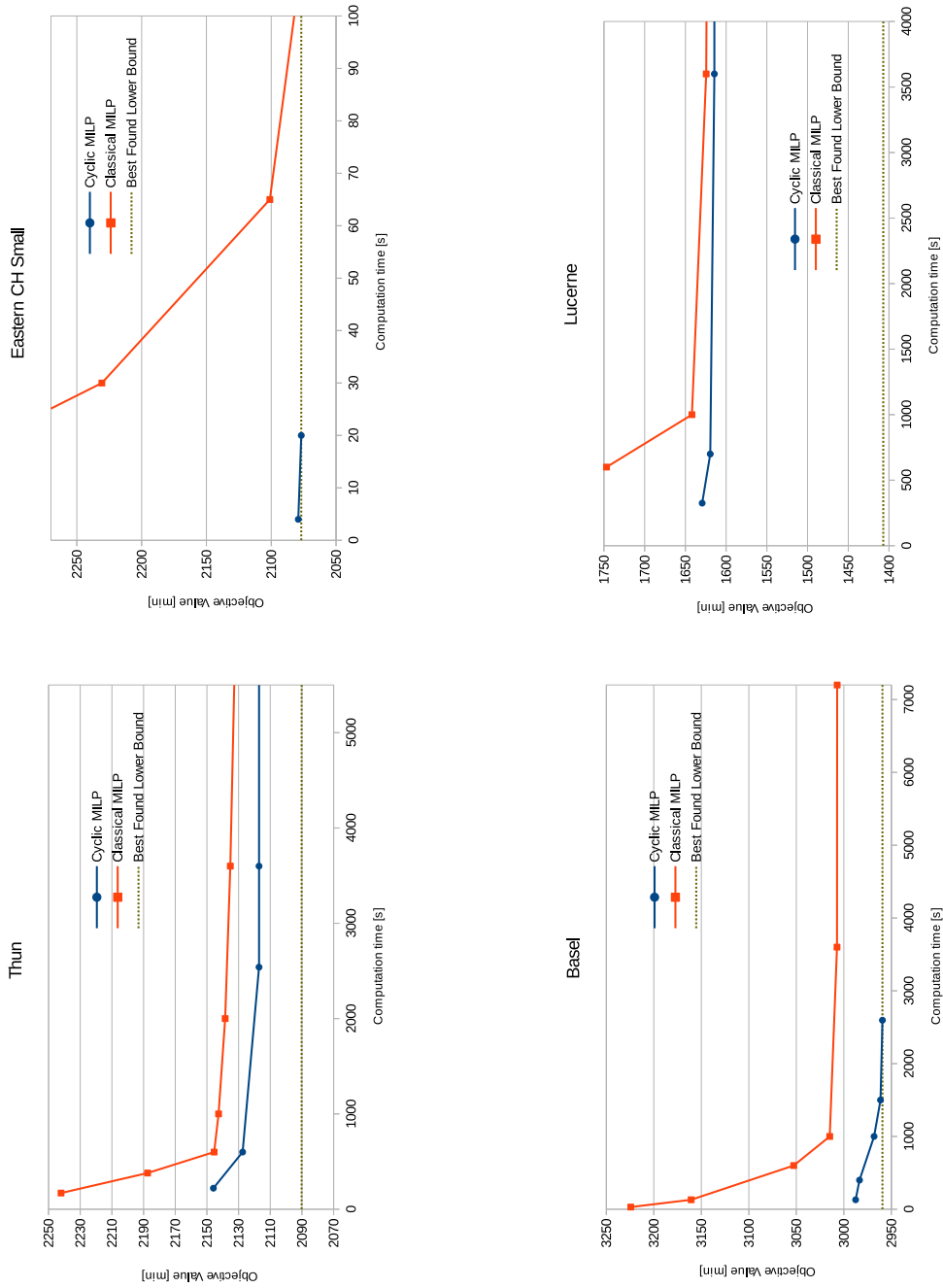


Figure 4.4: Comparison of the optimization progress for the cyclic and classical MILP formulation

Although more solutions could be found with the classical formulation in the same time, the solutions found over the cyclic MILP seem to be better. Especially the lower bound and with this the relative MIP gap is much better for the cyclic MILP formulation. Therefore earlier observations and comparisons to both MILP formulations are further underlined through our computations, also for different models and a new version of CPLEX. An exceptional case is the Thun-Basel model. In this case better solutions with shorter computation times could be found over the classical MILP formulation. But still the relative MIP gap of the classical MILP could never be reached. For further computations of this thesis we use the cyclic MILP formulation.

4.4.3 Comparison of different cycle bases

In this subsection, we test different strictly fundamental cycle bases and their influence on the computational performance for our models. We compare four types of cycle bases defined out of different spanning trees. For the construction of the spanning trees, we use the algorithm of Kruskal, which adds edges one by one out of a sorted list to the spanning tree as long as they do not close a cycle. Varying the sorting strategy of the list, we therefore get different spanning trees. As already studies of Liebchen showed [Liebchen and Peeters, 2009] and we discussed in Section 4.3.2, cycle bases with small cycle widths lead to a better performance.

We therefore also compare the distribution of cycle widths for our cycle bases and we use the sum of all cycle widths $S_w(\mathcal{C}_B) = \sum_{C \in \mathcal{C}_B} w_C$ as a key figure for the whole cycle basis \mathcal{C}_B . In a first test setting, we compare the computational performance for all cycle bases with our smallest model (Eastern Switzerland small). Subsequently we concentrate on the comparison of the best two cycle bases for three further models (Lucerne, Basel, Thun).

In the following we specify the ordering of the list of constraints to construct the spanning trees and therefore the cycle bases:

Sorting small intervals

All constraints $a \in A$ are ordered according their interval sizes $u_a - l_a$, starting with the constraint having the smallest interval size.

Train and Constraint ordered

In this list all constraints are ordered according to their constraint type and affiliation to different trains. In a first part of the list all trip, dwell and frequency constraints are ordered train by train and trains are ordered according their train numbers. Subsequently, all connection, time dependency and slot constraints follow in an order as they are defined in the model construction. In the third part all headway constraints are collected.

Sorting large intervals

This is the inverse list of the first one (sorting small intervals).

Shuffle

For this list, all constraints are shuffled randomly.

The first list leads to a minimum spanning tree according to the size of edge intervals. Its corresponding cycle basis \mathcal{C}_{B1} is also suggested as a good and simply constructible cycle basis in the literature [Caimi, 2009], [Liebchen et al., 2008]. Using knowledge about the structure of an average PESP graph, the second list, leading to cycle basis \mathcal{C}_{B2} , ensures to have all constraints of a train line as a continuous path in the spanning tree. These paths of train lines then are connected over frequency, connection, time dependency and slot constraints which in general have smaller edge intervals than headway constraints. Only if several trees remain, they are connected to a spanning tree with the help of headway constraints. To see the influence of this strategy, of using cycle bases with small cycle widths, on computational performance two further lists, not respecting this idea are defined. As a first one we directly use the complementary strategy of constructing a maximum spanning tree according to the edge interval sizes. And as a second example, we use a list of constraints with an arbitrary shuffled sequence. We denote the corresponding cycle bases by \mathcal{C}_{B3} and \mathcal{C}_{B4} .

Table 4.4 shows the result of the first test comparing all cycle bases for the smallest problem instance. For all computations a relative mipgap tolerance of 3% is used in CPLEX. The value t_{FF} is the computation time CPLEX needs to find a first feasible solution and t_{gap} the computation time until the mipgap tolerance of 3% is reached. As soon as t_{gap} exceeds a time limit of one hour the computation is stopped. With $obj(t)$ and $gap(t)$ we denote the currently best found objective value (passenger travel time), respectively the currently reached mipgap at time t . For the shuffled list the average of five different computations is taken.

	\mathcal{C}_{B1}	\mathcal{C}_{B2}	\mathcal{C}_{B3}	\mathcal{C}_{B4}
$t_{FF}[s]$	3.7	4.43	2250	159.8
$obj(t_{FF})[min]$	2083.5	2079.1	2155.6	2174.86
$gap(t_{FF})[\%]$	0.42	0.21	9.34	10.46
$t_{gap}[s]$	3.7	4.43	3600	3600
$obj(t_{gap})[min]$	2083.5	2079.1	2084.3	2090.64
$gap(t_{gap})[\%]$	0.42	0.21	6.24	6.894
$S_w(\mathcal{C}_B)[-]$	2256	2807	2906	3302.4

Table 4.4: Comparison of the computational performance for four different cycle bases testing Eastern Switzerland small

For this smaller problem instance computational performance already differs a lot depending on the choice of cycle basis. As it was to be expected, the third and fourth cycle basis lead to a worse performance. While we find a first feasible timetable in less than 5 seconds in the first two cases, we need several minutes for the other two. But not only finding a first solution differs, especially the whole optimization process works very slowly for the third and fourth cycle basis. Comparing the sum of cycle widths there is an interesting point: Although the first cycle basis has much smaller cycle widths, the computational performance compared to the second one is comparably efficient, even leading to a slightly better solution in quite the same amount of time.

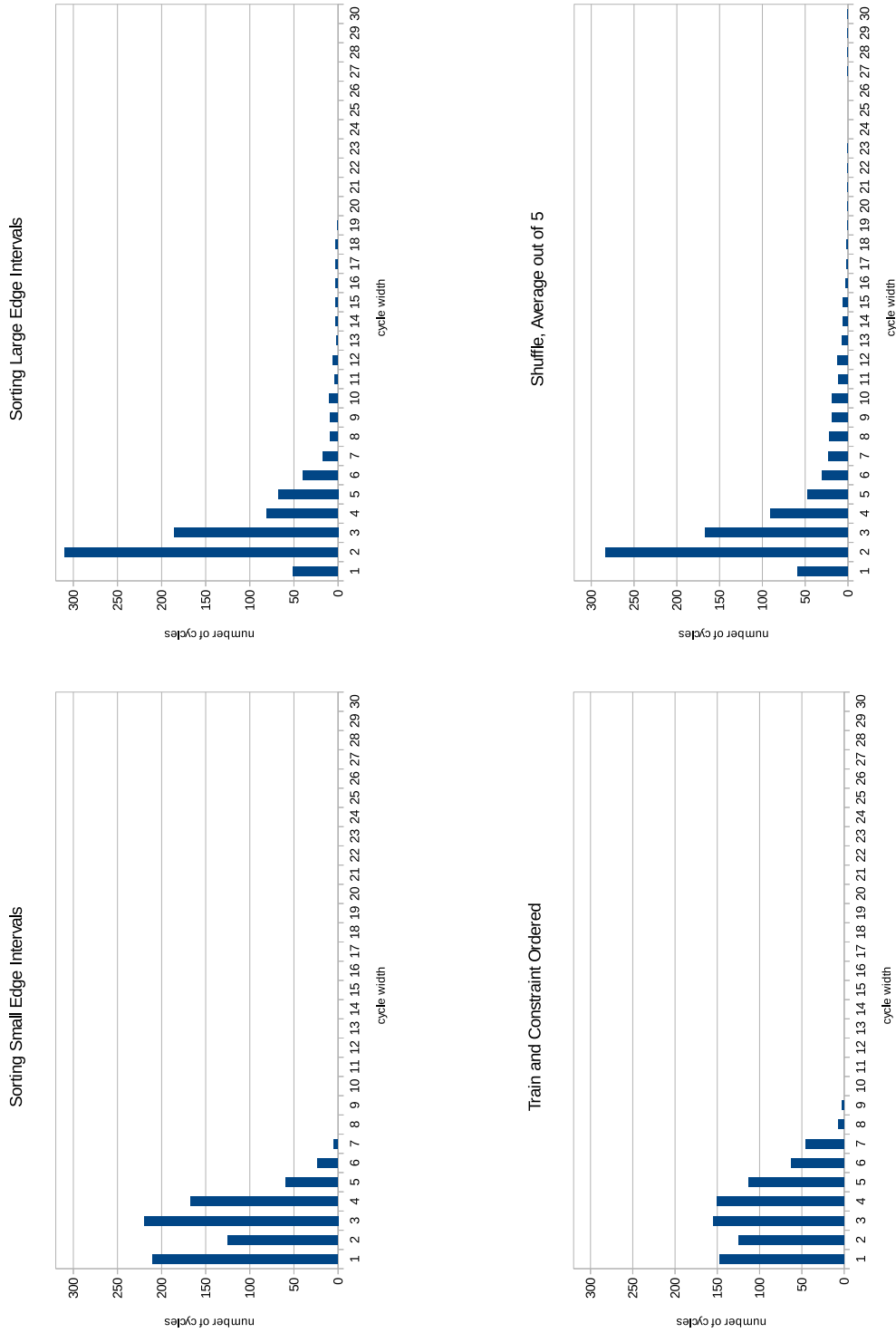


Figure 4.5: Distribution of cycle widths for four different cycle bases

In Figure 4.5 the distribution of cycle widths is illustrated. We can easily observe that in the case of \mathcal{C}_{B3} and \mathcal{C}_{B4} longer cycles exist. For \mathcal{C}_{B4} even a cycle appears with a width of 30 in one iteration. For \mathcal{C}_{B2} the longest cycle has a width of 9 and in the case of \mathcal{C}_{B1} there is even no larger cycle width than 7.

Summarizing the result of the first test we can once more confirm a considerable influence of the choice of cycle basis on the computational performance of CPLEX to solve and optimize a PESP instance and an interesting comparison of the first two cycle bases.

In a second test we further compare the difference of the first two cycle bases and their computational performance. Table 4.5 shows the result of the second test.

	Lucerne, \mathcal{C}_{B1}	Lucerne, \mathcal{C}_{B2}	Basel, \mathcal{C}_{B1}	Basel, \mathcal{C}_{B2}	Thun, \mathcal{C}_{B1}	Thun, \mathcal{C}_{B2}
t_{FF}	1325	325	152.5	130	266.5	220
$\text{obj}(t_{FF})$	1628.7	1629.4	2986.5	2987.6	2147.7	2145.8
$\text{gap}(t_{FF})$	1.5	1.54	1.34	1.56	2.93	2.77
$\overline{S_w}(\mathcal{C}_B)$	6867	9884	7651	13333	9469	12041

Table 4.5: *Further comparison of the computational performance for the better two cycle bases \mathcal{C}_{B1} and \mathcal{C}_{B2} with the three medium size models Lucerne, Basel and Thun*

As in the first test for each instance the sum of cycle width is considerably larger for \mathcal{C}_{B2} . Nevertheless, in all three cases we find a solution faster, using the second cycle bases. We therefore use cycle basis \mathcal{C}_{B2} for all further computations of this thesis.

5 Decomposition methods

5.1 Introduction

Since optimizing a PESP instance is an NP -hard problem, computation times can grow exponentially with the problem size in the worst case and can lead to intractable problems for MILP solvers. On the other hand, splitting a PESP instance into smaller subproblems reduces computational complexity for each separated subproblem considerably. If we were able to coordinate the solution of the subproblems to a global feasible or even optimal solution with a small number of iterations, we could gain an acceleration of the overall computation time or we could even make it possible at all to find a feasible solution for a larger PESP instance. Furthermore we could use parallel computation to solve the subproblems.

This leads to the idea of using decomposition methods to solve the PESP optimization problem. As an introduction to the topic, we review the general idea of decomposition methods in optimization theory and explain in more details the principle of Benders decomposition in the next section. Based on similar ideas we introduce an own decomposition out of a graph decomposition of the PESP graph in Section 5.3. We split the original PESP formulation into subproblems and a master problem to coordinate the subproblems in Section 5.3.1. Subsequently, we study the special case of splitting a PESP graph through a certain track section in Section 5.3.2. We introduce two heuristic solution methods to solve the master problem in Section 5.3.3 and test it on a small problem instance in Section 5.4.

5.2 Decomposition in optimization theory

The idea of decomposition methods in optimization theory is quite as old as the optimization theory itself. Although in the beginning standard linear programs were in the centre of research, the amount of memory and computational effort needed to solve these optimization problems were an obstacle for larger applications in practice at this time. Thus first decomposition methods for linear programs were already published in the 1960s about ten years after the introduction of linear programs. To these first methods belong the well known Dantzig-Wolfe Decomposition [Dantzig and Wolfe, 1960], Benders Decomposition [Benders, 1962] and Lagrangian Optimization. Often they are the basis of more specialized and sophisticated decomposition methods developed later on. Therefore, we want to introduce the technique that is behind these methods to use it as a starting idea for our decomposition in the next section. Since it can be shown that the three mentioned decomposition methods are based on the same technique if they are applied to different

representations of the problem (primal LP, dual LP, Lagrangian dual LP) [Lim, 2010], we concentrate on one of them: The Benders decomposition.

The main idea behind a decomposition method is breaking a larger problem up into smaller ones, which can be solved separately. Studying an application and trying to adapt this idea, two difficulties remain:

- How should a problem be split into subproblems? And
- how can solutions of the subproblems be merged into a global solution of the original optimization problem?

In the case of Benders, the definition of the decomposition is based on splitting the set of variables. In particular, a set of so called *complicating variables* y is removed to reduce problem complexity of the remaining optimization problem considerably. The remaining optimization problem is then called the *subproblem*. For the coordination of the subproblems solutions to a global optimal solution, a so-called *master problem* is defined. By iteratively solving the subproblem, constraints for the master problem are defined leading to an optimal variable assignment for the complicating variables. These constraints are based on the dual reformulation of the subproblem and a description of its feasible region by an enumeration of all extreme points and extreme rays. For further details let us define the following optimization problem:

$$\text{Minimize} \quad c^\top x + f^\top y \quad (5.1)$$

$$\text{subject to} \quad Ax + By = b \quad (5.2)$$

$$x \geq 0, \quad x \in \mathbb{R}^p, \quad y \in Y \subset \mathbb{R}^q \quad (5.3)$$

The set of variables y can be a special type of variables making the problem difficult, as for example integer variables in a mixed integer linear program, or a set of variables whose removal would lead to a block-structured constraint matrix A . Fixing the variables y in the original formulation leads to the subproblem (5.4).

$$\text{Minimize} \quad c^\top x \quad (5.4)$$

$$\text{subject to} \quad Ax = b - By \quad (5.5)$$

$$x \geq 0, \quad x \in \mathbb{R}^p \quad (5.6)$$

The optimal objective value of this subproblem is denoted by $q(y)$, which leads to a first formulation of the master problem (5.7).

$$\text{Minimize} \quad f^\top y + q(y) \quad (5.7)$$

$$\text{subject to} \quad y \in Y \quad (5.8)$$

If the subproblem is unbounded from below for some $y \in Y$, then the master problem is unbounded, too. Assuming boundedness for the subproblem, $q(y)$ can also be defined by the dual reformulation of the subproblem (5.9).

$$\text{Maximize} \quad \alpha^\top (b - By) \tag{5.9}$$

$$\text{subject to} \quad A^\top \alpha \leq c \tag{5.10}$$

$$\alpha \text{ unrestricted} \tag{5.11}$$

The feasible region of this dual subproblem is independent of the fixed y . Furthermore, we assume this region to be non-empty, otherwise the original problem would be unbounded or infeasible. Then we can describe the feasible region by enumerating all its extreme points $(\alpha_p^1, \dots, \alpha_p^I)$ and extreme rays $(\alpha_r^1, \dots, \alpha_r^J)$, leading to the second reformulation of the subproblem (5.12).

$$\text{Minimize} \quad q \tag{5.12}$$

$$\text{subject to} \quad (\alpha_r^j)^\top (b - By) \leq 0 \quad \forall j = 1, \dots, J \tag{5.13}$$

$$(\alpha_p^i)^\top (b - By) \leq q \quad \forall i = 1, \dots, I \tag{5.14}$$

$$q \text{ unrestricted} \tag{5.15}$$

This reformulation only uses one variable, but in general has a huge amount of constraints, since there are usually exponentially many extreme points and extreme rays. The idea of the Benders decomposition is to use this reformulation in the master problem, but only with a subset of these constraints. Solving this relaxed master problem leads to a candidate (y^*, q^*) for an optimal solution. Using y^* to solve the subproblem, we can test whether it is a global optimal solution ($q(y^*) = q^*$) or whether it is close enough to the optimal solution. If this is not the case, we add a new constraint to the master problem corresponding to the extreme point or extreme ray we found by solving the dual subproblem. These additional constraints also are called *Benders optimality*, respectively *Benders feasibility cuts*. In a worst case all constraints of 5.12 have to be produced to find the global optimal solution. But even in this case it is possible to solve the optimization problem with a finite number of iterations.

The separation of different subproblems also allows us to run an optimization in parallel on different processors, a further starting motivation to introduce decomposition methods. But also without parallelization, splitting a large problem into smaller components and solving them independently reduces the size of branch and bound trees, and therefore the amount of memory needed to solve a problem. This fact is often the reason to introduce decomposition methods if known solution methods and used processors reach a limit of memory.

5.3 Geographical decomposition of the PESP

Using the idea of decomposition methods for the PESP is not completely new. For example Odijk introduces a constraint generation method in [Odijk, 1996], which in fact

is a kind of Benders decomposition applied to the PESP decision problem. All integer variables are separated in a master problem, which is subsequently enlarged with new feasibility cuts generated out of extreme rays of the dual subproblem containing all continuous variables. Unfortunately, this method could not help to significantly reduce the computation time to solve the PESP decision problem. Also Lindner uses decomposition methods in [Lindner, 2000] for an extended PESP model. Since in his case the subproblem is similar to the PESP decision problem, his approach is not comparable to our ideas. The main idea of our approach is the introduction of a block angular structure using cuts in the PESP graph and the definition of a corresponding decomposition method.

5.3.1 Definition of the decomposition

In this section, we define this decomposition approach for the extended cyclic MILP, introduced in Chapter 4.3.3. Motivated from common decomposition approaches in optimization, introduced in Section 5.2, we try to separate variables of our MILP to reach a block structure in the describing matrix of the MILP with as few coupling variables as possible. This separation is based on a cut in the PESP graph which we introduce in the next subsection. Subsequently, we construct a suitable cycle basis to reduce the number of coupling variables, discuss the reached block structure and introduce corresponding subproblems and a master problem to coordinate the subproblems for the decomposition.

Definition of a cut-graph

Let $G = (V, A)$ be a connected PESP graph with integer edge bounds $l(a), u(a) \in \mathbb{N}, \forall a \in A$ and $A_{cut}^0 \subset A$ a set of edges splitting G in $k \geq 2$ components G_1, \dots, G_k when the edges A_{cut}^0 are removed from G . We denote all incident vertices to A_{cut}^0 by $V_{cut}(A_{cut}^0) = \bigcup v_{ij}^{cut}$, where $0 \leq j < k$ describes a number corresponding to the component containing this vertex, and $i, 0 \leq i < n_j$ enumerates all n_j incident vertices belonging to component j , as illustrated in Figure 5.1.

Definition 8. The induced subgraph defined over all incident vertices $V_{cut}(A_{cut}^0)$ is called **cut-graph** $G_{cut}(A_{cut}^0)$. We denote the edge set of this cut-graph by $A_{cut}(A_{cut}^0)$.

Independence and cycle basis

After splitting a PESP graph into k different components, we want to use this decomposition to also split the extended cyclic MILP. Since every tension variable $x_a, a \in A$ directly corresponds to an edge $a \in A$ of the PESP graph and every integer cycle variable $q_C, C \in \mathcal{C}_B$ to a cycle of the PESP graph, we want to use this relation to separate the variables of the MILP. We define \vec{x}_{cut} as a vector collecting all tension variables x_a corresponding to an edge $a \in A_{cut}$ of the cut-graph. All remaining tension variables x_a are collected in a vector $\vec{x}_i, 1 \leq i \leq k$ corresponding to the component in which edge a is contained. Similarly we define vectors $\vec{q}_{cut}, \vec{q}_1, \dots, \vec{q}_k$ collecting the integer cycle variable. The vector \vec{q}_{cut} contains every cycle variable q_C corresponding to a cycle C completely contained in the cut-graph $G_{cut}(A_{cut}^0)$. And the vectors $\vec{q}_i, 1 \leq i \leq k$ contain the remaining cycle variables corresponding to cycles contained in the union of the

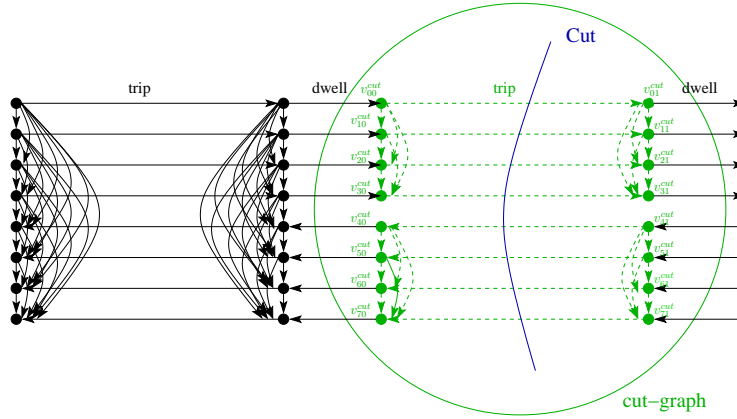


Figure 5.1: *Illustration of a cut-graph. All green vertices and green dashed edges are part of the cut-graph.*

component i and the cut graph $G_{cut}(A_{cut}^0)$. To ensure that every variable of the MILP is contained in exactly one vector, we have to construct a cycle basis \mathcal{C}_B that does not contain edges of two different components outside the cut-graph $G_{cut}(A_{cut}^0)$:

$$\forall C \in \mathcal{C}_B : \exists i \in \mathbb{N}, 1 \leq i \leq k : \{a \mid a \in C\} \subseteq G_i \cup G_{cut} \quad (5.16)$$

In this section, we introduce the construction of a strictly fundamental cycle basis satisfying Condition 5.16 and we show that we therefore get a block structure as illustrated in Figure 5.2.

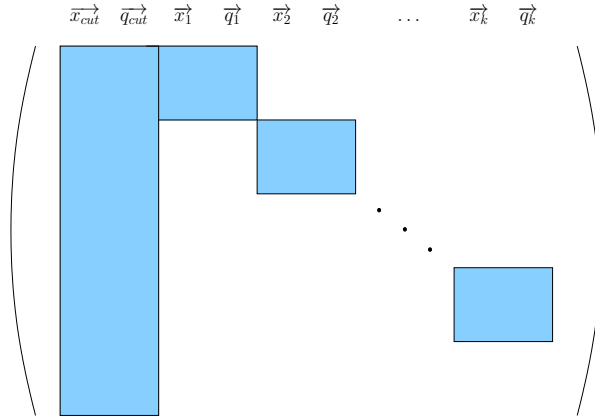


Figure 5.2: *Desired block structure for the extended cyclic MILP*

Definition 9. Let G_{cut} be a cut-graph for a PESP graph $G = (V, A)$. We assume G_{cut} to be connected, otherwise we connect it by introducing pseudo PESP constraints not restricting any solution. Let H_{cut} be a spanning tree of G_{cut} and H_G a spanning tree of G containing H_{cut} . Then we denote any strictly fundamental cycle basis constructed out of H_G by $\mathcal{C}_B(G_{cut})$.

Theorem 3. *A strictly fundamental cycle basis $\mathcal{C}_B(G_{cut})$ satisfies Condition 5.16.*

Proof. We prove Theorem 3 by contradiction. Let $C \in \mathcal{C}_B(G_{cut})$ be a cycle of this cycle basis containing edges from at least two different components outside the cut-graph ($G_i \setminus G_{cut}, G_k \setminus G_{cut}, i, k \in \{1, \dots, n_j\}$) and a_{chord} the chord edge closing this cycle in H_G . This chord edge a_{chord} has to be part of a component outside the cut-graph, say $G_i \setminus G_{cut}$, otherwise cycle C would be completely contained in G_{cut} with the given definition of $\mathcal{C}_B(G_{cut})$. Every other edge of C , apart from a_{chord} , is part of the spanning tree

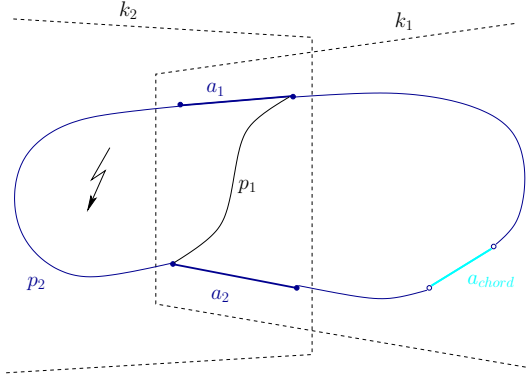


Figure 5.3: Contradicting cycle in H_G

H_G . Since the edges of C belong to at least two different components, there have to be at least two edges $a_1, a_2 \in C, a_1 \neq a_2$ belonging to the cut-graph G_{cut} , which do not share any common incident vertex. The incident vertices as a part of the cut-graph G_{cut} are connected over a path p_1 completely contained in the spanning tree H_{cut} . Since our considered cycle C contains edges of a second component $G_k \setminus G_{cut}$, there has to be a path $p_2 \neq p_1$ starting from an incident vertex of a_1 and ending at an incident vertex of a_2 over edges of the second component contained in H_G . The existence of both paths leads to a cycle in H_G contradicting the assumption that H_G is a spanning tree. The situation is depicted in Figure 5.3. \square

Since the choice of an integer cycle basis for the extended cyclic MILP does not influence the solution of the PESP, we assume $\mathcal{C}_B(G_{cut})$ to be our cycle basis of the MILP from now on for this chapter. Theorem 3 ensures that every integer variable $q_C, C \in \mathcal{C}_B(G_{cut})$ is contained in exactly one of the vectors $\vec{q}_{cut}, \vec{q}_1, \dots, \vec{q}_k$. To show that we can reach the desired block structure, we have to ensure that no constraint containing variables of \vec{x}_i and $\vec{x}_j, i \neq j$ exist at the same time. Recapping the set of constraints contained in the extended cyclic MILP we have the following three types of constraints:

bounds on tensions	$l(a) \leq x_a \leq u(a)$	$\forall a \in A$
cycle constraints	$\sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a = Tq_C$	$\forall C \in \mathcal{C}_B(G_{cut})$
non collision cycles	$\sum_{a \in C^+} x_a - \sum_{a \in C^-} x_a = 0$	$\forall C \in \mathcal{C}_N$

Constraints on tension bounds only contain one variable and therefore never violate this rule. Every non-collision cycle is composed of four edges. Even if there is an edge among

these four edges contained in the cut set A_{cut}^0 , it is not possible that two of the remaining three edges belong to two different components outside the cut graph $G_{cut}(A_{cut}^0)$. With Theorem 3 we can further ensure that no cycle constraint contains variables of \vec{x}_i and $\vec{x}_j, i \neq j$ at the same time.

Decomposition of the extended cyclic MILP

The introduced block structure of Figure 5.2 allows us to use similar ideas as in the Benders decomposition.

Definition 10. We refer to the original MILP by $GP(\vec{x}_{cut}, \vec{q}_{cut}, \vec{x}_1, \vec{q}_1, \dots, \vec{x}_k, \vec{q}_k)$ and call it the *global problem*.

If we fix all coupling variables \vec{x}_{cut} and \vec{q}_{cut} our MILP decomposes in k separated smaller MILPs, which we call from now on subproblems.

Definition 11. We define the i th *subproblem* as a MILP containing all constraints on the variables of the vectors $\vec{x}_i, \vec{q}_i, 1 \leq i \leq k$ and an objective function f_i restricted on these variables. We refer to it as $SUB_i(\vec{x}_i, \vec{q}_i, \vec{x}_{cut}, \vec{q}_{cut}), 1 \leq i \leq k$.

Similarly to the Benders decomposition, we define a master problem over the coupling variables $\vec{x}_{cut}, \vec{q}_{cut}$ to control the coordination of the subproblems.

Definition 12. Let P_X, P_Q be the feasible region of \vec{x}_{cut} and \vec{q}_{cut} defined over all constraints of the original global problem only containing variables of the cut-graph and f_{cut} the objective function reduced on these variables. Then we define the *master problem* as the optimization problem of finding an assignment for \vec{x}_{cut} and \vec{q}_{cut} minimizing $\sum_{1 \leq i \leq k} f_i + f_{cut}$, so that $\vec{x}_{cut} \in P_X, \vec{q}_{cut} \in P_Q$ and all constraints of $SUB_i, 1 \leq i \leq k$ are satisfied.

Since subproblems do not contain integer variables in a standard Benders decomposition, only a few approaches exist introducing similar ideas for mixed integer linear subproblems. For example John Hooker introduced a generalized duality, called inference duality, to adapt the idea of cutting planes for the master problem [Hooker and Ottosson, 1995]. The implementation of this kind of approach is very sophisticated and needs a specific and deep knowledge about the structure of the subproblems. Furthermore, it does not guarantee an efficient elimination of possible assignments for the master problem. Instead of using a generalized duality to find good assignments for the coupling variables in the master problem so-called no-good constraints can also be used [Trick, 2010]. They have their origin in the constraint satisfaction theory [Tsang, 1993] and exclude variable assignments in the master problem directly if they lead to infeasibility.

To work with similar ideas as in the case of no-good, constraints we use the following Lemma introduced by Odijk allowing us to restrict our defined master problem on integer assignments also for \vec{x}_{cut} .

Lemma 1. [Odijk, 1994] *Let I denote an instance of PESP with integer tension bounds $l(a), u(a), \forall a \in A$ and an integer time period T . If I admits some feasible tension values $x_a \in [l(a), u(a)]^A$ then it also admits an integer feasible solution $x'_a \in \{l(a), \dots, u(a)\}^A \cap \mathbb{Z}^A$*

Therefore, we are able to subsequently eliminate already studied assignments of \vec{x}_{cut} and \vec{q}_{cut} in the master problem. And if we end up in a worst case, a complete enumeration of all assignments, we can still ensure that we finish our computations after a finite number of iterations. To have a better impression on the polyhedron defined in the starting master problem, we study the set of possible assignments for \vec{x}_{cut} and \vec{q}_{cut} for a special type of cut graph in the next section.

5.3.2 Cuts through track sections

To keep the master problem of our decomposition as small as possible we are looking for cut-graphs G_{cut} with a small number of edges. At the same time the size of the largest subproblem should differ enough from the original problem to profit in the running time. In graph theory such a cut is called minimum balanced cut [Andreev and Räcke, 2004]. Since finding such a cut in general is *NP*-hard [Garey et al., 1976], we try to use specific properties of our application. Using the typical structure of a PESP graph for a train scheduling problem we know that headway edges often appear in large graph cliques. Therefore we avoid to choose a cut through headway edges and there remain two reasonable cuts:

- a cut through a track section, and
- a cut through a station.

Of course a complete graph cut can consist of a combination of both types. In this section we concentrate on cuts through a track section (trip edges). We assume that all edges contained in A_{cut}^0 are trip edges.

A trip edge connects the departure event of a certain train at a station with the arrival event of the same train at the next station. If two trains run on the same track for this section the incident vertices of their trip edges are connected over headway edges (see Figure 5.4). Together with the trip edges they close non-collision cycles (Section 4.3.3) to avoid collisions and overtaking on this track. In addition to the headway edges there could also be frequency and further separation edges between arrival and departure events in the same station. But for this subsection we assume that we only have headway edges with a fixed common headway time h and all trains have the same lower and upper trip times t_l, t_u .

We choose a spanning tree H_{G_B} as in Figure 5.4, which contains only one constraint out of the cut set A_{cut}^0 to define a cycle basis $\mathcal{C}_B(G_{cut})$ and study the corresponding decomposition of the MILP. The following theorem provides a bound on the number of possible assignments for the integer variables \vec{q}_{cut} :

Theorem 4. *Let n be the number of trains running on a common track, so that no three headway times sum up to more than the main period T ($3h < T$), G_{cut} a cut-graph to a cut set A_{cut}^0 which contains exactly all trip constraints of the trains on this track, and $\mathcal{C}_B(G_{cut})$ a strictly fundamental cycle basis corresponding to the spanning tree $H_{G_{cut}}$, which contains only one constraint out of the cut set A_{cut}^0 . Then the number m of possible assignments of integer variables \vec{q}_{cut} is bounded by $m \leq (n - 1)!$.*

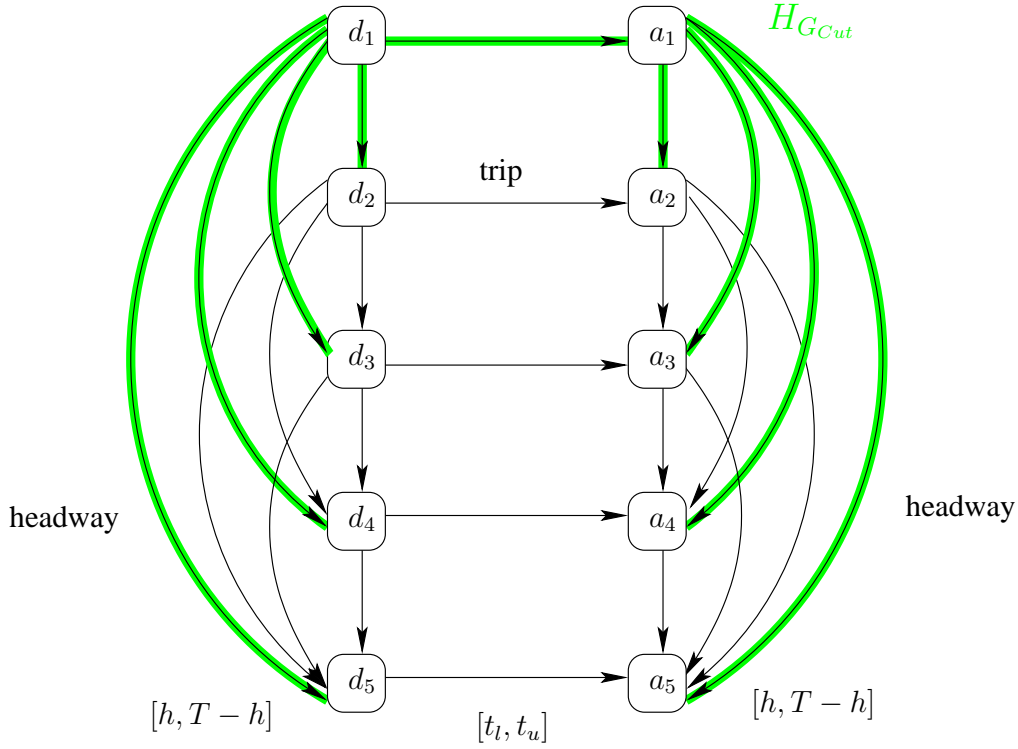


Figure 5.4: Example of a trip cut with five trains running in the same direction

Proof. With the given spanning tree $H_{G_{cut}}$ we generate two different types of cycles in the cut-graph's cycle basis $\mathcal{C}_B(G_{cut})$:

Cycle composed of three headway constraints

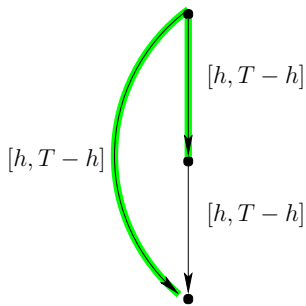


Figure 5.5: H-Cycle

Every headway edge, not contained in $H_{G_{cut}}$ closes a cycle C which contains exactly three headway constraints as in Figure 5.5. Let q_C be the integer variable corresponding to this cycle C , then we can bound q_C by $a \leq q_C \leq b$ with:

$$a = \left\lceil \frac{3h - T}{T} \right\rceil = 0, \text{ since } 3h \leq T, h > 0$$

$$b = \left\lfloor \frac{2T - 3h}{T} \right\rfloor = 1, \text{ since } 3h \leq T, h > 0$$

Cycle composed of two headway and two trip constraints

Every trip constraint not contained in $H_{G_{cut}}$ closes a cycle out of two headway and two trip constraints (see Figure 5.6).

These cycles coincide directly with the non-collision cycles. Therefore, we require that every integer variable q_C belonging to such a cycle has to satisfy $q_C = 0$.



Figure 5.6: Trip cycles of two trains running in the same / opposite direction

Let $\{q_{C_1}, \dots, q_{C_j}\}$ be the set of integer variables belonging to the headway cycles corresponding to one end of the track in G_{cut} ($j = (n - 1)!$), $\{q_{C_{j+1}}, \dots, q_{C_l}\}$ the integer variables of the non-collision cycles ($l - j = (n - 1)$) and $\{q_{l+1}, \dots, q_m\}$ the integer variables of the headway cycles formed by the headway constraints corresponding to the other end of the track. ($m - l = (n - 1)!$).

Considering a cycle containing two trip and two headway constraints, all not contained in $H_{G_{cut}}$, the corresponding integer variables $\{q_{C_{h_1}}, q_{C_{t_1}}, q_{C_{h_2}}, q_{C_{t_2}}\}$ have to satisfy the non-collision constraints $q_{C_{t_1}} + q_{C_{h_2}} - q_{C_{t_2}} - q_{C_{h_1}} = 0$ or $q_{C_{t_1}} + q_{C_{h_2}} + q_{C_{t_2}} - q_{C_{h_1}} = 0$ depending on train directions. Since $q_{C_{t_1}} = q_{C_{t_2}} = 0$ the two integer variables of the headway constraints $q_{C_{h_1}}, q_{C_{h_2}}$ have to be equal. Hence, in order to find an upper bound for m , it suffices to include the set $\{q_{C_1}, \dots, q_{C_j}\}$ in our further computations.

Bijjective relation between values of \vec{q}_C and train sequences

Let q_C be an integer variable out of the set $\{q_{C_1}, \dots, q_{C_j}\}$. Then, we already know that q_C can only have value 0 or 1. We now show a relation between the value of q_C and train sequences.

Suppose the cycle corresponding to q_C connects three events e_1, e_2, e_3 over three headway constraints x_1, x_2, x_3 as shown in Figure 5.7. The value of q_C is $\frac{x_1 + x_2 - x_3}{T}$, thus:

$$\begin{aligned} q_C = 0 &\Leftrightarrow x_1 + x_2 - x_3 = 0 \\ q_C = 1 &\Leftrightarrow x_1 + x_2 - x_3 = T \end{aligned}$$

Furthermore, there are six possible orders of the occurrences of e_1, e_2, e_3 if we consider a fixed time period from 0 to T , as indicated in Figure 5.7. Here we can observe that q_C equals 0 as long as we keep the sequence $e_1 - e_2 - e_3$ modulo starting edge. If we do one transposition in this order (i.e. $e_1 - e_3 - e_2$) we get $q_C = 1$.

Thus q_C fixes the sequence of three events modulo starting edges. If there are more arrival and departure events corresponding to the same track and station, we have such a q_C for every group of three events. one which in total fixes the sequence of every event.

On the other hand, we can deduce all variables $q_C \in \{q_{C_1}, \dots, q_{C_k}\}$ if we know the sequence of all departure and arrival events of the n trains we consider in Theorem 2. This result was also shown by Leon Peeters [Peeters, 2003], who studied the same cycles

composed of three headway edges, which he called safety triangles, in a section of his dissertation.

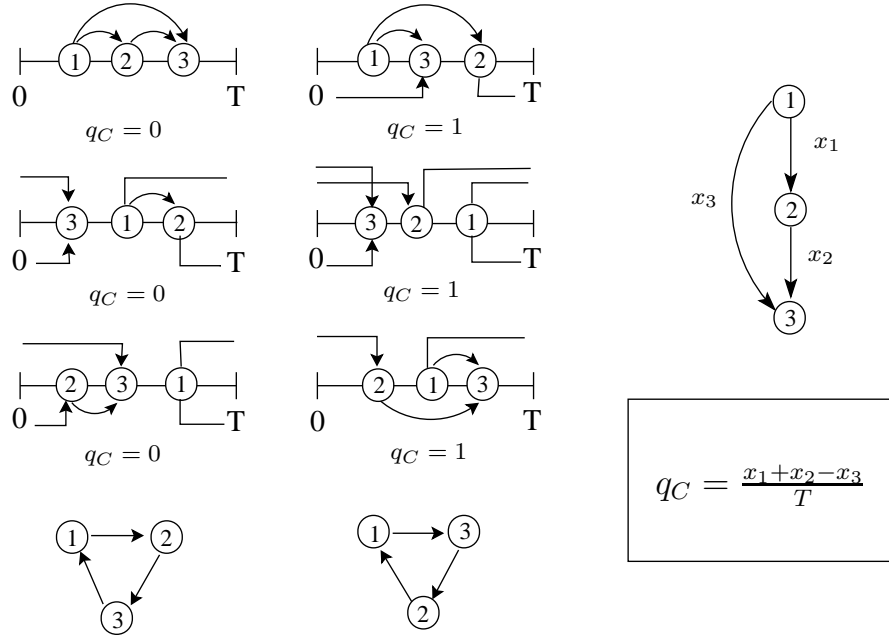


Figure 5.7: Possible orders of the occurrences of three events in a fixed time period 0 to T

With this bijective relation between the values of q_C and train sequences, we can bound m by the number $(n - 1)!$ of possible train sequences. If there are no further constraints between the considered arrival and departure times and no different trip and headway times, this bound is tight. \square

Using this bound for the possible assignments of \vec{q}_{cut} in case of a cut through one section, we can generalize it to a cut through several sections as stated in the following corollary:

Corollary 1. *Let m be the number of possible assignments for the integer variables of a cut-graph G_{cut} defined by a cut through j track sections with n_i trains per track ($1 \leq i \leq j$). Then we can bound m by*

$$m \leq \prod_{i=1}^j (n_i - 1)!,$$

as long as no sum of three headway times h is not larger than the total time period T .

Even without the assumption of equal trip and headway times, the number of possible assignments of \vec{q}_{cut} is still related to the number of possible train sequences.

Remark 1. In a general trip cut the number of possible assignments for \vec{q}_{cut} corresponds to the number of possible train sequences on the considered track.

If there are time constraints separating several pairs of trains and therefore reducing the number of possible train sequences, the number of possible assignments for \vec{q}_{cut} gets smaller as well.

Corollary 2. *Let m be the number of possible assignments of the integer variables of a cut-graph G_{cut} defined by a cut through a track with an even number n of trains. If there are $\frac{n}{2}$ disjoint pairs of trains having a time distance of 30 minutes, then we can bound m by*

$$m \leq \left(\frac{n}{2} - 1\right)! \cdot 2^{\left(\frac{n}{2}-1\right)},$$

as long as no sum of three headway times h is larger than the total time period T .

Illustrating this influence of half hour periods for every train, Table 5.1 gives some examples for a track with 2 to 10 trains per hour.

Number of trains	Possible train sequences without separation	Possible train sequences with separation
2	1	1
4	6	2
6	129	8
8	5040	48
10	362880	284

Table 5.1: Influence of train separation constraints on the possible number of train sequences

After showing dependencies of assignments for \vec{q}_{cut} and train sequences, we want to study the possible assignments for \vec{x}_{cut} . Lemma 2 gives a first bound.

Lemma 2. *Let G_{cut} be the cut-graph of a general trip cut, n the number of trains running on a common track, with h, t_l and t_u common headway and trip times for all trains and $T \in \mathbb{N}$ the total time period. Suppose the sequence of trains, i.e. \vec{q}_{cut} , is fixed. Then the number m of feasible assignments for \vec{x}_{cut} is bounded by the following threshold:*

$$m \leq \binom{T - n(h - 1) - 1}{n - 1} \cdot (t_u - t_l + 1)^n$$

Before we prove Lemma 2 we state the following remark based on the fact that for every set of tension variables corresponding to a cycle in the PESP graph, a tension variable is already fixed after assigning a value to the remaining variables of this cycle.

Remark 2. The number of possible assignments of \vec{x}_{cut} is equal to the number of possible assignments of \vec{x}_{0Cut} , where \vec{x}_{0Cut} is a vector collecting a subset of all tension variables of \vec{x}_{cut} corresponding to a spanning tree $H_{G_{cut}}$ of the cut-graph G_{cut} .

Proof. To prove Lemma 2 let us consider \vec{x}_{0Cut} corresponding to a spanning tree containing all trip edges as illustrated in Figure 5.8.

The number of possible assignments for all tensions corresponding to the trip edges is smaller than or equal to $(t_u - t_l + 1)^n$. This threshold is not necessarily reached if the

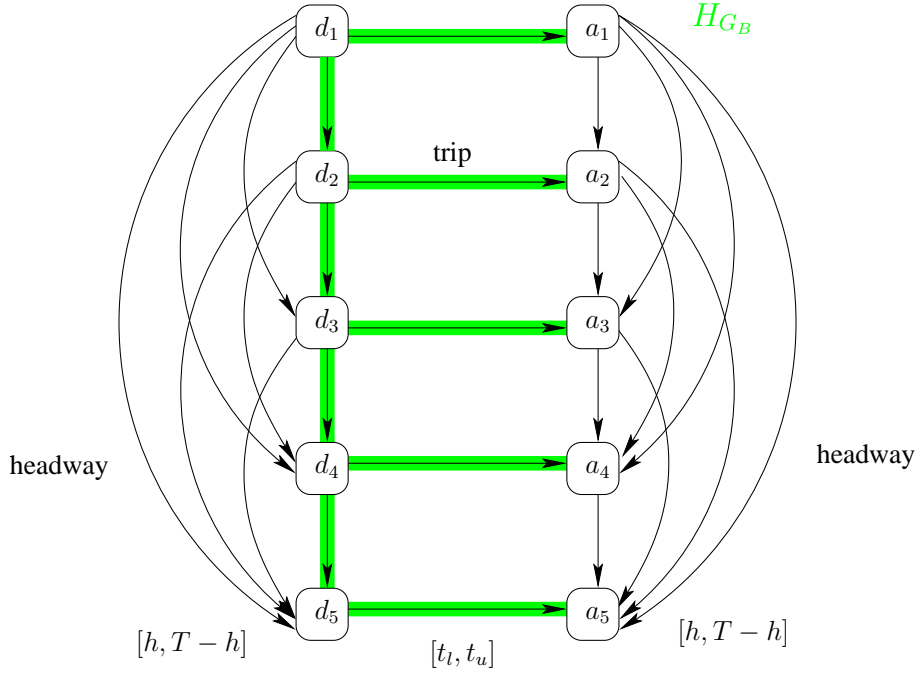


Figure 5.8: Illustration of the spanning tree depicted by green edges chosen to study the number of possible assignments for all tension variables in the cut-graph

interval size $t_u - t_l$ is large and non-collision cycles could therefore avoid certain combinations of trip time assignments. It remains to show that the number of possible assignments for all tension variables of $\overrightarrow{x_{0Cut}}$ corresponding to headway constraints is smaller than or equal to $\binom{T-n(h-1)-1}{n-1}$. For this we consider the following combinatorial problem: We fix a first train departure event on minute 0 and study the number of possibilities to schedule the remaining $(n-1)$ trains with the given sequence and required headway time h on integer departure minutes. All together, there are T possible departure minutes. One is already occupied by fixing the first train. Furthermore, we can reduce the number of possible departure minutes by $(h-1) \cdot n$ since we have to ensure a minimal time difference of h minutes between every consecutive train. Therefore, we get the combinatorial problem of finding the number of possibilities to choose a group of $(n-1)$ departure minutes out of $(T - n(h-1) - 1)$ possible departure minutes, which leads to the given binomial coefficient $\binom{T-n(h-1)-1}{n-1}$. \square

The following table shows how fast the number of possible assignments of $\overrightarrow{x_{cut}}$ grows with the number of trains, even if we fix trip times. Using a headway time of $h = 2$ minutes and a main period $T = 60$ minutes, we can have up to 30 trains. Approaching this maximum capacity of the track section, the number of assignments of $\overrightarrow{x_{cut}}$ decreases again down to one possibility, whereby the number of possible assignments for $\overrightarrow{q_{cut}}$ in this case reaches a number exceeding every practicability for an enumeration in an algorithm.

In Table 5.2 we can see that in a very general case an enumeration of all possible assignments of $\overrightarrow{q_{cut}}$ in practice is only realizable for a small number of trains. In the case of $\overrightarrow{x_{cut}}$

Number of trains	Possible number of assignments for \vec{x}_{cut}	Possible number of assignments for \vec{q}_{cut}
2	57	1
3	1'540	2
5	316'251	24
10	2'054'455'634	362'880
20	68'923'264'410	121'645'100'408'832'000
25	131'128'140	620'448'401'733'239'439'360'000
28	31'465	1'088'886'950'418'352'160'768'000'000
30	1	8'841'761'993'739'701'954'543'616'000'000

Table 5.2: Exponential growth of possible assignments for variables in a general cut-graph

we should avoid any enumeration. The next subsection introduces two algorithmic approaches to deal with the coordination problem of our decomposition by cutting through a track section.

5.3.3 Heuristics for the coordination of the master and two subproblems

In this section, we introduce two algorithms to find an adequate assignment of the variables \vec{q}_{cut} and \vec{x}_{cut} of the master problem in the case of two subproblems. The most simple exact method would be a complete enumeration of every integer variable assignment. As we saw in the last subsection, especially for \vec{x}_{cut} , this would exceed computation times for practical applicability. Therefore, we introduce two heuristics to find a feasible assignment for \vec{x}_{cut} , after the fixation of an assignment for \vec{q}_{cut} .

Heuristic space search

Instead of a complete enumeration of all feasible assignments for the master variables \vec{x}_{cut} we want to test a set of \vec{x}_{cut} assignments distributed over all \vec{x}_{cut} assignments, feasible to the corresponding lower and upper bounds given by the Odijks cuts (see Thm 2). For this we consider the space spanned over all these \vec{x}_{cut} assignments, which we call from now on X_{cut} -Space. We distribute *maxIt* points in the X_{cut} -Space and lead the \vec{x}_{cut} assignment for the first subproblem in the direction of such a point.

Altogether we do the following steps: We start with an arbitrary solution of the first subproblem and try to find a global solution by fixing the first subproblem's integer variables \vec{q}_1 and \vec{q}_{cut} in the global problem. Solving this global problem has a running time comparable to solving the second subproblem. If we cannot find a feasible global solution, we influence the solution of the first subproblem over an objective function with the goal to move the \vec{x}_{cut} assignment of the first subproblems solution in a new part of the X_{cut} -Space. We do this by adding the minimum of the Euclidean distance between \vec{x}_{cut} and one of the defined points to its objective function. In this way the original linear objective function now includes quadratic terms, but can be solved with commercial solvers in a

comparable amount of time. As long as we cannot find a feasible global solution, we repeat this step for a next point. If we run all $maxIt$ iterations without finding a feasible global solution, we stop without knowing whether a global feasible solution exists. To prove infeasibility with this algorithm we would need a corresponding point in the X_{cut} -Space for every possible assignment of \vec{x}_{cut} that directly leads us to an enumeration over all possible integer \vec{x}_{cut} assignments. The algorithm is summarized with a pseudo code in Algorithm 1.

Algorithm 1 *Heuristic space search to find feasible global solutions*

Input: Integer variable \vec{q}_{cut} of the cut-graph, maximum number of iterations $maxIt$
Output: Feasible Global Solution with $\vec{q}_{cut} = \vec{q}_{cut}$, if the algorithm finds one after $maxIt$ iterations
boolean foundGSolution := false;
objPoints := List of $maxIt$ points distributed in the X_{cut} -Space;
Solve $Sub1(\vec{x}_1, \vec{q}_1, \vec{x}_{cut}, \vec{q}_{cut})$;
if $Sub1$ is feasible **then**
 $\vec{q}_1 := \vec{q}_1$ of $Sub1$'s solution;
Solve $GP(\vec{x}_1, \vec{q}_1, \vec{x}_{cut}, \vec{q}_{cut}, \vec{x}_2, \vec{q}_2)$;
if GP feasible **then**
Print global solution;
foundGSolution = true;
else
for all \vec{x}_{cutObj} in objPoints **do**
Solve $Sub1(\vec{x}_1, \vec{q}_1, \vec{x}_{cut}, \vec{q}_{cut})$ with the additional objective function minimizing the Euclidean distance between \vec{x}_{cut} and \vec{x}_{cutObj} ;
 $\vec{q}_1 := \vec{q}_1$ of $Sub1$'s solution;
Solve $GP(\vec{x}_1, \vec{q}_1, \vec{x}_{cut}, \vec{q}_{cut}, \vec{x}_2, \vec{q}_2)$;
if GP feasible **then**
Print global solution;
foundGSolution = true;
break;
end if
end for
if not foundGSolution **then**
print "No global solution found."
end if
end if
else
print "There is no global solution";
end if

Hyperplane heuristic

Instead of partitioning the X_{cut} -Space in advance, in this method we want to subsequently introduce separating hyperplanes to split the X_{cut} -Space forcing new \vec{x}_{cut} assignments

for both subproblems. These hyperplanes are based on the following idea: If we fix all integer variables \vec{q}_i and \vec{q}_{cut} for one subproblem, we can project the polyhedron of the corresponding LP's solution space in the X_{cut} -Space and get a connected polyhedron in the X_{cut} -Space. If we can find such a projected polyhedron for the other subproblem, overlapping in at least one point, we have a feasible global solution. Thus we can translate our goal to find two overlapping polyhedrons in the X_{cut} -Space.

We start our search with an arbitrary solution of the first subproblem. To this solution we fix all integer variables, consider the projected polyhedron P_1 in the X_{cut} -Space and test the existence of a polyhedron to the other subproblem overlapping P_1 . We do this test again with the help of the global problem, where we fix all integer variables of the first subproblem. If a solution exists, there has to be an overlapping and we directly have a global feasible solution. On the other hand, if this global problem is infeasible, there is no polyhedron of the other subproblem overlapping P_1 .

We consider an arbitrary solution of the second subproblem together with its corresponding polyhedron P_2 and do the same step in the other direction. If we do not find a polyhedron of the first subproblem overlapping P_2 , we construct a separating hyperplane between the two disjoint polyhedrons P_1 and P_2 . We continue our search on both sides of the hyperplane separately. This means, for both subproblems we introduce additional linear constraints requiring the coupling master variables \vec{x}_{cut} to lie on the desired side of the hyperplane. In this way we ensure that we consider new integer assignments for the two subproblems, which further correspond to other \vec{x}_{cut} assignments, since they have to lie in an other part of the X_{cut} -Space.

Restarting the whole procedure on both sides of the hyperplane and continuing to add further hyperplanes in case of disjoint polyhedrons, our number of considered subspaces grows exponentially. But as Theorem 5 shows, the procedure will stop after a finite number of steps.

Algorithm 2 *Hyperplane Heuristic to find feasible global solutions*

Input: Integer variable \vec{q}_{cut} of the cut-graph
Output: Feasible Global Solution with $\vec{q}_{cut} = \underline{q}_{cut}$, if the algorithm finds one after *maxIt* iterations
boolean foundGSolution := false;
int iterationNumber := 0;
hyperPlanes := empty list;
LookForGlobalSolution(hyperPlanes);

Algorithm 3 *LookForGlobalSolution(hyperPlane)*

```

iterationNumber++;
Solve Sub1( $\vec{x}_1, \vec{q}_1, \vec{x}_{cut}, \vec{q}_{cut}$ ) together with all hyperPlane constraints;
if Sub1 is feasible then
   $\vec{q}_1 := \vec{q}_1$  of Sub1's solution;
  Solve GP( $\vec{x}_1, \vec{q}_1, \vec{x}_{cut}, \vec{q}_{cut}, \vec{x}_2, \vec{q}_2$ );
  if GP feasible then
    Print global solution;
    foundGSolution = true;
  else
    Solve Sub2( $\vec{x}_{cut}, \vec{q}_{cut}, \vec{x}_2, \vec{q}_2$ ) together with all hyperPlane constraints;
    if Sub2 is feasible then
       $\vec{q}_2 := \vec{q}_2$  of Sub2's solution;
      Solve GP( $\vec{x}_1, \vec{q}_1, \vec{x}_{cut}, \vec{q}_{cut}, \vec{x}_2, \vec{q}_2$ );
      if GP feasible then
        Print Global Solution;
        foundGSolution = true;
      else
        Solve Sub1( $\vec{x}_1, \vec{q}_1, \vec{x}_{cut}, \vec{q}_{cut}$ ) with the additional objective function mini-
          mizing the Euclidean distance to the solution of Sub2;
         $x_{C0} :=$  solution value of  $\vec{x}_{cut}$ ;
        Solve Sub2( $\vec{x}_{cut}, \vec{q}_{cut}, \vec{x}_2, \vec{q}_2$ ) with the additional objective function mini-
          mizing the Euclidean distance to the solution of Sub1;
         $x_{C1} :=$  solution value of  $\vec{x}_{cut}$ ;
         $N :=$  normal hyperplane through the middle of  $x_{C0}$  and  $x_{C1}$ ;
        if iterationNumber < maxIt and not foundGSolution then
          LookForGlobalSolution(hyperplanes + N(left)Constraint);
          LookForGlobalSolution(hyperplanes + N(right)Constraint);
        end if
      end if
    end if
  end if
end if
if not foundGSolution then
  print "No global solution found.";
end if

```

Theorem 5. *The Hyperplane Heuristic method terminates after finitely many iterations.*

Proof. Lemma 2, similar argumentation, but also the fact that there are only finitely many assignments for \vec{q}_i , show that also the possible number of polyhedrons of both subproblems is bounded by a number n . In a worst case, we have n disjoint polyhedrons and therefore an infeasible case. To ensure infeasibility we have to do the Hyperplane Heuristic procedure as long as in each considered subspace there remains only polyhedrons of one subproblem. We reach this at the latest if there is at least one polyhedron in each subspace.

How do we reduce the number of polyhedrons adding a new hyperplane? On both sides of a new hyperplane we reduce the number of polyhedrons by at least one. In a worst case every other polyhedron aside from the two considered disjoint polyhedrons can lie on both sides of the hyperplane (the hyperplane splits every other polyhedron in two halves). In such a case the upper bound for this step could be tight for one iteration step. Thus the number of hyperplanes we have to introduce is bounded by $\sum_{i=0}^{n-1} 2^i = 2^n - 1$ \square

high for a complete enumeration of all \vec{q}_{cut} assignments in a practical application. Furthermore all reached objective values over the decomposition methods were worse than the objective value of the first feasible solution found with the global optimization. Testing different ideas to use the results as starting solutions for the global optimization, some benefits could be shown using the best solution determined over the heuristic space search algorithm out of a few randomly chosen \vec{q}_{cut} assignments. Nevertheless a generalization of the decomposition ideas for larger problem cases would require enlarging the number of subproblems and the size of the cut-graph. The complexity of the methods, together with only marginal benefits for the whole optimization process in the test case and unsolved strategies for infeasible instances, motivated to redefine our decomposition ideas. In the next section we introduce another approach simplifying the connection of more than two

5.4 Computational results

The heuristics introduced in Section 5.3.3 are tested on a smaller problem instance already used in earlier research at ETH Zurich [Caimi, 2009]. This model therefore does not belong to the seven introduced test instances in Section 3.4. This is the case because the algorithms in this chapter were developed and evaluated before we received provided data from SBB. The test region contains the triangle Lucerne, Zug and Arth-Goldau in central Switzerland and is based on Switzerland's passenger timetable of the year 2007. Trip and dwell times, as well as connection constraints were reengineered from public timetable and general assumptions on headway times and driving paths were taken. However, from a mathematical point of view the corresponding PESP model is comparable to the others introduced in Section 3.4. Furthermore, all used edge bounds in the corresponding PESP graph are integer, and therefore satisfy our integer assumptions on edge bounds used in this chapter. The considered test region is not large, but contains several changes between single and double track lines, as well as a mixture of different train types. In total there are 46 trains visualized in the line map of Figure 5.9.

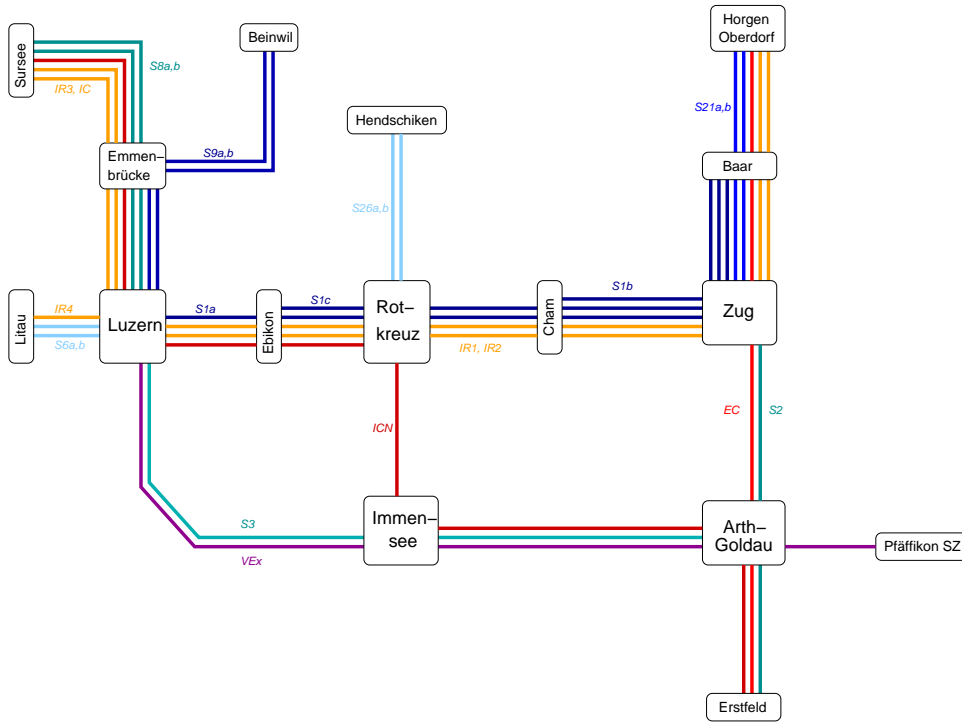


Figure 5.9: Line map of the test instance

The infrastructure is modelled very roughly (Figure 5.10). As soon as there are two tracks, trains are supposed to be separated by their direction. Using Theorem 4 three cuts through infrastructure sections were defined with a boundary for $\overrightarrow{q_{Cut}}$ assignments as small as possible. The sections used for the cuts are illustrated in Figure 5.10 together with the infrastructure.

The cuts split the global MILP in two subproblems with the following number of variables:

$$\text{Cut 1: } \begin{array}{lll} |\overrightarrow{x_{Cut}}| = 43 & |\overrightarrow{x_1'}| = 815 & |\overrightarrow{x_2'}| = 467 \\ |\overrightarrow{q_{Cut}}| = 9 & |\overrightarrow{q_1'}| = 312 & |\overrightarrow{q_2'}| = 173 \end{array}$$

$$\text{Cut 2: } \begin{array}{lll} |\overrightarrow{x_{Cut}}| = 49 & |\overrightarrow{x_1'}| = 1077 & |\overrightarrow{x_2'}| = 199 \\ |\overrightarrow{q_{Cut}}| = 6 & |\overrightarrow{q_1'}| = 408 & |\overrightarrow{q_2'}| = 72 \end{array}$$

$$\text{Cut 3: } \begin{array}{lll} |\overrightarrow{x_{Cut}}| = 31 & |\overrightarrow{x_1'}| = 687 & |\overrightarrow{x_2'}| = 607 \\ |\overrightarrow{q_{Cut}}| = 9 & |\overrightarrow{q_1'}| = 272 & |\overrightarrow{q_2'}| = 219 \end{array}$$

To solve our test instances we used CPLEX 12 and a Lenovo IBM Intel(R) Core(TM) i7 CPU notebook (2.67 GHZ, 3.9 GB RAM). Solving the test instance without decomposition to a gap of 1% takes more than 7 hours. But after 540 seconds there is a first feasible

solution with a total travelling time of 1935 minutes. After 7 hours the minimized total travelling time is 1886.5 minutes.

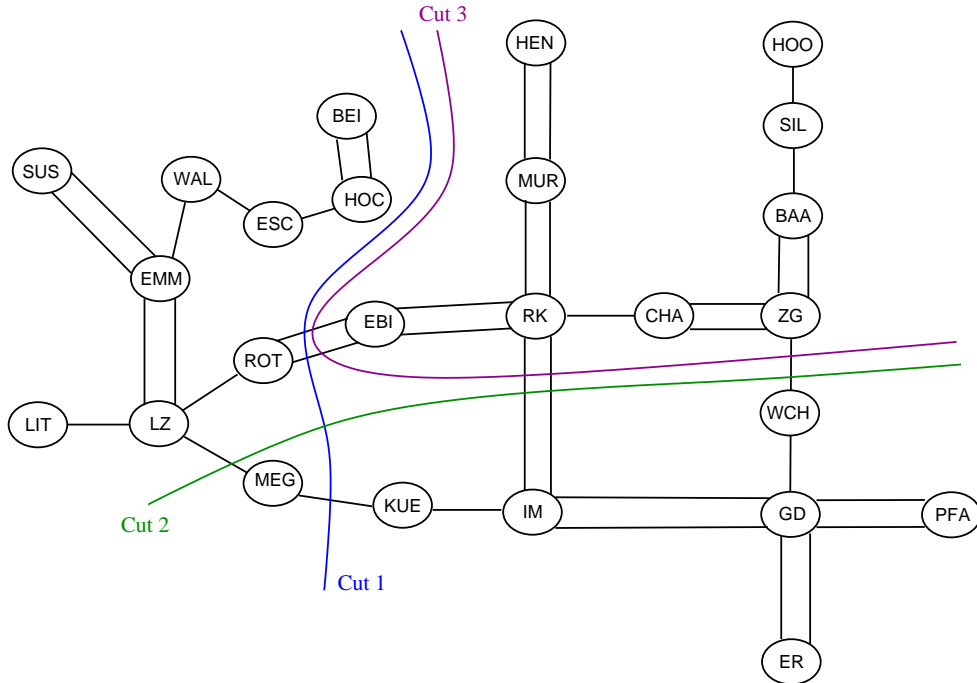


Figure 5.10: Infrastructure of the test instance with three cuts to test the geographical decomposition

To cut 1 and cut 3 there are 216 possible assignments for \vec{q}_{Cut} which are feasible in the cut-graph. And 36 assignments are feasible in the cut-graph of cut 2. The following two subsections show computational results for the heuristic decomposition algorithms tested to all three cuts. In both cases a fixed threshold is used for the maximal number of iterations allowed in the decomposition. To evaluate the loss of feasible global solutions through this breaking off criteria, we tested all assignments of \vec{q}_{Cut} for the global problem.

5.4.1 Hyperplane heuristic

For the hyperplane heuristic we used a threshold of 15 iterations. Table 5.3 shows indicators of our computational results running the hyperplane heuristic for every assignment of \vec{q}_{Cut} . In case of cut 2 and 3 several \vec{q}_{Cut} assignments exist, for which the global problem is infeasible. The hyperplane heuristic already recognized them solving the subproblems separately before ending with a complete iteration. In these cases we counted 0 iterations for the result. Computation times were distributed very widely. For the first cut we exceeded the global computation time to find a first feasible solution of 540 seconds 45 times. In the case of cut two this happened twice, once with 1161 seconds and a second one, an extreme case, with all of 5 hours computation time. For the third cut all computation times were shorter than 540 seconds. While having some benefits for the average computation time to find a first feasible solution, only a few computations lead

to an objective value of a quality comparable to the first feasible solution of the global problem.

Hyperplane heuristic	Cut 1	Cut 2	Cut 3
number of computations leading to a global solution	182	26	62
number of computations without finding a global solution	34	10	155
number of $\overrightarrow{q_{Cut}}$ assignments without having a global solution	0	9	36
average objective value in minutes	2014	1987	2013
min objective value in minutes	1939.5	1933.5	1903
max objective value in minutes	2170.5	2085	2119
average computation time in seconds	311	560	155
min computation time in seconds	2	0	0
max computation time in seconds	1436	17874	491
average number of iterations	6.13	2.25	10.07
min number of iterations	1	0	0
max number of iterations	15	15	15

Table 5.3: Results of the hyperplane heuristic for all three cuts

5.4.2 Heuristic space search

For the heuristic space search algorithm we used an upper bound of 10 iterations for each computation. We always started our computations over the larger subproblem. Compared to the hyperplane heuristic, the average computation time per $\overrightarrow{q_{Cut}}$ assignment is considerably smaller and not that widely distributed. But the average objective value in all cases loses in quality furthermore. Table 5.4 shows all indicators of our computations. The global computation time to find a first feasible solution was exceeded only for cut 2. In this case it was exceeded even four times with more than 1700 seconds.

Heuristic space search	Cut 1	Cut 2	Cut 3
number of computations leading to a global solution	179	25	111
number of computations without finding a global solution	37	11	105
number of $\overrightarrow{q_{Cut}}$ assignments without having a global solution	0	9	36
average objective value in minutes	2028	1999	2033
min objective value in minutes	1941.5	1935	1950
max objective value in minutes	2122.5	2089.5	2170
average computation time in seconds	25.4	298	10.7
min computation time in seconds	3	0	0
max computation time in seconds	198	3521	106
average number of iterations	3.47	4	3.6
min number of iterations	1	1	1
max number of iterations	10	10	10

Table 5.4: Results of the heuristic space search for all three cuts

5.4.3 Expected computation time to find a solution of a certain quality

To have a better impression of the two heuristic methods in comparison to the original global problem directly solved with CPLEX, in this subsection we compute expected computation times to find a solution with a certain optimality gap. For each of the two methods we choose the sequence of fixed \vec{q}_{Cut} assignments randomly uniformly distributed out of every possible enumeration and run the method until we either find a solution satisfying our condition on objective value, or we reach the end of our list.

Let Z_{It} be a random variable describing the number of necessary iterations, e the number of \vec{q}_{Cut} assignments leading to a solution with an objective value of desired quality and d the number of remaining \vec{q}_{Cut} assignments. Then, we can compute the expected number of Z_{It} with the following sum:

$$E[Z_{It}] = \sum_{k=1}^{d+1} \frac{\binom{d}{k-1} \cdot e}{\binom{e+d}{k}} = \sum_{k=1}^{d+1} \frac{d! \cdot e \cdot (e+d-k)!}{(d-k+1)! \cdot (e+d)!} \cdot k.$$

To acquire the expected computation time, we have to include the computation times for every \vec{q}_{Cut} iteration. These computation times are uniformly distributed among the computation times for all e successful \vec{q}_{Cut} assignments and among all d remaining \vec{q}_{Cut} assignments. Therefore let CT_e be the average computation time among all successful \vec{q}_{Cut} assignments and CT_d the average computation time among all remaining \vec{q}_{Cut} assignments. We can determine the expectation value of the computation time Z_{CT} with the following sum:

$$E[Z_{CT}] = \sum_{k=1}^{d+1} \frac{d! \cdot e \cdot (e+d-k)!}{(d-k+1)! \cdot (e+d)!} \cdot (CT_e + (k-1) \cdot CT_d)$$

	15 %	10 %	5 %	3 %
Global Optimization	500	500	500	1261
Hyperplane heuristic with cut 1	371	407	1875	33848
Hyperplane heuristic with cut 2	750	779	1448	6736
Hyperplane heuristic with cut 3	544	574	2762	16743
Heuristic space search with cut 1	31	36	325	2760
Heuristic space search with cut 2	420	437	1229	5365
Heuristic space search with cut 3	21	25	232	-

Table 5.5: *Expected computation time in seconds to find a global solution of different qualities measured by their gap, compared to the computation time to solve the global problem with the given MipGap tolerance directly*

Table 5.5 shows all expected computation times to solve the global problem until a gap of 15, 10, 5 and 3% is reached with the two decomposition methods for every cut and

compares it to the global optimization problem directly solved with CPLEX to the corresponding gap.

5.4.4 Decomposition method to find a starting solution

The results of the last subsection show that our decomposition method can often find a first feasible solution faster than CPLEX. But to find a solution with a small optimality gap, the original CPLEX method remains stronger. Therefore, we try to combine the advantages of both methods in this section, taking solutions of the decomposition method as starting solutions in normal CPLEX. For every cut and heuristic method we tested three ideas and compared them to the optimization process without decomposition.

In a first step we start our heuristic methods without enumeration. Thus we do not fix a given train sequence, we just leave this decision to the first subproblem. Figure 5.11 illustrates the results. As soon a global solution is found with a given method, the graphic shows the total passenger travel time and its minimization depending on the computation time.

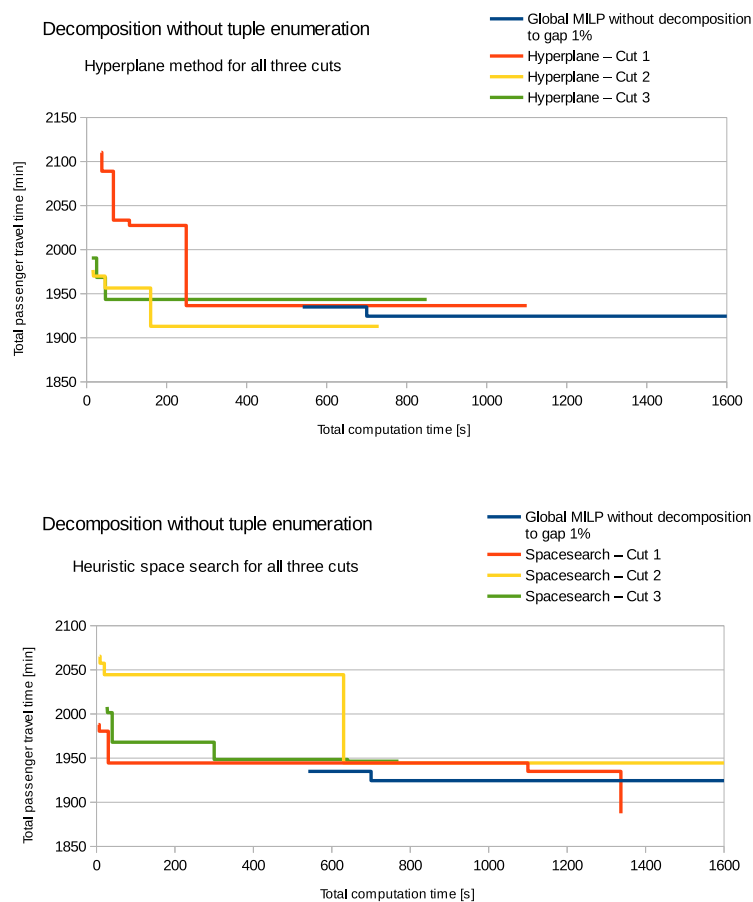


Figure 5.11: *The two heuristic methods without enumeration used to find a starting solution for global optimization*

For the second test, the first feasible solution out of the enumeration is taken as a starting solution for CPLEX. The results illustrated in Figure 5.12 show a faster detection of a first feasible solution in all cases, but a worse optimization starting from the given solution.

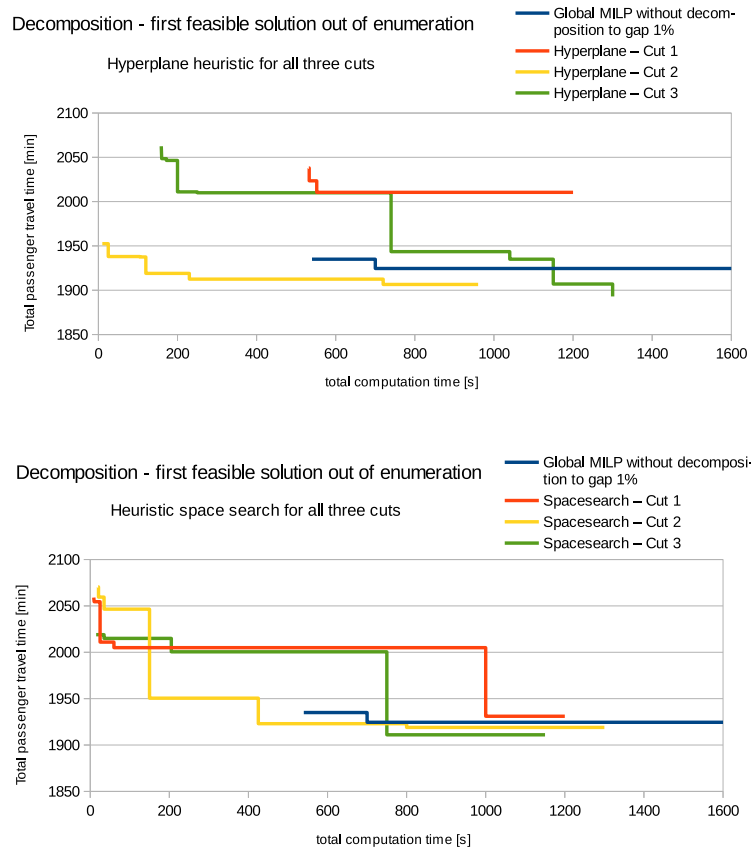


Figure 5.12: First feasible solution of the two heuristic methods used as a starting solution for global optimization

To improve the objective value in a third test, we compute the first five feasible solutions of the enumeration and use the best of them to start the optimization of the global problem afterwards. Figure 5.13 illustrates the computational results. In the case of the Hyperplane Heuristic and two cuts the computation time to construct these first five feasible solutions exceeds the computation time of the global method to find a first feasible solution. But the optimization starting from the best found solution quickly leads to equal, or even slightly better solutions compared to the global optimization. In case of the Spacesearch Heuristic for all three cuts, a better solution could be found in shorter computation times compared to the global solution method.

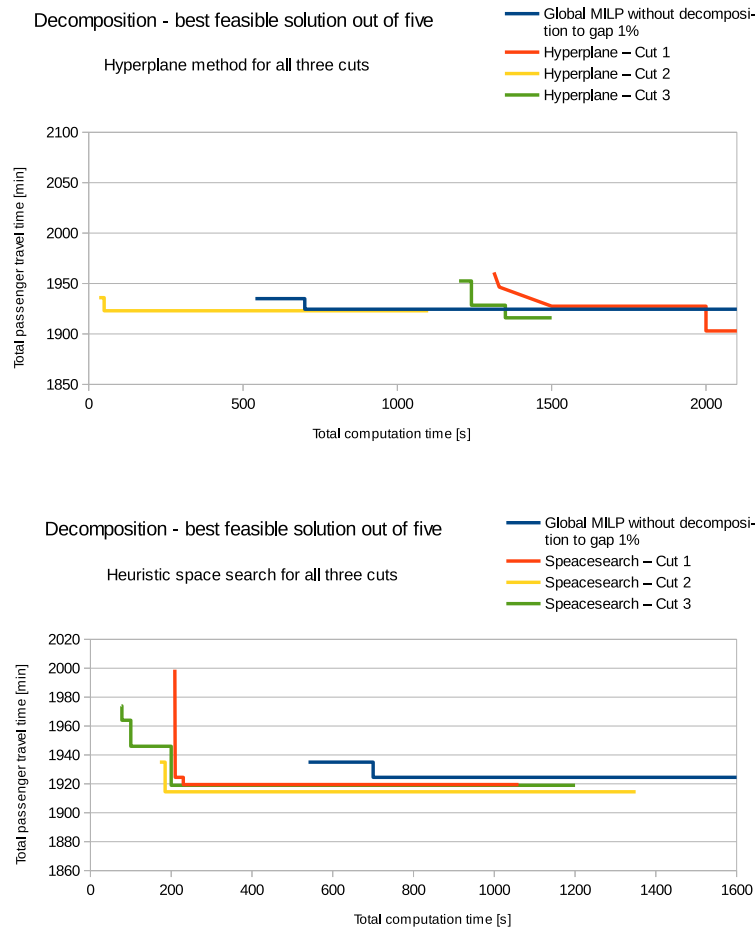


Figure 5.13: *Best of the first five feasible solutions of the two heuristic methods used as a starting solution for global optimization*

5.4.5 Summary of the results

The computational results of the introduced decomposition methods and the given test model show benefits in the computation time to detect first feasible solutions. For quite all fixations of the tuple \vec{q}_{cut} , the Heuristic Space Search as well as the Hyperplane Heuristic found global feasible solutions faster than the original global solution method. But for the largest part of all fixations of \vec{q}_{cut} the objective value was worse than the one reached with the global optimization method. We have not been able to answer which fixation of \vec{q}_{cut} leads to the best objective value. In addition to the number of possible fixations will be non-practicable to enumerate for larger models. But using the constructed timetables as a starting solution in the global optimization process can improve the overall optimization process. Furthermore, the experience out of the computations show considerable improvements of computational performances by the fixation of already a small amount of variables in the timetabling problem. This is also used as a starting motivation to the research of the next chapter.

6 Sequential decomposition

6.1 Introduction

To simplify the introduction of a higher number of subproblems and to allow more flexibility between them, we introduce a second algorithmic approach to decompose the PESP in this chapter. Instead of cutting the PESP model strictly geographically, we separate groups of train lines and schedule them sequentially. This approach is motivated from planning practice. The introduction of back iterations and flexible fixation constraints for already scheduled train lines ensure that we find a feasible solution if one exists. Computational results even show promising benefits in accelerating computation times to find solutions close to optimality especially for large instances.

In order to motivate this second decomposition method, Section 6.2 summarizes known decomposition ideas of the manual timetabling process. Subsequently, Section 6.3 introduces the theoretical framework of the second decomposition method. It describes the algorithmic approach in Subsection 6.3.1 and discusses some mathematical properties of the approach in Subsection 6.3.2. Section 6.4 shows the results of the second decomposition method. It is partitioned in five subsections and starts with Subsection 6.4.1 introducing different partitions of train lines used for the computational study. Subsequently Subsection 6.4.2 discusses different types of fixation constraints. Then Subsection 6.4.3 shows results and studies the influence of the starting time fixation margin on the overall computation time and timetable quality. A subsequent subsection, Subsection 6.4.4 gives an overview of the experience with infeasible timetabling problems in connection with the introduced decomposition method. Finally, Subsection 6.4.5 compares the results of the decomposition method with the original global solution method using no decomposition.

6.2 Decomposition in the manual planning process

Due to the high complexity and amount of work, manually planned traffic systems are only adapted and optimized locally [Weidmann, 2011a]. Spatial and settlement development, new strategies from public authorities, systematization of the network and timetable, new linkages between different traffic systems, as well as cooperation of train operating companies lead to changes in functional requirements, which further on can require adaptations of the infrastructure. Those adaptations of functional requirements and of the infrastructure are then realized and included sequentially into an already existing timetable. Therefore, timetables often grow historically. Furthermore, fixed train hierarchies among different train types, regional bounded case studies and the ongoing liberalization of the railway market further promote sequential procedures. Such procedures we can compare to a decomposition of the whole timetabling problem. Also timetabling

studies concentrating on single corridors or on main station areas can be counted to such a manual decomposition. In addition to the decomposition into different geographical regions and train lines, also a decomposition into different model granularities can be found. Depending on the planning stage, timetables are constructed with varying time granularities and infrastructure considerations. Thus in an early planning stage also manual planners often work with a granularity of one minute, aggregated infrastructure and simplified speed profiles visualized in time timetable graphs. Often passenger trains and freight trains are planned sequentially, starting with all passenger trains, since their functional requirements are known much earlier.

6.3 Sequential decomposition for the PESP

The idea of adapting heuristic methods from manual planning practice to solve the PESP is not new. Already in [Nachtigall and Voget, 1996] a greedy approach fixing one train line after the other was stated as a fast method to find first feasible solutions. Furthermore, Leon Peeters introduced the so-called cycle fixation heuristic in his dissertation [Peeters, 2003] and showed promising results for the practical optimization of cyclic railway timetables. He worked with groups of train lines scheduled sequentially while fixing the sequence of previously scheduled trains over integer cycle variables. The approach was tested for the largest model considered in his dissertation which could not be optimized to optimality. They could find a feasible timetable in a reasonable amount of time which had a better quality, measured by the objective value, compared to solutions of the constraint programming solver CADANS. But, in case of infeasibility during the sequential timetable construction, there is no backtracking introduced to ensure finding a feasible solution if one exists.

In this chapter we introduce an algorithmic approach motivated from manual planning practice using similar ideas as Peters. Slightly generalized methods to fix already scheduled trains together with a backtracking ensure the detection of feasible solutions and even allow decisions on compromises between fast computation times and optimization. We decompose the PESP model over the partition of train lines into $p > 1$ disjoint groups. The algorithm iteratively enlarges the PESP model adding the next group of train lines while fixing already scheduled train lines by adding a margin of size t_w , $0 \leq t_w \leq T$ around their scheduled departure time from previous iteration. This idea of additional fixation constraints allows a continuous transition from strong sequential timetabling ($t_w = 0$) to a complete synchronous optimization approach ($t_w = T$). The approach of a strong sequential timetabling also can be compared to the so-called asynchronous railway simulation framework, where train paths are fixed sequentially.

The concept was originally aimed at study the influence of hierarchical train prioritizations in timetable construction via the periodic event scheduling problem. During first studies it resulted to be an efficient way to accelerate finding good feasible solutions for the PESP optimization model at the same time. In Subsection 6.3.1, the algorithm for the sequential decomposition is introduced. Subsequently computational experience and

a mathematical background are discussed in Subsection 6.3.2 and 6.4. Section 6.3 is partially based on [Herrigel et al., 2013].

6.3.1 Algorithmic approach

Let $G = (V, A)$ be a PESP graph with vertex set V and edge set A . The corresponding optimization problem of $G = (V, A)$ together with an objective function formulated as the extended cyclic MILP introduced in Chapter 4.3.3 will be referred to as the *global optimization problem*. To introduce a partition of all train lines in the global optimization problem we denote the set of train line numbers contained in the original PESP model by L . Consequently we are able to define the three main parameters for our decomposition algorithm:

A **partition number** p fixing the number of sets we want to use,

A **partition function** $f_{prio} : L \rightarrow \{1, \dots, p\}$ assigning a priority number $f_{prio}(l) \in \{1, \dots, p\}$ to every train line number $l \in L$,

and a **time margin**, $t_w, 0 \leq t_w \leq T$ describing a degree of fixation for already scheduled train lines.

Corresponding to the given partition function f_{prio} a sequence of PESP graphs $G_1 = (V_1, A_1), G_2 = (V_2, A_2), \dots, G_p = (V_p, A_p)$ is defined. The first PESP graph G_1 contains all vertices belonging to arrival and departure events of the train lines $f_{prio}^{-1}(1)$ which are contained in the first group of the partition function together with the zero time event v_0 as introduced in Chapter 3.2. The edge set of the first PESP graph then is defined as the subset $A_1 \subset A$ containing all edges of A having both end vertices in A_1 . In other words, the PESP graph G_1 is the induced subgraph of G over all events V_1 . Every further PESP graph G_i is defined as the induced subgraph of G over all events V_i , whereby the event set V_i subsequently is enlarged by set of events corresponding to the next group of train lines $V(f_{prio}^{-1}(i))$. Therefore,

$$V_i := \bigcup_{1 \leq j \leq i} V(f_{prio}^{-1}(j)) \cup \{v_0\}, \quad 1 \leq i \leq p, \text{ and}$$

$$G_1 = (V_1, A_1) \subset G_2 = (V_2, A_2) \subset \dots \subset G_p = (V_p, A_p) = G = (V, A).$$

To each PESP graph G_i of the sequence G_1, \dots, G_p a corresponding extended cyclic MILP is defined and optimized, as further explained below. In this sequence of MILPs tension variables corresponding to edges already existing in previous PESP graphs occur in every following MILP again. To reduce the solution space of these variables, fixation constraints $A_{fix}^{(1)}, \dots, A_{fix}^{(p-1)}$ are introduced. They fix all event times of already scheduled

train lines up to the given time margin t_w . Let $(\Pi)^{(i-1)}$ be the set of all scheduled event times π_j solving the last MILP defined out of $G^{(i-1)}$, then $A_{fix}^{(i)}$ is defined as

$$A_{fix}^{(i)} := \bigcup_{j:v_j \in V^{(i-1)}} (a_{fix}^j = (v_0, v_j)),$$

$$\text{with bounds } [l(a_{fix}^j), u(a_{fix}^j)] = [\pi_j^{(i-1)} - \frac{t_w}{2}, \pi_j^{(i-1)} + \frac{t_w}{2}].$$

This set of fixation constraints also reduces the solution space for integer cycle variables as will be discussed in Section 6.3.2.

In terms of a PESP graph partition $G_1 = (V_1, A_1) \subset G_2 = (V_2, A_2) \subset \dots \subset G_p = (V_p, A_p)$ and sets of fixation constraints $(A_{fix})^{(1)}, \dots, (A_{fix})^{(p-1)}$ the decomposition algorithm can be described in four steps as shown in Algorithm 4.

Algorithm 4 *Sequential decomposition heuristic for PESP*

Step 0: Start with $i = 1$;

Step 1: Construct the PESP graph $G_i = (V_i, A_i)$;

Step 2: Add the set of fixation constraints $(A_{fix})^{(i-1)}$ to A_i , if $i > 1$;

Step 3: Define the extended cyclic MILP with a fundamental cycle basis out of a spanning tree containing all fixation constraints $(A_{fix})^{(i-1)}$, if $i > 1$. Solve the corresponding MILP;

Step 4:

if last MILP is feasible **then**

if $i < p$ **then**

 Define $A_{fix}^{(i)}$;

 Return to *Step 1*;

end if

if $i == p$ **then**

print global feasible solution;

end if

else

if $i == 1$ **then**

print "Global problem is infeasible.";

else

if Current fixation margin $< T$ **then**

 Adapt $A_{fix}^{(i-1)}$ by enlarging the fixation margin by two (lower bound -1 , upper bound $+1$);

 Return to *Step 3*

else

print "Global problem is infeasible.";

end if

end if

end if

The algorithm contains two loops. An outer loop is repeating Step 1 to Step 4 for each PESP graph of the sequence G_1, \dots, G_p , as long as we do not discover infeasibility. The second inner loop is a backtracking loop and tries to solve the corresponding MILP by enlarging all fixation constraints by two as long as it cannot find a solution and the fixation margin is still smaller equal the main Period T . While the number of outer loops, which have to be run to find a feasible global solution, is constant and equals p , the number of backtracking loops can vary from zero to $\lceil \frac{T-t_w}{2} \rceil$ for every outer loop.

Figure 6.1 shows a flow chart summarizing the procedure.

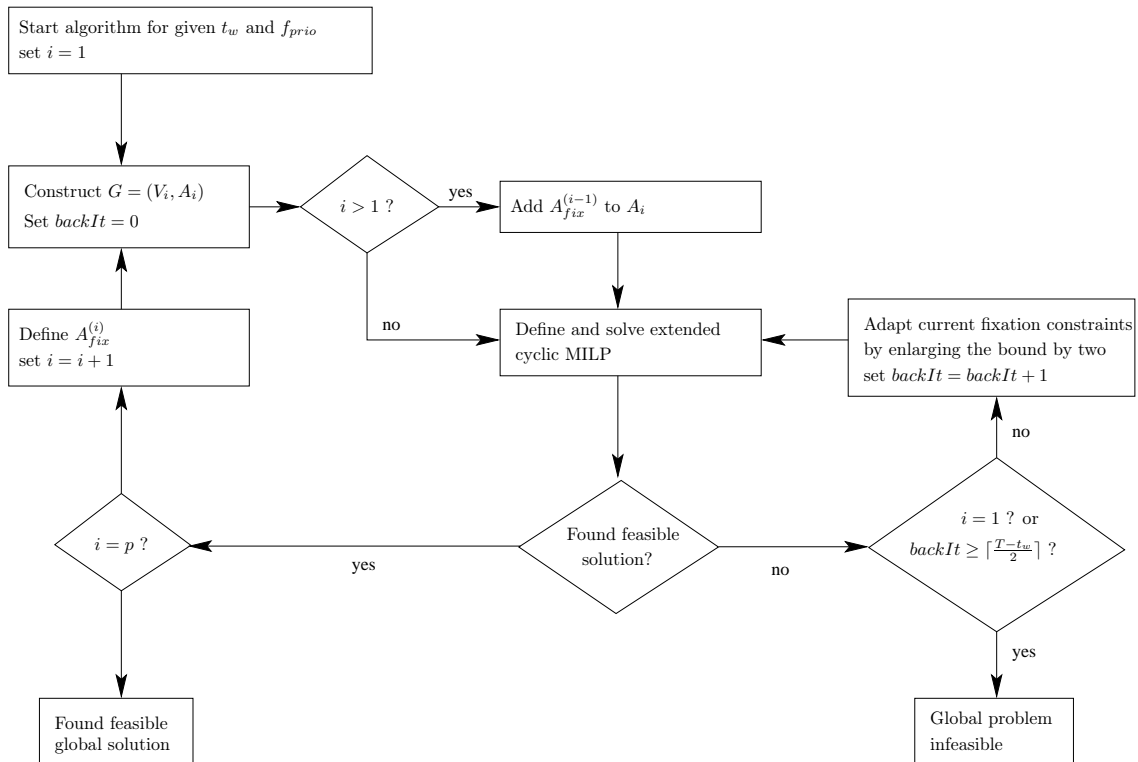


Figure 6.1: Flow chart of the sequential decomposition algorithm

6.3.2 Mathematical properties of the decomposition

In this subsection the mathematical background of the ideas used in the algorithmic approach of Subsection 6.3.1 is discussed.

Correctness of the algorithm

Theorem 6. *Algorithm 4 ends with a feasible global solution, if one exists. If no feasible timetable exists, it finishes its computations after a finite number of iterations and states the infeasibility.*

Proof. To show the correctness of Algorithm 4 we have to ensure that in case of a global feasible situation as well as in a global infeasible situation the algorithm returns a correct result. The problem which is solved in the last iteration has the feasible set of the global

problem as a subset of its feasible set. Therefore, it is not possible that the algorithm could find a feasible solution if the global problem is infeasible. It remains to show that we always find a feasible solution if one exists. For this, we ensure that we never return infeasibility if the global problem is feasible. With the given maximum number of loop iterations, and the consecutive guarantee to end up with one of both outputs, we can ensure to end up with a feasible solution. The algorithm states global infeasibility in two cases:

First case Already the first subproblem is infeasible. Since we do not add any fixation constraint in the first iteration, we found an infeasible subproblem of the global problem. The global problem has to be infeasible as well.

Second case The iterator *backIt* reaches its threshold $\lceil \frac{T-t_w}{2} \rceil$. In this case we tried to solve the problem with a fixation margin of $t_w + 2 \cdot \lceil \frac{T-t_w}{2} \rceil \geq T$. Since every PESP constraint having an interval bound larger than or equal to the main period does not restrict any solution, no fixation constraint influences the solution anymore. The only constraints having an influence on the solution are original constraints of $G = (V, A)$. If we already detect infeasibility of a subset of all constraints, the global problem has to be infeasible as well.

□

Choice of cycle basis In this paragraph we discuss the decision of including all fixation constraints in the spanning tree to construct the fundamental cycle basis for the extended MILP formulation (see Step 3 of Algorithm 4). The main motivation to force this inclusion is to keep the average number of edges per cycle small, which improves solution time of the corresponding cyclic MILP.

Lemma 3. *Let \mathcal{C}_B be a fundamental cycle basis for the PESP graph $G = (V_i, A_i \cup A_{fix}^{(i-1)})$, $i > 1$ in iteration i of the Algorithm 4 constructed out of a spanning tree H containing all fixation constraints $A_{fix}^{(i-1)}$. Then*

- 1) every edge $a \in A_{i-1} \setminus H$ closes a cycle $C \in \mathcal{C}_B$ closed by exactly three edges, and
- 2) the integer variable q_C in the described MILP formulation of Algorithm 4 to such a cycle C is fixed as long as

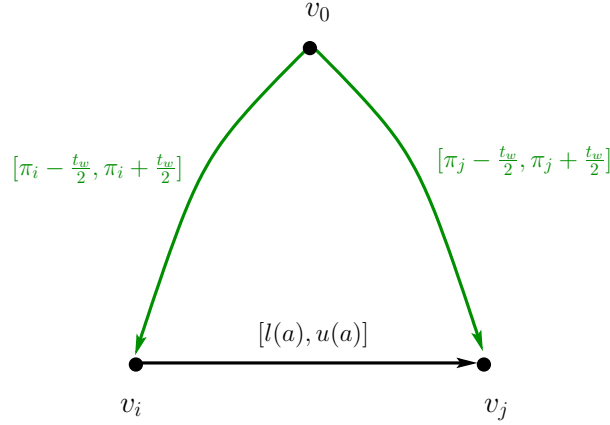
$$u(a) - l(a) < T - t_w.$$

Proof.

The first statement of Lemma 3 directly follows from the fact that both end vertices of an arbitrary edge $a \in A_{i-1}$ are connected to the zero time event v_0 in H . Therefore, the corresponding cycle is composed of these two fixation constraints and the edge a .

For the proof of the second statement we use the cycle cut inequality introduced by [Odijk, 1997].

Let v_i and v_j be the end vertices of edge a , x_a the corresponding tension variable in iteration i and π_i, π_j be the scheduled event time of the previous iteration $i - 1$. Using Odijk's inequality q_C has the following bounds:


 Figure 6.2: Cycle of \mathcal{C}_B in A_{i-1}

$$\begin{aligned}
 a_C &= \left\lceil \left(\left(\pi_i - \frac{t_w}{2} \right) + l(a) - \left(\pi_j + \frac{t_w}{2} \right) \right) \cdot \frac{1}{T} \right\rceil \\
 &= \left\lceil \left(\left(\pi_i - \frac{t_w}{2} \right) + l(a) - \left(\pi_i + x_a + \frac{t_w}{2} \right) \right) \cdot \frac{1}{T} \right\rceil \\
 &= \left\lceil \frac{l(a) - x_a - t_w}{T} \right\rceil \leq q_C,
 \end{aligned}$$

$$\begin{aligned}
 b_C &= \left\lfloor \left(\left(\pi_i + \frac{t_w}{2} \right) + u(a) - \left(\pi_j - \frac{t_w}{2} \right) \right) \cdot \frac{1}{T} \right\rfloor \\
 &= \left\lfloor \left(\left(\pi_i + \frac{t_w}{2} \right) + u(a) - \left(\pi_i + x_a - \frac{t_w}{2} \right) \right) \cdot \frac{1}{T} \right\rfloor \\
 &= \left\lfloor \frac{t_w - x_a + u(a)}{T} \right\rfloor \geq q_C.
 \end{aligned}$$

The bound a_C is 0, as long as $t_w - l(a) + x_a < T$ and b_C is 0 if $t_w - x_a + u(a) < T$. From $l(a) \leq x_a \leq u(a)$ and the given condition of the second statement we can deduce that $0 = a_C \leq q_C \leq b_C = 0$. \square

Remark 3.

- Usually most trip, dwell, separation and frequency constraints have a time interval of less than 10 min. Therefore there are several fixed integer cycle variables belonging to cycles closed over constraints of these types out of A_{i-1} , as long as $t_w \leq T - 30$.
- Using a preprocessing step reducing $u(a) - l(a)$ to $2 \cdot t_w$, which we assume is done by CPLEX, even all cycles closed by constraints of A_{i-1} are fixed for $t_w < \frac{T}{3}$

Decomposition of the PESP graph

To discuss the applied decomposition of the PESP graph we introduce two types of constraints:

Definition 13. Let $G = (V, A)$ be a PESP graph, $f_{prio} : L \rightarrow \{1, \dots, p\}$ be a partition function of its train lines and $V(f_{prio}^{-1}(i)), 1 \leq i \leq p$ all vertices of $G = (V, A)$ corresponding to the set of train lines added in the i^{th} iteration.

We denote the set of edges $A(f_{prio}^{-1}(i)) \subset A, 1 \leq i \leq p$ connecting vertices of the set $V(f_{prio}^{-1}(i)) \cup \{v_0\}$ by *inner constraints*, and

all remaining edges having their end points in two vertex sets corresponding to different groups of train lines are referred to as *binding constraints*

As the name already tells, binding constraints propagate restrictions through already scheduled trains to the next group of train lines we want to schedule. Since groups are scheduled sequentially and without knowledge about further groups, it could be advantageous to find a feasible solution for the next iteration if we had as few binding constraints as possible and if we had binding constraints with larger interval bounds. This leads us to the idea of decomposing a PESP graph over a group partition of train lines running close to each other from a geographical point of view. Such a decomposition would also be convenient from a practical side, as well as economical seen from implementation complexity. In Section 6.4 we discuss and compare two different approaches for the group partition. Considering PESP constraints and their typical size of intervals in practice, we can split them into three groups summarized in Table 6.1.

	Constraint Type	Interval Size
small	trip, dwell, frequency, slot, overall	less than 10 min
medium	connection, separation	between 10 and $T - 10$ min
large	headway	at least $T - 10$ min

Table 6.1: Typical interval sizes of PESP constraints in practice

With the given PESP graph decomposition used in Algorithm 4 all intervals of binding constraints are of medium and large size. With the idea of collecting train lines running on similar sections we especially reduce the number of separation constraints, but also some connection constraints in the set of binding constraints. Furthermore, we can enlarge the number of headway constraints belonging to inner constraints choosing an adequate geographical partition of train lines. All constraints having a typical interval size of less than 10 minutes are inner constraints belonging to blocks considered in one iteration and bounded by t_w in later iterations.

Studying the MILP we solve in iteration i of Algorithm 4 we can show a block structure, as indicated in Figure 6.3, building the following collection of tension and cycle variables in vectors:

- $\overrightarrow{x_{bind,i}}$: tension variables of binding constraints of iteration i
 $\overrightarrow{x_{fix,i}}$: tension variables of fixation constraints of iteration i
 $\overrightarrow{x_{inn,i}}$: tension variables of inner constraints of iteration i
 $\overrightarrow{q_{sched,i}}$: cycle variables corresponding to cycles closed by edges $a \in A_{i-1}$
 $\overrightarrow{q_{new,i}}$: remaining cycle variables

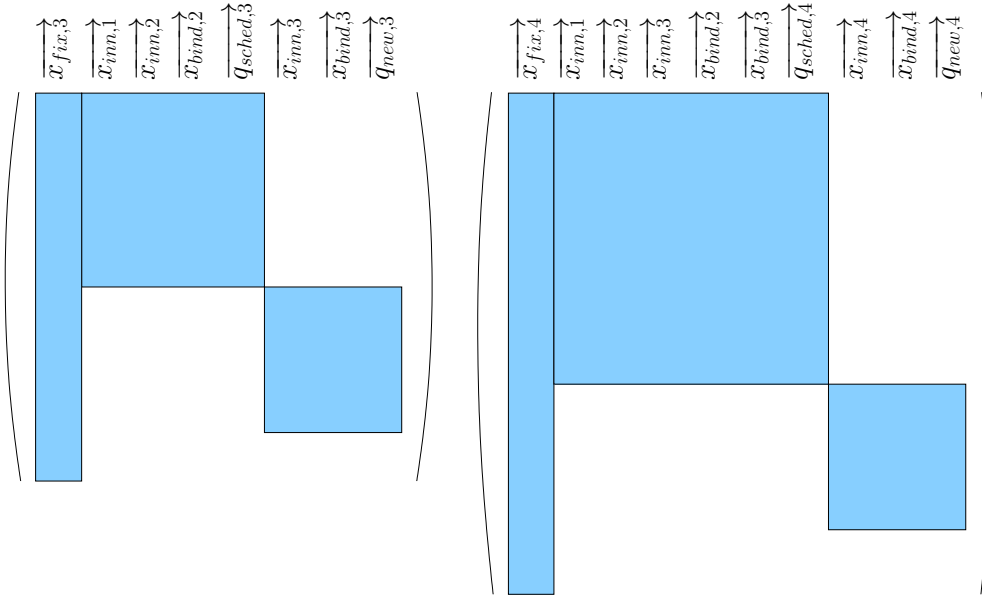


Figure 6.3: Blockstructure to the sequential decomposition heuristic for iteration 3 and 4

6.4 Implementation and computational results

In this section, we evaluate the sequential decomposition heuristic introduced in the last section over the three largest PESP models further described in Section 3.4. In a first subsection, we define two different train line partitions. A first one is motivated from planning practice, whereby the second one is more or less randomized, serving as a control group to test whether the first partition brings advantages for the solution method. For both group partitions the number of inner and binding constraints is discussed and an adequate number of groups used for both partitions and every model is fixed. A second subsection introduces two small variations of the fixation constraints and tests them for the three models. The third subsection discusses different parameter settings for all three models and their influence on computation time and timetable quality. It gives an overview of the distribution of the overall computation time over different iterations and it also shows the average number of back iterations needed for all computations. Subsequently, the fourth subsection presents the experience with infeasible

PESP models concerning the sequential decomposition. And a last subsection summarizes the results and compares it to the global solution methods without decomposition.

6.4.1 Choice of train line partitions

In this section we introduce two different methods to define train line partitions. A first method, called *geographical partition* (Geo) is motivated from planning practice. In a first step all fast trains are scheduled and then in all subsequent steps, groups of regional trains, partitioned according to their geographical location, are added to the schedule. A more detailed definition is given in the next paragraph. The geographical group partition is compared to a second partition method, called *alternating partition* (Alt), which should serve as a control group to test whether a procedure adapted from planning practice could bring benefits with the application to our sequential decomposition approach. In the alternating group partition trains are mixed according to their geographical location and no separation between fast and regional trains is added. Further descriptions are given in the second paragraph of this section, followed by observations of the number of inner and binding constraints for both group partitions as well as a discussion of an adequate number of groups for the partition.

Geographical partition (Geo)

The set of all fast trains, as defined in Table 2.1, build the first group of the geographical partition. This separation according to the train type is on the one hand side motivated as an adaption of hierarchical procedures of the manual timetabling. But on the other hand side it could be advantageous to include them in a first step of the decomposition because of their longer length of the itineraries. Adding long distance trains at the end of the sequential method is assumed to be more difficult than a shorter regional train.

For the partition of all regional train lines into $p-1$ groups, according to their geographical location, an automated heuristic is used. The problem of partitioning a given set of train lines into $p-1$ groups of equal size, so that all train lines of the same groups are closest to each other, measured by a certain distance function, can be formulated as a $(p-1)$ -means clustering problem [Vattani, 2011]. Our used distance function is called *similarity table* and is defined in Definition 14.

Definition 14. Let L be the set of train lines and $l_i, l_j \in L$ two arbitrary train lines. Then the *similarity table* $S \in [0, 1]^{|L| \times |L|}$ is defined as

$$S(l_i, l_j) = \frac{\text{number of common operation points on the itineraries of } l_i \text{ and } l_j}{\text{number of operation points on the itinerary of } l_j}$$

Since solving a k -means clustering problem is an NP -hard optimization problem [Vattani, 2011], we use a simple heuristic method to define an adequate partition. In a first step $p-1$ trains, called *starting trains*, sharing as few common operation points on their itinerary as possible, are separated. The first train of the list of train lines is chosen as a first starting train. Then for each train the sum of similarity to all trains already

included in the list of starting trains is considered and a train having the smallest value for this sum is added to the starting trains, as long as the group of starting trains contains less than $p - 1$ trains.

In a second step all remaining regional trains are distributed to the starting trains, such that all groups have approximately the same size (measured by the total number of corresponding PESP events) and trains sharing a common group do have a sum of similarity values as large as possible. In each step the smallest group is determined and a train with the largest sum of similarity to all already contained trains of this group is added to the group, until no regional train is left.

In order to influence the geographical group construction manually the opportunity to define all starting trains and their sequence manually is added. This allows a better distribution of the starting trains to different regions of the network and also enables us to determine a region where the algorithm has to start with the timetable construction. If a train partition with this additional option of fixing the starting trains manually is used in later parts of this chapter, we will refer to the partition method by the *manual geographical partition* (GeoMan).

Alternating partition (Alt)

In contrast to the geographical group partition the alternating partition simply distributes all train lines to p different groups of approximately the same size. As in the first partition the size of groups is given by the total number of PESP events. And in each step a next train line of the list of all train lines is added to the currently smallest group. The ordering of the list of trains is given by the sequence in which trains are collected out of data for the PESP model. Trains therefore are mixed according to their geographical location and over all train types.

Inner constraints and binding constraints

In this paragraph, the number of inner and binding constraints for both partition methods and different group sizes is tested for different problem instances. Table 6.2 shows the percentage of binding constraints in each case.

In all examples, there are more binding constraints in the case of the alternating group construction. Since we grouped regional trains using similar infrastructures in the geographical group partition, there should be more headway constraints counted to the inner constraints for this partition. In the case of the EasternCH model, the smallest difference between the two partitions can be observed. The reason for this phenomenon could be the strong interlinking of quite every train line around Zurich main station. To prevent disadvantageous partitions in a second attempt we use the manual definition of the geographical group partition to fix a set of starting trains and the sequence of the groups. This manual definition is chosen so that the algorithm starts to construct the timetable first around Zurich main station with all longer regional trains offering a cross-city link and ends in the peripheral area of eastern Switzerland. The corresponding percentages

Model	p	Geo (GeoMan)	Alt
Thun-Basel	5	35.97%	55.63%
	6	35.17%	59.43%
	7	37.73%	60.31%
	8	37.46%	59.78%
Eastern CH	7	50.10% (38.48%)	59.37%
	8	59.87% (45.07%)	60.82%
	9	51.34% (45.21%)	61.88%
German-speaking CH	13	40.54%	63.30%

Table 6.2: Percentage of binding constraints for different group partitions

of binding constraints of this manual geographical group partition (GeoMan) is given in brackets in Table 6.2 and show a clear reduction of binding constraints.

Choosing adequate number of groups

After the definition of two group partition methods, we want to discuss the influence of the number of groups on the overall performance and solution quality of the sequential decomposition approach. Figure 6.4 visualizes computational results for the example of the geographical and alternating group partition, using a starting fixation $t_w = 12$ minutes applied to the Thun-Basel model and represents our general observations we made corresponding to variations of the number of groups.

Using a geographical group partition with two groups, which means that all fast trains are scheduled in a first step followed by all regional trains together in a second step, the computational performance of the decomposition is not better than the global solution using the cyclic MILP formulation to solve the same problem in one step. To find a first feasible solution it takes more than 5 hours computation time and the optimization afterwards sticks, so that the second iteration of the decomposition breaks with the time limit of 24 hours without reaching the desired MipGap of 0.5% in the last iteration. In case of the alternating group partition, which distributes all fast and regional trains mixed in two groups of equal size, a first feasible solution is found after 30 minutes. The optimization of this solution down to a MipGap of 0.5% takes another 5 hours of computation time. Using three and four groups for the geographical group partition already brings a considerable improvement in computational performance. Both computations can be finished with a total computation time of less than 20 minutes and with a MipGap which is lower than the one reached with the global solution method using the cyclic MILP formulation. For the alternating group partition the computation time decreases as well using 3 and 4 groups. But in case of 3 groups it still exceeds the limit of 20 minutes with a total computation time of 28 minutes. Enlarging the number of groups further on for both group partitions improves the overall computation time to values of about 5 minutes and less. Up to a number of 9 groups the overall computation time of the geographical group partition seems to augment slightly again, but having a look at larger numbers of groups up to 20 one can observe that the overall computation times fluctuate in an interval of about 2 to 6 minutes. Also observing the reached MipGap for

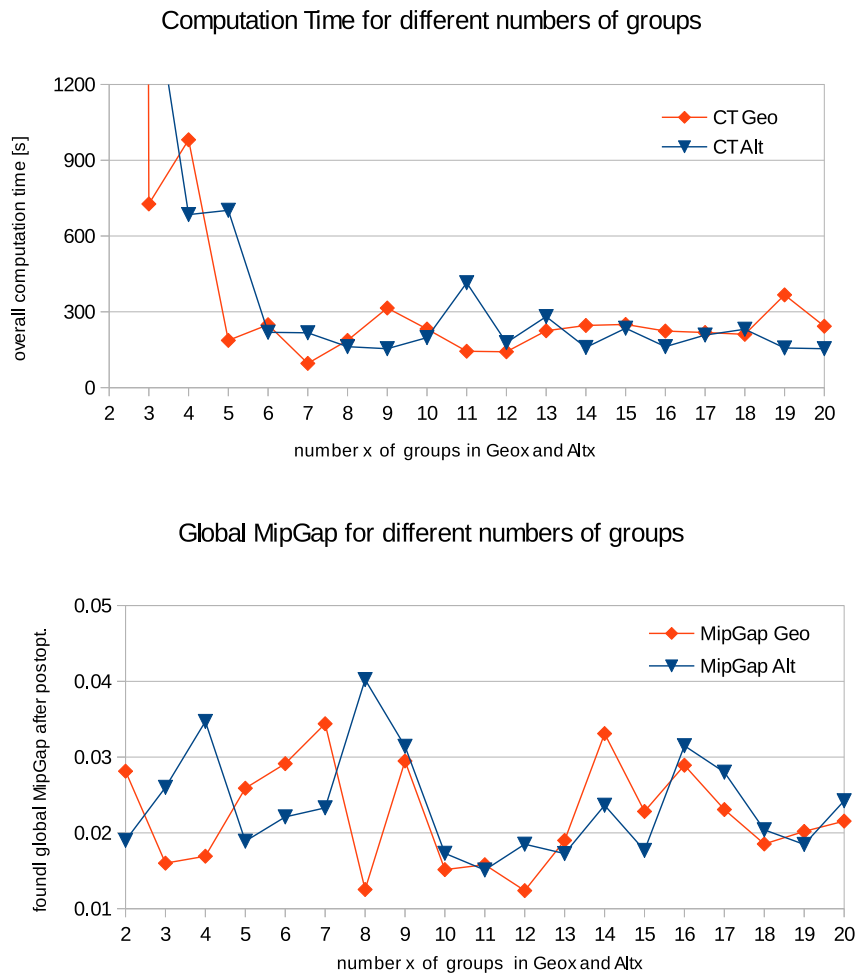


Figure 6.4: *Overview of the computational results to the variation of group numbers for the sequential decomposition heuristic at the example of the geographical and alternating group partition applied to the Thun-Basel model with a starting time fixation margin of $t_w = 12$.*

all computations no clear trend corresponding to the number of groups used for the group partition can be observed. Thus the only conclusion we want to fix at this place is that starting with a maximal average group size of about 450 events for the regional trains, the computational performance of the sequential decomposition for all our models is by trend better than using larger groups.

This early observations motivate us to fix the number of groups used for the Thun-Basel model to at least 5, for the EasternCH model to at least 7 and for the GermanSpeakingCH to 13. To keep the number of computations for subsequent evaluations in a manageable framework, we restrict the number of groups to a variation between this lower bound and three subsequent larger numbers for the Thun-Basel model and the EasternCH model. For the GermanSpeakingCH we even fix the number of groups to 13 and do not compare further number of groups.

6.4.2 Variation of fixation constraints

Besides the choice of a partition number p and an adequate train line partition function f_{prio} also the time margin t_w to fix already scheduled trains plays an important role in our sequential decomposition approach. The influence of the choice of a certain margin t_w to start Algorithm 4 on the overall computation time and timetable quality is discussed in Subsection 6.4.3. In this section we introduce two slightly adapted fixation methods for the algorithm, motivated from manual timetabling, to compare it to the original fixation.

These adapted fixation methods are both based on the idea that already scheduled trains, which run in a different geographical region than the new trains added in the next iteration, do not need the same amount of flexibility as other trains running close to the new trains. Therefore, the size of the fixation interval t_w^{Sim} for trains running far away from the new group of trains can be reduced up to half of the original interval size t_w . The measurement on how close two trains run to each other is defined once more over the similarity table already used in Section 6.4.1. For each already scheduled train line l_1 a train line l_2 of the new group sharing the most common operational points on its itinerary is determined. The value $S(l_1, l_2)$ of the similarity table to this train is then used to define the fixation margin t_w^{Sim} . As soon as $S(l_1, l_2)$ is larger than 0.5 the original interval size t_w is used. If there is no common operation point ($S(l_1, l_2) = 0$), t_w^{Sim} is reduced to $\frac{t_w}{2}$. In the case of a similarity value between 0 and 0.5, a linear proportional value between t_w and $\frac{t_w}{2}$ is used. Thus

$$t_w^{Sim} = \begin{cases} t_w & \text{if } S(l_1, l_2) \geq 0.5 \\ t_w \cdot (0.5 + S(l_1, l_2)) & \text{if } S(l_1, l_2) < 0.5 \end{cases}.$$

If back iterations are necessary in Step 4 of Algorithm 4 all interval sizes are enlarged by $\frac{t_w^{Sim}}{t_w} \cdot 2$ in case of the first method, called *sequential decomposition with fixation over similarity* (Sim). In the case of the second method, the similarity is only considered to define the first fixation margin. If back iterations are necessary, all margins are enlarged equally by two (lower bound minus one, upper bound plus one). Therefore, the second method is called *sequential decomposition with starting fixation over similarity* (SimS).

The two fixation heuristics Sim and SimS are tested for different sequential decompositions and compared to the original fixation defined in Algorithm 4. The results for different models and group partitions are similar. Figure 6.5 gives an overview of the results of the geographical decomposition of the Thun-Basel instance over a partition in 8 groups. For each fixation strategy, we test different values of the starting fixation t_w . We furthermore use a MipGap of 3% for all iterations except the last one (*inner iterations*), for which (*last iteration*) we decrease the MipGap to 0.5%. For the original strategy (Geo8) the total computation time starts to exceed a threshold of 10 hours for a starting t_w of 34 minutes. The first graphic shows clear advantages of the overall computation time in case of the two fixation heuristics Sim and SimS for a larger starting t_w ($t_w \geq 14$). As soon as a solution is found over a decomposition strategy, this solution is used as a starting solution for the global cyclic MILP. And a postoptimization of 20 seconds over the global MILP is used to further optimize the result and to get a feedback on the global

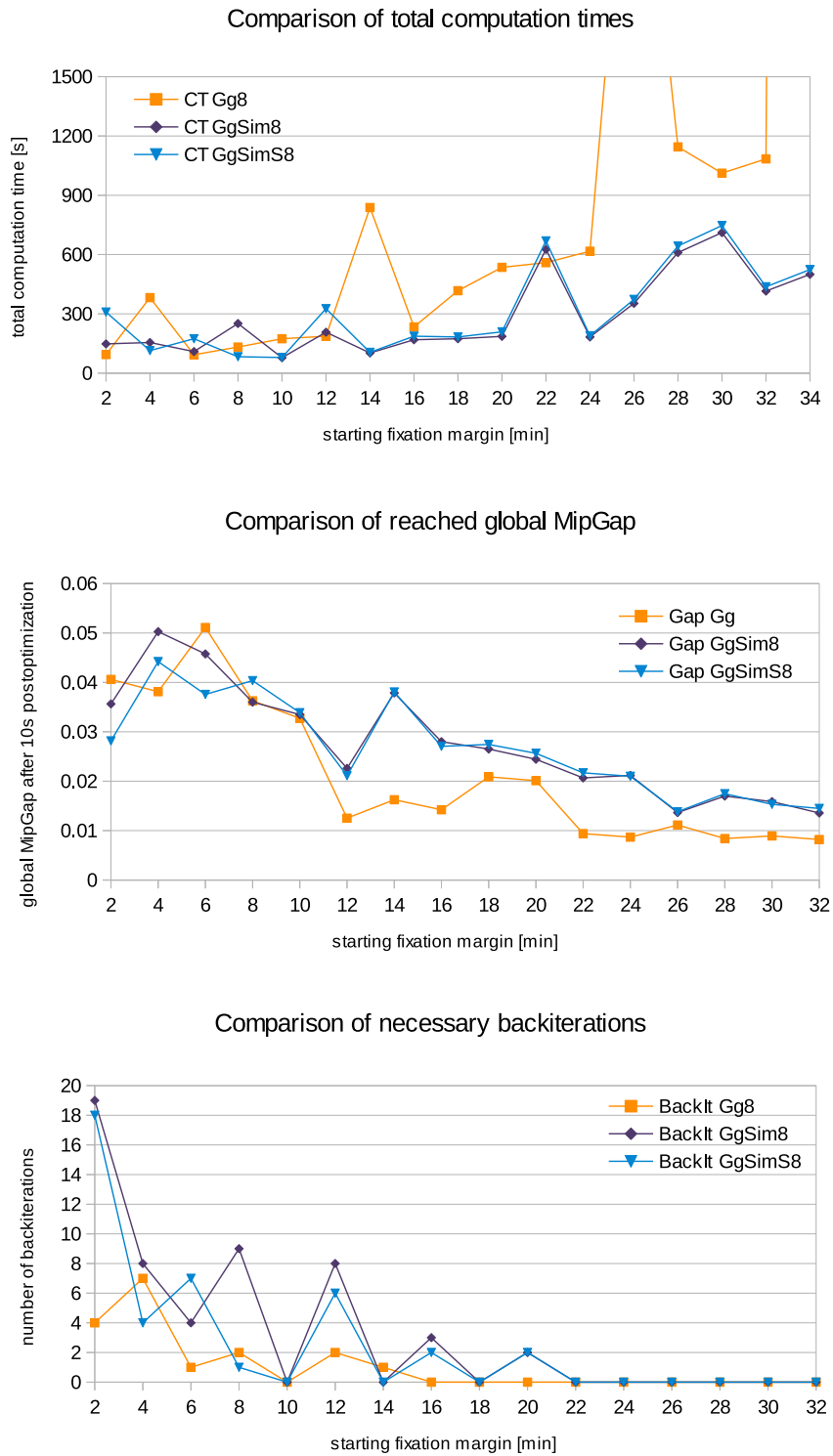


Figure 6.5: Comparison of the fixation heuristics between Sim, SimS and the original fixation defined in Algorithm 4 for the example of the geographical decomposition of the Basel-Thun instance with 8 groups.

MipGap of the found solution. In the second diagram, the found global MipGap of every computation is given. It shows that the original fixation heuristic leads to better results starting from a fixation $t_w \geq 12$. Also the necessary number of back iterations in the last diagram shows benefits for the original fixation heuristic. Comparing Sim and SimS, the differences in all three graphics are only slight. The most remarkable difference can be found in the number of necessary back iterations. The fixation heuristic Sim tends to need more back iterations.

Summarizing the observations to all test instances, the original fixation strategy as defined in Algorithm 4 should be prioritized, as long as the total computation time is acceptable. In the case of very large models, the application of the fixation heuristic Sim or SimS could come into consideration further discussed at the end of the next section. But the risk of loss in timetable quality has to be kept in mind.

6.4.3 Influence of the fixation margin on computation time and quality

For our computational studies we run several group partitions, use different methods to define fixation constraints and test various values for starting fixation margins t_w . Table 6.3 gives an overview of all computations we run for our three largest models (Thun-Basel, EasternCH and German-SpeakingCH). For the smaller two instances we tested all sizes of t_w with a distance of 2 minutes starting from $t_w = 0$ up to a size of t_w where a time limit of total 24 hours of computation time is reached. This is often the case starting from $t_w = 26$ to $t_w = 46$, depending on the group partition and the choice of fixation. For the largest model, we test some specific values of t_w ($t_w = 4, 8, 12$ and 16). The group partitions ending with *In2* are explained later in this section. The last column of Table 6.3 shows the used MipGap tolerances (inner iterations, last iteration) for each model.

Model	f_{prio} with Fixation Strategy	p	MipGap
Thun-Basel	Geo, GeoSim, GeoSimS	5, 6, 7, 8	(3,0.5)
	GeoIn2		
	Alt, AltSim, AltSimS		
	AltIn2		
EasternCH	Geo	8, 9, 10	(3,1)
	Alt		
	GeoSimS	7, 8, 9	(3,1)
	GeoMan, GeoManSimS AltSimS		
German-SpeakingCH	Geo	13	(4,3)
	Alt		

Table 6.3: Overview on parameter settings for the sequential decomposition

As it can be observed already in Figure 6.5 computation times and qualities of timetables fluctuate over different starting fixations t_w . But the results show also some trends which can be observed with the enlargement of t_w . Thus for example the number of back iterations clearly decreases with larger sizes of t_w . Furthermore, timetable quality, as well as computation time, increase. For small values of t_w a MipGap of about 4% can be observed, whereby it comes down to 1% for larger values of t_w . For values of $t_w \leq 12$ minutes all computation times are less than 5 minutes. Then computation times grow and twice exceed the upper limit of 20 minutes for $t_w = 26$ and $t_w = 34$. But starting with $t_w = 12$ there are several computations with good timetable qualities (MipGap around 1%) and still short computation times.

Computations of the Thun-Basel model

The same trends can be observed in all our computations. Figure 6.6 shows average values for the geographical and alternating group partitions (Geo, GeoSim, GeoSimS and Alt, AltSim, AltSimS) applied to the Thun-Basel instance. Besides an affirmation of all trends mentioned in the last paragraph, the diagrams further show interesting differences between the geographical and the alternating group partition. As it makes sense out of the definition of the two group partitions, using the alternating group partition we need more back iterations than in the case of the geographical group partition. Also computation times slightly seem to be worse for the alternating group partition. In case of the MipGap the alternating group partition shows better results up to a starting fixation of $t_w = 12$. This could be the case because of the back iterations. For the alternating group partition and small fixation degrees more of the double of the amount of back iterations are necessary than for the geographical partition and the same value of t_w . This leads to considerably larger fixation intervals during all computations and optimizations allowing more flexibility for timetable construction and therefore better solutions. Starting from $t_w = 12$ the geographical partition slightly reaches better objective values except for $t_w = 28$.

Distribution of computation time over different iterations

An interesting observation out of Figure 6.6 is also the fact that the number of back iterations does not seem to enlarge computation times considerably. So for example the alternating group partition with starting $t_w = 4$ needs in average 23 back iterations, but only has an average computation time of about 5 minutes, comparable to the geographical group partition, which uses 4 back iterations in average. A deeper insight into the distribution of total computation time on different iterations is given by Figure 6.7 for the example of the alternating group partition *Alt8* applied to the Thun-Basel model. It shows the amount of time used for all back iterations, the expenditure of time for the last iteration distributed in three parts (time until first feasible solution is found, time to optimize this solution up to a MipGap of 1% and time to further reduce the MipGap from 1% to 0.5%), and the computation time for all remaining iterations. Figure 6.7 clearly confirms the observation on back iterations. In all cases, they use a small amount of

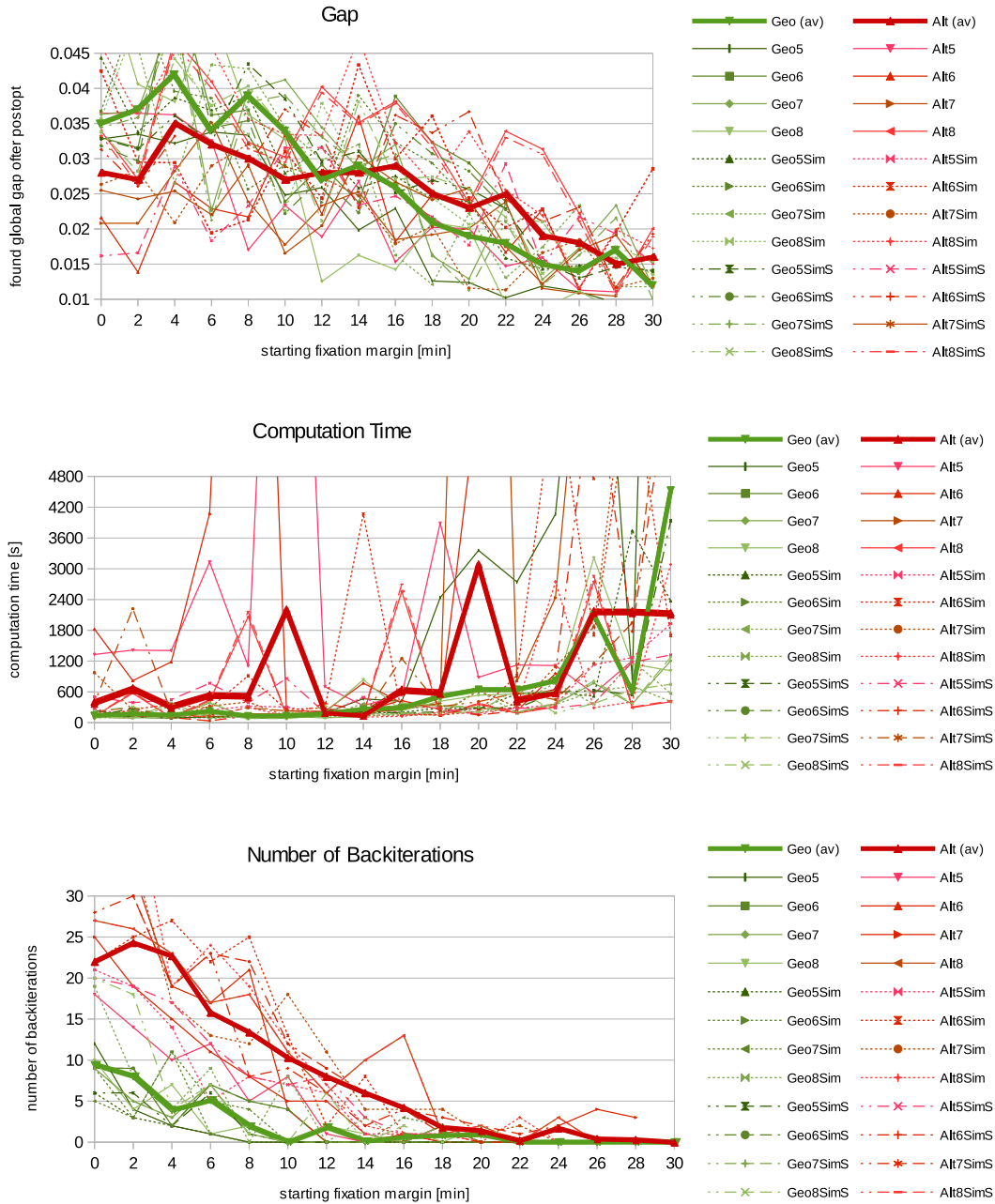


Figure 6.6: Values on the total computation time, reached global MipGap and the necessary number of back iterations to all computations (Geo, GeoSim, GeoSimS and Alt, AltSim, AltSimS) together with their average for the Thun-Basel model.

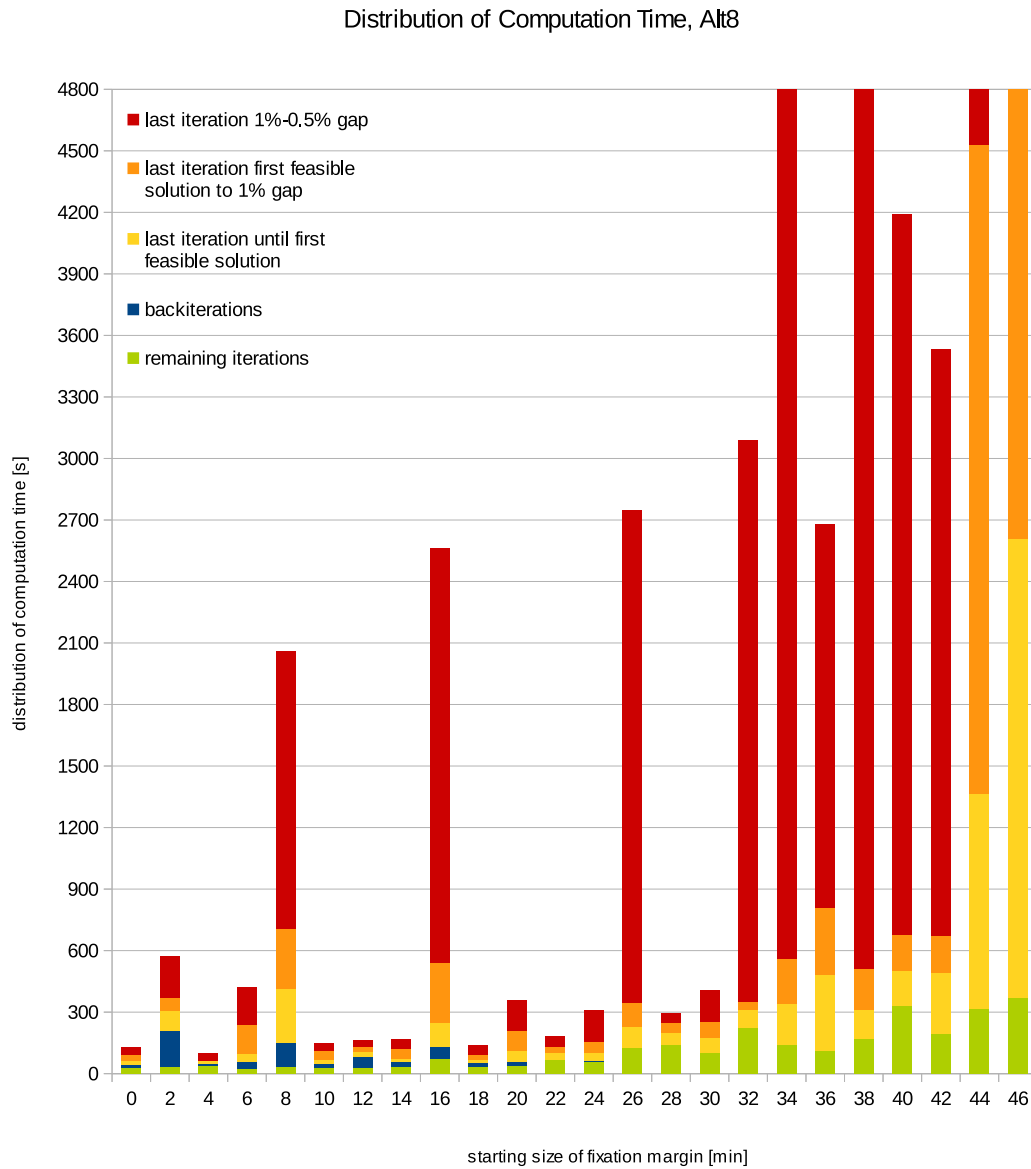


Figure 6.7: Typical distribution of computation time over different iterations of the sequential decomposition approach for the Thun-Basel model using an alternating group partition with 8 groups and different values for the starting fixation margin t_w

time compared to the other iterations. In more than 96% of all computations applied to the Thun-Basel model, the average computation time for back iterations is less than 10 seconds.

Figure 6.7 further shows that the most time consuming iteration for every computation is the last iteration. And it seems also to be the most critical iteration related to the fluctuations in computation time. Often the reduction of the relative MipGap from 1% to 0.5% in the last iteration takes a huge amount of time and does not lead to a considerable improvement of the objective values, which means that CPLEX mainly enlarges the lower bound for the objective values and cannot find better integer solutions. Thus the MipGap tolerances have to be chosen carefully. They improve the average quality of timetables, but they can lead to large fluctuations in computation time. As it can be seen out of table 6.3, the MipGap tolerances are adapted for the larger two problem instances (EasternCH and German-SpeakingCH).

A further possibility to reduce high fluctuations in computation time is the introduction of a breaking condition, for example if the number of back iterations in the last iteration exceeds a certain limit. In the case of EasternCH we use a limit of 10 back iterations. This limit is reached in 5 computations (GeoSimS7 with $t_w = 2, 14$, GeoSimS9 with $t_w = 6$, AltSimS7 with $t_w = 0$, and Alt10 with $t_w = 4$). Especially in the last iteration, when all trains are included in the scheduling problem, back iterations and their connected increase of the fixation interval are very time consuming. Thus from a computational point of view, a reordering of the groups in the sequential decomposition method could be advantageous if a lot of back iterations appear in the last iteration. This idea is further discussed in connection to the manual geographical group partition for the EasternCH model (GeoMan) later on in this section.

Group Fusions

Concerning the fact that for all computations the last iteration takes the largest amount of computation time, the idea of merging groups with short computation times appears obvious. Figure 6.8 shows results of the extreme case of a fusion of all groups except of the last for the geographical group partition in the case of the Thun-Basel model. The corresponding group partitions are indicated with the end *In2*. The results show a much more stable performance compared to the original defined partition over different values of the starting fixation t_w . A reduction of the number of iterations reduces the risk of ending in different local minima for each value of starting t_w of the sequential approach. Using only two iterations together with the defined back iterations leads to repetitions of the same computations over several choices of t_w . As long as there are back iterations for Geo5In2, Geo6In2 and Geo7In2 the reached MipGap and used computation times are constant. Enlarging the size of t_w in each step reduces the necessary number of back iterations by one. The group partition Geo6In2 shows a much better performance than the original partition Geo6. With a MipGap of about 1% it finds a timetable of a quality which is never reached using the partition Geo6 for all computations from $t_w = 0$ to $t_w = 20$. And with a computation time of about 10 minutes, the computational

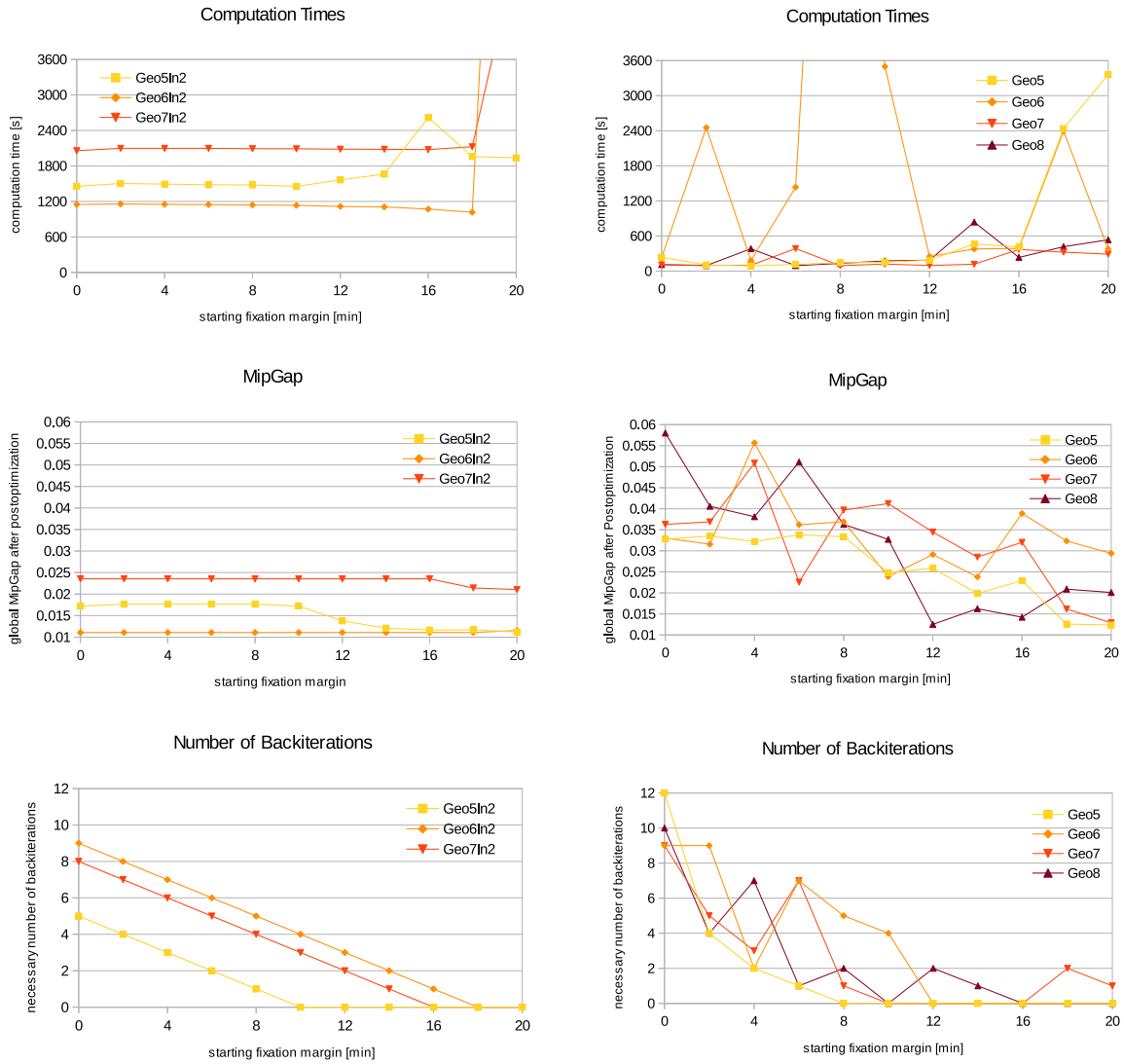


Figure 6.8: Results of the computational performance for an adapted group partition Geo5In2, Geo6In2 and Geo7In2 using a fusion of all first $n - 1$ groups of the original geographical group partition in n groups.

performance is also better than the average computation time for Geo6, which fluctuates considerably. The partition Geo5In2 reaches a MipGap close to 1% starting from $t_w = 14$. Enlarging the size of the first group with Geo7In2 and Geo8In2 increases computation time and the MipGap. In case of Geo8In2, the computation for $t_w = 0$, including 3 back iterations even takes more than 9 hours, and is therefore left out in Figure 6.8. Both Geo7In2 and Geo8In2 lead to a MipGap of more than 2% for all tested computations.

Summarizing our observations of the group fusions they can improve optimization as well as computational performance as long as the group sizes do not exceed a certain limit. A group partition with decreasing sizes of groups over all iterations could be interesting in general. Starting with an originally defined group partition using equal sizes of groups, one could start to combine groups. In a first step enlarging the first group as long as the computation time of the first iteration stays acceptable from a practical point of view. Then also the remaining groups could be further combined.

Computations to the EasternCH model

Repeating our computations to the original defined geographical and alternating group partitions for the EasternCH model we enlarge the relative MipGap tolerance of the last iteration from 0.5% to 1% and introduce a breaking condition in the last iteration if there are more than 10 back iterations. Both ideas are already mentioned and discussed earlier in this section. Figure 6.9 shows results of the average values of the computation time, the reached global MipGap and the number of back iterations are similar to the one of Figure 6.6 for the Thun-Basel model.

As in the case of the computations of the Thun-Basel model, the number of back iterations for the geographical group partition is smaller than for the alternating group partition. But observing the results of the average computation times and the reached global MipGap the geographical group partition does not show the same advantages as in the case of the Thun-Basel model. Also looking back to Table 6.2 the number of binding constraints is not reduced by the same amount for the geographical group partition as in the Thun-Basel model. Studying the geographical group partition for EasternCH and the corresponding planning situation in detail, an inappropriate constellation can be found: There are long regional train lines passing critical dense infrastructure elements around Zurich main station which are planned in the last step of the decomposition. In planning practice, one would start timetable construction in this critical region and the time slots of the corresponding regional trains would be fixed in an early step of this construction, influencing the timing of peripheral regional trains. To adapt this idea from planning practice in a better way we use the idea of fixing starting trains for the geographical group partition, mentioned in Section 6.4.1. The corresponding group partition we call the manual geographical group partition *GeoMan*.

The results in Figure 6.9 show that with this idea the necessary amount of back iterations, especially in the last iteration, can be reduced considerably. Having less back iterations

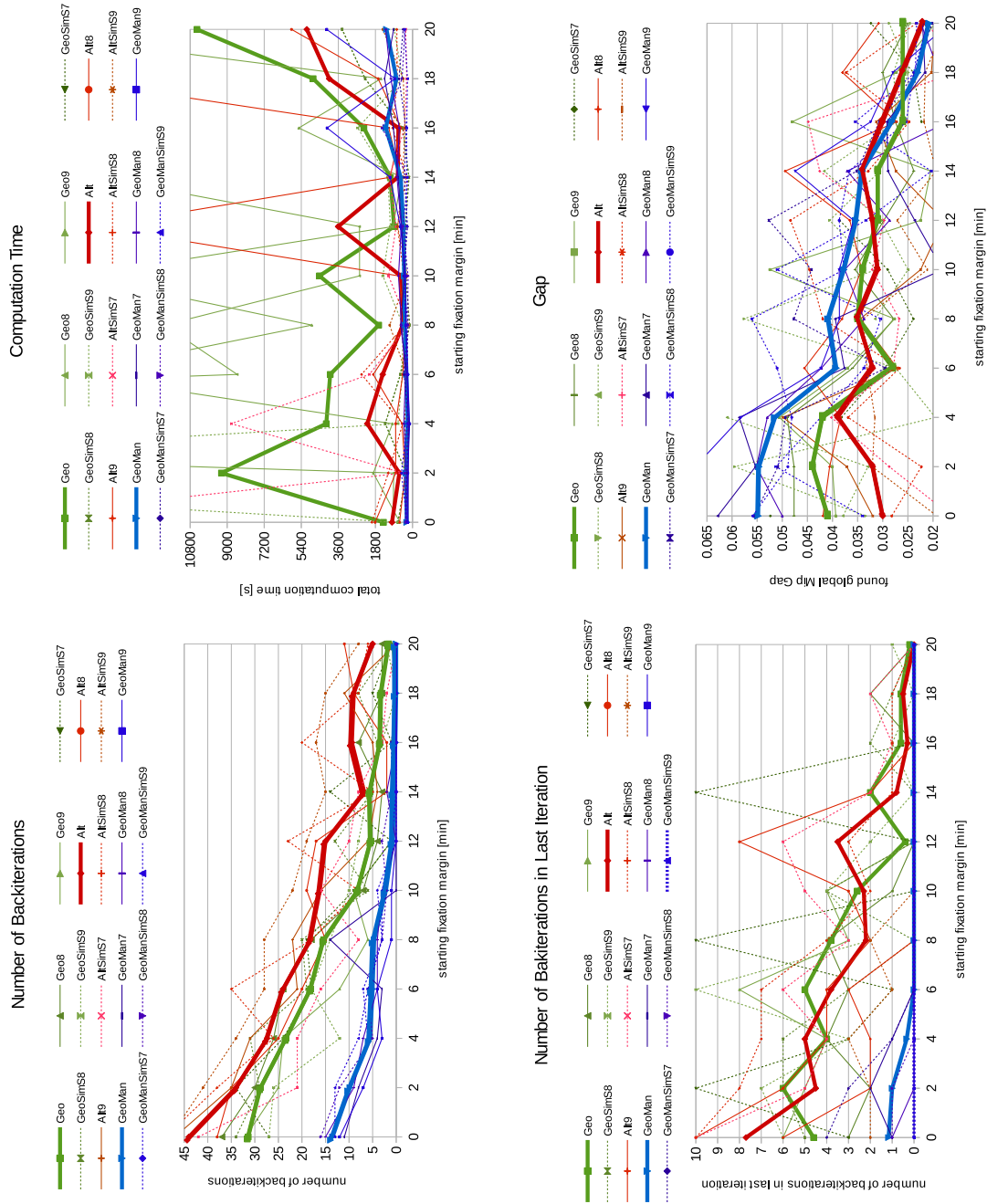


Figure 6.9: Values on the total computation time, reached global MipGap and the necessary number of back iterations to all computations (Geo(Man), Geo(Man)Sim, Geo(Man)SimS and Alt, AltSim, Alt-SimS) together with their average for the EasternCH model.

in the last iteration of the decomposition also reduces the total computation time. Having a look at the reached global MipGap, the new group partition GeoMan constructs worse timetables for small starting fixations t_w . This also has a connection to the smaller number of back iterations and the therefore stricter sequential time fixation over the iterations. But starting with a fixation around $t_w = 10$, the MipGap of the group partition GeoMan starts to become comparable to the other two group partitions and for larger fixations it even seems to get slightly better. Therefore, once more the adaption of ideas from planning practice brought interesting improvements to our algorithms.

Computations of the GermanSpeakingCH model

Our observations of the computational performance for the Thun-Basel and the EasternCH model are used to fix parameters for our largest model German-SpeakingCH. Figure 6.10 shows the results.

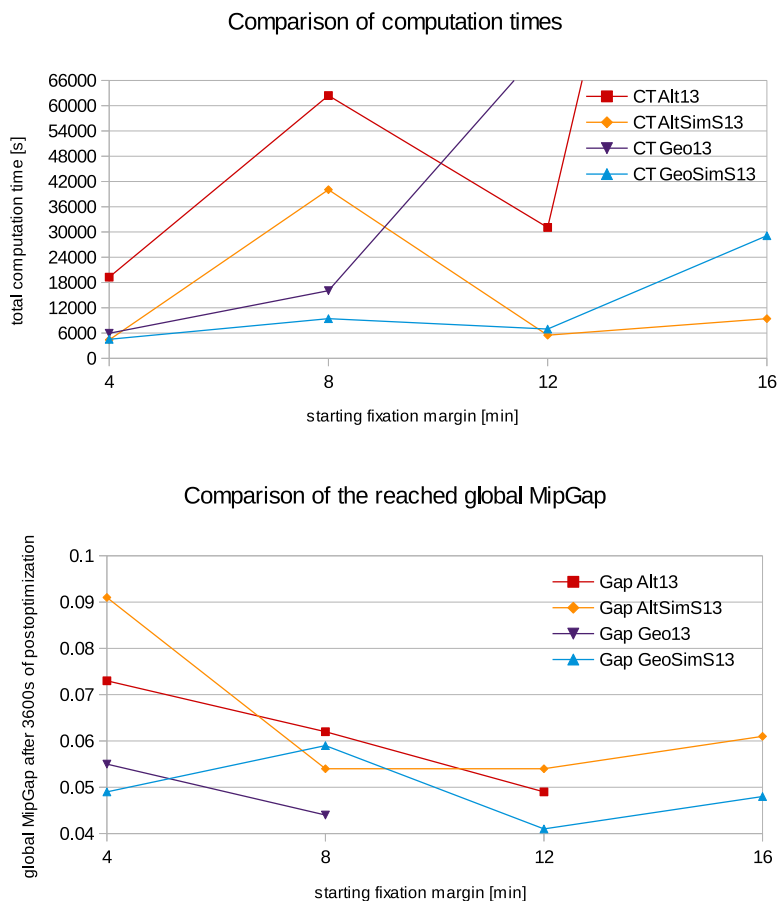


Figure 6.10: Overview on the results of the sequential decomposition for the largest test instance German-SpeakingCH with different parameter settings

For all computations a time limit of 24 hours is used. In case of the alternating group partition this time limit is reached in the last iteration without finding a feasible solution for starting $t_w = 16$. And for the geographical group partition in both cases ($t_w = 12, 16$) the time limit is reached already in the eleventh iteration without finding any feasible integer solution. For all remaining parameter settings a global solution is found within the given time limit. In several cases the global solution is even found and optimized in about 2 hours. Using the time fixation strategy SimS brings clear advantages in computation times for both group partitions and $t_w = 12, t_w = 16$. For the postoptimization a time limit of one hour (included in the total computation time) is used.

6.4.4 Experience with Infeasibility

Until now we only discussed the performance of our sequential decomposition algorithm for feasible problem instances. What happens if no feasible timetable exists for a given timetabling problem? This question is a very important and interesting question connected to the PESP in general.

Typical procedures as relaxing or even leaving out all functional requirements in a first step to test whether they are the reason for infeasibility and to allow longer dwell times to enable further changes in train sequences are well known and already used in practice [Opitz, 2009]. If a timetabling problem after these measures is still infeasible the timetabling software DONS for example provides a further elegant possibility often used by planners [Herrigel et al., 2010]: It allows us the switching off of headway constraints for chosen critical infrastructure sections. With this idea planners can test whether their conjectures for the reason of the infeasibility is true and therefore simplify the adaption of the timetabling input to find a feasible timetable.

In this section we discuss the problem of infeasible PESP instances in connection with the sequential decomposition approach and study three different, concrete examples.

Model	Cyclic MILP	Classical MILP	Seq. Decomp.
InfRes	5 min	> 12 h	25 min
InfConn	40 s	> 12 h	11 min
InfSep	14 s	> 12 h	3.4 h

Table 6.4: *Computation times for three infeasible test instances using the cyclic and the classical global MILP formulation and one example for the sequential decomposition.*

A first infeasible problem instance, referred by *InfRes*, is based on the original Thun-Basel model, but requires a trip time reserve which is not possible on a critical single track line between Thun and Lucerne. With the required trip time reserve trains cannot cross at the given crossing stations anymore and the timetabling problem becomes infeasible. For a second instance, *InfConn*, we reduce the upper bound of all connection constraints in the Thun-Basel model until the instance gets infeasible. This happens starting with an upper

bound of 10 minutes. In a third example, *InfSep*, we require a too strong train separation. We use the EasternCH model and reduce flexibility in train separation from 5 minutes down to 1 minute. Table 6.4 shows the overall computation times to discover infeasibility with the global cyclic MILP, the classical MILP formulation and one example of a sequential decomposition. For the Thun-Basel model a geographical decomposition with 7 groups (Geo7) is used and for the EasternCH example the manual geographical decomposition in 8 groups (GeoMan8). In all three cases a starting fixation $t_w = 2$ is chosen.

The cyclic MILP formulation shows impressive short computation times for all infeasible instances compared to the corresponding feasible problem instances and also to the classical MILP formulation. To have a better impression of the computation times of the sequential decomposition Table 6.5 gives further details on the computation times of different iterations of the decomposition.

Model	Number of subproblems	Number of the infeasible subproblem	Total CT for feasible subproblems	Av. CT for last subproblem BackIt 1-10	Av. CT for last subproblem BackIt 11-20	Av. CT for last subproblem BackIt 21-26	CT for last BackIt
InfRes	7	5	21 s	5 s	13 s	21 s	18 min
InfConn	7	7	93 s	24 s	30 s	30 s	1 s
InfSep	8	5	47 s	37 s	108 s	150 s	160 min

Table 6.5: *Distribution of the computation time for the considered sequential decomposition of all three infeasible test instances*

To prove infeasibility of a test instance over the sequential decomposition method all time fixation constraints have to be enlarged until they do not restrict any feasible solution. In case of a starting fixation $t_w = 2$ and the original defined strategy on fixation constraints (not Sim and SimS), 28 back iterations are necessary to prove infeasibility for one subproblem. The short computation times for all previous feasible subproblems and the first main part of all back iterations for the infeasible subproblem, allow a conjecture on infeasibility already at an earlier time observing the process of the sequential decomposition. If the conjecture of infeasibility appears, it could make sense to start the cyclic MILP formulation to confirm this infeasibility within a short time. Compared to the global MILP formulation, the sequential decomposition, especially with a geographical group partition,

has an essential advantage: the information on the infeasible subproblem helps to find the reason for infeasibility. So for example for the instance InfRes the geographical region of the fifth subproblem, which contained train lines in the South of the model between Thun and Lucerne, directly leads us to the critical single track line, where the chosen trip time reserve causes a conflict.

6.4.5 Comparison to the global solution method

After a discussion of comparisons of the computational performance and solution quality among different solution strategies based on the sequential decomposition approach itself, in this section we want to compare them with the global solution method using the cyclic and classic MILP.

A main instrument to estimate the solution quality of a timetable constructed over the sequential decomposition approach is the postoptimization step already mentioned in previous sections. After finding a feasible global timetable over the decomposition approach, this timetable is used as a starting solution for the cyclic global MILP and is optimized further on. In case of the two smaller instances, we run this computation for 20 seconds and for the largest model we even run the postoptimization for 1 hour. This postoptimization step allows, on one hand, proof of feasibility of the found global solution and on the other hand it provides a global MipGap to estimate quality of the found solution.

As it could already be observed in Section 4.4.2, the solution method based on the global cyclic and classical MILP formulations start to have difficulties with problem sizes as we have in our three largest models. In case of the Thun-Basel model, at least a solution can be found within the given time limit of 24 hours. After about 7 hours of computation time using the cyclic MILP formulation CPLEX finds a first feasible solution with a global gap of 2.65%. And as we can see from Table 4.3 this relative MipGap cannot be improved after 17 hours of optimization. Using the classical MILP formulation CPLEX can find a first feasible solution already after 5000 seconds of computation time and it can optimize this solution considerably faster than over the cyclic formulation. Figure 6.11 visualizes the optimization progress. This clear advantage of using the classical MILP instead of the cyclic one, is an exceptional case among all computations we run in connection with this thesis.

Figure 6.11 further visualizes the results reached over the application of our sequential decomposition approach using the geographical and alternating group partition with the classical time fixation strategy (Geo, Alt). In total it includes 68 computations for the alternating group partition and 70 computations for the geographical group partition. Compared to the optimization progress over the classical MILP formulation there are 3 parameter settings for the alternating group partition (Alt5 ($t_w = 10$), Alt6 ($t_w = 8$), Alt7 ($t_w = 20$)) and two parameter settings for the geographical group partition (Geo6 ($t_w = 8$ and $t_w = 22$)), which perform worse. They need more computation time to find a solution, which has a larger MipGap than the one which would be found using the global classical MILP formulation. But for the remaining 97%,

96% for the geographical and alternating group partition respectively, the sequential decomposition finds faster better solutions than both global solution methods. In 86%, 82% respectively, even a solution is found before any first feasible solution is found over the global solution methods. And for 41%, 31% respectively, a timetable is found which already has a lower MipGap than the one reached by the global solution method until its time limit. Using the best found solution over the sequential decomposition method and a postoptimization of 10 hours we could determine the best solution for the Thun-Basel model with an objective value of 4712.39 and a relative MipGap of 0.32%.

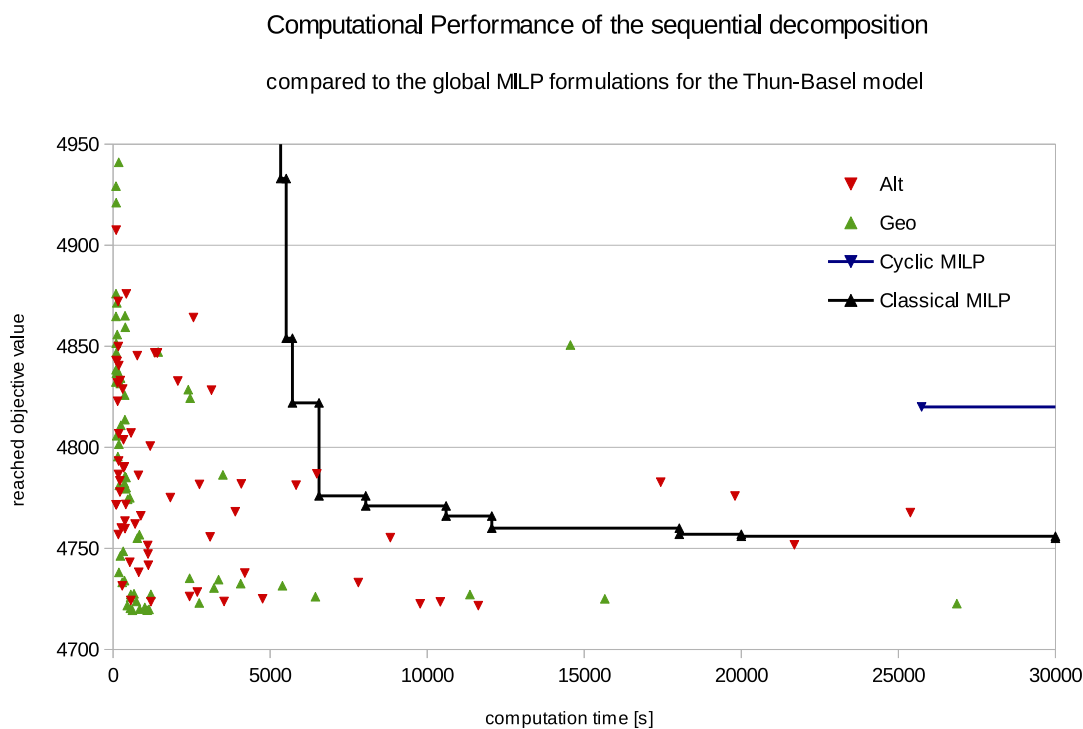


Figure 6.11: Comparison of the computational performance of both global solution methods to the sequential decomposition results for the Thun-Basel model.

In the case of our two largest models EasternCH and GermanSpeakingCH we were not able to find any solution over both global solution methods within the given time limit of 24 hours. Thus every solution found over the sequential decomposition method in the previous section already is an improvement compared to the global solution methods. Table 6.6 gives an overview of the results, in addition to the one visualized in Figure 6.9. Using the manual geographical group partition for the sequential decomposition approach with a small starting time fixation t_w between 0 and 6 minutes, we already find a first feasible timetable in 4 minutes. Even an optimized timetable to an average MipGap of 1.8% can be reached for the same group partition and a larger starting time fixation t_w

around 16 – 22 minutes. Therefore, these computational results show clear benefits in using the sequential decomposition to solve and optimize a larger PESP problem.

Starting fixation t_w	Av. CT Geo	Av. Gap Geo	Av. CT GeoMan	Av. Gap GeoMan	Av. CT Alt	Av. Gap Alt
0 – 6	67 min	4.1%	4.3 min	5.0%	28 min	3.4%
8 – 14	49 min	3.3%	9.8 min	2.9%	87 min	2.9%
16 – 22	2.8 h	2.5%	35 min	1.8%	109 min	2.3%
> 22			92 min	2.0%	113 min	2.4%

Table 6.6: *Overview of the average computational performance of the sequential decomposition for the EasternCH model*

With the intent to enlarge the model size to reach also a limit for the sequential decomposition, we create the GermanSpeakingCH model, as a fusion of all models and some new connecting regions between them. But already after a small adaption of the used MipGap tolerances we are able to solve the model over several parameter setting as visualized in Figure 6.10. Thus the sequential decomposition seems to be a development of a heuristic solution method going in an interesting direction to create new solution methods for the optimization of the PESP, being also able to solve large and dense timetabling problems in practice.

7 Synthesis

7.1 Summary of results

This thesis concentrates on the advancement of algorithms to the automated construction of periodic railway timetables by decomposition methods. It embeds the algorithms in the current planning process and suggests connections to already existing planning software. Furthermore, it reviews detailed instructions on how a timetabling problem can be modelled with the used algorithms. It summarizes and compares different solution methods to solve and optimize the timetabling problems.

The basic algorithmic approach fixed for this thesis is based on a mixed integer linear programming formulation and uses a commercial MIP solver. The algorithms are implemented in a separated tool. With the help of real data directly originating from a software used at Swiss Federal Railway to construct their annual timetable different timetabling problems in several sizes are defined. A timetable evaluation software is used to confirm reasonability of the constructed timetables for practice. Furthermore, the same software can be used to fix timetable supplements providing a certain degree of timetable stability over the reduction of the average delay propagation.

The elaborated test models are used to fine-tune the algorithmic approach accelerating computational performance to construct and optimize timetables. Different parameter settings of the commercial solver and cycle bases for the MILP formulation are tested. With this improved choice of parameters our smallest timetabling problem including 63 trains and 2310 constraints can be solved and optimized up to a MipGap tolerance of 3% in 5 seconds. The three medium size instances including up to 100 trains and about 5'000 – 6'000 constraints are solved with computation times between 3 and 5 minutes. But for our three largest models including 130 up to 300 trains and more than 10'000 constraints the chosen algorithmic approach starts to have difficulties. For the smallest of the three at least a feasible timetable can be found after 7 hours of computation time. But the subsequent optimization stuck completely and can not detect any other solution until the limit of computation time of 24 hours is reached. In the case of the two largest models, no solution at all can be found with a limit of 24 hours of computation time.

To deal with such larger problem instances, two decomposition approaches are introduced in this thesis. A first one, called geographical decomposition, is motivated from known decomposition methods of the optimization theory. It is based on a geographical partition of the timetabling problem into subproblems defined over graph cuts in the model description. The special case of cuts through track sections is discussed and two heuristic iterative algorithms are introduced to solve two subproblems iteratively until a feasible or

even optimized global solution of the timetabling problem can be found. Both algorithms are tested for a small problem instance. They both show benefits in finding faster first feasible global timetables. But they can not accelerate the optimization of the global timetabling problem.

The second decomposition method introduced in this thesis is motivated from manual planning practice. It is based on a partition of all train lines into p groups and uses an iterative algorithm subsequently adding a next group of train lines to the timetabling problem whereby previously scheduled train lines are fixed up to a given time margin. The additional introduction of backiterations ensures the detection of a feasible global solution, if one exists. Comprehensive computational evaluations for different train line partitions and time fixation strategies can show essential benefits in finding first feasible global solutions as well as in the subsequent optimization process. For timetabling models we can not find any feasible global solution with the fine-tuned original solution method in 24 hours of computation time, we find first feasible global solutions with an average computation time of 4 minutes and optimized solutions to an average MipGap of 1.8% within half an hour of computation time.

7.2 Conclusion, discussion

With this thesis the macroscopic timetabling approach of [Caimi, 2009] could be brought further to practice. The provided data from SBB allowed a more detailed model granularity for the timetable construction including corresponding train itineraries. Over the automated inclusion of technical trip times, minimal dwell and train preparation times, a list of the most important connections with minimal transfer times as well as train and track specific line and junction headway times, the model construction could also be accelerated. Therefore, it was possible to enlarge the model regions and to fix different timetabling problems. The connection of our algorithms to the timetable evaluation software OnTime allowed proof of a certain degree of reasonability of our constructed timetables and could thereby further interest from practice.

Our observations of the computational performance of the basic algorithmic approach for our different model sizes could confirm conjectures stated in [Caimi, 2009]. Starting from a certain size and complexity of the timetabling problem, the algorithmic approach starts to have difficulties in optimizing timetabling instances or even in detecting any feasible timetable. The introduced approaches based on decomposition methods could bring essential advantages and experience to deal also with larger problem sizes for timetable optimization problems. The second decomposition heuristic with a parameter setting between a strong sequential and a pure synchronous planning leads to strong benefits in computational performance. Compared to a pure synchronous approach, total computation time could be reduced considerably and compared to a strong sequential method, timetable quality could be improved considerably. Although their benefits were tested for concrete test models defined for Switzerland, the introduced decomposition ideas can also be used for more general applications.

Fixations and constraints for train arrival and departure times early in timetable construction have already been exploited to reduce computation times in different PESP solution methods, especially to solve the PESP decision problem, as for example over the constraint programming approach. However the combination of such ideas with the optimization of a PESP instance together with backiterations is not known to the authors in previous research. The idea seems to be very effective accelerating the timetable optimization process and should be kept into consideration for further research.

7.3 Further research

Further research could concentrate amongst others on the objective function. The introduction of a more sophisticated objective function for example including passenger flows could improve the overall benefit of the approach. Influences of different objective functions on the running time, also in the case of the decomposition approaches, would be interesting.

The definition of an overall support framework defining methods including refinements of the model granularity along different planning stages, as soon as train itineraries are known in more detail, would bring an essential benefit for practice. To this framework also adaptations of the objective function along the considered time horizon could be added and an approach using iterative fixations to reduce complexity approaching the microscopic model granularity could be defined.

After the introduction of such a new overall framework for the refinement of the model granularity, the last step, the translation of the timetable to a complete microscopic timetable, as defined in [Caimi, 2009], could be tested and, if necessary, adequate backiterations to the macroscopic planning level could be elaborated.

For the introduced geographical decomposition method several research questions remained open, for example the definition of an iterative method combining more than two subproblems to a global feasible or even optimal timetable. Or further examinations of more general graph cuts could be studied. And strategies leading quickly to the detection of fixations for the integer cut variables, allowing good global timetables could improve this decomposition idea essentially.

Also the sequential decomposition heuristic could be advanced with further research. For example, the influence of varying group sizes could be studied in further detail. The authors conjecture that a subsequent reduction of the group sizes over the different timetabling steps of the sequential decomposition could bring benefits for computation times and solution quality. But also the definition of a paralleled approach, starting with different starting time fixation margins to find a best first feasible solution in a short time for further optimization could be very interesting.

7.4 Recommendations for application

For this research the model construction to different timetabling problems was mainly used to test all defined algorithms and to evaluate the corresponding computational performance. Now having an algorithmic approach, allowing also the optimization of larger problem instances in acceptable running times for practice, a step further to the practical application could follow.

For this an even closer collaboration with practitioners would be necessary. The application of the algorithmic approach for a specific larger case study could bring further important experience for practitioners as well as for developers. As soon as practitioners trust the algorithms and they can follow the principles the algorithms use to construct and optimize a timetable out of their defined timetable scenario, the connection of the algorithmic approach to existing planning software could be started. With additional options in an adequate user interface to accelerate the definition of a timetabling instance the handling with the given approach further could be accelerated and improved.

Glossary

Blocking time Time interval in which a section of track is allocated to the exclusive use of one train and therefore blocked to all other vehicles [Pachl, 2008].

Buffer time Additional time distance, if two trains run with a larger time distance as the minimum line headway.

CADANS Subsystem of DONS solving a PESP model.

Capacity Maximum number of train that may be operated using a defined part of the infrastructure at the same time as a theoretical limiting value that is not reached in practice [Pachl, 2008].

Commercial stop of a train Station where this train stops to board and alight passengers.

Commercial timetable Timetable with all arrival and departure times of all trains at commercial stops.

Component of a graph Subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the subgraph.

Connection Adjustment of the arrival of a train at a station to the departure of another train at the same station to enable a transfer from the first to the second train for passengers.

CPLEX Commercial solver owned and distributed by IBM to solve mixed integer linear programs.

Cut in a graph Set of edges in a graph which split the graph in at least two components if they are removed.

Cycle Closed path in which the first and last vertices are the same.

Cycle basis A set of simple cycles which forms a basis of the cycle space of a graph in graph theory.

DONS Automatic timetabling system, designer of network schedules used in the Netherlands.

Dwell time Total time between the arrival and the departure of a train in a station.

Frequency Regular time distance between trains of one train line.

Fundamental cycle basis Cycle basis for which there exists a spanning tree such that exactly each cycle closed by a chord of the spanning tree is part of the cycle basis.

Graph A pair consisting of a finite set of vertices and a finite set of pairs of vertices called edges.

Graphical timetable Visualization of the location of each train for a given infrastructure section as a function in time.

Hyperplane Subspace of one dimension less than its ambient space.

Induced subgraph Subset of all vertices of a graph together with all adjacent edges.

Integral cycle basis Cycle basis in which every non-basis cycle is an integer linear combination of the cycles of the cycle basis.

Integrated fixed interval timetable A symmetric periodic timetable for which a network of IFIT hubs is defined, where all trains provide connections in all directions to the first symmetry minute.

Itinerary Sequence of tracks a train uses to drive through the railway network.

Junction Location in the network where train itineraries can converge or diverge.

Line map Visualization of train lines on a geographical map illustrating all direct connections for customers.

Link Connection between two nodes together with all tracks in between.

Macroscopic Node-link-models that contain aggregated information on nodes and links [Pachl, 2008].

Mesoscopic Node-link-models as syntheses of microscopic and macroscopic infrastructure models [Pachl, 2008].

Microscopic Node-link-models that contain, depending on the purpose, the highest possible level of details on nodes and links [Pachl, 2008].

Minimum line headway Minimum time distance two consecutive driving trains can have such that no blocking times intersect.

MIP gap Gap between the best integer objective and the objective of the best node remaining.

Netgraph Combination of a line map and a commercial timetable visualizing all larger stations as small boxes together with all arrival and departure times for the lines passing these stations.

NeTS Timetabling software “Netzweites Trassen-System” used by SBB to construct timetables at different planning stages.

Node Representation of an arbitrary location in a network.

OD-matrix The origin-destination-matrix describes all demands of passenger trips between every pair of origin and destination in a network.

OnTime Timetable evaluation software developed and distributed by VIA Consulting and Development GmbH and traFIT Solutions GmbH.

Operation point Location in the network used to describe a timetable.

Partial periodic timetable A periodic timetable with small adaptations at peak- and off-peak hours.

Path Sequence of edges which connect a sequence of vertices which are all distinct from one another.

Periodic timetable A timetable which repeats itself during each period.

Service intention Description of the offer a railway company would like to tender to the customers during one day.

Spanning tree Subgraph of a graph that connects all vertices but includes no cycle.

Station Operating points in the railway network where trains can stop to board and alight passengers.

Subgraph Subset of all edges of a graph together with all adjacent vertices.

Symmetric periodic timetable A periodic timetable for which all trains have a common symmetry axis.

TAKT Automatic timetabling system developed at TU Dresden and used by Deutsche Bahn AG.

Technical driving time Minimum driving time a train can reach with ideal conditions and maximum power.

Track occupation diagram Visualization of time slots which are used by trains at a platform of a station.

Train dispatching time Time necessary for train operation after closing all doors until an train departs.

Train line Representation of a set of trains serving the same stations regularly over a day.

Train path Part of the capacity of the railway infrastructure which is necessary to schedule or to run a train with a requested speed profile [Pachl, 2008].

Train separation Separation in time of trains belonging to different train lines, but sharing a similar offer.

Train type Classification of trains corresponding to their commercial function to improve comprehensibility for customers.

Viriato Commercial software from SMA and Partners Ltd. for service and operational planning.

Zero time event Time event of a PESP model to which the time zero is scheduled before any calculations.

Bibliography

- [Spo, 2014] (2014). Gleispläne schweiz. www.sporenplan.nl.
- [Andreev and Räcke, 2004] Andreev, K. and Räcke, H. (2004). Balanced Graph Partitioning. *ACM Computing Surveys, New York*.
- [BAV, 2011] BAV (2011). Taktvoll durch die Schweiz. www.deutschland-takt.de. Presentation of Hauke Fehlberg, Sektionschef Planung.
- [BAV, 2014] BAV (2014). Bahnreform. <http://www.bav.admin.ch/bahnreform/>.
- [Benders, 1962] Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numer Math*, 4(1):238–252.
- [Bickel et al., 2010] Bickel, T., Cosanday, E., and Akermann, H. (2010). Deciding for the right timetable production system. *EURAILmag Business & Technology*, 22:202–205.
- [Borndörfer et al., 2010] Borndörfer, R., Erol, B., Graffagnino, T., Schlechte, T., and Swarat, E. (2010). Aggregation methods for railway networks. Technical report, ZIB Berlin.
- [Bücker and Seybold, 2012] Bücker, T. and Seybold, B. (2012). Stochastic modelling of delay propagation in large networks. *Journal of Rail Transport Planning & Management*, 2:34–50.
- [Bueker, 2010] Bueker, T. (2010). *Ausgewählte Aspekte der Verspätungsförtpflanzung in Netzen*. PhD thesis, RWTH Aachen University.
- [Burkolter, 2005] Burkolter, D. (2005). *Capacity of Railways in Station Areas using Petri Nets*. PhD thesis, ETH Zurich.
- [Cadarsó and Marn, 2012] Cadarsó, L. and Marn, . (2012). Integration of timetable planning and rolling stock in rapid transit networks. *Annals of Operations Research*, 199(1):113–135.
- [Caimi, 2009] Caimi, G. (2009). *Algorithmic decision support for train scheduling in a large railway network*. PhD thesis, ETH Zurich.
- [Caimi et al., 2009a] Caimi, G., Fuchsberger, M., Laumanns, M., and Schüpbach, K. (2009a). Periodic railway timetabling with event flexibility. *Networks*. accepted for publication.
- [Caimi et al., 2009b] Caimi, G., Laumanns, M., Schüpbach, K., Wörner, S., and Fuchsberger, M. (2009b). The periodic service intention as a conceptual frame for generating timetables with partial periodicity. In *Proceedings of the 3rd International Seminar on Railway Operations Modelling and Analysis (RailZurich 2009)*, Zurich, Switzerland.
- [Daduna and Vo, 1993] Daduna, J. R. and Vo, S. (1993). Practical experiences in schedule synchronization. In *the Sixth International Workshop on Computer-Aided Scheduling of Public Transport*. 39:55.
- [Dantzig and Wolfe, 1960] Dantzig, G. and Wolfe, P. (1960). Decomposition principle for linear programs. *Oper Res*, 8(1):101–111.
- [De Fabris et al., 2014] De Fabris, S., Longo, G., Medeossi, G., and Pesenti, R. (2014). Automatic generation of railway timetables based on a mesoscopic infrastructure model. *Journal of Rail Transport Planning & Management*, 4:2–13.
- [European Commission and Transport, 2006] European Commission, D.-G. f. E. and Transport (2006). Ertms - delivering flexible and reliable rail traffic.

- [Fioole et al., 2006] Fioole, P., Kroon, L., Maróti, G., and Schrijver, A. (2006). A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research*, 174 (2):1281–1297.
- [Franke et al., 2013] Franke, B., Seybold, B., Bueker, T., Graffagnino, T., and Labermeier, H. (2013). Ontime - network-wide analysis of timetable stability. In *Proceedings of 5th International Seminar on Railway Operations Modelling and Analysis - RailCopenhagen 2013*.
- [Fuchsberger, 2012] Fuchsberger, M. (2012). *Algorithms for railway traffic management in complex central station areas*. PhD thesis, ETH Zurich.
- [Garey et al., 1976] Garey, M., Johnson, D., and L., S. (1976). Some simplified NP-complete graph problems. *Theoret. Comp. Sci.*, 1(3):237–267.
- [Goerigk and Schöbel, 2013] Goerigk, M. and Schöbel, A. (2013). Improving the modulo simplex algorithm for large-scale periodic timetabling. *Computers & Operations Research*, 40 (5):1364–1370.
- [Grossmann et al., 2012] Grossmann, P., Hoelldobler, S., Manthey, N., Nachtigall, K., Opitz, J., and Steinke, P. (2012). Solving periodic event scheduling problems with sat. *Advanced Research in Applied Artificial Intelligence*, 7345:166–175.
- [HaCon, 2014] HaCon (2014). Train Planning System (TPS) Fahrplankonstruktion und -management. <http://www.hacon.de/tps>, Hannover.
- [Herrigel et al., 2010] Herrigel, S., Heydenreich, B., Middelkoop, D., Kroon, L., Den Dulk, J., Van De Burgwal, L., Wojtkowski, A., Vromans, M., Giesing, S., and Schnuurman, S. (2010). Meeting at prorail to the comparison of dons and currently used pesp solution methods used at ifor, eth zurich. Internal Report.
- [Herrigel et al., 2013] Herrigel, S., Laumanns, M., Weidmann, U., and Nash, A. (2013). Hierarchical decomposition methods for periodic railway timetabling problems. *Transportation Research Record*, 2374:73–82.
- [Herrmann, 2005] Herrmann, T. (2005). *Train Routings through Station Areas and Stability of Timetables*. PhD thesis, ETH Zurich.
- [Hooker and Ottosson, 1995] Hooker, J. and Ottosson, G. (1995). Locig-based benders decomposition. *Mathematical Programming*, 96:2003.
- [Huerlimann et al., 2004] Huerlimann, D., Nash, A., Schuette, J., and Krauss, V. (2004). Rarail - a standard data interface for railroad applications. *Comprail Dresden*.
- [Hürlimann, 2002] Hürlimann, D. (2002). OpenTrack, Simulation of railway networks. <http://www.opentrack.ch/>.
- [IBM, 2014] IBM (2014). Ilog cplex optimizer 12.5. www.ibm.com.
- [im Auftrag des BAV, 2011] im Auftrag des BAV, S. B. (2011). *Offizielles Kursbuch Schweiz*. Stämpfli AG, www.fahrplanfelder.ch.
- [Janecek et al., 2010] Janecek, D., Weymann, F., and Schaer, T. (2010). LUKS - Integriertes Werkzeug zur Leistungsuntersuchung von Eisenbahnknoten und -strecken. *Eisenbahntechnische Rundschau (ETR)*, 1:744–747.
- [Klemm and Stemme, 1988] Klemm, W.-D. and Stemme, W. (1988). Schedule synchronization for public transit networks. In Daduna, J. R. and Wren, A., editors, *Proceedings of the fourth International workshop on Computer-Aided Scheduling of Public Transport (CASPT 1988)*, volume 308, pages 327–335. Springer.
- [Kroon, 2008] Kroon, L. (2008). Fahrplanoptimierung mit mathematischen Modellen bei den Niederländischen Eisenbahnen. *Eisenbahntechnische Rundschau*, 6:359–362. In German.
- [Kroon et al., 2009] Kroon, L., Huisman, D., Abbink, E., Fioole, P.-J., Fischetti, M., Maroti, G., Schrijver, A., Steenbeek, A., and Ybema, R. (2009). The New Dutch Timetable: The OR Revolution. *INTERFACES*, 39(1):6–17.

- [Kroon et al., 2006] Kroon, L. G., Dekker, R., Maróti, G., Retel Helmrich, M., and Vromans, M. J. C. M. (2006). Stochastic improvement of cyclic railway timetables. Technical Report ERS-2006-067-LIS Revision, Erasmus Research Institute of Management (ERIM).
- [Kroon and Fischetti, 2001] Kroon, L. G. and Fischetti, M. (2001). Crew scheduling for netherlands railways destination: Customer. *S. Vo, J. R. Daduna, eds. Computer-Aided Scheduling of Public Transport. Springer, Berlin*, pages 181–201.
- [Liebchen, 2004] Liebchen, C. (2004). Symmetry for Periodic Railway Timetables. In *Electronic Notes in Theoretical Computer Science*, volume 92, pages 34–51.
- [Liebchen, 2006] Liebchen, C. (2006). *Periodic Timetable Optimization in Public Transport*. PhD thesis, TU Berlin.
- [Liebchen, 2008] Liebchen, C. (2008). The first optimized railway timetable in practice. *Transportation Science*, 42(4):420–435.
- [Liebchen and Möhring, 2007] Liebchen, C. and Möhring, R. (2007). The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables - and Beyond. In Geraets, F. et al., editors, *Algorithmic Methods for Railway Optimization*, LNCS 4359, pages 3–40. Springer.
- [Liebchen and Peeters, 2002] Liebchen, C. and Peeters, L. (2002). Some practical aspects of periodic timetabling. In Chamoni, P., Leisten, R., Martin, A., Minnemann, J., and Stadtler, H., editors, *Operations Research Proceedings 2001*. Springer, Berlin.
- [Liebchen and Peeters, 2009] Liebchen, C. and Peeters, L. (2009). Integral cycle bases for cyclic timetabling. *Discrete Optimization*, 6(1):98–109.
- [Liebchen et al., 2008] Liebchen, C., Proksch, M., and Wagner, F. H. (2008). Performance of algorithms for periodic timetable optimization. In M. Hickman, P. Mirchandani, and S. Vo (Eds.), *Computer-aided Systems in Public Transport, Lecture Notes in Economics and Mathematical Systems, Springer Berlin Heidelberg*, 600:151–180.
- [Liebchen and Stiller, 2008] Liebchen, C. and Stiller, S. (2008). Delay resistant timetabling. *Public Transport*. Online first.
- [Liebchen and Swarat, 2008] Liebchen, C. and Swarat, E. (2008). The second chvatal closure can yield better railway timetables. In Fischetti, M. and Widmayer, P., editors, *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.
- [Lim, 2010] Lim, C. (2010). Relationship among benders, dantzig-wolfe, and lagrangian optimization. *Wiley Encyclopedia of Operations Research and Management Science*.
- [Lindner, 2000] Lindner, T. (2000). *Train Schedule Optimization in Public Rail Transport*. PhD thesis, TU Braunschweig.
- [Lüthi, 2009] Lüthi, M. (2009). *Improving the Efficiency of Heavily Used Railway Networks through Integrated Real-Time Rescheduling*. PhD thesis, ETH Zurich.
- [Middelkoop, 2010] Middelkoop, A. D. (2010). Simulatie op het spoor proraail vervoer en dienstregeling. <https://afdelingen.kivi.nl/media-afdelingen/DOM100000211/verslagen2010>. Presentation.
- [Middelkoop and Bouwman, 2001] Middelkoop, A. D. and Bouwman, M. (2001). SIMONE: Large scale train network simulations. In Peters, B., Smith, J., Medeiros, D., and Rohrer, M., editors, *Proceedings of the 2001 Winter Simulation Conference*, pages 1042–1047.
- [Mittelman, 2014] Mittelman, H. (2014). Mixed integer linear programming benchmark. <http://plato.asu.edu/ftp/milpc.html>.
- [Nachtigall, 1993] Nachtigall, K. (1993). Exact solution methods for periodic programs. *Hildesheimer Informatik-Berichte 14/93, Universitat Hildesheim*, 14/93.

- [Nachtigall, 1998] Nachtigall, K. (1998). Periodic Network Optimization and Fixed Interval Timetables. Habilitation Thesis, University Hildesheim.
- [Nachtigall, 2014] Nachtigall, K. (2014). Taktfahrpläne mit takt. Chair of traffic flow science, TU Dresden.
- [Nachtigall and Opitz, 2008] Nachtigall, K. and Opitz, J. (2008). Solving periodic timetable optimisation problems by modulo simplex calculations. In Fischetti, M. and Widmayer, P., editors, *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.
- [Nachtigall and Voget, 1996] Nachtigall, K. and Voget, S. (1996). A genetic algorithm approach to periodic railway synchronization. *Computers & Operations Research*, 23(5):453–463.
- [Netcetera, 2014] Netcetera (2014). NeTS - Netzkapazität maximal ausnutzen. www.netcetera.com, Zurich.
- [Noordeen, 1995] Noordeen, M. (1995). *Stability analysis of cyclic timetables for a highly interconnected rail network*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne.
- [Odijk, 1994] Odijk, M. (1994). Construction of periodic timetables, Part 1: A cutting plane algorithm. *Technical Report, TU Delft*.
- [Odijk, 1996] Odijk, M. (1996). A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B*, 30(6):455–464.
- [Odijk, 1997] Odijk, M. (1997). *Railway Timetable Generation*. PhD thesis, Delft University of Technology, Netherlands.
- [Odijk et al., 1997] Odijk, M., Zwaneveld, P., and Hooghiemstra, J. (1997). Decision support systems help railned to search for "win-win" solutions in railway network design. Discussion paper, Tilburg University, Center for Economic Research.
- [Ohrimenko et al., 2007] Ohrimenko, O., Stuckey, P., and Codish, M. (2007). Propagation = lazy clause generation. *Principles and Practice of Constraint Programming*, 4741:544–558.
- [Opitz, 2009] Opitz, J. (2009). *Automatische Erzeugung und Optimierung von Taktfahrplänen in Schienenverkehrsnetzen*. PhD thesis, TU Dresden.
- [Pachl, 2008] Pachl, J. (2008). *Railway, Timetable & Traffic*, chapter Timetable Design Principles, pages 9–42. Eurailpress.
- [Peeters, 2003] Peeters, L. (2003). *Cyclic Railway Timetable Optimization*. PhD thesis, Erasmus University Rotterdam.
- [Peeters and Kroon, 2001] Peeters, L. and Kroon, L. (2001). A cycle based optimization model for the cyclic railway timetabling problem. In Voß, S. and Daduna, J., editors, *Proceedings Computer-Aided Scheduling of Public Transport (CASPT 2000)*, volume 505, pages 275–296. Springer, Berlin.
- [Pham, 2013] Pham, V. (2013). The liberalization of rail transport in the european union. *Economics Honors Papers. Paper 10*. <http://digitalcommons.conncoll.edu/econhp/10>.
- [RhB, 2014] RhB (2014). RhB Streckennetz. <https://www.rhb.ch/de/service-souvenirs/streckennetz>.
- [Schrijver and Steenbeck, 1994] Schrijver, A. and Steenbeck, A. (1994). Dienstregelontwikkeling voor Railned (timetable construction for Railned). Technical report, C.W.I. Center for Mathematics and Computer Science, Amsterdam. In Dutch.
- [Schrijver and Steenbeck, 1994] Schrijver, A. and Steenbeck, A. (1994). Dienstregelontwikkeling voor Railned: Rapport CADANS 1.0. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, the Netherlands. In Dutch.
- [Serafini and Ukovich, 1989] Serafini, P. and Ukovich, W. (1989). A mathematical model for periodic scheduling problems. *SIAM J. Disc. Math.*, 2(4):550–581.

- [Siefer and Radtke, 2005] Siefer, T. and Radtke, A. (2005). Railway - simulation: Key for better operation and optimal use of infrastructure. In *RailDelft 2005*.
- [SMA, 2014] SMA (2014). Viriato - software for railways. www.sma-partner.ch, Zurich.
- [Trick, 2010] Trick, M. (2010). Combinatorial benders approaches to hard problems. In *INFORMS/ALIO*.
- [Tsang, 1993] Tsang, E. (1993). *Foundations of Constraint Satisfaction*. London, Academic Press.
- [UTP, 2014] UTP, V. (2014). Facts & arguments 2014 / 2015 in favour of swiss public transport.
- [Vattani, 2011] Vattani, A. (2011). k-means requires exponentially many iterations even in the plane. *Discrete Comput Geom*, 45:596?616.
- [Voorhoeve, 1993] Voorhoeve, M. (1993). Rail scheduling with discrete sets. *Unpublished report, Eindhoven University of Technology, The Netherlands*.
- [Weidmann, 2011a] Weidmann, U. (2011a). *System- und Netzplanung, Band 1.1, System- und Netzplanung des Personenverkehrs, Vorlesungsskript*. Institut for Transport Planning and Systems.
- [Weidmann, 2011b] Weidmann, U. (2011b). *Systemdimensionierung & Kapazität, Band 2.1, Grundlagen der Produktions- und Ressourcenplanung, Vorlesungsskript*. Institut for Transport Planning and Systems.
- [Wong et al., 2008] Wong, R. C. W., Yuen, T. W. Y., Fung, K. W., and Leung, J. M. Y. (2008). Optimizing Timetable Synchronization for Rail Mass Transit. *Transportation Science*, 42 (1):57–69.
- [Zwaneveld and Kroon, 1995] Zwaneveld, P. J. and Kroon, L. G. (1995). Stations: final report of phase 1. Technical Report 201, Rotterdam School of Management.

Curriculum Vitae

Personal Data

Name: Sabrina Herrigel-Wiedersheim
Date of birth: August 31, 1985
Place of birth: St. Gallen, Switzerland
Nationality: Swiss
Marital status: married
Children: Levin Herrigel (2013)

Education

2010–2014 Doctoral student at the Institute for Transport Planning and Systems (2011-2014), Department of Civil, Environmental and Geomatic Engineering and at the Institute for Operations Research (2010-2011), Departement of Mathematics, ETH Zurich
Dissertation: "Algorithmic decision support for the construction of periodic railway timetables" under the supervision of Prof. Dr. Ulrich Alois Weidmann, Prof. em. Dr. Hans-Jakob Lüthi, Prof. Dr. rer. nat. habil Karl Nachtigall and PD. Dr. Marco Laumanns
2004–2009 Master in mathematics at ETH Zurich
1998–2004 High School in St.Gallen, Switzerland

Experience

2009 Trainee at SMA, consulting and software company for railway system planning in Zurich, Switzerland
2006–2009 Teaching Assistant at the Departement of Mathematics, ETH Zurich