DISS. ETH NO. 28736

PRACTICAL AUTOMATED RAILWAY SCHEDULING – EXPLOITING DECOMPOSITION, INFEASIBILITIES AND RECURRENT SITUATIONS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES
(Dr. sc. ETH Zurich)

presented by

FLORIN LEUTWILER
MSc ETH ME

born on 21 November 1993
citizen of Reinach AG, Switzerland

accepted on the recommendation of

Prof. Dr. F. Corman, examiner
Prof. Dr. P. Pellegrini, co-examiner
Prof. Dr. S. Feuerriegel, co-examiner

2022

ABSTRACT

Railway Scheduling is one of the most important tasks in railways. The successful operation of a railway system directly depends on the timetable. The timetable defines all processes in the operation of railway and defines both efficiency and quality of service for the railway. Operators of a railway are interested in designing timetables as optimal as possible, in order to use existing resources of the railways system as efficiently as possible. An optimal timetable minimizes costs, provides maximum transport capacity, and enables railway operators to offer high-quality services to their customers. The design of a timetable is a highly complex task, which at present time can only be solved by a subdivision. Many of the existing railway networks are so large, that their operators are forced to divide the planning of a network-wide timetable into several sub-steps. Often, the planning of a network-wide timetable is divided by the geography of the network, into local scheduling problems. In consequence of the division, computed timetables do often not correspond to the optimal network-wide timetable. Capacity losses are the result. This thesis focuses on methods and algorithms for the automation of railway scheduling. The focus lies on the efficiency and scalability of new methods and algorithms, to compute an optimal timetable for large-scale planning problems. The thesis is divided into three major parts, each of which addresses different aspects of railway scheduling through different methods.

The first part of this thesis is dedicated to the development of a decomposition approach to compute a timetable. In contrast to today's methods of dividing the problem, an optimal timetable is to be calculated by the novel method. Decomposition approaches many times show an improved scalability compared to centralized approaches, which motivates the development of such a methodology for scheduling. A comprehensive analysis highlights existing decomposition approaches in the area of railway scheduling and discusses strengths and weaknesses of the different decompositions and solution methodologies. Based on the analysis, a novel geographic decomposition is developed. Necessary theory for the novel decomposition is developed and translated into a practical implementation. Experiments show an increased scalability of the new method, compared to existing approaches.

The second part of the thesis focuses on the enhancement of the decomposition method developed in the first part. Statistical knowledge about

recurrent situations in railway operations is exploited to reduce the computational complexity of the existing decomposition method. It shows, that recurrent situation can be partially learned, and the decomposition method can be accelerated. The accelerated decomposition method is particularly suitable for time-critical applications, such as the real-time adjustment of schedules due to unforeseen disturbances.

The third part of the thesis investigates an alternative decomposition approach, compared to the first part of the thesis. The focus lies on the efficiency of a methodology. By a particular decomposition of the scheduling problem, the problem can be separated into two characteristic problems. In particular, the search for the optimal timetable is separated from the question of feasibility. It shows that an optimal timetable can be found by means of a problem widely known in the literature, which can be solved extremely efficiently. In practical experiments, the novel decomposition method proves highly efficient for problems of railway scheduling.

This thesis introduces several new approaches and enhancements for the automation of railway scheduling. Newly developed methods and algorithms show improved runtime and scaling behavior compared to existing approaches. The results of this thesis provide a foundation for the future development of fully automated systems for railway scheduling, and a tool for computing network-wide optimal timetables.

ZUSAMMENFASSUNG

Die Fahrplanplanung ist eine der wichtigsten Aufgaben der Bahn. Der erfolgreiche Betrieb einer Bahn steht in direktem Zusammenhang zum Fahrplan. Der Fahrplan definiert alle Abläufe im Betrieb einer Bahn und definiert damit die Effizienz als auch die Servicequalität des Bahnbetriebes. Die Betreiber einer Bahn sind daran interessiert Fahrpläne optimal zu entwerfen, um bestehende Resources der Bahnanlage möglichst effizient einzusetzen. Ein optimaler Fahrplan minimiert Kosten, bietet maximale Transportkapazität und ermöglich es den Bahnbetreibern dem Kunden ein qualitativ hochwertiges Angebot zu bieten. Der Entwurf eines Fahrplans ist eine hoch komplexe Aufgabe, welche zu heutigen Zeit nur durch eine Aufteilung überhaupt lösbar ist. Viele der bestehenden Bahnnetze sind so gross, dass deren Betreiber gezwungen sind, die Planung eines netzweiten Fahrplans in mehreren Teilschritten durchzuführen. Dabei wird oftmals die Planung des gesamten Netzes, in geographische Gebiete aufgeilt. Eine Zerteilung der Planung hat zur Konsequenz, dass resultierende Fahrpläne für das gesammte Netz oftmals nicht dem optimal möglichen Fahrplan entsprechen. Kapazitätseinbussen sind damit unvermeidlich. Die vorliegende Arbeit konzentriert sich auf Methoden und Algorithmen zur Automatisierung der Planung von Fahrplänen im Bereich der Eisenbahn. Im Fokus liegt die Effizienz und Skalierbarkeit neuer Methoden und Algorithmen, um auch für grosse Planungsprobleme eine optimale Lösung zu errechnen. Die Arbeit ist unterteilt drei Teile, welche jeweils durch verschiedene Methoden unterschiedliche Aspekte der Fahrplanplanung adressieren.

Der erste Teil dieser Arbeit widmet sich der Entwicklung eines Dekompositionsansatzes zur Errechnung eines Fahrplanes. Bei dem neuen Ansatz wird, im Gegensatz zu heutigen Methoden, ein optimaler Fahrplan errechnet. Dekompositionsansätze zeigen oft eine verbesserte Skalierbarkeit gegenüber zentralisierten Ansätzen, was die Entwicklung einer solchen Methodik motiviert. In einer umfangreichen Analyse werden bestehende Dekompositionsansätze im Bereich der Fahrplanplanung aufgezeigt und die Stärken und Schwächen der verschiedenen Dekompositionen und Lösungsmethodiken diskutiert. Auf Basis der Analyse wird eine neuartige geographische Dekomposition entwickelt. Die dazu notwendige Theorie wird hergeleitet und in einer praktischen Implementierung umgesetzt. Expe-

rimente zeigen eine erhöhte Skalierbarkeit der neuen Methodik gegenüber bestehenden Methoden.

Der zweite Teil der Arbeit fokussiert die Weiterentwicklung der Dekompositionsmethode aus dem ersten Teil. Dabei wird statistisches Wissen über wiederkehrende Situationen in Bahnbetrieb ausgenutzt, um den Rechenaufwand der bestehenden Dekompositionsmethode zu reduzieren. Es zeig sich, dass wiederkehrende Situation teilweise erlernt werden können, und damit die Dekompositionsmethode beschleunigt werden kann. Die optimierte Dekompositionsmethode ist besonders geeignet für die zeitkritische Applikation der Echtzeitanpassung von Fahrplänen aufgrund unvorhergesehener Störungen.

Der dritte Teil der Arbeit untersucht einen alternativen Dekompositionsansatz, im Vergleich zum ersten Teil der Arbeit. Dabei steht die Effizienz der Methodik im Vordergrund. Durch eine gezielte Dekomposition der Planungsproblems eines Fahrplans, lässt sich das Problem in zwei Teilprobleme aufspalten. Die Suche nach dem qualitativ optimalen Fahrplan wird von der Frage der Machbarkeit separiert. Es zeigt sich, dass ein qualitativ optimaler Fahrplan mittels eines in der Literatur weitaus bekannten Problems gefunden werden kann, welches äusserst effizient lösbar ist. Durch die Separierung der zwei Teilprobleme kann ein hoch effizienter Algorithmus zur Fahrplanplanung geschaffen werden.

Die vorliegende Arbeit zeigt mehrere neue und aufeinander aufbauende Ansätze zur automatisierten Fahrplanplanung. Neu entwickelte Methoden und Algorithmen weisen eine verbesserte Laufzeit und Skalierung im Vergleich zu bestehenden Ansätzen in der Fahrplanplanung. Die Resultate der Arbeit bieten ein Fundament zur Entwicklung von vollautomatisierten Systemen zur Fahrplanplanung, und somit einem Werkzeug zur Berechnung netzweit optimaler Fahrpläne.

ACKNOWLEDGEMENTS

My most sincere gratitude goes to Francesco Corman for his guidance and support throughout my entire journey of creating this thesis. Our long and exhaustive arguments challenged me in many aspects and lead to what the thesis is today. Thank you for showing me the importance of explaining my own thoughts to others and the beauty of colors! I am most grateful for the support and feedback in the process of writing all my papers. Thanks for all the long hours of discussions over all the little details of my work.

Further gratitude goes to all of the team Flux at SBB. From the beginning on I felt very welcome and I was supported with any help whenever it was needed. Thanks to all the "monkeys" of Flux for showing me how to write proper code and taking the time to explain me all the tools of this world. A big thanks goes also to the three "elephants" of Flux, Ambra Toletti, Julian Jordi, Kaspar Schüpbach. Our discussions about all the nasty, yet fascinating facets of timetabling and all the small details deeply hidden in all the algorithms of timetabling were always a great joy and the thesis would not be the same without them.

A special thanks goes to Gabrio Caimi. It was only through your enthusiasm for research that this thesis became possible. Your motivation and enthusiasm for my research was always a great motivator.

I would also like to thank Paola Pellegrini and Stefan Freuerriegel for taking the time to be my co-examiners and sharing interest in my research.

A big thanks goes to all past and present members the TS Group at IVT. Despite my rare presents in Zurich, it was always great fun to hang out with all of you during the many hours of coffee breaks, lunch and other happenings. A thank you to Beda for showing me the joy of running in the mountains.

Finally, I would like to thank my family and friends for supporting during the time of my thesis and for distracting me from my work with all kinds of adventures from time to time.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

1.1 MOTIVATION

Importance of Railways and Railway Traffic Management

Railways have always played an important role in the society and economy of modern countries. In public transport, railways provide fast, safe and efficient service for medium and long distance travels. In economy, railways provide a safe and cost effective way for national and international freight transport. In many European countries railways are a substantial mode of transport to the population, e.g., Germany or Switzerland the modal split in 2019 shows that 6.2% [31] and 13% [28] of all passenger kilometers where made made by railways with total of 2 931 Mio. [31] and 641 Mio. [29] passengers in the year 2019. A significant part of the populations need for travel is satisfied by railways. From an industrial point of view, in the year of 2019, German Railways and the Swiss Federal Railways served respectively 123 066 Mio. ton kilometers [30] and 9 908 Mio. ton kilometers [27] of freight transport. With these numbers railways make up to 10.6% [30] and 37% [27] of the total freight transported in these countries. Indisputably, the above numbers underline the importance of railways to public transport and economy.

Challenges of Railway Traffic Management

In railways, enormous efforts in planning are necessary to operate railway systems smoothly and at the same time, assure a satisfactory level of service to all customers. The summary of planning in railway systems is known as railway traffic management (see Section 1.2). Modern countries with well developed railway systems face major challenges in railway traffic management, organizing and successfully operating their system [6]. The sheer amount of infrastructure and resources to be acquired, maintained, operated and controlled within such large systems forces the operators of railway systems to split railway traffic management into several phases and multiple planning steps in each single phase. The phases are know as the strategic, tactical and operational phase or in other words the design,

planning and control phase [8]. In the strategic (design) phase, future available resources of the railway system are planned together with a concept of usage. In the tactical (planning) phase, a plan for the operation of available resources is designed. For the profitability of a railway it is crucial that all resources are operated as economically as possible. In the operational (control) phase, the railway system is operated according to results of the tactical phase. Real-time control of the system deals with unforeseen disturbances.

Often even a single planning step inside a single phase of railway traffic management (e.g. the design of a timetable) is too complex, when considered on the railway network of an entire company or country, such that these problems cannot be tackled as a whole [6]. In consequence, many of the current practices in railway traffic management are to split up any problem of planning (often geographically), and address the separated parts individually, to then manually or computer-aided, merge local results back together to a consistent network-wide solution of the problem [49, 60]. While this process is able to provide solutions for the extremely complex problems of railway traffic management, the process is in general suboptimal. Most often local results are merged using rules of practice, i.e., by heuristics or practical experience of human planners (e.g., high-speed trains have precedence over suburban trains, first come first serve strategies etc. [34, 41]). These practices, while providing satisfactory results, clearly neglect a large set of possible solutions, some of which are likely to perform better than results of current practice. It is a major challenge in the organization of railways to improve current practices beyond rules of practice and find best possible solutions to all the problems of railway traffic management. Only then the benefits of railways for society and economy can be ensured and increased in the future.

Automation in Railway Scheduling

Railways experience in recent years a continuously rising demand for passengers and freight transport, which must be matched with a increasing capacity of railways, to maintain and improve the existing levels of service. In many cases, the capacity of a railway can be increased through the acquisition of new network resources, e.g., infrastructure or trains. The acquisition of new resources is in general a rather cost expensive way to increase the capacity of a railway and increases the amount of infrastructure to be maintained. In contrast to the acquisition new resources, many railway companies, including the Swiss Federal Railways [58], see a more

cost effective lever on capacity in the current practices of railway traffic management, in particular practices for railway scheduling, which consider the design and adjustment of railway timetables. Optimizing the operation of existing resources, increases, in a very cost effective way, the capacity of railway systems and allows to meet the future demand. Improvements on current practices require new methodologies to tackle the problems of scheduling in railway traffic management and getting one step closer to the optimal utilization of railway resources [6].

Considering the computational aspects of problems in railway scheduling, most of them are proven to be NP-Hard and require an exponentially growing solving-effort with the growth in size of a problem [6]. The size itself of a problem may increase due to an increased level of detail or simply due to an increased number of resources considered. Current algorithms of academia are in general unable to handle problems of railway scheduling for large-scale networks, e.g., those of entire countries, without significant simplifications of the problem. A clear need for algorithms handling larges and more detailed instances of railway scheduling exists [39]. In either way, the scalability of scheduling problems in railway traffic management is in direct conflict to the needs of railway operators.

With a more integrated manner of solving scheduling problems in railway traffic management, railway operators aim for an improved performance and more capacity from their system, to meet future demands.

This thesis proposes novel methodologies and algorithmic approaches, for problems of railway scheduling, i.e., timetabling and rescheduling, in railway traffic management. With methodologies of decomposition, the exploration of recurrent situations and the insights gained by infeasibilities, novel algorithms for the problems of railway scheduling are developed to handle larger problems more efficiently than approaches of the existing literature. Overall, this thesis is a contribution towards the automation of railway scheduling and a step towards a cost effective way for railway operators to match the future raising demand.

1.2 BACKGROUND

Railway traffic management is the summary of all organization, planning and control processes necessary to operate a railway system. So far, the only practically successful way to handle the challenges of railway traffic management is to separate the process into a hierarchy of sub process, each

itself individually manageable by available technologies and organizational processes of today. In this section, a background on problems of railway traffic management is given with a particular focus on the problems addressed in this thesis, that are the problems of railway timetabling and railway rescheduling. The process of railway traffic management can be divided into three sequential phases [44]:

Strategical Phase

The first phase of railway traffic management is referred as the strategical planning phase. Processes within the strategical phase are characterized by a long term time horizon and in general a lager number of involved stakeholders, i.e., railway infrastructure and operator companies and numerous governmental institutions. [8] describes the strategical phase as the phase of resource acquisition. Strategical planning itself can be divided in two subphases.

The first subphase of strategical planning considers the network planning, i.e., the study of available resources, e.g., railway infrastructure or rolling stock, and planning of future changes or acquisitions in resources; hence the term resource acquisition. Network planning is a very strategic, politically influenced kind of planning, performed on a abstract level, such that in general it is not addressed by academia.

The second subphase of the strategical planning is the planning of railway lines, i.e., the line planning. A railway line describes a railway service in terms of served stations, frequency and capacity of the service. Each railway line has an origin, a destination and intermediates stations to be served. The basis for the line planning involves the elements of railway infrastructure, demand and political requirements. Political requirements are often minimal required capacities (e.g., in terms of available seats) for particular lines, in consequence of the governments duty to provide public transport to all citizens. In the literature of railway traffic management, researchers proposed different mathematical models for the problem of line planning. Most models can either be categorized as an integer [15, 33, 54] or (non-linear) mixed integer [9, 10] mathematical program. On these models, different objectives for the mathematical optimization have been proposed. Approaches like [15] or [33] optimize from an operators point of view, the operational cost of the line plan in terms of numbers of lines and number of coaches used to satisfy the demand; other approaches present an objective from a passengers point of view (e.g., [9], [54]).

At the end of the strategical planning phase a line plan exists. The line plan specifies a set of railway lines to be operated in the future and on future railway infrastructure. In the next planning phase of railway traffic management, outgoing from the line plan, a schedule (timetable) for the operation of train services is computed.

Tactical Phase

The second phase of railway traffic management is referred as the tactical phase. [8] describes this phase as the resource allocation phase. In this phase railway resources, i.e., infrastructure elements, rolling stock and railway staff, are allocated to the train services specified in the line plan. The tactical phase can be further divided into three subphases.

In the first subphase of tactical planning, railway infrastructure resources are allocated to the train services. This process is often referred as timetable planning or simply timetabling. The result of timetabling is a plan of operation for all train services together with a plan for the allocation of railway infrastructure to individual train services over time. For more details on timetabling see Section 1.2.1.

Subsequent, in the second phase of tactical planning, rolling stock is allocated to the train services in the timetable. The major contribution of rolling stock planning is an optimized plan for the circulation of rolling stock, to generate transportation capacity (e.g., in terms of seats available) as demanded by the line plan, while minimizing the amount of shunting operations. Input to rolling stock planning is an a-priory computed line plan and a corresponding timetable, together with information on the available rolling stock and its initial distribution over the railway network. Rolling stock circulations are often designed to end with a rolling stock distribution on the network, that is identical to the initial situation; such circulation can in general be reapplied with only minor additional shunting efforts. Approaches in the literature on rolling stock planning can be categorized by the units to be assigned to individual lines of the line plan and whether underway combining of those units is considered or not. Approaches as [1], [50] or [11] assign complete locomotive-hauled units to train services considered in the timetable. Approaches such as [17] or [25] consider the individual assignments of locomotives and coaches into train units. In approaches such as [50], assigned train units can only be combined or separated at the start or end of a line in the line plan, i.e., no underway combining is considered. Differently, approaches such as [17]

consider the underway combination of single elements into train units, also with regards to the time necessary to reorder trains units. Objectives for the optimization in rolling stock planning are often chosen either to minimize the operational cost or to minimize passenger discomfort. Operational costs are represented through fixed and variable costs [4], shunting actions [50], the number of units used [55] or seat shortage with respect to the expected demand [1, 25]. The result of rolling stock planning is a sequence of trips for all rolling stock units, i.e., a rolling stock circulation.

In the last subphase of tactical planning, the crew scheduling is performed, where the railway staff is assigned to the train services considered in the timetable. The crew schedule is planned based on the rolling stock circulation and the timetable. Each train has to be assigned a driver as well as a minimal amount of crew members necessary to operate the train. Crew scheduling usually includes, besides the constraint of all shift being covered, constraints of working regulations to assure a legal working environment for the staff. Possible such constraints are (meal) breaks, minimal and maximal duration of shifts (e.g., [2, 59]), variation in the schedule [37] or schedules according to the location of staff and depots [36]. Crew scheduling is in generally modeled as an integer problem, in particular a problem of set covering. The problem of set covering is in general extended by additional constraints due to the above discussed legal and operational aspects. Most commonly, the objective minimized in crew scheduling is the total cost of staff deployed.

Other elements of the tactical phase are possibly the planning of maintenance (e.g., [43]) and the planning of shunting operations (e.g., [26]).

The result of the tactical phase are schedules for train services, rolling stock and crew. In the next phase of railway traffic managements, these plans are executed on the railway network.

Operational Phase

The last phase of railway traffic management is referred as the operational phase. [8] describes this phase as the resource consumption phase. For the successful operation of a railway network, it is crucial to handle delays and disruptions efficiently. To deal with delays, operators of a railway network perform real-time rescheduling of railway operations, where the existing timetable, computed during the tactical phase is replanned under consideration of real-time information, e.g., delays of trains or disruptions of the network. Such replanning is in general referred to as real-time rescheduling or dispatching. Real-time rescheduling reduces delays in the sense of

reestablishing the original timetable and preventing existing delays from spreading over the railway network. Insights on rescheduling are given in section 1.2.2.

Figure 1.1 gives an overview on all the individual phases and planning steps of railway traffic management.

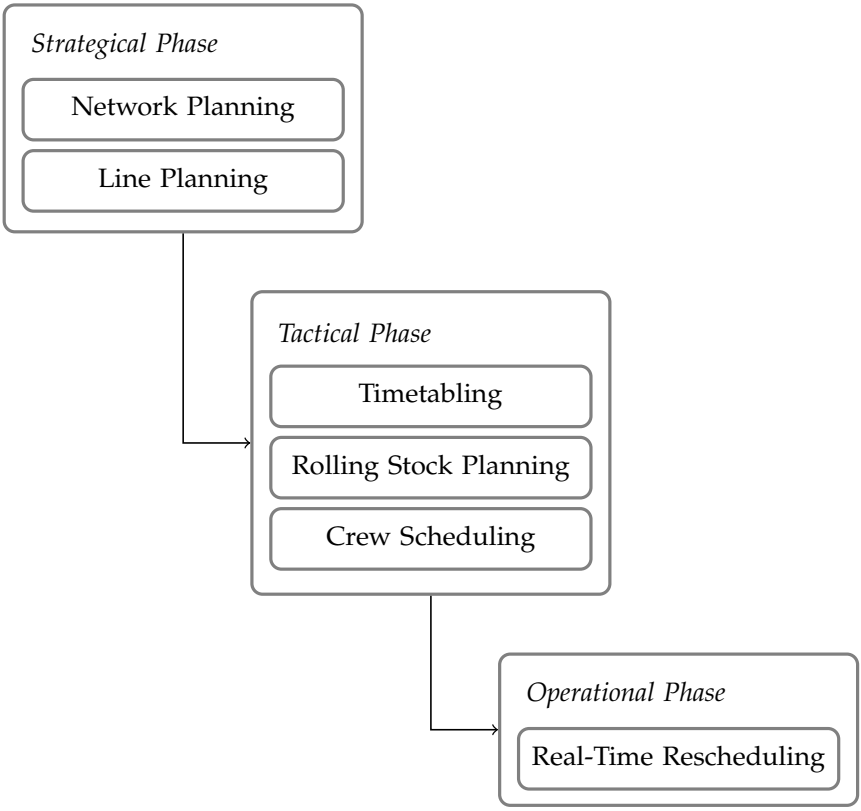

FIGURE 1.1: Overview of Planning Steps in Railway Traffic Management.

1.2.1 *Timetabling*

In the first subphase of tactical planning, a line plan is converted into a detailed schedule of operations for all train services operating on the railway infrastructure. This process is generally referred to as timetable planning or

simply timetabling. Aside from constraints of the infrastructure, timetabling also includes constraint from a safety system, the dynamics of the rolling stock and of course the line plan.

Timetabling is in general performed on either of two abstraction levels of the railway network. Macroscopic timetabling is a simplifying approach where the railway network is reduced to stations and lines, which yields a significant decrease in computational complexity [12, 40]. In macroscopic timetabling, railway traffic is scheduled on an abstract level in terms of arrivals and departures at important nodes of the railway network. A more detailed modeling is the microscopic representation of a railway network. The infrastructure in microscopic models is considered at the level of block sections, i.e., sections of few hundred meters of railway track. The level of detail in microscopic timetabling corresponds to a level of detail used in safety systems for collision avoidance [5, 14]. In microscopic timetabling, entry and exit times for all trains over all traversed block section are scheduled. While a macroscopic approach simplifies the timetabling problem, only a microscopic approach can guarantee the operability (in terms of safety) of the resulting timetable.

Orthogonal to micro-macro, timetabling can be categorized into periodic and non-periodic. Periodic timetabling [48] intends to simplify the timetabling process by rolling out a short schedule, e.g., for few hours of operation, over a longer time horizon, e.g., a day or several days. Consequentially the time horizon for the actual computation can be shortened in periodic timetabling, reducing the overall size of the problem. While the size of a problem is generally reduced in periodic timetabling, periodicity constraints introduce an additional complexity to the periodic formulation of the timetabling problem. In non-periodic timetabling, the entire time horizon is scheduled at once. As such non-periodic timetabling is in general a more complex computational problem, but in advantage over periodic timetabling, non-periodic operations such as irregular freight trains can directly be included in the planning process. Differently, in periodic timetabling, non-periodic operations are usually manually inserted into the periodic timetable after the computations.

In the literature on railway timetabling a variety of modeling techniques are proposed. The most common approach is the formulation of a mixed-integer mathematical program, often under the usage of big-M constraints [21, 56]. Other popular approaches are graphical formulations, where the constraints of a timetabling problem are graphically represented, e.g., in the alternative graph of [45] for job-shop scheduling. Discrete event models [23,

57] formulate operations to be planned as discrete events, which, in combination with forward simulation or max-plus algebra, are used to derive a timetable. Constraint programming models [53] propose an alternative formulation, where the problem of timetabling is specifically formulated in the language of a specific class of solving techniques, i.e., constraint programming algorithms.

All of the above models are solved with respect to a variety of objectives. Probably the most popular objective is the minimization of the total time of operation [21]. In this case, from an operators point of view, the time span from the first operation till the last operation over all train services is minimized. In case the line plan also inherits temporal bounds on the operation of train services, the objective can be reformulated as minimizing deviations, i.e., delays, with respect to the temporal bounds of line plan [23]. Other examples of objectives in the literature are energy related objectives [14] or in case delaying a train is not an option, the absolute number of trains services included in the final timetable [13].

An important aspect in the problem of timetabling is the routing of trains over the network. Many of the models discussed above consider for each train service a fixed route, given as an input to the problem of timetabling. Railway networks are often structured as such, that only within the closer perimeter of a station, routing alternatives are available, and longer connections between hubs consists of single or double tracks; this makes the assumption of fixed routes often very reasonable, especially in the case macroscopic models. As the counter part, designated approaches of the literature address in specific the problem of routing train services in areas of stations or other areas with routing alternatives [62]. Extended approaches [12, 46] include the question of routing directly into the problem of scheduling, where the direct inclusion in general proposes a significant overhead with respect to computational complexity.

The result of timetabling is a detailed plan of operations for all train services to be operated, i.e., a timetable. The plan of operations determines exact departure and arrival times of train services at stations and other strategically important nodes of the railway network. Implicitly, the timetable also defines routes and orders of train services on the railway infrastructure.

1.2.2 *Real-Time Rescheduling*

All planning steps before the operational phase of railway traffic management are in general done long term in advance before the actual operation,

e.g., days or months in advance. Once a detailed schedule of operation is established, it must be continuously updated during the operation, to adapt the schedule to unforeseen events, i.e., smaller disruptions or delays. Updating an existing schedule under consideration of real-time information is referred as (real-time) rescheduling; numerous examples are given in [24]. Rescheduling considers often the same microscopic representation of the railway network as microscopic timetabling, making these two problems almost identical in their constraints. Rescheduling is usually performed under very strict time limitations to enable real-time control of the railway system. To achieve a solution within limited time, rescheduling typically focuses on very small geographic areas in the railway network to keep the size of rescheduling problems at a moderate size.

Models for rescheduling coincide with those of timetabling. (Mixed) Integer programming [51], alternative graph [19] and discrete event models [20] are present in the literature of rescheduling. Depending on the approach, the model includes rerouting possibilities [22, 35] or not [19]. The models are optimized with respect to recovering an optimal system state. With an operators point of view, the goal is to minimizing the total or maximal delay in the system [52], the number of kept connections or the on-time arrival of trains [20]. Further similar objectives exist in the literature (see e.g. [24]). With a passengers point of view, objectives as [22] or [3] optimize earliest passenger arrival or the total number of served passengers. For broader overview see [18] or [24].

The results of real-time rescheduling is an adapted version of the timetable, in general with adapted travel times of train services and possibly different routings and orderings for train services. The adapted schedule is used by the railway dispatcher to control the railway system towards a desirable state.

1.3 RESEARCH QUESTIONS

This thesis is dedicated to the problem of railway scheduling. The overarching research question of this thesis is:

How can the scalability and efficiency of algorithms in railway scheduling be improved?

The main research question of this thesis is detailed in four different subordinate research questions, which are to be answered within the scope of this thesis. The four research question of this thesis are:

Q1: *How can problems of railway scheduling be decomposed?*

For many different problems faced in applications of science and engineering approaches of decomposition have proven of great value for large-scale problems as they exploit particular structures in the underlying mathematical optimization problem [16]. Also in the field of railway scheduling, decomposition approaches have proven their value (e.g. [40, 42]). With a detailed study of the literature on decomposition in problems of railway scheduling, it is to understand different ways to decompose a problem of railway scheduling. Decompositions of the literature should be analyzed to understand the structure of problems in railway scheduling and how these can be exposed for a decomposition of the problem. Methods of the literature should be characterized by individual strengths and weaknesses to understand their suitability to different problems of scheduling.

Q2: *Can decomposition improve the scalability of algorithms for railway timetabling?*

Scalability is a major issue in railway timetabling. With more scalable algorithms larger problems of timetabling can be addressed and global solutions can be assembled from fewer, larger solutions, improving the overall quality of timetables. Algorithms for railway scheduling with good scalability are required. In contrast, problems of railway timetabling are NP-Hard, making scalability a major challenge. With this research question it is to understand, whether a decomposition approach can lead to an algorithm for railway timetabling with improved scalability and by which larger instances of railway timetabling can be addressed, compared to algorithms of today. Findings from research question Q1 should be used as a basis.

Q3: *Can statistical learning of recurrent situations improve existing approaches of OR for the time-critical problem of real-time railway rescheduling?*

The application of real-time rescheduling for railway is a crucial part in the operation of a railway system. Within short time, adaptations of timetables with high quality are required to keep the stability and the level of service of a railway at a high level, at all times. In the application of rescheduling, very often recurrent situations occur. Often timetables are repeated hourly or daily and similar situations occur during every repetition

of the timetable. This research question is to address the question, whether statistical knowledge about recurrent situations in railway operations can be exploited to improve existing algorithms of OR for railway rescheduling; possibly the findings of research question O2. An improvement in computational performance will increase the practical applicability of algorithms to the problem of real-time rescheduling.

Q4: *How can proofs of infeasibility be used to design efficient algorithms for railway timetabling?*

Proofs of infeasibility have been used, and proven valuable in many applications of mathematical optimization (e.g., Decomposition [32] or Optimization [47]). As a proof of infeasibility exposes insights to the underlying mathematical optimization problem, it can be used to strengthen an existing mathematical formulation or provide insights (e.g., lower bounds on the objective) during the process of solving a problem of mathematical optimization. With this research question it is to understand, whether proofs of infeasibility can be used in the design of algorithms for railway timetabling, leading to more efficient algorithms that those of today. It should be analyzed how knowledge on a problem of timetabling, gained through a proof of infeasibility, can be exploited for efficiency. The goal is to create an novel algorithms based on proofs of infeasibility, that provides optimal or near optimal solutions in short time.

1.4 RESEARCH CONTRIBUTION

Relevance to Society

This thesis contributes with novel methodologies and algorithms for the problems of railway scheduling. Novel methods and algorithms can be used in software tools for automated railway scheduling, either to design new timetables or adapt timetables in real-time. Automated railway scheduling brings several benefits for railway operators and railway passengers in comparison to practices of today.

Automation in the design of railway timetables (i.e., timetabling) has already proven to bring great benefit to railway operators (e.g., [38]). Timetables computed by computer algorithms often prove superior to a human designed timetable, resulting in more available capacity on the same infrastructure [40]. On a local scale, algorithms for railway timetabling are able to evaluate the full spectre of possible timetables compared to practices of today, and provide best possible solutions. On a global, network-wide

scale, efficient and scalable algorithms can over come the limitations of today's practices, where in the process of merging local timetables, global optimal timetables are lost and with it, potential network capacity. Algorithms with better scalability, such as those developed in this thesis, can help to overcome losses in merging local solutions, by reducing the number of geographical splits in the planning problem, resulting in an increased network capacity. An increase in capacity not only benefits railway operators, but also railway passengers. With an increased capacity through improved timetabling, railway operators can achieve the necessary capacity with less cost intensive and time consuming infrastructure acquisitions. Reduced infrastructure and operational costs for railway operators reflects in lower ticket pricing for railway customers. Furthermore, an increased capacity through better timetabling can help to raise the level of service for passengers in a cost effective manner. In summary, an automation in railway timetabling brings mutual benefits to railway operators and passengers, underlining the relevance of this thesis.

Automation for real-time adjustments existing timetables (i.e., rescheduling) improves the real-time control of railway systems. Highly efficient algorithms as those of this thesis, provide quickly and reliably optimal or near optimal adaptations of the existing timetable to the current situation of the network. Algorithms can relieve human dispatches in time-critical situation, providing solutions reliably, within short time. Furthermore, scalable algorithms can address larger areas of the railway network quickly to better estimate the impact of rescheduling actions on the entire network. Consequentially, with more efficient, more scalable algorithms the stability and reliability in the operation of a railways will improve, lowering costs for railway operators and raising the level of service for passengers. With novel methodologies, this thesis supports the development of automated rescheduling, relevant for practice.

Relevance to Science

The present thesis is first and foremost relevant to the field of research in railway scheduling. The thesis introduces novel methodologies for the decomposition of problems in railway scheduling and provides a strong basis for further research. Proposed methodologies generalize beyond the applications shown in this thesis and may be used by other researchers to create a variety of decompositions in the field of railway scheduling. Decomposition in railway scheduling is today an active field of research as possible key concept to tackle scalability issues of railway scheduling

problems and thus to tackle a major obstacle when putting results of academia into practice [7].

Beyond the scope of railway applications the relevance of this thesis extends to a much broader spectrum of scheduling problems. Scheduling problems arise in a variety of applications in the industry and are of great interest to researchers in the field of operations research (OR) and mathematics [61]. The present thesis proposes novel methodologies for the decomposition of scheduling problems that are not limited to applications of railways. Rather, the novel methodologies of this thesis are provided on general description of the scheduling problem, which motivates the extension of findings to other fields of research.

1.5 METHODOLOGY

Data

The research of this thesis was conducted in close collaboration with the Swiss Federal Railways (SBB). Real-life data (instances) on the problem of railway timetabling were provided by SBB for the experiments in this thesis. Provided data are specifications and boundary conditions originally used for the design of the current railway timetable of Switzerland, in operation at the time this research has been conducted. All instances of railway timetabling provided by SBB are reduced versions of the original planning problem and are temporal and geographic excerpts from the original timetable of Switzerland; the original timetable covers a full day of operation in the entire country. As real-life data has been used for the experiments in this thesis, all computational results provided, reflect the actual traffic on the railway of Switzerland.

Data provided by SBB is devoted to timetabling on a microscopic representation of the railway network, i.e., timetabling on the level of block sections, used by the safety system of the railway. Therefore, all problems of railway scheduling, i.e., timetabling or rescheduling, addressed throughout this thesis are conducted for a microscopic representation of the railway network. The provided data further contains detailed information about the dynamic interactions of rolling stock and infrastructure. For every combination of a train, i.e., a particular combination of rolling stock material, and block section in the infrastructure, a specific minimal running time at the resolution of seconds is provided. Also, the data of SBB includes additional safety requirements, that are different from most of the literature. For each pair of train and block

section, a safety corridor is specified as chain of subsequent block sections, which must be free of other traffic, to allow for a safe emergency stop. Finally, data provided by SBB includes service aspects. For each train, operational requirements are given, which specify the stopping time and minimal transfer times to other trains at every station served by the train.

Validation

For the experiments of this thesis, a software tool was provided by SBB, to validate solutions (timetables) for their operational correctness. The software tool evaluates a timetable for physical correctness, safety and satisfaction of operational requirements. For all trains included in a timetable, scheduled events are validated for physical feasibility with regard to train dynamics on the particular traveled infrastructure. Furthermore, a timetable is validated for safety as emergency safety corridors are validated to exist for all trains on all traveled block sections. Finally, the timetable is validated for operational requirements. The timetable must satisfy minimal and maximal stopping times as well as minimal and maximal transfer times to meet the required level of service.

All computational results of this thesis have been evaluated and proven to be operationally correct through the software tool provided by SBB.

1.6 ORGANIZATION

The present thesis is organized in three main parts, plus an additional appendix. The main parts of this thesis are discussed over four chapters, that go along with four publications and the four research questions of Section 1.3. Figure 1.2 provides an overview of the organization, showing respective chapters and the corresponding research question of Section 1.3 answered in each chapter. Part II and Part III of the thesis are independent. Part II continues research results of Part I.

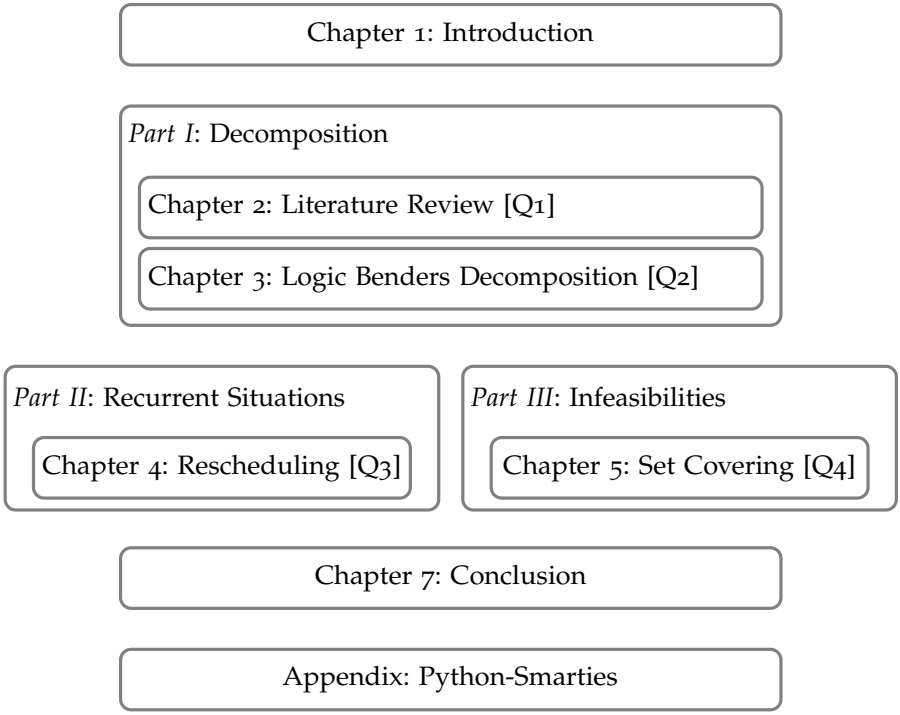

FIGURE 1.2: Overview of the structure of this thesis.

Part I: Decomposition

The largest part of research conducted in this thesis is devoted to a decomposition approach for the problem of railway timetabling. Results from Part I of this thesis are further extended in other parts of this thesis. The research in Part I is separated into two chapters, answering research question Q1 and Q2 from Section 1.3 respectively:

Chapter 2 answers research question Q1. In Chapter 2 an exhaustive review is conducted on the existing literature of decomposition approaches for railway scheduling (i.e., timetabling and rescheduling). In Chapter 2, we identify domains of decompositions in the literature and discuss corresponding advantages and disadvantages of different domains. We categorize and analyze different techniques of decomposition, and study properties of the resulting decomposed problem of railway scheduling. Finally, different methods are reviewed, which are used to address the

decomposed formulations of railway scheduling. In the review, we identify strengths and weaknesses of different decomposition approaches and point out potential research gaps, some of which motivated further research conducted in Chapter 3.

Chapter 3 answers research question Q2. In Chapter 3 a novel logic-based Benders decomposition is developed for the problem of railway timetabling. The problem of timetabling is geographically decomposed in a way, different from the existing literature, based on the ideas of a logic Benders decomposition. Theory for a novel logic Benders cut is developed to account for the special structure of the proposed decomposition. An exhaustive series of experiments empirically shows the benefits of the proposed decomposition, compared to existing approaches. With a second series of experiments, limitations of the proposed decomposition approach are exposed; computational benefits of the decomposition are shown to rely on particular structural properties of the underlying scheduling problem. Conventional approaches prevail over the novel decomposition if particular structural properties are absent in a problem of timetabling.

Part II: Recurrent Situations

Chapter 4 answers research question Q3. In Chapter 4 recurrent situations in the problem of railway rescheduling are exposed through methods of statistical learning. In this chapter, results from Chapter 3 are extended to the problem of real-time railway rescheduling. Methods of statistical learning expose patters in recurrent situations of railway rescheduling, which can be used to computationally accelerate the decomposition approach proposed in Chapter 3. With an enhanced decomposition approach it is possible to handle problems of rescheduling under strict time limitations, as it is the case for real-time railway traffic management (i.e. real-time rescheduling). With an exhaustive series of experiments, improvements and practical limitations of statistical learning in combination with a decomposition approach are shown.

Part III: Infeasibilities

Chapter 5 answers research question Q4. In Chapter 5 a computationally efficient approach for the problem of railway timetabling is proposed, based on infeasibilities of the problem. A series of incremental heuristics is introduced to address the problem of railway timetabling in a highly

efficient manner. A decomposition, different from the decomposition of Chapter 3, incrementally exposes infeasibilities in the problem of railway timetabling, to translate the optimization in timetabling into a problem of set covering. The detection of infeasibilities is based on implementations developed during the research of Chapter 3, i.e., Python-Smarties. Incremental heuristics are proposed to address the problem of set covering and provide a highly efficient method to compute near optimal solutions for the problem of railway timetabling. We prove the efficiency of incremental heuristics in a series of experiments and provide empirical evidence for the performance regarding computational time and solution quality of our novel approaches.

The thesis concludes in a discussion in Chapter 6, where achieved results of the thesis are critically assessed and put into perspective with practice.

Appendix

In the course of this thesis a Boolean Satisfiability Solver (Python-Smarties) has been developed alongside the theoretical contributions of the thesis. The solver provides the algorithmic backbone for all experiments conducted in the scope of this thesis. In the appendix of this thesis (Appendix A) we provide background information on concepts and methods from the literature, which have been combined into the implementation of Python-Smarties.

REFERENCES

1. Abbink, E., van den Berg, B., Kroon, L. & Salomon, M. Allocation of Railway Rolling Stock for Passenger Trains. *Cent. Discuss. Pap.* **43** (2002).
2. Abbink, E. J., Albino, L., Dollevoet, T., Huisman, D., Roussado, J. & Saldanha, R. L. Solving large scale crew scheduling problems in practice. *Public Transport* **3**, 149 (2011).
3. Adenso-Díaz, B., Oliva González, M. & González-Torre, P. On-line timetable re-scheduling in regional train services. *Transportation Research Part B Methodological* **33**, 387 (1999).
4. Alfieri, A., Groot, R., Kroon, L. & Schrijver, A. Efficient circulation of railway rolling stock. *Transportation Science* **40**, 378 (2006).

5. Bešinović, N., Goverde, R. M., Quaglietta, E. & Roberti, R. An integrated micro-macro approach to robust railway timetabling. *Transportation Research Part B Methodological* **87**, 14 (2016).

6. Borndörfer, R., Grötschel, M. & Jäger, U. in *Production Factor Mathematics* 95 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010).

7. Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M. & Schlechte, T. Recent success stories on integrated optimization of railway systems. *Transportation Research Part C Emerging Technologies* **74**, 196 (2017).

8. Borndörfer, R., Schlechte, T. & Swarat, E. *Railway track allocation - Simulation, Aggregation, and Optimization* in *Proceedings of the 1st International Workshop on High-Speed and Intercity Railways* (2012), 53.

9. Bussieck, M. R., Kreuzer, P. & Zimmermann, U. T. Optimal lines for railway systems. *European Journal of Operations Research* **96**, 54 (1997).

10. Bussieck, M. R., Lindner, T. & Lübbecke, M. E. A fast algorithm for near cost optimal line plans. *Mathematical Methods of Operations Research* **59**, 205 (2004).

11. Cacchiani, V., Caprara, A. & Toth, P. A Lagrangian heuristic for a train-unit assignment problem. *Discrete Applied Mathematics* **161**, 1707 (2013).

12. Caimi, G., Chudak, F., Fuchsberger, M., Laumanns, M. & Zenklusen, R. A New Resource-Constrained Multicommodity Flow Model for Conflict-Free Train Routing and Scheduling. *Transportation Science* **45**, 212 (2011).

13. Caimi, G., Burkolter, D., Herrmann, T., Chudak, F. & Laumanns, M. *Design of a railway scheduling model for dense services* in *Networks and Spatial Economics* **9** (2009), 25.

14. Caimi, G., Fuchsberger, M., Burkolter, D., Herrmann, T., Wüst, R. & Roos, S. Conflict-free train scheduling in a compensation zone exploiting the speed profile. *Proceedings ISROR RailZurich* **161**, 1 (2009).

15. Claessens, M. T., Van Dijk, N. M. & Zwaneveld, P. J. Cost optimal allocation of rail passenger lines. *European Journal of Operations Research* **110**, 474 (1998).

16. Conejo, A., Castillo, E., Mínguez, R. & García-Bertrand, R. *Decomposition Techniques in Mathematical Programming* (Springer, 2005).

17. Cordeau, J. F., Desaulniers, G., Lingaya, N., Soumis, F. & Desrosiers, J. Simultaneous locomotive and car assignment at VIA Rail Canada. *Transportation Research Part B Methodological* **35**, 767 (2001).

18. Corman, F. & Meng, L. A Review of Online Dynamic Models and Algorithms for Railway Traffic Management. *IEEE Transactions on Intelligent Transportation Systems* **9**, 1 (2014).

19. D'Ariano, A., Pacciarelli, D. & Pranzo, M. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operations Research* **183**, 643 (2007).

20. De Schutter, B. & Van Den Boom, T. *Model predictive control for railway networks* in *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings* **1** (2001), 105.

21. Dessouky, M. M., Lu, Q., Zhao, J. & Leachman, R. C. An exact solution procedure to determine the optimal dispatching times for complex rail networks. *IIE Transactions* **38**, 141 (2006).

22. Dollevoet, T., Huisman, D., Schmidt, M. & Schöbel, A. Delay Management with Rerouting of Passengers. *Transportation Science* **46**, 74 (2012).

23. Dorfman, M. J. & Medanic, J. Scheduling trains on a railway network using a discrete event model of railway traffic. *Transportation Research Part B Methodological* **38**, 81 (2004).

24. Fang, W., Yang, S. & Yao, X. A Survey on Problem Models and Solution Approaches to Rescheduling in Railway Networks. *IEEE Transactions on Intelligent Transportation Systems* **16**, 2997 (2015).

25. Fioole, P. J., Kroon, L., Maróti, G. & Schrijver, A. A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operations Research* **174**, 1281 (2006).

26. Freling, R., Lentink, R. M., Kroon, L. G. & Huisman, D. Shunting of Passenger Train Units in a Railway Station. *Transportation Science* **39**, 261 (2005).

27. Für Statistik, B. *Güterverkehr* `https://www.bfs.admin.ch/bfs/de/home/statistiken/mobilitaet-verkehr/gueterverkehr.html` (2022).

28. Für Statistik, B. *Pendlermobilität in der Schweiz 2019* `https://dam-api.bfs.admin.ch/hub/api/dam/assets/17164643/master` (2022).

29. Für Statistik, B. *Verkehrsleistungen im Personenverkehr* https://www.bfs.admin.ch/bfs/de/home/statistiken/mobilitaet-verkehr/querschnittsthemen/oeffentlicher-verkehr.html (2022).

30. Für Verkehr und digitale Infrastruktur, B. *Eisenbahngüterverkehr* https://www-genesis.destatis.de/genesis/online?sequenz=statistikTabellen&selectionname=46131#abreadcrumb (2022).

31. Für Verkehr und digitale Infrastruktur, B. *Verkehr in Zahlen 2020* https://www.bmvi.de/SharedDocs/DE/Publikationen/G/verkehr-in-zahlen-2020-pdf.pdf?__blob=publicationFile (2022).

32. Geoffrion, A. Generalized Benders Decomposition. *Journal of Optimization Theory and Applications* **10**, 237 (1972).

33. Goossens, J. W., van Hoesel, S. & Kroon, L. A branch-and-cut approach for solving railway line-planning problems. *Transportation Science* **38**, 379 (2004).

34. Herrigel, S., Laumanns, M., Nash, A. & Weidmann, U. Hierarchical Decomposition Methods for Periodic Railway Timetabling Problems. *Transportation Research Record* **2374**, 73 (2013).

35. Keita, K., Pellegrini, P. & Rodriguez, J. A three-step Benders decomposition for the real-time Railway Traffic Management Problem. *Journal of Rail Transport Planning & Management* **13**, 100170 (2020).

36. Kroon, L., Huisman, D., Abbink, E., Fioole, P., Fischetti, M., Maróti, G., Schrijver, A., Steenbeek, A. & Ybema, R. The New Dutch Timetable: The OR Revolution. *Interfaces* **39 (1)**, 6 (2009).

37. Kroon, L. & Fischetti, M. Crew scheduling for Netherlands railways "destination: customer". *Computer-aided scheduling of public transport*, 181 (2001).

38. Lamorgese, L. & Mannino, C. An Exact Decomposition Approach for the Real-Time Train Dispatching Problem. *Operations Research* **63**, 48 (2015).

39. Lamorgese, L., Mannino, C. & Natvig, E. An exact micro–macro approach to cyclic and non-cyclic train timetabling. *Omega (United Kingdom)* **72**, 59 (2017).

40. Lamorgese, L., Mannino, C. & Piacentini, M. Optimal Train Dispatching by Benders'-Like Reformulation. *Transportation Science* **50**, 910 (2016).

41. Lenze, W. & Nießen, N. Quality in rail timetabling - A detailed review of stakeholders' interests compared with current practices in Germany. *Journal of Rail Transport Planning & Management* **20**, 100282 (2021).

42. Luan, X., De Schutter, B., Meng, L. & Corman, F. Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. *Transportation Research Part B Methodological* **141**, 72 (2020).

43. Luan, X., Miao, J., Meng, L., Corman, F. & Lodewijks, G. Integrated optimization on train scheduling and preventive maintenance time slots planning. *Transportation Research Part C Emerging Technologies* **80**, 329 (2017).

44. Lusby, R. M., Larsen, J., Ehrgott, M. & Ryan, D. Railway track allocation: Models and methods. *OR Spectr.* **33**, 843 (2011).

45. Mascis, A. & Pacciarelli, D. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operations Research* **143**, 498 (2002).

46. Meng, L. & Zhou, X. Simultaneous train rerouting and rescheduling on an N-track network: A model reformulation with network-based cumulative flow variables. *Transportation Research Part B Methodological* **67**, 208 (2014).

47. Morgado, A., Heras, F., Liffiton, M., Planes, J. & Marques-Silva, J. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints* **18**, 478 (2013).

48. Odijk, M. A. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B Methodological* **30**, 455 (1996).

49. Palmqvist, C.-W., Olsson, N. O. & Winslott Hiselius, L. The Planners' Perspective on Train Timetable Errors in Sweden. *Journal of Advanced Transportation* **2018** (2018).

50. Peeters, M. & Kroon, L. Circulation of railway rolling stock: a branch-and-price approach. *Computers & Operations Research* **35**, 538 (2008).

51. Pellegrini, P., Marlière, G. & Rodriguez, J. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B Methodological* **59**, 58 (2014).

52. Pellegrini, P., Marlière, G. & Rodriguez, J. *Real time railway traffic management modeling track-circuits* in *ATOMOS 2012, 12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems* (2012), 12p.

53. Rodriguez, J. A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B Methodological* **41**, 231 (2007).

54. Schöbel, A. & Scholl, S. *Line planning with minimal traveling time* in *5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'05)* (2006).

55. Schrijver, A. Minimum Circulation of Railway Stock. *CWI-Quarterly* **6**, 205 (1993).

56. Tornquist, J. & Persson, J. A. *Train traffic deviation handling using tabu search and simulated annealing* in *Proceedings of the 38th annual Hawaii international conference on system sciences* (2005), 73a.

57. Van den Boom, T. J. J. & De Schutter, B. Modelling and control of discrete event systems using switching max-plus-linear systems. *Control Engineering Practice* **14**, 1199 (2006).

58. Voelcker, M. *Der Weg zur automatisierten Kapazitätsplanung und Steuerung* `https://www.mtrail.ch/pdf/EI_4_19_mtrail-saved.pdf` (2022).

59. Wang, Y., Zhang, Z., Huisman, D., D'Ariano, A. & Zhang, J. A Lagrangian Relaxation Approach Based on a Time-Space-State Network for Railway Crew Scheduling. *Computers & Industrial Engineering*, 108509 (2022).

60. Watson, R. Train planning in a fragmented railway-a British perspective (2008).

61. Zhang, J., Ding, G., Zou, Y., Qin, S. & Fu, J. Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing* **30**, 1809 (2019).

62. Zwaneveld, P. J., Kroon, L. G., Romeijn, E. H. & Salamon, M. Routing Trains Through Railway Stations: Model Formulation and Algorithms. *Transportation Science* **30**, 181 (1966).

2

A REVIEW OF PRINCIPLES AND METHODS TO DECOMPOSE LARGE-SCALE RAILWAY SCHEDULING PROBLEMS

Leutwiler, F., & Corman, F. (submitted). A Review of Principles and Methods to decompose Large-Scale Railway Scheduling Problems. Manuscript submitted to European Journal of Transportation and Logistics on 12.2021.

This is a pre-print (2nd revision) version, which is currently under revision in European Journal of Transportation and Logistics.

Key findings:

- The literature on decomposition in the railway sector can be categorized by the domain, principle and coordination method of decomposition.
- Particular suitable applications are identified for individual domains, principles and methods of decomposition.
- Important practical aspects in decomposition are discussed and research gaps are identified.

Author contributions: Review of literature: FL; Conduction of analysis and synthesis: FL; Manuscript preparation and review: FL, FC.

ABSTRACT

Providing punctual, reliable and performant services to customers is one main goals of railway network operators. The railway scheduling problem is to determine, ahead of time (timetabling), a plan describing the timing of the operations in a railway network, or updating such plan during operations (rescheduling). By optimization and automation, it is possible to operate more trains on the network, closer to the infrastructure capacity. Especially when the scale and complexity of the scheduling problem is increasing, for large-scale networks and multiple interconnected problems, this is of great value for network operators. When planning or adjusting railway operations becomes increasingly complex, modern scheduling algorithms can bring significant performance and economic benefits. In this survey we review approaches in the state of the art for the problems of railway scheduling. We show how the many different approaches of decomposition proposed in the literature of railway scheduling can be categorized into two general principles. We study different solution methods and identify a list of open topics for dealing with large-scale problems for future research.

2.1 INTRODUCTION

Railway operations are normally following a predetermined plan (timetable), which specifies times for arrival, departure and passing at stations. Such plan can be designed well ahead of time (timetabling) or adjusted during operations, when delays occur (rescheduling). When following the timetable, trains should not encounter congestion or result in unsafe situations with regards to infrastructure or other traffic on the network. The timetable is also the key to an efficient usage of railway resources and infrastructure. Careful planning of resource usage, especially for large-scale railway networks and dense traffic, enables a railway system to achieve high performance [30].

The timetable can be designed (offline) a few days up to months before actual operation, normally based on a given planned demand for railway services. The objectives are often to maximize the amount of services running, to fulfill the demand, to ensure a limited deviation from an ideal offer which could be politically determined. During operations, the timetable is adjusted, when delays occur, and real-time deviations need to be taken into account. The adjustment to unforeseen delays and disturbances is often targeting the minimization of deviations between the offline timetable and what is actually produced; or a quick recovery of the offline plan. This

problem is known as railway rescheduling. Both problems deal with the time allocation of railway services, considering scarce resources such as infrastructure capacity or vehicles. They identify choices in timing of services; their explicit relative order or sequence; and the route that trains follow in the network or at stations. The present paper considers both problems, under the unifying term of railway scheduling.

In railway scheduling, operations of trains are generally described by means of events. One event can represent the start or end of an operation. Typical operations are traversals of trains over the railway network, or dwell processes at stations. The problem can be considered at different levels of detail, resulting in two typical models, macroscopic and microscopic. In macroscopic railway scheduling, railway stations are abstracted into nodes, such that the network is represented in a set of nodes and lines. A single operation of a train in macroscopic modelling is the traversal of a train from one station to another. In microscopic railway scheduling, the railway network is considered in segments of few hundred meters of railway track, i.e., block sections, which provide enough precision in control, also for safety systems. A single operation of a train in microscopic modelling is the traversal of a single block section. The time of an event in railway scheduling is generally represented either in continuous or time-indexed form. In case of indexed time, often a time-space graph is used which connects points, i.e., events, in the indexed space according to operations. Points outside of the graph are neglected for scheduling. Extensive surveys on models of railway scheduling can be found, e.g., in [22], [11].

From a computational point of view, railway scheduling has been proven NP-Hard in many different versions of the problem (e.g., [47], [44], [14]). Scalability issues for large-scale problems are thus unavoidable and it is crucial to keep these issues under control through the design of novel methods. Current practice in the industry is to split up (often geographically) a large-scale scheduling problem and schedule separated parts individually. Manually or computer-aided, the local timetables are computed and afterwards merged together to a consistent (i.e., globally feasible) network-wide timetable. The merging process is in general suboptimal. Often such process is a kind of heuristic, rule-based process (e.g., high-speed trains have priority over suburban trains, first come first serve strategies etc.), where resource utilization potential is lost. Due to the heuristic merging, it is unavoidable that not all possible scheduling solutions are considered and evaluated.

Complementary to those suboptimal processes stands a continuously increasing demand in public transport and freight transport on railway

systems, which requires an increasingly higher level of performance from railways. While the acquisition of new network resources, e.g., infrastructure or vehicles, is a possible way to increase performance, it is in general very expensive, slow and with little flexibility. Many railway companies, e.g., the Swiss Federal Railways, instead focus on improving the utilization of existing resources. Railway operators show an increasing interest in novel methods, which solve the railway scheduling problem at an optimal level [2]. If large-scale practical instances could be solved to optimality, the existing resources could be used more efficiently, going beyond the shortcomings of today's practices and matching the potential already identified in academia, e.g. in [34].

A promising direction towards handling scalability issues of large-scale railway scheduling problems is the idea of decomposition. Similar to current practices, the problem is separated into smaller subproblems. Different from current practice, appropriate mathematical methods can reliably deliver good solutions, or even guarantee to find an optimal one. Decomposition has already been proven promising in numerous planning problems for railways (see e.g., [3], [34]). Still a gap between academia and industry exists. The approaches of academia have been evaluated on scenarios of increasing complexity up to 80 trains over 600 block sections [19], 150 train over 968 block sections [40] or 130 trains over 53 block sections [34]. Real life large-scale networks, like the one of Switzerland, operate more than 10000 trains per day over more than 5000 block sections. The need for novel methodologies to further scale up the complexity in automated railway scheduling becomes evident, and motivates our review on principles and solution methods of decomposition for railway scheduling.

In this review, we analyze the literature of decomposed railway scheduling and provide insights on the individual decomposition principles and solution methods, highlighting similarities and differences between different existing approaches in the literature. This review is thought to be interesting and useful for both researchers and practitioners. For researchers, we comprehensively analyze the railway scheduling problems in the literature, identifying which specific aspects and mathematical structures have been shown to be particularly suitable for decomposition. We also identify and motivate alternative mathematical descriptions of the problem. Moreover, we can identify combinations of mathematical models, decomposed according to some principles, and solution methods. This identifies some combinations which seem promising, even though have not been studied yet. For practitioners, we discuss how decompositions based on

physical characteristics of the problem have been more popular, possibly because they can be more directly understood. Nevertheless, there are other approaches possible, which can be identified. We believe our review of solution methods enables the discovery of new practical applications as we review advantages of different methods.

We believe, with a coherent discussion of mathematical models, decomposition principles, and solution methods, we can inspire further development from an academic point of view and motivate possible industrial application of those concepts. This review focuses on the problems of railway timetabling and rescheduling, i.e., respectively the design and adjustment of a timetable for the operation of a railway system. Many other problems are also relevant, in a supply of railway services: determining which infrastructure is needed, how to structure lines of the required services, which vehicles should be available, and for which services they should be used, how much personnel is required and on which services it should be driving. In the same spirit of the decomposition that this paper discusses, those problems are most often treated independently, due to different time scales, and different stakeholders and objectives. In this review, we assume that the input to a scheduling problem is given, from a solution to some problems upstream, and we can assume that the scheduling problem will influence other problems downstream. We will neglect upstream and downstream problems (such as line planning, timetable rolling stock and crew scheduling), and focus only on timetabling and rescheduling.

The remainder of this review is structured as follows. In Section 2.2 we review different domains of decomposition in the literature and analyze decompositions in the literature from a railway point of view. In Section 2.3 we discuss principles of decomposition to understand the decompositions of Section 2.2 from a mathematical point of view. In Section 2.4 we review solution methods of mathematical optimization from the literature to understand how a decomposed problem is addressed to determine a globally feasible (and optimal) solution. A discussion on strengths and weaknesses of different decompositions, but also on future research needs, is given in the final Section 2.5.

2.2 DECOMPOSITION IN RAILWAY SCHEDULING

Decomposition is a technique to solve mathematical problems, especially beneficial when dealing with large-scale instances, which generally show issues of scalability. A decomposition will operate on an original large-scale

problem, reformulating it into a set of different, smaller and possibly multiple problems. We call *decomposition principle*, the way an original problem is reformulated into multiple problems. Any of those problems can be solved with specific *solution methods*, which need to solve the problems themselves, but also harmonize their solution into something which satisfies the constraints of the original problem. The decomposition is the combination of a decomposition principle and a solution method.

We label a decomposition as *hierarchical*, if one problem has responsibility of the coordination, and results in one-to-many (vertical) communication to many other problems. This identifies a *master* problem with the responsibility of coordination (hierarchically higher) and (possibly) many *subproblems* (hierarchically lower). The subproblems do not need to be of the same mathematical type and structure, and most often, they are not. Moreover, the subproblems communicating with a single master can also have same, or different, mathematical type and structure. With a small abuse of notation, we might call subproblems the entire set of problems in a decomposition, or just those problems, that, compared to a master problem, are at the same lower level.

We label a decomposition as *decentralized*, if it results in many-to-many (horizontal) communication between problems, that are of the same mathematical type and structure. We call those problems to be at the same level.

The two types of decompositions are not exclusive and can appear both in some complex approaches. Moreover, boundary cases exists which can be hard to classify. Depending on specific characteristics, we can call the multiple problems of one level as subproblems (which highlights that they are constrained by some other problem), and some other as masters (which highlights that the problem is coordinating some other problems). If there are multiple levels, it can be that the subproblem for one master is actually itself a master for another subproblem. Finally, we call *centralized* the original, non-decomposed problem.

Decomposition is often motivated by special structural properties of the original problem, which lead the subproblem(s) to be a problem significantly easier to be solved than the original problem. For instance, it may generate, from one large problem, numerous smaller subproblems that are computationally much easier to solve (e.g., due to a superlinear relation between instance size and complexity). Decomposition is advantageous in case the computational benefits from solving the decomposed problems prevails over the additional efforts of coordinating them. We here identify

as typical the structure of a single master and multiple subproblems, but some approaches might be considered having a single subproblem; and/or no master problem.

A *coordination scheme* and a corresponding *solution method* is the way to solve the resulting problems of a decomposed reformulation. A coordination scheme is an unavoidable necessity in decomposition to achieve globally feasible (and optimal) solutions. Most solution methods are iterative schemes, where alternately master and subproblems are solved. Through the coordination scheme, solutions of the subproblems have an influence on the master problem to find feasible and optimal solutions. The combinations of principles to decompose and solution methods to coordinate result in the variety of decompositions we can find in the literature.

In this section we review decomposition approaches, looking at *domains* and aspects by which academic approaches have decomposed the problem of railway scheduling. Domains classify structures in the mathematical model of the scheduling problem, which are exposed for a decomposed reformulation of the problem. We use the concept of domains, to link structures of the mathematical model to physical or mathematical characteristics of the scheduling problem and identify possible advantages and disadvantages for different domains. In our review, we identified four domains of decomposition in the literature: the geographic, temporal, entity and generic domain. Within each domain, few aspects can be identified, based on which an original problem is separated into master problem and subproblems. While the geographic, temporal and entity domain have strong ties to the physical application of railways, we categorize a decomposition as generic if the related structure exploited has ties to general mathematical optimization. Still for most of the decompositions in the generic domain, a railway related interpretation can be made. In general, the geographic, temporal and entity domain can be seen as special cases of the generic domain. Table 2.1 summarizes the domains and aspects in the decompositions of the literature. We will show later in Section 2.3 how particular decompositions are achieved at a mathematical level.

2.2.1 *Geographic Domain*

In decomposition approaches of the geographic domain (GEO), the original (centralized) scheduling problem is separated such that different subproblems correspond to different geographic areas, based on the underlying structure of the railway network. Events in the same subproblem correspond

to operations which take place in the same area. The majority of geographic decompositions in the literature propose areas containing several stations, which generally happen to be of uniform size. The master problem in such decompositions considers the traffic in-between areas. A smaller number of geographic decompositions propose a decomposition by lines and stations. In this case, each individual station is defined as an individual area, corresponding to an individual subproblem; one large area contains what might remain of the entire railway network, defining the master problem.

2.2.2 *Temporal Domain*

In decomposition approaches of the temporal domain (TMP), the original scheduling problem is separated into subproblems, each of which covers a reduced time period, such that all together cover the temporal span of the original problem. The master problem in such decompositions considers the railway traffic at temporal boundaries. For a temporal decomposition it is necessary to know, for each event, a time window when it can take place, such that events can be assigned to a particular time period (i.e., a subproblem). In rescheduling, such assignment can be done using the original timetable. Reduced time periods may be overlapping in time, such as [56] or be disjoint such as [40]. The consequences of such choice will be analyzed in Section 2.3. Decompositions in the temporal domain are structurally very similar to decompositions in the geographic domain (see also Section 2.3). We can consider the concept of neighboring subproblems as those sharing some events or having events that are somehow constrained to events of each other. In this case, geographic decompositions have a variable number of neighbors, depending on the topology. Temporal decomposition instead have at most two neighboring subproblems (one earlier, one later).

2.2.3 *Entity Domain*

In decomposition approaches of the entity domain (ENT), the original scheduling problem is separated into subproblems, each of which considers the scheduling a single entity. Entities can be any arbitrary set of railway resources (e.g., a train, or an infrastructure element). The master problem in such decomposition contains the interactions amongst entities. Possible entities identified in the literature are individual trains as individual entities or a group of trains as a single entity; there exists also the case of a

Domain	Aspect	Publications
Geographic (GEO)	Geographic Areas	[19], [17], [18], [28] [48], [52], [40], [42] [54]
	Lines and Stations	[32], [34], [33]
	Junctions	[37]
Temporal (TMP)	Overlapping Periods	[56]
	Disjoint Periods	[40]
Entity (ENT)	Single Train	[51], [8], [50], [7] [41], [9], [10], [40] [6], [12], [13], [5] [49], [4], [21], [48] [46], [45], [29], [14] [23], [15]
	Group of Trains	[26], [38]
	Infrastructure	[4], [5]
Generic (GEN)	Ordering Binaries	[31], [27], [38], [28]
	Routing Binaries	[27], [38], [57], [20]
	Macroscopic Scheduling	[1]
	Duration of Operations	[53]

TABLE 2.1: Decompositions in Railway Scheduling.

decomposition where one entity is the set of all trains and a second entity is the set of all block sections in the network.

For the majority of decompositions in the domain of entities, individual trains are considered as individual entities. In this case, each train is scheduled in a single subproblem. Other decompositions in the domain of entities define a group of trains as a single entity. In this case, a single subproblem concerns the scheduling of all trains in the respective group. Grouping of trains can be conducted based on, e.g., priority [26] or train-type [38]. A

special case in the domain of entities are [4, 5] where exactly two entities are defined. One entity represents all trains of the scheduling problem and a second entity represents all block sections of the scheduling problem. In this decomposition, trains and infrastructure usage are optimized independently and the master problem is to assure the match between them. In such decomposition, trains are scheduled in one subproblem and their interaction (on block sections) is considered in another subproblem. Therefore, trains can be scheduled individually each for one subproblem; the consistency at the level of block section, including possible conflicts or penalties is performed by the other type of subproblem.

2.2.4 *Generic Domain*

In decomposition approaches of the generic domain (GEN), the original scheduling problem is separated into subproblems, based on the structure of the underlying mathematical optimization problem, not directly motivated by the physical railway system. In the generic domain, the literature shows exclusively decompositions, which exploit structures related to particular types or classes of variables. Generic decompositions exploiting structures in the constraints of a scheduling problem could not be found. In case of structures related to variables, subproblems are defined by means of variables to be optimized in the subproblem. Variables not in the subproblems are optimized in the master problem. Differently from the previous domains, where subproblems are scheduling problems and often numerous, in the generic domain it is possible that subproblems are not scheduling problems and only a single subproblem is considered.

As an interesting fact, all decompositions in the generic domain reviewed for this work, with the exception of [53], are based on time-continuous formulations. Furthermore, compared to other domains the generic domain is the most diverse domain, where subproblems related to different aspects substantially differ in problem-type and complexity. We discuss this aspect in more detail in Section 2.3.1.

2.3 PRINCIPLES OF DECOMPOSITION

From a perspective of mathematical optimization, a problem suitable for decomposition should have a special structure, which causes the decomposed problems to be significantly simpler than the original problem. If

such structure is absent, a decomposition is usually not beneficial with respect to computational speed.

Assuming a problem can be written as a (linear) mathematical optimization problem in standard form, i.e., $min\ c^T x\ \ s.t.\ \ Ax \geq b$, the structure of the constraint matrix A may highlight a possible decomposition. If such matrix has a block-diagonal structure, the individual blocks can be solved independently, leading naturally to a decomposition into multiple problems of smaller size. Often, problems that are suitable for a decomposition show an almost block-diagonal structure. That is, apart from a few entries in the constraint matrix, all others are within a block-diagonal structure. In this case, the problem can be reformulated, considering entries outside of the block-diagonal structure in the master problem and each block as an individual subproblem. The resulting subproblems, which are smaller in size can effectively be parallelized and are often much easier to solve (e.g., due to a superlinear relation between instance size and complexity).

In alternative to block-diagonality, a problem is suitable for decomposition if it can be separated into master and subproblem, such that the subproblem (even without a block-diagonality) becomes of a different problem type, that by itself is significantly simpler to solve (e.g., the subproblem belongs to P instead of NP as the original problem).

In both cases, a coordination scheme (used by a solution method) makes sure the individual solution(s) of master and subproblem(s) are modified accordingly, to result in a solution for the original (centralized) problem.

Based on those two special properties (natural decomposition into smaller problems; and easier class of problems), which can be exploited for a decomposition, we review in the following and more extensively the way the approaches in the state of the art use those properties, to result in effective decompositions. Two big classes of approaches exist: the principle of decomposing by *complicating variables* and the principle of decomposing by *complicating constraints* [16]. We review both in the subsections below with respect to the literature.

2.3.1 *Complicating Variables*

In the principle of complicating variables, the term "complicating variables" refers to a subset of variables in an optimization problem, which is known to cause a significant part of its computational complexity. If the complicating variables are set to constant value, the residual optimization problem over

Publication	Domain	Description	Interpretation (Master / Sub.)
[32], [34] [33], [36]	GEO	Variables of traffic between stations.	Scheduling between stations/ Scheduling a station.
[19], [18] [17], [52] [50], [28]	GEO	Variables of traffic between areas.	Scheduling between areas/ Scheduling an area.
[48]	GEO	Variables of traffic between areas only of a single train.	Scheduling a single train/ Scheduling an area.
[37]	GEO	Variables of traffic between areas of different junctions.	- /Scheduling a junction area.
[56]	TMP	Variables of traffic between time periods.	- /Scheduling a time period.
[26], [38]	ENT	Variables related to groups of trains.	- /Scheduling a group of trains.
[51], [7] [8], [49]	ENT	Variables related to individual trains.	- /Scheduling a single Train.
[1]	GEN	Variables related to macroscopic scheduling model.	Macroscopic scheduling/ Microscopic scheduling.
[31], [38] [27], [28]	GEN	Variables related to precedence of trains.	Ordering of trains/ Scheduling of trains.
[27], [38] [57], [20]	GEN	Variables related to routing decisions of trains.	Routing of trains/ Scheduling of trains.
[53]	GEN	Variables related to durations of operations.	Setting durations/ Scheduling with given durations.

TABLE 2.2: Complicating Variables in Decomposed Railway Scheduling.

the remaining variables is computationally significantly simpler, either due to a natural decomposition or an easier problem class, e.g. P instead of NP.

In a decomposition of complicating variables the optimization over the complicating variables is separated from the optimization of the remaining, i.e., non-complicating, variables. The non-complicating variables are optimized in a subproblem, while considering the complicating variables as fixed. The optimization over the complicating variables is done in the master problem. We summarize different identifications of complicating variables from the literature (together with a railway specific interpretation of master and subproblem) in Table 2.2 and report on the resulting decomposition in Table 2.3. In most publications, complicating variables have not been identi-

fied explicitly as such by the authors themselves. We instead systematically apply this concept of complicating variables, as we believe that it helps in bringing different decompositions to a common denominator. In Table 2.3, we report on the type of master and subproblem, where the type “Scheduling" refers to a scheduling problem (explicitly including ordering decisions), as such a mixed-integer linear problem. In case ordering is excluded, we denote it by “(LP)" to indicate the non-integrality of such problem. The types “Ordering" and “Routing" refer to integer linear problems; integrality is required to describe the ordering and routing decisions of railway scheduling respectively. Other types used in Table 2.3 are self-explanatory. The last column reports respectively the type of master and subproblem.

In the geographic domain, we see by Table 2.2 that such decompositions are achieved by fixing (i.e., determining as complicating) variables related to traffic passing the borders between different areas, stations or junctions. Complicating variables describe the sequence and timing of traffic over borders, therefore the constraint matrix shows a block-diagonal structure for non-complicating variables and multiple subproblems can be identified. We see such phenomena in Table 2.3 as all decompositions in the geographic domain show as many subproblems as areas, stations or junctions, which are in general independent. We further see in Table 2.3 multiple approaches with no master problem. In such cases complicating variables are assigned to subproblems; and subproblems then are dependent on each other. In Section 2.4, we will see that particular solution methods can be used, which result in independent solution processes for the subproblems, such that they can be solved in parallel. A special case is the approach of [50], where complicating variables are further divided into smaller groups, in particular one group for each train. The result are multiple master problems, as many as trains. Those problems are independent, because interactions among trains is handled in the subproblems, which consider the traffic inside different areas.

In the temporal domain, we see similar considerations as in the geographical domain. Instead of geographic areas, time periods are used to identify complicating variables (see Table 2.2); complicating variables are events between subsequent time periods. In our review of the literature we discovered only one related publication, where complicating variables describe traffic operating beyond the temporal boundary of two subsequent time periods. The single decomposition found keeps the complicating variables in the subproblem, such that subproblems remain dependent (see Table 2.3).

In the entity domain, we see decompositions where complicating variables are a subset of variables related to an entity. In particular, complicating

		Master Problem			Subproblem		
Publication	Domain	*N*	Indp	Type	*N*	Indp.	Type
[32], [34] [33]	GEO	1	-	Scheduling	# Stations	Yes	List Coloring
[36]	GEO	1	-	Scheduling	# Stations	Yes	Scheduling
[50], [28]	GEO	-	-	-	# Areas	No	Scheduling
[19], [18] [17], [52]	GEO	1	-	Scheduling	# Areas	Yes	Scheduling
[48]	GEO	# Trains	Yes	Shortest Path	# Areas	Yes	Scheduling
[37]	GEO	-	-	-	# Junctions	No	Scheduling
[56]	TMP	-	-	-	# Time Periods	No	Scheduling
[26], [38]	ENT	-	-	-	# Train Groups	No	Scheduling
[51], [49]	ENT	-	-	-	# Trains	No	Scheduling
[7], [8]	ENT	-	-	-	# Trains + # Stations	No	Scheduling
[1]	GEN	1	-	Scheduling	1	-	Scheduling (LP)
[31]	GEN	1	-	Ordering	1	-	Scheduling (LP)
[27]	GEN	2	No	Routing Ordering	1	-	Scheduling (LP)
[28]	GEN	-	-	-	# Train Groups	No	Scheduling
[38],	GEN	2	No	Ordering Routing	1	-	Scheduling
[57]	GEN	1	-	Scheduling with Routing	1	-	Scheduling with Maintenance
[20]	GEN	1	-	Routing	1	-	Scheduling
[53]	GEN	1	-	Scheduling Durations (LP)	1	-	Scheduling with fixed Durations

Indp.: Independent Problems.

TABLE 2.3: Decompositions of Complicating Variables in Railway Scheduling.

variables are those variables related to an entity, which appear in a constraint together with variables of another entity. Entities used for the identification of complicating variables are single trains or groups of trains (see Table 2.2). Depending on the entities used, different decompositions are reported in Table 2.3. In the literature, no decomposition on complicating variables in the entity domain has been found, where a master problem is used and sub-

problems are independent due to an entity related block-diagonal structure in the subproblem. In the literature, complicating variables for the entity domain are in general kept inside the subproblems, such that these remain dependent. We will see that few publications apply solution methods, which are able to treat such subproblems independently by appropriate heuristics (see Section 2.4). A possible reason for no exploitation of block-diagonality may be that complicating variables in the entity domain are in general numerous, leading to a large master problem and are usually hard to identify.

In the generic domain, we find four classes of complicating variables in the literature (see Table 2.2). In the literature, complicating variables are identified either related to macroscopic scheduling, ordering or routing of trains, or the duration of operations (see Table 2.2). Variables for the order of trains exclusively appear in continuous-time models, such that any decomposition exploiting such variables can only be applied to a continuous-time model of railway scheduling. Furthermore, depending on the particular railway scheduling problem addressed, e.g., including of routing, different decompositions are reported in Table 2.3. In case of complicating variables related to ordering and routing, the master problem is a pure integer linear program, to optimize only over the binary variables related to ordering or routing decision. In case of [1] the master is a macroscopic scheduling problem and in case of [53] the master is a scheduling problem with no binary decisions, i.e., a linear program (LP). We can see in Table 2.3 that all proposed decompositions in the generic domain exploit not a block-diagonal structure in the subproblem, but a simplification of the problem class, such that there is always a single subproblem. Exceptions are [53], where the subproblem cannot be simplified and [28] where complicating variables are kept in the subproblem and optimized in different groups (see Section 2.4). In two publications (i.e., [27], [38]), complicating variables have further been grouped into different sets. In both cases binary variables related to routing decisions and ordering decisions are addressed separately.

2.3.2 *Complicating Constraints*

In the principle of complicating constraints the term “complicating constraints" refers to subset of constraints in an optimization problem, which are the main cause for its computational complexity. If the complicating constraints are removed from the problem, the optimization over the remaining, non-complicating constraints is computationally, significantly simpler, either

Publication	Domain	Description	Interpretation (Master / Sub.)
[54], [40] [42]	GEO	Constraints in-between different areas.	Coordination of Areas/ Scheduling of an Area
[40]	TMP	Constraints in-between different time periods.	Coordination of Time Periods/ Scheduling of a Time Period
[6], [9] [14], [15]	ENT	Constraints of capacity between individual trains (AB-TSG).	Coordination of Trains/ Scheduling a single Train
[12], [13] [10]	ENT	Constraints of capacity between individual trains (PB-TSG).	Coordination of Trains/ Scheduling a single Train
[45], [41]	ENT	Constraints of capacity between individual trains (TI).	Coordination of Trains/ Scheduling a single Train
[21], [40]	ENT	Constraints of capacity between individual trains (TC).	Coordination of Trains/ Scheduling a single Train
[29], [46] [50]	ENT	Constraints of capacity between individual trains.	Coordination of Trains/ Scheduling a single Train
[4], [5]	ENT	Constraints of capacity between trains and block sections (PB-TSG).	Coordination of Blocks and Trains/ Scheduling of Blocks and Trains

(AB/PB)-TSG: Arc/Path-Based Time-Space Graph Model.
T(I/C): Time-Indexed/Continuous Model.

TABLE 2.4: Complicating Constraints in Decomposed Railway Scheduling.

due to a block-diagonal structure in the constraints and/or as the problem belongs to a different complexity class of problems, e.g., P instead of NP.

In the decomposition of complicating constraints, the optimization over the complicating constraints (in the master) is separated from the optimization over the non-complicating constraints (in the subproblem). We summarize complicating constraints identified in decompositions approaches of the literature in Table 2.4 and report on the corresponding decompositions in Table 2.5. Similar to the case of complicating variables, in most publications, complicating constraints have not been identified explicitly as such by the authors themselves. We instead systematically apply this concept of complicating constraints, as we believe that it helps in bringing different decompositions to a common denominator. In Table 2.5, we report on the type of master and subproblems in the decomposition, where we refer by “Penalty Update" to a linear optimization problem over penalty parameters

used to coordinate subproblems. We refer by “Set Packing" to an integer program with the basic structure of a set packing problem and possibly additional constraints, and by “Scheduling" to a mixed-integer problem, likewise as for Table 2.3. “Conflict Detection" and “Utility Evaluation" refer to an evaluation of solutions rather than to an optimization problem; the remaining types of problems in Table 2.5 are self-explanatory.

In the geographic domain, we can identify complicating constraints as the constraints between the different geographic areas (see Table 2.4). Areas might be defined differently in different publications, but all with the same goal of exploiting a block-diagonal structure leading to multiple subproblems. We can see the result of the block-diagonal structure in Table 2.5 as all subproblems in such decompositions are independent.

In the temporal domain, decompositions are very similar to decompositions of the geographic domain. The only difference to the geographic domain is that complicating constraints are not identified between different areas, but different time periods (see Table 2.4). As in the geographic domain, a block-diagonal structure is exposed, leading to independent subproblems (see Table 2.5).

In the entity domain, for all publications in the literature, we can exclusively identify constraints of network capacity as the complicating constraints. In this case, a train related block-diagonal structure is exposed; this leads to multiple smaller, independent subproblems, one for each individual train. In Table 2.4, we identify different complicating constraints related to network capacity, in relation to different underlying models of railway scheduling. In arc- and path-based models using a time-space graph, and also general time-indexed models, network capacity constraints result in clique constraints. Each clique is the set of arcs, paths, or time-indices, which would lead to a conflict on a single particular block section if used simultaneously in the timetable. In time-continuous models, network capacity constraints are often modeled as disjunctive precedence constraints between pairs of trains on each resource. The number of possible pairs for n trains is at most n^2, such that the amount of network capacity constraints in time-continuous models is in the worst case $n^2 * m$, where m is the number of resources. For time-indexed models, capacity constraints usually grow linearly with the number of resources, i.e., a single capacity constraint for each resource, in general resulting in smaller amount of capacity constraints. Special cases are the decompositions of [29, 46, 50]. In these approaches, network capacity constraints are not explicitly considered, but network

		Master Problem			Subproblem		
Publication	Domain	N	Indp.	Type	N	Indp.	Type
[54], [40] [42]	GEO	1	-	Penalty Update	# Areas	Yes	Scheduling
[40]	TMP	1	-	Penalty Update	# Time Periods	Yes	Scheduling
[6], [9] [14], [15]	ENT	1	-	Penalty Update	# Trains	Yes	Shortest Path
[12], [13]	ENT	1	-	Set Packing	# Trains	Yes	Trajectory Planning
[10]	ENT	1	-	Set Packing	# Trains	Yes	Shortest Path
[45]	ENT	1	-	Penalty Update	# Trains	Yes	Shortest Path*
[41]	ENT	1	-	Penalty Update	# Trains	Yes	Least-Cost Path*
[40]	ENT	1	-	Penalty Update	# Trains	Yes	Scheduling
[21]	ENT	1	-	Scheduling	# Trains	Yes	Utility Evaluation
[29], [46]	ENT	1	-	Conflict Detection	# Trains	Yes	Scheduling
[50]	ENT	#Areas	Yes	Conflict Detection	# Trains	Yes	Scheduling
[4], [5]	ENT	1	-	Penalty Update	#Trains + #Block Sections	Yes	Shortest Path

Indp.: Independent Problems.
*: Time dependent.

TABLE 2.5: Decompositions of Complicating Constraints in Railway Scheduling.

capacity is validated after the scheduling process of individual trains. We argue this is the same rationale as complicating constraints.

The entity domain is the most commonly used domain for decompositions based on complicating constraints. A possible reason to this might be the fact that in decompositions based on train entities on time-indexed formulations, subproblems are not only independent, but also reduce to shortest, least-cost or similar path-based problem (see Table 2.5), which in general can be solved extremely efficiently. In absence of a time-space graph, i.e., in general time-indexed models, a time-dependent shortest path problem results. Hybrid situations occur when the time-space graph is

updated iteratively according to possible train trajectories computed by a train dynamics model [12, 13]. A special case are the decompositions in [4] and [5], which are based on a reformulation of a time-space graph model. Additional variables are introduced to model the occupation of a block section by train, such that complicating constraints can be identified as those constraints between the variables of scheduling (events) and variables of occupation. The result are two subproblems, where one is to schedule all trains and another to schedule all occupations of block sections. In both these subproblems a block-diagonal structure is exposed and both problems naturally decompose into a shortest path problem for scheduling each train and each occupation of a block section individually.

The literature in decomposed railway scheduling does not show any decomposition based on complicating constraints in the generic domain.

2.4 SOLUTION METHODS IN DECOMPOSED RAILWAY SCHEDULING

Complementary to the principles of decomposition, different solution methods are used to solve and coordinate master and subproblems back to a solution, valid for the original (centralized) problem. In the following, we review solution methods considered in the literature of decomposed railway scheduling. In general we will differentiate between exact methods and heuristic methods.

In the literature, we can identify purely hierarchical solution methods; purely decentralized solution methods; and a small group which has both characteristics at once.

For decompositions showing a hierarchical aspect, problems of the decomposition (master and subproblems) are arranged in hierarchical levels. In general problems of the same type are (considered to be) on the same level. In hierarchical solution methods, problems on the same level are not coordinated directly with each other. A (master) problem on a hierarchically higher level can be identified, responsible for coordination (i.e., one-to-many (vertical) communication). For decompositions showing a decentralized aspect, in general the problems (often of the same type) are on a single level and are coordinated directly with each other (i.e., many-to-many (horizontal) communication). No specific problem within the decentralized structure can be identified as responsible for coordination of others.

The way of coordination directly determines which problems can be solved in a specific order, or can be solved at the same time, i.e. parallelized. In decentralized solution methods, the problems can be solved in arbitrary

order. Instead, in hierarchical solution methods problems of the same level can be solved independently (and parallelized), but different levels must be addressed in some specific (partial) order, e.g., a master problem has to be solved before/after all subproblems. In this sense, hierarchical solution methods are partially parallelizable, as the master must be solved only after all subproblems are solved, which means the total execution time depends on the computation time of both the master, and the slowest subproblem. Decentralized solution methods can achieve full parallelization.

We review purely hierarchical solution methods in Section 2.4.1, purely decentralized solution methods in Section 2.4.2, and we discuss in Section 2.8 solution methods which have a combination of both aspects. Furthermore, we review in Section 2.4.4 a small group of methods that are non-iterative and apply only unidirectional coordination over the problems of a decomposition.

2.4.1 *Purely Hierarchical Solution Methods*

In purely hierarchical solution methods, problems of a decomposition are solved in a hierarchy over multiple levels, where in general the master problem(s) of a decomposition are considered hierarchically higher than the subproblem(s).

In hierarchical solution methods, coordination is achieved exclusively via different levels of hierarchy to establish consistency over all problems of a decomposition. The coordination between hierarchical levels (vertical communication) is in general bidirectional; from higher to lower hierarchies and vice versa.

In hierarchical decomposition, the coordination from higher to lower levels imposes some aspect of the solutions of hierarchically higher problems onto problems of lower level. We find two different schemes for this in the literature, either enforcing by constraints, or guiding through adjusted penalties. For simplicity we refer respectively to those concepts as *strict imposition* or *soft imposition*. With the exception of [21], the strict imposition are a result of the principle of complicating variables and soft impositions are the result of the principle of complicating constraints. In solution methods applying strict impositions, solutions of higher levels affect directly the solution space of problems of lower levels. In other terms, solutions of higher hierarchy result in determining some values for variables, to be imposed: complicating variables are fixed in the subproblems. In solution methods applying a soft imposition, solutions of higher levels are consid-

Publications	Imposition	Coordination Scheme	Solution Method	Optimal
[32], [34] [33], [36]	Strict	Constraint Generation	Logic Benders	Yes
[31]	Strict	Constraint Generation	Classical Benders	Yes
[27]	Strict	Constraint Generation	Three-Layer Benders	Yes
[18]	Strict	Constraint Generation	Benders-Like	Yes
[1]	Strict	Constraint Generation	Constraint Adaptation	No
[5], [4], [10]	Soft	Column Generation	Dantzig-Wolfe	Yes
[12], [13]	Soft	Column Generation	Dantzig-Wolfe	No
[6], [41] [45], [54] [9]	Soft	Penalty Function	Lagrangian Relaxation	No
[14], [15]	Soft	Penalty Function	Relax-and-Cut	No
[42], [40]	Soft	Penalty Function	ADMM	No
[19], [17] [52], [57] [20], [28] [53]	Strict	Solution Passing	Multi-Shot	No
[21], [46]	Strict	Negotiation	Auction	No

TABLE 2.6: Hierarchical Solution Methods in decomposed in Railway Scheduling.

ered as parameters of a penalty term in the objective of problems of lower levels, guiding in this way their solutions process. Soft impositions are used in decompositions of complicating constraints to substitute complicating constraints through coordinating penalties.

Strict imposition has the advantage that solutions of the master problem and subproblem(s) are always consistent, building together a feasible solution for the original (centralized) problem. Therefore, once solutions for master and subproblems have been found, a solution for the original problem is given. As a drawback, strict imposition must consider the possibility that the constraints imposed on the subproblems by the master make them infeasible. Therefore, the solution process needs to handle infeasibility of subproblems (e.g., [18], [34], [36]). Moreover, it is often the case that no intermediate solution is available until a master solution has been found, for which also a feasible solution to all subproblems exists.

Soft imposition does not constrain subproblems. Therefore, subproblems are in general always feasible but not necessarily consistent with each other and thus not feasible with regard to the original (centralized) problem. As a consequence of soft impositions, especially for problems with integer variables (e.g., railway scheduling), it is often difficult to achieve consistency between subproblem solutions. In many publications, heuristics are used to recover a feasible solution for the original problem, from the inconsistent solutions of subproblems (e.g., [6], [14], [42]). As an advantage, these heuristics can also be used to determine a solution at any intermediate point of the solution process.

The coordination from lower levels to higher levels in hierarchical solution methods provides feedback from subproblem(s) to the master problem(s). For such coordination we can identify in the literature five types of coordination schemes: *constraint generation*, *column generation*, *penalty functions*, *solution passing* and *negotiation*.

In Table 2.6 we summarize hierarchical solution methods of the literature and report the type of imposition, coordination scheme, particular solution method and possible optimality of the method. We discuss the coordination schemes of the literature in more detail below.

2.4.1.1 *Coordination by Additional Constraints: Constraint Generation*

In the coordination scheme of constraint generation, subproblems report feedback to the master problem in form of additional constraints, which are not part of the original problem. In general, these constraints are incrementally generated and considered throughout iterations of sequentially solving master and subproblems. The generated constraints are meant to represent the feasible space and optimal solutions of subproblems, using only variables of the master problem. Constraint generation is exclusively used for decompositions based on complicating variables and thus all related methods use strict impositions.

Logic Benders [39] and classical Benders decomposition [25] are well-known schemes from the field of mathematical optimization. Benders decomposition is a constraint generation scheme with proven optimality. Generated constraints, i.e., Benders cuts, are based on proofs of infeasibility and suboptimality of subproblems, which can be used inside the master problem. While in classical Benders decomposition a clear procedure is given to generate new constraints, in logic Benders decomposition the actual procedure for constraint generation has to be designed individually for each specific application. The authors, which proposed such logic Benders de-

composition in [39] simply propose guidelines for the constraint generation, rather that an exact procedure. For the logic Benders decomposition in [32–34] the authors propose a novel constraint for their particular subproblem, which is to schedule traffic at a station. In [36], a logic Benders decomposition is used, where subproblems share the same mathematical structure with the master and the centralized problem. For the Benders decomposition, a novel type of constraint is introduced, based on the infeasibility of a general railway scheduling problem. Differently, [31] applies classical Benders decomposition to railway scheduling. In [27] a three-layer Benders decomposition has been introduced, where the master problem of a classical Benders decomposition is split into two layers. One master problem determines optimal routing in railway scheduling, and a subordinate master problem is to determine optimal precedences in railway scheduling. In [18] the authors designed constraints for a subproblem that is the scheduling of a geographic area, which has the same spirit of Benders decomposition as they address infeasibility and suboptimality in subproblems similarly. In [1] the master problem is a macroscopic scheduling problem, which includes macroscopic precedence constraints. These constraints are iteratively adapted based on the analysis of the subproblem, that is a microscopic scheduling problem.

Benders decomposition is particularly suitable for a decomposition, where the subproblems have large solutions spaces, that are rather independent from the solution space of the master problem. In this case only few additional constraints are necessary to represent the solution space of the subproblem, in the master; quick convergence can be expected. A railway related example is the decomposition of [34] where a subproblem relates to a station with many alternative routes for passing trains, all taking the same amount of time to pass. In such case, scheduling the trains outside the stations (which would be the master problem) is basically independent of routes used by trains in the station. A single constraint for the master problem would be sufficient to represent travel times of all routes in the station to the master.

2.4.1.2 *Coordination by Additional Solutions: Column Generation*

In the coordination scheme of column generation, subproblems report solutions to the master problem, which result in additional variables, i.e., columns in the constraint matrix of the master problem. Column generation can only be applied to decompositions in complicating constraints; and all related methods use soft impositions only. Dual values from a solution of the master problem are used for penalties in the subproblems (soft im-

position) to generate appropriate new subproblem solutions, i.e., columns in the master. Approaches of column generation typically work iteratively adding columns to the master problems based on the incumbent solutions of the subproblems. Those latter are guided through iterative updates of the penalty in their objective, based on the incumbent master solution. The only solution method used for column generation in all reviewed works is the Dantzig-Wolfe reformulation.

Dantzig-Wolfe reformulation [55] is an approach to reformulate the solution space of a mathematical optimization problem through its vertices, i.e., solutions at the boundary of the solution space. The optimization problem after a Dantzig-Wolfe reformulation is to find an optimal convex combination of vertices. The vertices are the columns of such optimization problem. The reformulation is in general addressed by a column generation, where new vertices are iteratively generated. In case of decomposition, the Dantzig-Wolfe reformulation is used to reformulate only the non-complicating constraints. In this case, generating new vertices (i.e., subproblem solutions) is then, per definition of complicating constraints, a significantly simpler problem. Further, if the non-complicating constraints inherit a block-diagonal structure, a single new vertex can be computed in parallel, over multiple independent subproblems.

In [4, 5, 10] a Dantzig-Wolfe reformulation is applied, where the column generation is carried out within a branch-and-bound scheme; these schemes are known as branch-and-price schemes, where pricing refers to the generation of new columns by appropriate penalties. Complicating constraints are capacity constraints, such that each column generated, that is a partial vertex and a solution of a subproblem, is the schedule of an individual train. Only with a branch-and-price scheme, the Dantzig-Wolfe reformulation and column generation is able to propose optimal solutions in case of mixed-integer programming, e.g., the railway scheduling problem.

In [12, 13], Dantzig-Wolfe reformulation is applied similarly, but column generation is performed in a heuristic manner. Therefore, it is possible that the columns, which are part of the optimal solution, are never generated; if this is the case, the optimal solution cannot be found when solving the master problem. The approaches in [12, 13] distinguish themselves from the literature, as they include train dynamics. Each column generated from a subproblem in such decomposition corresponds to a dynamically feasible train trajectory. [13] considers in comparison to [12] strengthened capacity constraints.

Column generation is effective if new columns, i.e., subproblem solutions, can be computed efficiently. In general, many different solutions from each subproblem (corresponding to columns in the master) are necessary, to make sure the master can find a combination of them that is an optimal solution to the original (centralized) problem. Therefore, it is crucial in column generation to compute subproblem solutions quickly. The most effective examples in railway scheduling are decompositions in the entity domain, where subproblem solutions are schedules of single trains and reduce to problems of shortest-path, which can be solved extremely efficient.

2.4.1.3 *Coordination by Objective Function: Penalty Functions*

In the coordination scheme of penalty functions, solutions of subproblems are considered as parameters of a penalty function in the objective of the master problem. In particular the inconsistency between solutions of subproblems (a mismatch in solutions of the subproblems, determining a global infeasibility) manifests as a penalty in the master problem. The master problem on the other hand is an optimization problem over variables used to parameterize penalty functions in the subproblems. This scheme is known as Lagrangian relaxation [24]. The penalty function in subproblems, parameterized by the master, directs the optimization of subproblems inside the respective feasible area, away from solutions potentially inconsistent with other subproblem solutions. Coordination by penalty functions can only be applied to decompositions in complicating constraints and is exclusively found in the literature, in the form of Lagrangian relaxation [24]; all methods of the coordination scheme of penalty functions use soft impositions only. Three different types are given: those applying the general scheme of Lagrangian relaxation, those using a relax-and-cut scheme, and those using ADMM (Alternating Direction Method of Multipliers).

In [6, 9, 41, 45] a classical Lagrangian relaxation is applied to the constraints of network capacity. For [6, 9, 45] the subproblem reduces to a (time-dependent) shortest path problem. In [41] maintenance constraints lead to a least-cost path problem. In all schemes above, the master problem computes a subgradient step for an update on variables used to parameterize penalty functions inside subproblems, based on the latest subproblem solutions. Often, e.g., in [6], subproblem solutions are first adapted to be consistent with each other (i.e., build a centralized solution), and then used for the subgradient step. These two steps of first ensuring feasibility, and then updating the penalty, increases the convergence amongst subproblem solutions. Still, the scheme of Lagrangian relaxation is known to converge

slowly; in fact, all related publications use some heuristic to compute a feasible upper bound solution at any intermediate point of the scheme. Most common is a heuristic where the penalty of a solution of the master problem is used to determine priorities to trains (subproblems); those are then, train by train, sequentially scheduled based on their priority. In [54] a Lagrangian relaxation has been proposed for a geographic decomposition. Complicating constraints at the border of geographic regions are relaxed. The solution scheme follows the standard Lagrangian relaxation scheme with a subgradient step to update penalty parameters (i.e., master variables). In case convergence cannot be achieved after a given number of iterations, a heuristic solution scheme is applied to coordinate the transitions of trains between different geographic areas.

In [14, 15], Lagrangian relaxation has been paired with a lazy constraints scheme into a relax-and-cut scheme. In this case, during the iterative adaptation of the penalty parameters (i.e. master variables), only complicating constraints that are violated by the latest subproblem solutions are considered. This allows to reduce the computation of the subgradient step in the master problem. Also here, heuristics are proposed to compute intermediate feasible solutions.

In [40, 42] the authors used the model of their previous publication, i.e., [41], to propose a decomposition in the geographic domain. To solve the decomposed problem, an alternating direction method of multipliers (ADMM) is applied. ADMM uses a Lagrangian relaxation of the complicating constraint to generate a penalty function for subproblems, but applies a different scheme to update the penalty parameters in the master. In particular, subproblems are solved sequentially; before the solution of each subproblem, the penalty parameters in the master are adapted based on the latest subproblem solutions. Like standard Lagrangian relaxation, ADMM has no guarantee to converge for problems of railway scheduling, and in fact a similar priority scheduling heuristics as in [14] has been used to compute intermediate feasible solutions. [40] extends the decomposition principle of [42] to different decompositions, and compares it to other heuristic coordination methods.

A coordination by penalty functions (Lagrangian relaxation) is most effective if subproblems can be solved efficiently. Often many updates of penalty parameters in the master problem are necessary to find consistent solutions for all subproblems and subproblems must be solved many times. The most effective examples in railway scheduling are, as for column generation, decompositions in the entity domain, where subproblem solutions

are schedules of single trains, extremely efficient to compute, by solving, e.g., a shortest-path problem.

2.4.1.4 *Coordination by Sharing Solutions: Solution Passing*

We denote as solution passing a coordination scheme in which subproblems report feedback by communicating (passing) their entire solutions to the master, where in the master, depending on the actual implementation of solution passing, some part of the reported solution(s) is fixed in its own optimization to coordinate with subproblems. Solution passing is exclusively applied to decompositions based on complicating variables; and all methods of solution passing use strict impositions. By our interpretation a single solution method has been identified in the literature. We name such method as multi-shot; multiple tries (shots) are undertaken to find a global solution. Each try is the sequence of first solving the master problem and subsequently all subproblems. The important part is, that each try is carried out with different initial conditions, i.e., a slightly different master problem. The initial conditions to be included in the master are the results of solutions of subproblems passed to the master problem from the previous try (shot), hence multi-shot in solution passing. Most often, one master problem and several subproblems are considered [17, 19, 20, 52, 57].

A special case on solution passing and multi-shot is the approach of [28]. Different from the approaches above, the decomposition of [28] has no clear master problem but only multiple dependent subproblems; subproblems are sequentially solved. The multi-shot characteristic in such approach is brought in, as after solving all subproblems in sequence, the sequence is resolved again from the beginning, but this time considering the solutions of the previous iteration.

A different special case is the approach of [53], where the decision at the master level is to determine the duration of all operations in the timetable; and a single subproblem evaluates if a feasible timetable exists for the durations determined by the master. At the master level, a multi-agent approach is used: each agent defines the duration of a single operation. We classify the scheme as solution passing, as agents adapt their solutions based on solutions of the subproblem.

All approaches of solution passing and multi-shot heuristically optimize different subsets of variables from the original scheduling problem in different problems of the decomposition. In [17, 19, 52] furthermore a block-diagonal structure is exploited. In general we understand the heuristics of this section to work best in cases where subproblems have large solution

spaces with little ties to (i.e., few variables fixed by) the master problem (or other subproblems). A large solution space increases the likelihood for a feasible solution in the subproblem while few variables fixed by the master solution decrease potential coordination effort.

2.4.1.5 *Coordination by Sharing Intentions: Negotiation*

In coordination schemes of negotiation, subproblems report feedback to the master problem in form of an intention. The intention represent the desire of a subproblem to take a certain solution, i.e., schedule. In an iterative process, the master problem allows or denies specific solutions based on previously shared intentions from subproblems. Based on the master's decision, subproblem update their intentions. In the literature we identified a single solution method in negotiation with purely hierarchical structure: auctions.

In auctions, subproblems report their intentions in form of bids. A bid represents the worth of a particular subproblem solution in perspective of the subproblem objective. The master problem acts as an auctioneer. The master receives all bids and then decides which subproblem wins the auction, and may take the solution the auction is held for. Auctions are carried out multiple times, till a solution for each subproblem has been found. Auctions may also consider only parts of the solution of a subproblem, e.g., particular departure or arrival times of trains. In general, all auction-based decomposition methods we discovered in the literature of railway scheduling are also agent-based. That is, each subproblem is represented by an agent which computes the bids, as well as the auctioneer is represented by an additional agent. Auction based methods belong to methods using strict impositions; the result of an auction is strictly enforced to the subproblems, such that only the winning subproblem is allowed for a particular schedule.

In [21] agents represent subproblems, each modelling a train; they bid on possible schedules provided by the master problem, i.e., the auctioneer. In [46] agents are trains bidding for particular departure or arrival times at stations.

Auction-based methods are a category of multi-agent methods where convergence to a feasible solution is guaranteed, through the structured interaction of auctioneers and bidding agents. In general, there is no guarantee that other methods based on multi-agents will converge, i.e., agree on a common consistent solution.

Publications	Imposition	Coordination Scheme	Solution Method	Optimal
[49], [37], [7] [8]	Strict	Negotiation	Multi-Agent	No

TABLE 2.7: Decentralized Solution Methods in decomposed Railway Scheduling.

2.4.2 *Purely Decentralized Solution Methods*

In purely decentralized solution methods, the master and/or subproblems can be solved in an arbitrary order. All problems which are at the same level in the decomposition can be coordinated in a direct exchange of information with each other (vertical communication). A recent, more extensive review on the topic of decentralized railway scheduling can be found in [43]. Such review includes also applications beyond railways, where decentralized solution methods have shown potential.

Different from hierarchical solution methods, in decentralized solution methods only methods using strict impositions are possible, where impositions are made on a single level of hierarchy. If otherwise soft impositions would be used, feasibility of a solution could not be guaranteed and centralized coordination would be necessary in some form, to recover a feasible solution for the original (centralized) problem (e.g., see [29]). As strict impositions are required, decentralized methods of the literature are exclusively applied to decompositions based on complicating variables. Decentralized solution methods are applied in particular to those decomposition in Table 2.3 where no master problem or multiple master problems are considered; hence vertical communication is required. In these cases, the solution methods enforce a certain degree of parallelism among the various subproblems.

Regarding the coordination schemes in decentralized decomposition, the literature shows only a single scheme, i.e., the scheme of negotiation, where in particular agents negotiate between each other. We summarize all decentralized approaches of the literature in Table 2.7.

2.4.2.1 *Coordination by Sharing Intentions: Negotiation*

The coordination scheme of negotiation is a coordination scheme that appears in both, hierarchical and decentralized decompositions. In hierarchical negotiation a master problem may only guide, reject, or restrict subproblem solutions in the process of negotiation. Instead, approaches in decentralized

negotiation have no master problem and subproblems negotiate directly, mutually restricting each other's solution by strict impositions. In the literature reviewed, all solution methods using negotiation on a decentralized decomposition are agent-based approaches. Agents negotiate by exchanging (details of) their (current best) solution.

In [7, 8, 37, 49] decentralized agent-based approaches are proposed. A full decentralization is possible due to strict impositions, i.e., in these approaches agents are not allowed to propose solutions with would conflict with solutions they received from other agents. In [7, 8, 49] agents represent trains. Each agent respects the solutions it has received from other agents, such that in case of a possible conflict, it determines for itself a different conflict-free solution accordingly. In [37] agents represent different geographic areas. Agents exchange entry and exit times of trains; each agent adapts its schedule according to exit and entry times of trains it receives from other agents. In general, the decision processes of agents above are designed to converge to a consensus in an iterative exchange and adaptation of solutions.

Decentralized agent-based approaches can be parallelized to a high extent, as each agent is almost independent from anything else in its execution. The parallelism can be implemented locally on a single machine or spread over multiple physical machines.

2.4.3 *Hierarchically Decentralized Solution Methods*

Hierarchically decentralized solutions methods have at the same time both aspect of hierarchical and decentralized decomposition. In these approaches, the problems of the decomposition show a hierarchical structure; within the hierarchical structure, some problems on the same hierarchical level are coordinated directly, in a decentralized manner.

In the literature, hierarchically decentralized solution methods have been found with both strict and soft impositions. For solution methods using strict impositions it is not necessary to have a single problem at the hierarchical top for coordination. Multiple problems can exist on the highest hierarchical level; these problems can be addressed in parallel. For solution methods using soft impositions it is necessary to have a single problem at the hierarchical top for coordination. We summarize hierarchically decentralized solution methods in Table 2.8.

2.4.3.1 *Coordination by Sharing Intentions: Negotiation*

In all the literature, hierarchically decentralized solution methods are either auction or multi-agent methods where agents negotiate towards a consistent solution.

Different from auction-based methods of Section 2.4.1.5, auction-based solution methods of this section have multiple auctioneers. In [48] agents represent trains, bidding at different auctions for entry and exit times into different areas of the network. Auctions are held for different geographic areas of the railway network. We consider the decomposition of [48] as a special case where neither auction agents, nor train agents communicate among agents of the same type. Rather global coordination is achieved as train agents participate at multiple auctions, i.e., the train agent assures consistency of a single train ride, and multiple train agents participate at a single auction, i.e., the auction guarantees no conflicts inside an area. In Table 2.3 we classified train scheduling as the master problems, and auctions as subproblems. In reality, there is no clear superiority of one the two of types of problems over the other, as train agents coordinate the result of different auctions but also auctions impose restrictions on train agents.

Different from the multi-agent methods of Section 2.4.2.1, in the multi-agent methods discussed in this section, agents are partially arranged in a hierarchy. The influence of decentralization is given as, on some level, problems are coordinated directly and vertical communication takes place. Either as on the highest hierarchical level multiple problems, i.e., agents, exist, which are coordinated directly. Or as on a lower hierarchical level, subproblems, i.e., agents, are coordinated directly with each other, partially without the input from hierarchically higher levels.

In [50] a multi-agent approach is proposed where agents represent geographic areas on a hierarchically higher level, and train agents on a hierarchically subordinate level. Train agents exclusively communicate with agents of areas, about available block sections for their operations. Agents of areas coordinate with each other to assure consistency for inter-area traffic.

In [29] a multi-agent approach is proposed where agents only represent trains plus a single coordination entity. Agents communicate directly to each other exchanging the incumbent best solutions, i.e., timetables. Train agents do not necessarily respect solutions received from other agents, but rather the received solutions are input to a penalty function, which is considered in the decision process of the receiving agent (soft impositions). Therefore, if the incentive for the agent is big enough, the agent may decide for a solution,

Publications	Impositions	Coordination Scheme	Solution Method	Optimal
[48]	Strict	Negotiation	Auction	No
[50]	Strict	Negotiation	Multi-Agent	No
[29]	Soft	Negotiation	Multi-Agent	No

TABLE 2.8: Hierarchically Decentralized Solution Methods in decomposed Railway Scheduling.

which could lead to a conflict with another agents. As a consequence, a final centralized recovery step is required, to adjust solutions of train agents, i.e., subproblems, for feasibility. The recovery step is a hierarchical element in the otherwise decentralized solution method. Building a schedule by a centralized (recovery) agent assures that no train agent may plan operations, which would lead to conflicts or deadlocks in the system.

Hierarchically decentralized solution methods combine the benefits of both hierarchical and decentralized methods. Ideally, such solution methods result in multiple problems equal in type (suitable for decentralization), where each subproblem is rather difficult to solve. Then, it makes sense to solve these individual problems by a second decomposition, i.e., a using a hierarchical solution method.

2.4.4 *No Coordination*

In our review of the literature, we also identify several decompositions with solution methods where no explicit coordination scheme is applied. We classify as single-shot approaches with no coordination. That is, after a sequence of all problems of a decomposition is solved, either a feasible solution is found, or it is required that the fundamentals of the scheduling problem are changed, either rule-based or by human interaction. No retry is undertaken using different initial conditions and we may consider these solution methods as non-iterative.

In the literature all solution methods of no coordination apply strict impositions, such that if a solution is found after a single shot, such solution is guaranteed to be feasible regarding the original (centralized) problem. As such all these methods are decompositions of complicating variables (see Table 2.3). Furthermore no solution method in this class proposes an

Publications	Imposition	Coordination Scheme	Solution Method	Optimal
[51], [26], [38] [56]	Strict	None	Single-Shot	No

TABLE 2.9: Non-coordinating Solution Methods in decomposed Railway Scheduling.

explicit master problem to exploit block-diagonality. Rather all subproblems are dependent on each other and solved sequentially.

In approaches of [26, 38, 51] different subproblems describe different trains or groups of trains. In [56] different subproblems relate to different time periods. An overview is given in Table 2.9.

Solution methods with no coordination exploit exclusively the super-linear decrease of complexity for decreasing problem size. Being each subproblem significantly smaller than the original problem, the sequence of subproblems is in general solved faster than the original problem. Despite the computational speedup, methods of no coordination are not guaranteed to provide any solution.

2.5 DISCUSSION

In this paper, we reviewed the academic literature discussing decomposed railway scheduling. We included early approaches, following mostly methods of mathematical optimization, but also more novel approaches based on data-driven solutions or agents.

We identified mathematical principles for decomposition, which depend on the structure of the mathematical models. A complementary analysis focused on solution methods; two properties have been identified, the presence of a hierarchy in the solution process, and the presence of decentralized communication within subproblems at the same level. For each type of structure, we identified different ways by which the decomposed problems are coordinated towards globally feasible and (possibly) optimal solutions. Here, we take a step back and reflect on the strength and potential for specific approaches.

Domain	Principle	# Complicating Elements	Reduction of Complexity
Geographic	CV	$\mathcal{O}$(Traffic between Areas)	Block-Diagonal
Geographic	CC	$\mathcal{O}$(Traffic between Areas)	Block-Diagonal
Temporal	CV	$\mathcal{O}$(Traffic between Time Periods)	Block-Diagonal
Temporal	CC	$\mathcal{O}$(Traffic between Time Periods)	Block-Diagonal
Entity	CV	$\mathcal{O}$(Single Entity)	Block-Diagonal
Entity	CC	$\mathcal{O}$(Capacity Constraints)	Block-Diagonal (& Problem Class)
Generic	CV	$\mathcal{O}$(Decisions)	Problem Class

TABLE 2.10: Characteristics of Decompositions in different Domains

2.5.1 *Decomposition in Different Domains*

Different decomposition domains often result in different sizes of master and subproblem, different complexities of these problems and also determine whether the subproblem(s) shows a block-diagonal structure. In the ideal decomposition, the complexity of the master and subproblem(s) is as low as possible. For the master problem, the complexity often directly relates to the amount of complicating variables or complicating constraints in a decomposition, i.e., the amount of variables or constraints in the master. For subproblem complexity, there is no similar easy pattern. We discussed ways of decomposition by which we may expose a block-diagonal structure in the subproblem or fundamentally change the class of complexity of the subproblem. The size alone does not determine the complexity of a subproblem, but rather the complexity strongly depends on the domain and principle of decomposition. It is therefore important to choose the right domain and principle for a specific problem at hand, to result in a performant decomposition approach. The ideal decomposition should have a small amount of complicating elements (variables or constraints), and the subproblems should display a block-diagonal structure, where blocks propose an optimization problem of an easier class than the original problem. We report in Table 2.10 on the characteristics of decompositions in different domains, as seen in the literature. We report for domain and principle an order of magnitude of the computational complexity as depending on complicating elements and for related subproblems; and how complexity is reduced, e.g. through a block-diagonal structure or a computational complexity problem class (e.g., P instead of NP).

Geographic and temporal decompositions are promising, under the aspects of small sized master problem and block-diagonality in subproblems, but subproblems remain in the same complexity class as the original problem. The number of complicating elements relates to the amount of railway traffic between different geographic areas or time periods, and is usually much lower for those two domains, than for the entity or generic domain (e.g., see [40]); moreover the subproblems show in general a block-diagonal structure. For temporal decompositions, one has to be careful when separating the original time horizon into smaller periods. As [40] shows, if arbitrary periods are chosen (regular in time against a very variable traffic, or irregular in time against a rather constant traffic), a temporal decomposition can result in large numbers of complicating elements, e.g., more than for a decomposition in the entity domain. A possible solution for temporal decompositions can be to adjust the length of time periods according to traffic density on the network over time. Choosing time periods for the decomposition dynamically, according to traffic density over time, we can minimize the amount of traffic operating in more than one time period and thus minimize ties (complicating elements) between different time periods. An interesting open challenge is to exploit recurrent patterns in both decompositions, for instance based on periodic timetables, to simplify the solutions. For instance, Benders cuts can be computed for a recurrent service pattern and then propagated to all relevant subproblems.

The entity domain is by far the most explored domain in decomposed railway scheduling. The decomposition with train entities shows in general very beneficial properties with many very small subproblem due to block-diagonality and moreover subproblems of a complexity class usually extremely simple (e.g., shortest path problems). Decompositions in the entity domain have also drawbacks, which include a very large number of complicating elements. In decompositions based on complicating constraints, capacity constraints are in general identified as complicating; those are especially numerous in dense traffic situation, resulting in increased coordination burden. Furthermore, the entity domain shows only few decompositions by means of complicating variables. We see here a big potential for novel approaches. If entities are trains or block sections (e.g., [4]), a specific analysis may identify the potential conflicts amongst entities and identify those entities with many conflicts, such that variables related to those entities can be isolated as complicating variables. A possible result can be a partition of the problem in a complexity core; and a set of

non-complicating entities, which have only little or almost no interference with each other, and can be therefore scheduled independently.

The generic domain results in very different approaches concerning the amount of complicating elements (variables) and the type of subproblem, in comparison to other domains. Often all or a subset of discrete decisions are identified as complicating. An example such as [31] shows, that if a decomposition is appropriately chosen, the structures in the railway scheduling problem can be exposed, which lead to impressive speed-ups. An open challenge in the generic domain is how to determine an appropriate choice of complicating variables to expose structures; several cases have already been reported, where exposed structure lead to a significantly reduced complexity of the subproblem. To this end, data driven methods based on clustering or other unsupervised learning, as well as domain transformation, can help solve the problem faster. Further, we did not come across any generic decomposition based on complicating constraints. It is important thus to analyze a railway scheduling problem for complicating constraints, whose removal can lead to a decomposition with favorable properties, i.e., few complicating constraints, block-diagonal or simple subproblems. In this sense, graph representations of the problem might be used to analyze connected components, and/or by flow or cut approaches.

In conclusion, no domain used for decomposition seems to outperform the other in terms of general potential. The specific characteristics of the railway scheduling problem to be solved can strongly suggest some or other approaches, depending on which variables and constraints it imposes at global and local level. We expect decompositions with a well balanced complexity between master and subproblems as favorable for large-scale applications.

2.5.2 *Important Aspects of Decompositions*

In the following we discuss a few important requirements and potentials for decompositions.

Parallelization. Computer architectures today are designed for efficient parallel computing, especially modern GPU's contain a very high number of cores, designed for highly parallelized computing. Still, whether such potential can be explored depends on the problem to be solved by the computer. With regards to parallel computing, decentralized solution methods (Section 2.4.2) have a clear advantage over hierarchical solution methods (Section 2.4.1). Hierarchical structures can also be parallelized, but the master (or single entity to which all subproblems communicate) must be run at once,

and after all subproblems have been executed. Instead, every decentralized structure features many subproblems that can be run in parallel, lacking the possible bottleneck of a single master problem. An important issue in the parallelization is of course the communication, which should be fast, with high bandwidth (in case of geographically dispersed subproblems) and pertinent, i.e., exchanging what is needed to achieve consistency, but not much more. Instead, for hierarchical solution methods, we can revisit Table 2.3 and 2.5 to identify cases of multiple independent subproblems. Those are in general very effective for parallel computing. In hierarchical solution methods the bottleneck remains the slowest subproblem in each iteration, as for each solution iteration, normally all subproblems must be solved. This would suggest to balance the complexity of all subproblems (rather than the amount of sections, length of time interval, or trains).

Dedicated Hardware. In case the task for a computer is always the same in terms of operations to be done, the usage of dedicated hardware such as GPU's or FPGA's can lead to substantial speedup in computation. Revisiting Table 2.3 and 2.5 to see which problems are faced during the different decompositions, we see two possible applications for dedicated hardware: In most publications reviewed on train-entity based decompositions, the subproblems reduce to problems of shortest path. Such problems are strongly structured and solved by efficient algorithms, which repeat only few operations many times. As such, these decompositions are very suitable for being solved via dedicated hardware. The second use case for dedicated hardware relates to agent-based approaches. Agents in such approaches are in general rule based, or taking actions based on machine learning algorithms. Both implementations require for the computation of an agents decision, several matrix multiplications, for which GPU's have proven extremely effective.

Explainability of Solutions. An important aspect in industrial applications is that the experts accept the computed solutions. Here, in general decomposition approaches could bring a benefit. Since solutions for the original problem are the assembly of different subproblem solutions, experts have the possibility to study also the subproblem solutions, and get more insight in the determination of a particular solution. Moreover, decomposition domains rooted in the real life problem (for instance based on geography, or time) are commonly used in real life, thus easier to understand by most operators. This could help for the acceptance of novel decomposition methods.

Fast Availability. An important aspect with respect to industrial applications is the fast availability of a first good solution, or even better, being extremely fast in determining the optimal solutions. There, a clear dif-

ference exists between decompositions of complicating constraints, and complicating variables. In decompositions of complicating variables, especially in the exact approaches, a subproblem determines a feasible and optimal solution given the latest master solution. It is likely that many iterations are necessary before a first feasible solution for a subproblem is encountered, making it hard to have a intermediate feasible solutions available quickly. The extreme case is that the first time a feasible solution for all subproblems has been found, is also the end of the algorithm, as it is the optimal solution (see e.g., [34]). In decompositions of complicating constraints, subproblems often report some possible solutions. Either one can find an intermediate feasible solution for the original problem based on given subproblem solutions, or one of the many heuristics in the literature can determine intermediate feasible, yet not optimal, solutions. This could be an advantage in real-time truncated solution processes.

Railway specific vs. Generic. Looking at Table 2.3 and 2.5 we see that many decompositions are inspired by traditional elements of railway, e.g., areas, junctions or trains. We see great potential in getting away from railway specific concepts and looking at the problem of railway scheduling in a more generic way. In other terms, studying and understanding the problem not by its physical aspects, but by exploiting its mathematical structure. An example of this idea is [31], where the variables in the mathematical problem are decomposed only by the type of variables, and not by domain understanding. The paper proposes to separate all integer from all non-integer variables, under a standard Benders decomposition approach, and reports a significant speedup. Generic decomposition approaches are a great opportunity to use the variety of generic tools which are increasingly made available in mathematical optimization.

Indexing, or continuous Relaxation. Many of the decompositions reviewed in this work rely for their decomposition on a time-indexed formulation of the railway scheduling problem. These formulations generally suffer under a more detailed time granularity, as the size of the solution space depends on the amount of time indices. In other terms, if operators desire timetables with high accuracy, i.e., a fine discretization of time, this becomes a big disadvantage for such models and decompositions. This mainly concerns decompositions where capacity constraints are identified as complicating constraints, as these decompose in time-indexed form. Continuous formulations suffer from a more detailed granularity only for numerical approximations, e.g., concerns of floating-point arithmetic, which can be needed in case of non linear constraints or objective functions. In contrast,

continuous formulations in general include big-M constraints to model the discrete decisions of a railway scheduling problem. These constraints often show a poor linear relaxation. Instead, in general good (tight) linear relaxations are crucial for the efficiency of many algorithms that are used to solve mixed-integer problems (railway scheduling).

Separability of objective function terms, extension to multi objective optimization. One important factor we found in most of the decompositions we reviewed, is that the objective should be separable in terms of variables, i.e., does not contain any bi-linear or similar terms. If this is not the case, a separation into master and subproblem is often not possible, or possible only partially. In fact, it must be ensured that the variables, which appear together in a nonlinear term in the objective, have to be in the same problem. This can be either any of master or subproblem.

No decomposition approach reviewed deals with multi-objective optimization. Nevertheless, we believe that many decomposition approaches are easily extendable and some even as particularly beneficial, compared to the non-decomposed problems. In multi-objective optimization, where different objectives are optimized sequentially, e.g., when using methods of column or constraint generation, the columns/constraints generated during the optimization over one objective could partially or maybe even fully be reused in the optimization of other objectives.

2.5.3 *Current Research Gaps, and Directions for Future Research*

The literature of decomposed railway scheduling shows many different promising decompositions. Nonetheless as shown in the introduction of this paper, a gap is still existing between the size of instances addressed in academia, and the size of instances, which the operators need to handle in real life, especially in microscopic scheduling. Most decompositions we reviewed, sacrifice to some extent optimality for the sake of speed. On the contrary, it is the main goal of railway operations to increase the capacity of their railway network through highly effective timetables. As such there needs to be a dialog between academia and industry how much optimality may be sacrificed.

Future research must address the issues of today's existing methods, which include the following: A clear issue amongst all works reviewed is the lack of studies on benchmark large-scale instances of railway scheduling. Many papers conclude stating that larger instances have been tackled by decomposition, compared to existing centralized approaches, but no exhaus-

tive experiments have been conducted on openly available actual large-scale instances. Several papers state the issue, that in case of a non-optimal approach, it is extremely difficult to estimate the quality, or optimality gap, of an intermediate solution. It thus can hardly be estimated if it is worth to continue the computation or stop with the incumbent best solution. A number of publications, especially in column generation approaches, denote the issue of RAM as a limiting factor towards large-scale instances. RAM could be expanded, but this comes still at a high resource cost. Agent-based approaches have been mostly tested and especially been trained only on medium sized instances (i.e., 15 trains over 4 areas [48], 20 trains over 20 station [49], 48 trains over 2 junctions [37]). To prove a practical applicability of agent-based approaches on larger instances (e.g., 150 trains and 1000 block sections as in [40]) further studies are necessary with an increased amount of agents, and associated complexity. The amount of data (either from recorded operations, or from simulation, or both) necessary to train the agents in case of large-scale instances becomes an additional, significant factor of complexity.

Furthermore, many agent-based approaches reviewed in this work rely on more or less complex rules to guide the underlying decision process. The possibility to automatically learn such rules based on the instance and the realization is very attractive: [29] and subsequently [8] use techniques from machine learning, in particular q-learning, to provide a decision process for agents. It is an interesting and yet open question whether other methods in machine learning, ranging from simple regression to sophisticated neural networks, can lead to an improved solution quality in railway scheduling. Here, the availability of benchmarks (such as the ongoing research at Swiss Federal Railways - SBB [35]) allows collaborative investigation and benchmarks of specific approaches, which are typically only solved for specialized types of problems.

In a different direction, we see the integration of Benders decomposition very promising. First, all publications using such an approach are published within the last 6 years, which identifies it as a promising new idea. Also as we discussed earlier, especially for logic Benders decomposition, the design of appropriate constraints can be particularly elaborated to outperform what can be achieved by following a simple procedure. In this direction, the seminal paper [39] proposes only a framework and the necessary requirements for the validity of appropriate constraints, but leaves the problem open, on how to design the constraints for different applications

such as railway scheduling. We believe there is large potential for further improvement beyond the ideas of [32] and [36].

Finally, we see great potential in more generic decompositions. Most of the decomposition approaches we reviewed in this work propose decompositions based on ideas backed with a physical interpretation. It is relatively easy to interpret geographic areas, time periods and the entity of a train. Instead, generic decompositions can be less easy to be interpreted, but might have computational advantages. In this case, specific subset of complicating variables or complicating constraints can be determined, that fit very simple structures, or balance the load of the subproblems. In this sense, generic decompositions might exploit approximate descriptions of complexity, and use the variables and constraints of each subproblem (i.e., its complexity) as a hyper-parameter to be determined. We believe it is possible to achieve in such way decompositions closer to the ideal of a small master problem and simple independent, numerous subproblems.

REFERENCES

1. Bešinović, N., Goverde, R. M., Quaglietta, E. & Roberti, R. An integrated micro-macro approach to robust railway timetabling. *Transportation Research Part B Methodological* **87**, 14 (2016).
2. Borndörfer, R., Grötschel, M. & Jäger, U. in *Production Factor Mathematics* 95 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010).
3. Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M. & Schlechte, T. Recent success stories on integrated optimization of railway systems. *Transportation Research Part C Emerging Technologies* **74**, 196 (2017).
4. Borndörfer, R. & Schlechte, T. *Models for Railway Track Allocation* in *7th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'07)* (2007).
5. Borndörfer, R., Schlechte, T. & Weider, S. Railway track allocation by rapid branching (2012).
6. Brännlund, U., Lindberg, P. O., Nõu, A. & Nilsson, J.-E. Railway Timetabling Using Lagrangian Relaxation. *Transportation Science* **32**, 358 (1998).

7. Bretas, A., Mendes, A., Chalup, S., Jackson, M., Clement, R. & Sanhueza, C. *Modelling railway traffic management through multi-agent systems and reinforcement learning* in *In Elsawah, S.(ed.) MODSIM2019, 23rd International Congress on Modelling and Simulation* (2019), 291.
8. Bretas, A. M., Mendes, A., Jackson, M., Clement, R., Sanhueza, C. & Chalup, S. A decentralised multi-agent system for rail freight traffic management. *Annals of Operations Research* (2021).
9. Cacchiani, V., Caprara, A. & Fischetti, M. A lagrangian heuristic for robustness, with an application to train timetabling. *Transportation Science* **46**, 124 (2012).
10. Cacchiani, V., Caprara, A. & Toth, P. A column generation approach to train timetabling on a corridor. *4OR* **6**, 125 (2008).
11. Cacchiani, V. & Toth, P. Nominal and robust train timetabling problems. *European Journal of Operations Research* **219**, 727 (2012).
12. Caimi, G., Fuchsberger, M., Burkolter, D., Herrmann, T., Wüst, R. & Roos, S. Conflict-free train scheduling in a compensation zone exploiting the speed profile. *Proceedings ISROR RailZurich* **161**, 1 (2009).
13. Caimi, G., Fuchsberger, M., Laumanns, M. & Lüthi, M. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. *Computers & Operations Research* **39**, 2578 (2012).
14. Caprara, A., Fischetti, M. & Toth, P. Modeling and Solving the Train Timetabling Problem. *Operations Research* **50**, 851 (2002).
15. Caprara, A., Monaci, M., Toth, P. & Guida, P. L. A Lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics* **154**, 738 (2006).
16. Conejo, A., Castillo, E., Mínguez, R. & García-Bertrand, R. *Decomposition Techniques in Mathematical Programming* (Springer, 2005).
17. Corman, F., D'Ariano, A., Pacciarelli, D. & Pranzo, M. Dispatching and coordination in multi-area railway traffic management. *Computers & Operations Research* **44**, 146 (2014).
18. Corman, F., D'Ariano, A., Pacciarelli, D. & Pranzo, M. Optimal inter-area coordination of train rescheduling decisions. *Transportation Research Part E Logistics and Transportation Review* **48**, 71 (2012).
19. Corman, F., D'Ariano, A., Pacciarelli, D. & Pranzo, M. Centralized versus distributed systems to reschedule trains in two dispatching areas. *Public Transport* **2**, 219 (2010).

20. D'Ariano, A., Corman, F., Pacciarelli, D. & Pranzo, M. Reordering and local rerouting strategies to manage train traffic in real time. *Transportation Science* **42**, 405 (2008).

21. D'Ariano, A. & Hemelrijk, R. Designing a multi-agent system for cooperative train dispatching. *IFAC Proceedings Volumes* **12** (2006).

22. Fang, W., Yang, S. & Yao, X. A Survey on Problem Models and Solution Approaches to Rescheduling in Railway Networks. *IEEE Transactions on Intelligent Transportation Systems* **16**, 2997 (2015).

23. Fay, A. Decentralized Railway Control Based on Autonomous Agents. *IFAC Proceedings Volumes* **33**, 83 (2000).

24. Fisher, M. L. An applications oriented guide to Lagrangian relaxation. *Interfaces* **15**, 10 (1985).

25. Geoffrion, A. Generalized Benders Decomposition. *Journal of Optimization Theory and Applications* **10**, 237 (1972).

26. Herrigel, S., Laumanns, M., Nash, A. & Weidmann, U. Hierarchical Decomposition Methods for Periodic Railway Timetabling Problems. *Transportation Research Record* **2374**, 73 (2013).

27. Keita, K., Pellegrini, P. & Rodriguez, J. A three-step Benders decomposition for the real-time Railway Traffic Management Problem. *Journal of Rail Transport Planning & Management* **13**, 100170 (2020).

28. Kersbergen, B., van den Boom, T. & De Schutter, B. Distributed model predictive control for railway traffic management. *Transportation Research Part C Emerging Technologies* **68**, 462 (2016).

29. Khadilkar, H. A Scalable Reinforcement Learning Algorithm for Scheduling Railway Lines. *IEEE Transactions on Intelligent Transportation Systems* **20**, 727 (2019).

30. Kroon, L., Huisman, D., Abbink, E., Fioole, P., Fischetti, M., Maróti, G., Schrijver, A., Steenbeek, A. & Ybema, R. The New Dutch Timetable: The OR Revolution. *Interfaces* **39 (1)**, 6 (2009).

31. Lamorgese, L. & Mannino, C. A non-compact formulation for job-shop scheduling problems in traffic management. *Operations Research* **67**, 1503 (2019).

32. Lamorgese, L. & Mannino, C. An Exact Decomposition Approach for the Real-Time Train Dispatching Problem. *Operations Research* **63**, 48 (2015).

33. Lamorgese, L., Mannino, C. & Natvig, E. An exact micro–macro approach to cyclic and non-cyclic train timetabling. *Omega (United Kingdom)* **72**, 59 (2017).

34. Lamorgese, L., Mannino, C. & Piacentini, M. Optimal Train Dispatching by Benders'-Like Reformulation. *Transportation Science* **50**, 910 (2016).

35. Laurent, F., Schneider, M., Scheller, C., Watson, J., Li, J., Chen, Z., Zheng, Y., Chan, S.-H., Makhnev, K., Svidchenko, O., Egorov, V., Ivanov, D., Shpilman, A., Spirovska, E., Tanevski, O., Nikov, A., Grunder, R., Galevski, D., Mitrovski, J. & Mohanty, S. *Flatland Competition 2020: MAPF and MARL for Efficient Train Coordination on a Grid World* 2021.

36. Leutwiler, F. & Corman, F. A logic-based Benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research* **303**, 525 (2022).

37. Liu, J., Chen, L., Roberts, C., Li, Z. & Wen, T. *A Multi-agent Based Approach for Railway Traffic Management Problems* in *2018 International Conference on Intelligent Rail Transportation (ICIRT)* (2018), 1.

38. Liu, L. & Dessouky, M. A decomposition based hybrid heuristic algorithm for the joint passenger and freight train scheduling problem. *Computers & Operations Research* **87**, 165 (2017).

39. Logic-based Benders decomposition. *Mathematical Programming* **96**, 33 (2003).

40. Luan, X., De Schutter, B., Meng, L. & Corman, F. Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. *Transportation Research Part B Methodological* **141**, 72 (2020).

41. Luan, X., Miao, J., Meng, L., Corman, F. & Lodewijks, G. Integrated optimization on train scheduling and preventive maintenance time slots planning. *Transportation Research Part C Emerging Technologies* **80**, 329 (2017).

42. Luan, X., Schutter, B. D., van den Boom, T., Corman, F. & Lodewijks, G. Distributed optimization for real-time railway traffic management. *IFAC-PapersOnLine* **51**, 106 (2018).

43. Marcelli, E. & Pellegrini, P. Literature Review Toward Decentralized Railway Traffic Management. *IEEE Intelligent Transportation Systems Magazine* **13**, 234 (2021).

44. Mascis, A. & Pacciarelli, D. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operations Research* **143**, 498 (2002).

45. Meng, L. & Zhou, X. Simultaneous train rerouting and rescheduling on an N-track network: A model reformulation with network-based cumulative flow variables. *Transportation Research Part B Methodological* **67**, 208 (2014).

46. Narayanaswami, S. & Rangaraj, N. A MAS architecture for dynamic, realtime rescheduling and learning applied to railway transportation. *Expert Systems with Applications* **42**, 2638 (2015).

47. Odijk, M. A. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B Methodological* **30**, 455 (1996).

48. Parkes, D. & Ungar, L. *An auction-based method for decentralized train scheduling* in *Proceedings of the fifth international conference on Autonomous agents* (2001), 43.

49. Perrachon, Q., Chevrier, R. & Pellegrini, P. Experimental study on the viability of decentralized railway traffic management. *WIT Transactions on the Built Environment* **199**, 337 (2020).

50. Proença, H. & Oliveira, E. *MARCS Multi-Agent Railway Control System* in *Ibero-American Conference on Artificial Intelligence* (2004), 12.

51. Shang, F., Zhan, J. & Chen, Y. *Distributed Model Predictive Control for Train Regulation in Urban Metro Transportation* in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018), 1592.

52. Sinha, S. K., Salsingikar, S. & SenGupta, S. An iterative bi-level hierarchical approach for train scheduling. *Journal of Rail Transport Planning & Management* **6**, 183 (2016).

53. Solving periodic timetabling problems with SAT and machine learning. *Public Transport* **13**, 625 (2021).

54. Toletti, A., Laumanns, M. & Weidmann, U. Coordinated railway traffic rescheduling with the Resource Conflict Graph model. *Journal of Rail Transport Planning & Management* **15**, 100173 (2020).

55. Vanderbeck, F. & Savelsbergh, M. W. A generic view of Dantzig-Wolfe decomposition in mixed integer programming. *Operations Research Letters* **34**, 296 (2006).

56. Zhan, S., Kroon, L. G., Zhao, J. & Peng, Q. A rolling horizon approach to the high speed train rescheduling problem in case of a partial segment blockage. *Transportation Research Part E Logistics and Transportation Review* **95**, 32 (2016).

57. Zhang, Y., D'Ariano, A., He, B. & Peng, Q. Microscopic optimization model and algorithm for integrating train timetabling and track maintenance task scheduling. *Transportation Research Part B Methodological* **127**, 237 (2019).

3

A LOGIC-BASED BENDERS DECOMPOSITION FOR MICROSCOPIC RAILWAY TIMETABLE PLANNING

Leutwiler, F. and Corman, F. (2022). A logic-based benders decomposition for microscopic railway timetable planning. European Journal of Operational Research, 303(2):525–540.

This is a post-print version of [25], differing from the published paper only in terms of layout and formatting.

Key findings:

- A logic-based Benders decomposition with a novel logic Benders cut for microscopic railway timetabling.
- An aggregation of logic Benders cuts to speed up the iterative process.
- Instances twice as large can be solved 40 times faster than by a commercial MILP solver.

Author contributions: Model and Decomposition: FL; Proofs FL, FC; Implementation: FL; Manuscript preparation and review: FL, FC.

ABSTRACT

Railway timetable planning is one of the key factors in the successful operation of a railway network. The timetable must satisfy all operational restrictions at a microscopic representation of the railway network, while maximizing transportation capacity for passengers and freight. The microscopic planning of a railway timetable is an NP-Hard problem, difficult to solve for large-scale railway networks, such as those of entire countries. In this work, we propose a logic Benders decomposition approach to solve the problem of microscopic railway timetable planning. Our decomposition exploits the typical structure of a railway with dense networks around major hubs and sparse connections in-between hubs. A logic Benders cut is designed, which we are able to compute effectively for all decomposed problems within our considered structure, using a SAT based algorithm we developed. Moreover, an aggregation scheme for Benders cuts is proposed to speed up the iterative process. Experiments on real-world cases of the Swiss Federal Railways show a clear improvement in scalability compared to a variety of benchmarks including centralized approaches.

3.1 INTRODUCTION

The detailed schedule of operations is a fundamental aspect for the successful operation of a railway network. The schedule has a direct influence on the performance of the railway system, e.g., amount of train services, number of transfer connections, costs; and due to unavoidable delays, their impact on the system punctuality. Providing punctual, reliable, and sufficient services to the customers is one of the main goals of railway operators. The operators are keen to schedule the operations of trains on the network, such that a planned schedule is robust to smaller disturbances, minimal in operational cost; and provides short travel times and sufficient capacity to passengers. Railway operators continuously increase their expectations towards the schedule of operations in order to absorb the rising demand. The operators are keen to maximize utilization of existing network infrastructure such that sufficient capacity can be provided with only minimal extensions to the existing railway network.

An approach pursued by railway operators to increase capacity is the use of large-scale optimization. Network-wide scheduling enables the coordination of operations on a network-wide scale and is a step towards maximizing the global utilization of the network.

The problem of scheduling railway operations, usually referred as railway timetable planning or railway timetabling, seeks to determine a schedule for train services operating in a railway by determining their movements on the network in time (arrival, departure, passing time, orders) and space (route followed). A feasible schedule specifies the start and end of every movement on the railway for a limited period in time, i.e., the planning horizon. The timetable must satisfy numerous external requirements from planning steps in railways that precede scheduling, but also due to technological constraints of the railway safety system. Those latter are approximated in macroscopic approaches, while microscopic approaches aim to keep all the relevant details. Detailed descriptions on different models and solution approaches to railway timetable planning can be found in [8], [5].

Railway scheduling is known as a NP-hard problem (e.g., [31]). As such we must expect scalability issues for problems of railway scheduling, especially in approaches which model the railway network at a high level of detail. In contrast, railway operators desire to solve, with high quality and short time, large network-wide scheduling problems to improve the transportation capacity of the railway. As a contribution to close the gap between technologically solvable problems and the desires of railway operators, we introduce in this paper a logic-based Benders decomposition. Decomposition approaches have been developed to solve large scale optimization problems. In decomposition, the original problem is split into smaller subproblems significantly easier to solve, at the expense of an additional coordination scheme by a master process to synthesize a global optimal solution. With our logic-based Benders decomposition, we tackle the microscopic railway timetable planning problem with a decomposition that can be tailored to specific instances and industrial needs. We introduce a new type of logic-based Benders cut to address the resulting complex subproblems. We propose an efficient implementation of our decomposition and emphasize the value of our contribution with promising results on a real-world test case of the Swiss Federal Railways (SBB).

This paper is structured as following. In Section 3.2 we review existing literature and identify the contribution. We introduce in Section 3.3 the microscopic railway timetable planning problem and formulate it as a disjunctive program in Section 3.4. The logic-based Benders decomposition is described in Section 3.5. Details on the algorithmic design and implementation are given in Section 3.6 and 3.7, with experiments on a real-world example in Section 3.8. We conclude in Section 3.9.

3.2 RELATED WORK

In the following we review specifically approaches of decomposition for railway scheduling. The literature on railway scheduling clearly goes beyond approaches of decomposition and we refer to the reviews [8], [5] and [13] for an in-depth review of general timetable planning and rescheduling in railways.

3.2.1 *Decomposition in Railway Scheduling*

The literature of decomposed railway scheduling can be categorized into the different classes of decentralized and hierarchical decompositions.

In decentralized decomposition approaches, an original problem is divided into multiple smaller subproblems on a single hierarchical layer. The decomposition is in general achieved through the relaxation of specific constraints in the original problem such that the remaining constraints have block diagonal structure. In such cases, the problem decomposes into subproblems optimizing over different subsets of the original variables. An iterative scheme coordinates the individual subproblems towards a global optimal solution. Throughout the iterations, individual subproblems are guided towards satisfying the constraints of the original problem, which initially have been relaxed. This can be done either by the generation of new constraints or by a penalty function. Some examples of decentralized decomposition are [6], [28], [10]. A recent review can be found in [29].

Hierarchical decompositions consider two or more hierarchical layers, where problems on a single layer are potentially independent. The hierarchical decomposition is generally achieved by partial optimization over subsets of the original variables. On each hierarchical layer, a different set of variables is optimized. Hierarchical decomposition is a scheme where subproblems are solved iteratively, top-down over the hierarchical layers of the decomposition. Solutions from higher layers are considered as fixed for the optimization in hierarchical subordinate layers. Coordination amongst solutions through the hierarchical layers is twofold. The higher layers coordinate the solution of the lower layers by giving them consistent boundary constraints, based on their optimal solutions. The lower layers report information to the higher layers, usually in form of additional constraints.

The number of hierarchical layers may vary by the approach. [23], [24] use a hierarchical decomposition with two layers. [20], [26] or [18] use three or more layers.

3.2.2 *Benders Decomposition*

The concept of Benders decomposition [4] belongs to the class of hierarchical decompositions and is a general scheme for decomposition in mathematical programming. Benders decomposition aims to tackle a mathematical optimization problem $\mathcal{Z}$ over two sets of variables y and x, where the optimization over the variables y is known to be particularly difficult. In other terms, optimizing only over x in $\mathcal{Z}$, with y being constant as $\bar{y}$, i.e., $\mathcal{Z}_{|y=\bar{y}}$, is significantly simpler in terms of computational effort. Benders decomposition builds an adequate representation of $\mathcal{Z}$ by an optimization problem considering only the complicating variables y, instead of optimizing over variables y and x jointly. The optimization over y without x represents the master problem in Benders decomposition; the optimization over x with fixed $y = \bar{y}$ represents the subproblem(s). Benders Cuts are constraints used to transform the master problem into an adequate representation of $\mathcal{Z}$. Cuts are generated in an iterative scheme. In every iteration, a current optimal solution of the master problem is evaluated as $\bar{y}$ on the subproblem(s) to generate new cuts. The optimality Benders cuts are constraints, generated to represent the objective of $\mathcal{Z}$ as a function of y only, based on $\mathcal{Z}_{|y=\bar{y}}$. The feasibility Benders cuts are constraints, generated to represent the feasible space of $\mathcal{Z}$ as a function of y only, based on $\mathcal{Z}_{|y=\bar{y}}$. In case the subproblem is a convex optimization problem with no integrality constraints, Lagrangian duality can be used to generate optimality and feasibility cuts. We refer to Benders decomposition using Lagrangian duality [14] as classical Benders decomposition.

[9] show that for 01-linear programming including big-M constraints, where the problem is decomposed into complicating variables y as integer (binary) and x as continuous, non-complicating variables [34], feasibility cuts in a Benders decomposition can be strengthened. In this particular decomposition, a master solution on y defines which of the big-M constraints must hold in the subproblem over x. If a subproblem is infeasible, the constraints causing infeasibility in the subproblem are identified. If any of those constraints can be associated with a master variable y, the combinatorial Benders cut is a feasibility Benders cut on those binary variables. The combinatorial Benders cut excludes the solution, which leads to infeasibility in the subproblem, from the space of feasible solutions of the master. While in theory any set of constraints, known to cause infeasibility in the subproblem, is sufficient to generate a combinatorial cut, [9] propose an optimization to find a set with particularly small support in the y variables, to get a stronger cut.

Benders decomposition has been further generalized using logic formalism in [27]. An appropriate logic formalism is used to formulate proofs of optimality and feasibility for a subproblem. The logic Benders cut is the result of determining under which conditions in the master such a proof remains valid for a subproblem [27]. We refer to *logic-based Benders decomposition* as the Benders decomposition using a logic formalism for generating Benders cuts. The ideas of [27] have been developed further in [19] for the problem of planning and scheduling. Specific logic-based infeasibility proofs are designed for the three objectives of minimizing cost, make-span and total tardiness in planning and scheduling. Similar ideas are proposed in [17] and [16] for scheduling and combinatorial optimization. Those latter works propose a constraint programming approach to detect the infeasibility of a subproblem and generate combinatorial cuts.

A three-layer Benders decomposition has been proposed for railway rescheduling in [20]. In their Benders decomposition, binary variables modelling the routing decisions in railway scheduling are identified as complicating variables y. A single subproblem $\mathcal{Z}_{|y=\bar{y}}$ is a mixed-integer linear program, with binary variables x_{01} corresponding to ordering decision of railway scheduling. The variables x_{01} are further fixed to the values $\operatorname{argmin}_{x_{01}} \mathcal{Z}_{|y=\bar{y}}$ to result in a modified subproblem, which is linear in the variables $x \setminus x_{01}$, and for which the theory of classical Benders cuts can be used.

A classical Benders decomposition with strengthened and lifted Benders cuts is proposed in [22] for the problem of microscopic railway rescheduling. They propose a big-M formulation for rescheduling and apply a classical Benders decomposition, following the approach of [34]. Feasibility cuts are strengthened in the same way as in [9]. In [22], the Lagrangian dual of the subproblem is a max-cost flow problem, where a cycle on the associated flow-graph proves infeasibility of the subproblem. The cycle is used to form a combinatorial Benders cut. A simple cycle is a set of constraints minimal in the support on y variables, identified by [9] as favorable. Optimality cuts are strengthened and lifted in a similar manner as feasibility cuts.

[23] and [24] propose a logic-based Benders reformulation for railway rescheduling. In both publications, a geographic logic-based Benders decomposition is proposed, where the master and subproblems are associated with the scheduling of lines and stations respectively. The scheduling of lines is performed on a simplified railway network where stations are represented as nodes (macroscopic). The scheduling of stations is considered on the real network at the level of blocks and signals (microscopic). For the scheduling

of stations, both works assume further that there is a separate inbound and outbound path for each platform, and those paths result in the same running time. Under those assumptions, for the microscopic consideration of the network, the subproblem is shown to be reducible to a list-coloring problem. In both publications, combinatorial Benders cuts are generated by logic proofs of infeasibility, in general based on an interval-intersection graph.

3.2.3 *Contribution*

In this paper we consider a specific type of railway scheduling problem, which we term microscopic railway timetable planning (MRTP). We formulate it as a job-shop scheduling problem in form of a disjunctive program [2] similar to [31]. We apply the concept of a logic-based Benders decomposition [27] for our disjunctive program. The contributions of this work are:

1) We propose a logic-based Benders decomposition for a disjunctive program, where the subproblem is a disjunctive program of feasibility. With a disjunctive program we include discrete decisions (variables) into the subproblem, which differentiates our approach from most of the literature (e.g., [9], [20], [22]). We do not exploit any form of reduction in treating the subproblem, which results in considering master and subproblem to be identical to the original problem in their structure of constraints, different from existing approaches (e.g., [24]).

2) We introduce a new type of logic Benders cut for the subproblem considered. The logic Benders cut is derived from a proof of infeasibility, where the proof itself is described on a graphical representation of our subproblem. Compared to similar literature on logic-based Benders decomposition for railway scheduling, e.g., [24], we consider a more general type of proof of infeasibility. The cut itself is a logic disjunction over terms, where each term is a precedence relation between two events of the railway operations considered; this is similar to the cuts of [23, 24].

3) We propose an efficient procedure for the detection of infeasibility in our subproblem, where the infeasibility is used to generate a logic Benders cut. The detection of infeasibility combines Boolean Satisfiability solving with a graph algorithm that is inspired by [11]. Our techniques are similar to the classical Benders decomposition of railway scheduling of [22]. An aggregation of logic Benders cuts is further proposed to speed-up the iterative solution process.

3.3 NON-PERIODIC MICROSCOPIC RAILWAY TIMETABLE PLANNING

We address in this work the problem of non-periodic microscopic railway timetable planning (MRTP), where real time information is not considered. The goal of MRTP is to compute a detailed plan of operations, i.e., a timetable, which associates start and end time to every operation. A solution must satisfy constraints due to a safety system (requiring a collision free ordering of trains), route settings, passengers, and kinematics. In general, computations must be finished within few hours.

In MRTP, we optimize over times, routes and orders. Times are modelled as continuous variables. Both ordering and routing decisions are modelled as discrete decision. A *discrete decision* is to select one *choice*, out of a finite set of possible choices, specific to the decision. Each choice results in a specific (set of) constraints to be satisfied by the solution.

The railway network is modeled in a microscopic manner. The network is considered divided into blocks, i.e., track sections of several hundred meters of track. We consider the passing of a train over such a block as a single operation. We associate a start and end event with each operation, which are the physical entry and exit of the train on the block.

An instance of MRTP is defined by a list of train services that are to be included in the timetable, based on an estimated demand for transportation. Each train service is specified by means of temporal and infrastructure limitations for its operations.

Temporal limitations specify the time period of operation for a train service. Relevant events (arrival/departure from stations, entry/exit from the network) of a train service are restricted through individual time windows to limit the operation of a train in time. By temporal limitations we establish frequencies of services, minimum transfer times for passengers and rolling stock circulations.

Infrastructure limitations specify the infrastructure to be used by a train service by means of available blocks. The interplay of train dynamics and infrastructure is abstracted to minimal durations for related operations. The route of a train service is the consecutive, interconnected sequence of blocks from the origin to the destination. In routing areas, the network infrastructure allows different alternative subpaths, i.e., route alternatives, along which a train can move between two points (blocks) in the network. Each route alternative is a different sequence of blocks. A routing decision is a discrete decision, which is to select one route alternative in the routing area for a train service passing such routing area. A single route alternative

is one possible (routing) choice in the routing decision. Different routing choices lead to different operations for a train service. A solution for MRTP requires to select a routing choice for each routing decision.

A resource conflict is the potential simultaneous occupation of the same block by two trains. Such a condition is considered illegal by the safety system of the railway. Whenever a resource conflict between a pair of train services exists, a discrete decision must be made, that is an ordering decision for the usage of the conflicting block. In an ordering decision, two possible (ordering) choices are available, i.e., either one train before the other or vice versa. A solution for MRTP specifies an ordering choice for each ordering decision, i.e., for each pair of trains in conflict on a block.

The solution of MRTP is a timetable, i.e., event times for all operations according to the chosen routes. Assuming that all services can be scheduled within their time windows, the performance metric of a timetable depends on the timing of relevant events. We define the runtime delay of a relevant event as the difference in time between the scheduled time in the solution, and the earliest possible point of scheduling (lower bound of the time window). Based on our industrial setup, we optimize for a timetable with the minimal sum of runtime delays, over all relevant events. This metric favors a timetable that is robust against smaller delays. An early scheduling (a smaller runtime delay) increases the chances that the event and any successive events will still fit in their time window in case of any delay or disturbance. We formally define MRTP as the problem to schedule all train services according to their specifications, determining suitable orders, routes and times, minimizing the sum of runtime delays over all relevant events.

3.4 A DISJUNCTIVE FORMULATION OF MRTP

In the following, we propose a disjunctive formulation [2] for the problem of MRTP introduced in Section 3.3, similar to the formulation of job-shop scheduling in [31]. In MRTP, a train d (a job) is represented by a sequence of operations (d, b) with b indicating the related block (machine). Compared to [31] where a job is a fixed sequence of operations, the sequence of operations in a job of MRTP is variable. Different routing choices for a train service result in different operations in a job. For this reason, MRTP generalizes the job-shop scheduling with blocking and no-wait constraints of [31] to the job-shop scheduling with multi-purpose machines of [7] and further.

In MRTP, operations cannot be interrupted and jobs cannot be paused, such that the end of an operation coincides with the start of a successive

operation. We define a time variable $t_{db} \in \mathbb{R}_+$ for the start of operation (d, b). A *precedence relation* $((d, b), (q, p))$ is a linear constraint to make sure t_{qp} happens at least $f_{db,qp} \in \mathbb{R}$ time units after t_{db}. We denote with A the set of all precedence relations in an instance of MRTP. Precedence relations are either *fixed* (A_f) or *selectable* (A_s).

Fixed precedence relations in A_f represent the constraints of MRTP that must hold in any case, such as minimal travel times outside of routing areas or the temporal constraints of time windows. Time windows require a variable $t_0 = 0$ as the origin of time. Selectable precedence relations A_s are precedence relations that can be selected or not, and respectively do or do not need to hold, depending on choices made for the discrete decisions in MRTP. A discrete decision l logically is associated with a disjunctive constraint. A choice c in the decision logically corresponds to a term in the disjunction of such constraint. A single term in the disjunction is a set of selectable precedence relations; we denote such set as *choice set* W_c. We denote as *decision set* D_l the set of all choice sets related to decision l. Using the concepts of choice sets W_c and decision set D_l, we can write the disjunctive constraint of a decision l as,

$$\bigvee_{W_c \in D_l} \bigwedge_{((d,b),(q,p)) \in W_c} \left(t_{qp} - t_{db} \geq f_{db,qp} \right). \tag{3.1}$$

Constraint (3.1) is satisfied, if for at least one choice set in the decision set, all its selectable precedence relations are satisfied. In case a decision set contains two choice sets, where each choice set contains exactly one precedence relation, the constraint (3.1) simplifies to the alternative precedence relation of [31].

We model an ordering decision for a pair of trains d, q using the same block b by a decision set with exactly two choice sets. Each choice set contains a single selectable precedence relation to constrain the timing of operations (d, b) and (q, b) such that either (d, b) is finished before (q, b) starts or vice versa.

We model a routing decision by a decision set, whose elements are choice sets; each choice set is individually associated with a single routing alternative. A choice set is the set of selectable precedence relations related to minimal travel times of blocks along the corresponding routing alternative. In routing areas, we must take into account that train services may share infrastructure only on certain route alternatives. Resource conflicts may thus depend on routing decisions. We model such dependency by introducing auxiliary variables for each operation on a routing alternative. Auxiliary

variables, instead of the original variables, are then used in the precedence relations of ordering decisions occurring on routing alternatives. Auxiliary variables have the same value as their associated original variables, in case the related routing alternative is chosen. To enforce such equality, we use appropriate selectable precedence relations, which only hold in case the related routing alternative is chosen. We thus add these selectable precedence relations between the original and auxiliary variables to the choice sets of routing alternatives, to make them dependent on routing choices. Figure 3.1 illustrates auxiliary variables in an example of two trains d and q conflicting on block b; for train d, the block b belongs to one of two routing alternatives between its operations (d,u) and (d,v). In the graph of Figure 3.1, time variables are reported as nodes and precedence relations are arcs. We will later formally introduce such graph as the *generalized disjunctive graph*. In the example of Figure 3.1, the two choice sets related to the routing alternatives of train d consist of the dashed (W_{11}, upper) and dotted (W_{12}, lower) colored arcs respectively. The precedence relations between the auxiliary variables $(d,b)_{in}, (d,c)_{out}$ and their originals $(d,b), (d,c)$ are added to the choice set of the related routing alternative, i.e., to W_{12}. Auxiliary variables are then used in the precedence relations of the ordering decision D_2 over block b.

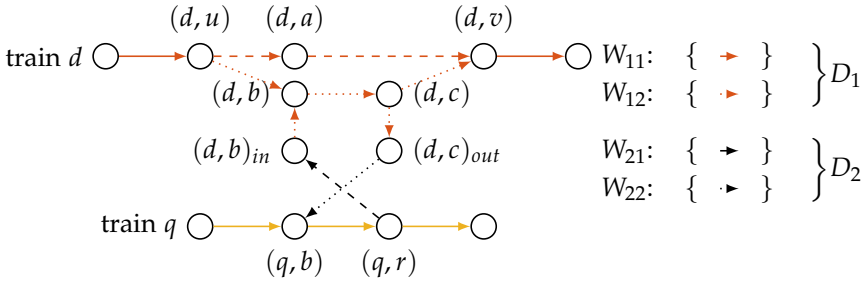

FIGURE 3.1: Explanation of routing for 2 trains and one routing area.

For the remainder of the paper, we simplify the notation of an operation (d,b) to i, if not stated otherwise. We define L as the set of all decisions in an

instance of MRTP. We can formulate the MRTP as the following disjunctive program:

$$\begin{aligned} \min \quad & \sum_{i \in H} t_i \\ \text{s.t.} \quad & t_j - t_i \geq f_{ij} \qquad (i,j) \in A_f \\ & \bigvee_{W_c \in D_l} \bigwedge_{(i,j) \in W_c} \left(t_j - t_i \geq f_{ij}\right) \quad l \in L. \\ & t_i \in \mathbb{R}_+ \; \forall \, t_i \end{aligned} \tag{3.2}$$

where H is the set of all relevant events associated with the objective of the problem of MRTP. Problem (3.2) is NP-hard. Indeed, in case L contains only alternative precedence relations, i.e., $|D_l| = 2 \; \forall l \in L$ and $|W_c| = 1 \; \forall \, W_c \in D_l$, Problem (3.2) can be reduced to Problem 3.1 (job-shop scheduling with blocking and no-wait) of [31], which was proven NP-hard in the same paper.

In the style of [31], we associate a node with each event time t_i in Problem (3.2). In this way, we can describe Problem (3.2) in a generalized disjunctive graph, that is the triple $G = (V, A_f, A_s)$, generalizing the ideas of [3]. V is the set of nodes, modelling the start of operations, origin of time, and auxiliary variables. We use the notion of precedence relations and arcs interchangeably, such that A_f, A_s and their union A are sets of fixed, selectable and all arcs in the context of the generalized disjunctive graph. Arc $(i,j) \in A$ has length f_{ij}. We assume no self-loops in A. Selectable arcs, similarly to selectable precedence relations, can be selected or not. Selectable arcs are grouped into the choice sets; choice sets are grouped into the decision sets of Problem (3.2). We define a subset of selectable arcs $\theta \subseteq A_s$ as a *selection* on the generalized disjunctive graph. Given a selection θ, $G(\theta) = (V, A_f \cup \theta)$ is a classical directed graph. We denote a selection as *complete* if it contains, for each decision l in L, all the arcs of at least one choice set, i.e., $\forall l \in L$, $\exists W_c \in D_l$ s.t. $W_c \subseteq \theta$. Otherwise, we denote a selection as *partial*. We define a selection as *consistent*, if $G(\theta)$ is free of positive length directed cycles.

Proposition 1. *Problem (3.2) has a feasible solution if and only if there exists a consistent, complete selection θ on the generalized disjunctive graph G of Problem (3.2).*

Proof. First we show that given a complete consistent selection θ on G, we can construct a feasible solution to (3.2). Let $V_0 \subseteq V$ be a set of "root" nodes that are pairwise unreachable in $G(\theta)$, i.e., such that no directed path exists in $G(\theta)$ connecting any two nodes $u, v \in V_0$. This means that the remaining nodes $V \backslash V_0$ are reachable from at least one node of V_0. We assign to each

"root" node in V_0 the value 0. For all nodes $v \in V \backslash V_0$, we assign a value that is either 0 or, if greater than zero, equal to the length of the longest path from any node in V_0 to v. All assigned values are finite because θ is consistent by assumption, i.e., the length of any longest path on $G(\theta)$ is finite. Moreover because θ is complete, the constructed solution satisfies all disjunctive constraints and is a feasible solution to (3.2).

Next we show the inverse implication. Given a feasible solution $\{\bar{t}_i, i \in V\}$ to (3.2), we prove that the selection defined by $\bar{\theta} := \{(i,j) \in A_s | \bar{t}_j - \bar{t}_i \geq f_{ij}\}$ is complete and consistent. Since $\{\bar{t}_i, i \in V\}$ is feasible, it satisfies all disjunctive constraints of (3.2) and hence $\forall l \in L, \exists W_c$ s.t. $W_c \subseteq \bar{\theta}$, which implies that $\bar{\theta}$ is complete. Moreover, we know by construction that all precedence relations in $G(\bar{\theta})$ are satisfied by $\{\bar{t}_i, i \in V\}$, which implies that $G(\bar{\theta})$ cannot contain any positive length directed cycle and $\bar{\theta}$ is also consistent. In fact, if $G(\bar{\theta})$ would contain a positive length cycle, the set of precedence relations in such cycle would be an infeasibility contradicting our assumption that $\{\bar{t}_i, i \in V\}$ is a feasible solution. □

In the rest of the paper, if clear from the context we refer to the generalized disjunctive graph simply as the disjunctive graph. An illustrative example of the disjunctive graph, choices and decisions is described in Appendix 3.10.

3.5 A BENDERS DECOMPOSITION FOR MRTP

In this section, we propose a Benders decomposition for Problem (3.2), where master and subproblem are disjunctive programs equal in the structure of constraints to the original Problem (3.2). For this decomposition, we introduce a logic feasibility Benders cut, inspired by [27].

With the logic Benders cut introduced, new ways of decomposing the MRTP become possible. We propose a geographic decomposition in Section 3.6. We motivate our Benders decomposition by the ability to represent substructures of MRTP with large solution spaces (subproblem) on a smaller set of variables (master) to reduce the computational complexity of the overall problem. We show that we can identify substructures (subproblems) that are independent and can be evaluated in parallel.

3.5.1 *A Decomposed Formulation of MRTP*

For clarity we refer to Problem (3.2) as the centralized problem ($\mathcal{C}$); we decompose it into a master problem ($\mathcal{M}$) and a subproblem ($\mathcal{S}$). In this

paper, we limit ourselves to a decomposition where the objective of $\mathcal{C}$ only depends on variables of $\mathcal{M}$, therefore $\mathcal{S}$ is only a problem of feasibility. In the following, we first show how to decompose Problem (3.2) into master and subproblem; and we later propose a condition to consider not a single but K independent subproblems.

To decompose Problem (3.2) we partition the constraints of $\mathcal{C}$, i.e., A, into $A_{\mathcal{M}}$ for $\mathcal{M}$ and $A_{\mathcal{S}}$ for $\mathcal{S}$. We partition A by partitioning A_f and A_s separately. Fixed precedence relations A_f can be partitioned arbitrarily into $A_{\mathcal{M},f}$ and $A_{\mathcal{S},f}$ for $\mathcal{M}$ and $\mathcal{S}$ respectively. Selectable precedence relations A_s are partitioned based on an arbitrary partition of L into $L_{\mathcal{M}}$ and $L_{\mathcal{S}}$; this can be made independently from the partition on A_f. The selectable precedence relations of $\mathcal{M}$ and $\mathcal{S}$ then are

$$A_{\mathcal{M},s} = \bigcup_{l \in L_{\mathcal{M}}} \bigcup_{W_c \in D_l} W_c; \qquad A_{\mathcal{S},s} = \bigcup_{l \in L_{\mathcal{S}}} \bigcup_{W_c \in D_l} W_c \tag{3.3}$$

respectively. Based on the partition of constraints, we introduce three sets of variables M, $M_{\mathcal{S}}$ and S. M are variables of $\mathcal{M}$ and S are variables of $\mathcal{S}$. Formally, we define the three sets as follows,

$$M := \{i,j \,|\, (i,j) \in A_{\mathcal{M}}\}; \quad S := \{i,j \,|\, (i,j) \in A_{\mathcal{S}}\}; \quad M_{\mathcal{S}} = M \cap S. \tag{3.4}$$

$M_{\mathcal{S}}$ are variables optimized in $\mathcal{M}$ and fixed in $\mathcal{S}$ according to Benders decomposition. In case not all relevant events H are in M, we simply extend M by the relevant events H, i.e., $M := \{i,j \,|\, (i,j) \in A_{\mathcal{M}}\} \cup H$. This ensures that the objective can fully be represented in $\mathcal{M}$.

The Benders scheme works in iterations, and numerous Benders cuts β are identified, on the subproblem, over the iterations $\Omega_\alpha = \{1, \cdots, \alpha - 1\}$. We define $\mathcal{M}^\alpha$ as the optimization problem $\mathcal{M}$ extended by all Benders cuts identified in all iterations Ω_α. We write $\mathcal{M}^\alpha$ in disjunctive form as,

$$\begin{aligned}
\min \quad & \sum_{i \in H} t_i \\
\text{s.t.} \quad & t_j - t_i \geq f_{ij} \qquad (i,j) \in A_{\mathcal{M},f} \\
& \bigvee_{W_c \in D_l} \bigwedge_{(i,j) \in W_c} (t_j - t_i \geq f_{ij}) \quad l \in L_{\mathcal{M}}. \\
& \beta^r \qquad \forall r \in \Omega_\alpha \\
& t_i \in \mathbb{R}_+ \, \forall t_i \in M
\end{aligned} \tag{3.5}$$

with β^r as the Benders cut identified at iteration r.

At each iteration α of the decomposition scheme, the solution $\mathcal{O}^{\alpha}_{\mathcal{M}} = \left\{\bar{t}^{\alpha}_i,\ i \in M\right\}$ of $\mathcal{M}^{\alpha}$ has to be evaluated on $\mathcal{S}$. We denote by $\mathcal{S}^{\alpha}$ the subproblem $\mathcal{S}$ dependent on $\mathcal{O}^{\alpha}_{\mathcal{M}}$ at iteration α of the Benders scheme:

$$\begin{aligned}
\min \quad & 0 \\
\text{s.t.} \quad & t_i = \bar{t}^{\alpha}_i && \forall i \in M_{\mathcal{S}} \\
& t_j - t_i \geq f_{ij} && (i,j) \in A_{\mathcal{S},f} \\
& \bigvee_{W_c \in D_l} \bigwedge_{(i,j) \in W_c} (t_j - t_i \geq f_{ij}) && l \in L_{\mathcal{S}} \\
& t_i \in \mathbb{R}_+,\ \forall t_i \in S.
\end{aligned} \tag{3.6}$$

In $\mathcal{S}^{\alpha}$, we rewrite the constraints $t_i = \bar{t}^{\alpha}_i\ \forall i \in M_{\mathcal{S}}$ as the following pair of inequality constraints, using the time origin $t_0 = 0$,

$$\begin{aligned}
& t_i - t_0 \geq \bar{t}^{\alpha}_i \quad \forall i \in M_{\mathcal{S}} \\
& t_0 - t_i \geq -\bar{t}^{\alpha}_i \quad \forall i \in M_{\mathcal{S}}.
\end{aligned} \tag{3.7}$$

We denote such constraints as *coordination constraints* (and similarly, we refer to *coordination arcs*). We denote the union of constraints $A_{\mathcal{S},f}$ and the coordination constraints (3.7) as $A^{\alpha}_{\mathcal{S},f}$. With the coordination constraints we are able to define the disjunctive graph for the subproblem as $G^{\alpha} := (S, A^{\alpha}_{\mathcal{S},f}, A_{\mathcal{S},s})$. We will later use the disjunctive graph of the subproblem to introduce our logic Benders cut.

Regarding the subproblem, we are able to further partition $A_{\mathcal{S}}$, in particular $A_{\mathcal{S},f}$ and $L_{\mathcal{S}}$, into K different sets $A_{k,f}$ and L_k. This allows to define, as above for a single subproblem, K different and independent subproblems $\mathcal{S}_k$. To achieve independence among the K subproblems, it must hold that:

$$\forall q, p \in K, q \neq p: \quad S_q \cap S_p \subseteq M_{\mathcal{S}}. \tag{3.8}$$

That is, any variable shared among different subproblems must be controlled, i.e., optimized, by the master. Otherwise, we cannot consider the subproblems independently. Despite the theory works on an arbitrary partition of constraints, a reasonable partition is crucial for the performance of a Benders decomposition. We later propose a decomposition for the railway scheduling problem in Section 3.6. In the following we design a feasibility Benders cut that is valid for disjunctive problems in the form of Problem (3.6). Optimality cuts are redundant as subproblems are problems of feasibility only.

3.5.2 *Determining Infeasibility*

From the theory on logic Benders cuts of [27], we know that any feasibility Benders cut β^α for $\mathcal{M}$ is valid, if and only if it is satisfied by any feasible solution of $\mathcal{C}$. We build a logic feasibility Benders cut from an infeasible substructure, i.e., an infeasibility proof for $\mathcal{S}^\alpha$; we will prove in Section 3.5.3 the validity of such cut. We motivate the use of an infeasibility proof as follows. If we know that there exists no solution for $\mathcal{S}^\alpha$, the set of values $\mathcal{O}^\alpha_\mathcal{M}$ is an infeasible partial solution for $\mathcal{C}$, and cannot be extended to any feasible solution $\mathcal{O}_\mathcal{C}$. Consequentially, a logic Benders cut is necessary to exclude $\mathcal{O}^\alpha_\mathcal{M}$ from the solution space of $\mathcal{M}$.

At each iteration of the Benders scheme we differentiate between two cases for $\mathcal{S}^\alpha$:

Case I: $\mathcal{S}^\alpha$ has a feasible solution $\mathcal{O}^\alpha_\mathcal{S} := \left\{\bar{t}^\alpha_i,\, t_i \in S\right\}$.

Case II: $\mathcal{S}^\alpha$ has no feasible solution and there exists an infeasibility proof $\mathcal{I}^\alpha$ for $\mathcal{S}^\alpha$.

If *Case I* applies, we can terminate the procedure and assemble a solution for $\mathcal{C}$ from the partial solutions $\mathcal{O}^\alpha_\mathcal{M}$ and $\mathcal{O}^\alpha_\mathcal{S}$. If *Case II* applies, we use an infeasibility proof for $\mathcal{S}^\alpha$ to generate a Benders cut and continue with the Benders procedure.

We use Proposition 1 to prove infeasibility of $\mathcal{S}^\alpha$ by showing that there exists no complete consistent selection on the related disjunctive graph G^α.

Definition 1. *(Infeasibility Proof for $\mathcal{S}^\alpha$)*
An infeasibility proof $\mathcal{I}^\alpha$ for $\mathcal{S}^\alpha$ is a set of simple directed positive length cycles, where each cycle is a subset of arcs of the related disjunctive graph G^α. It holds for $\mathcal{I}^\alpha$, that for all possible complete selections θ on G^α, at least one cycle of $\mathcal{I}^\alpha$ is completely in $G^\alpha(\theta)$.

Since a positive length cycle in $G^\alpha(\theta)$ can already be caused by a partial selection θ, the cardinality of $\mathcal{I}^\alpha$ is generally much lower than the number of possible selections on G^α. An inconsistent partial selection, whose inconsistency is proven by a single cycle, proves the inconsistency of any complete selection that extends such partial selection. In Section 3.6 we will show that we are able to generate an infeasibility proof by finding a positive length cycle for each complete selection θ on G^α, evaluating mainly partial selections.

In case a cycle $\gamma \in \mathcal{I}^\alpha$ involves at least one coordination arc, the length of the cycle depends on $\mathcal{O}^\alpha_\mathcal{M}$. We use such characteristic to derive a logic Benders cut in the next section. Goal of the Benders cut β^α is to constrain $\mathcal{M}^\delta$ of subsequent iterations $\delta > \alpha$ such that a feasible solution to $\mathcal{S}^\delta$ may exist.

3.5.3 *A Logic Benders Cut*

We here introduce a logic Benders cut based on the infeasibility proof of Section 3.5.2. In case at least one cycle in the infeasibility proof $\mathcal{I}^\alpha$ depends on the master solution, we are able to generate a valid logic Benders cut to constrain $\mathcal{M}^\delta$, $\delta > \alpha$. Our logic Benders cut is a constraint that, if satisfied by the master, avoids that the infeasibility proof $\mathcal{I}^\alpha$ applies to the related subproblem. An infeasibility proof $\mathcal{I}^\alpha$ is avoided on G^δ, $\delta > \alpha$ and thus in the subproblem, if there exists at least one cycle in $\mathcal{I}^\alpha$ that is either not fully contained in the arcs of G^δ, or has non-positive length when considering the arcs of G^δ. In case of our decomposition, where each cycle on G^α is also a cycle on G^δ, we can only change a cycle by its length over the iterations. In particular, we change the right-hand side of some coordination constraints (and thus the length of related coordination arcs) by an amount sufficient to make cycles non-positive in their length, and avoid infeasibilities in further iterations. An infeasibility independent from the master proves infeasibility of the centralized problem.

Our logic Benders cut is designed as a logic disjunction over terms, where each term is associated with a precedence relation between variables of $\mathcal{M}_\mathcal{S}$. A single precedence relation in the disjunction is designed based on a single cycle $\gamma \in \mathcal{I}^\alpha$. Of interest for generating such precedence relations are only those cycles in $\mathcal{I}^\alpha$, which contain at least one coordination arc. All other cycles are independent from $\mathcal{O}^\alpha_\mathcal{M}$ and hence can be neglected.

The single precedence relation generated from a single cycle γ restricts variables $M_\mathcal{S}$ and in consequence $\mathcal{O}^\delta_\mathcal{M}$. Assume that the precedence relation from γ is satisfied by $\mathcal{O}^\delta_\mathcal{M}$. We design the precedence relation such that, in this case, when we consider the set of cycles $\mathcal{I}^\alpha$ on G^δ, at least the cycle γ, used to generate the precedence relation, has non-positive length. Due to the changes in the graph, it might also be that other cycles in $\mathcal{I}^\alpha$ have non-positive length. We derive such a precedence relation by summing left- and right-hand side of all constraints that are arcs in the cycle γ and that are not coordination constraints. We denote those constraints by $\bar{\gamma} := \gamma \cap A_\mathcal{S}$. The purpose is to express the inconsistency described by a single positive length cycle only on the variables $M_\mathcal{S}$ such that it can be used in $\mathcal{M}$. When summing left- and right-hand side of the precedence relations that are arcs in $\bar{\gamma}$ we get the expression:

$$\sum_{(i,j) \in \bar{\gamma}} t_j - t_i \geq \sum_{(i,j) \in \bar{\gamma}} f_{ij}. \tag{3.9}$$

We may simplify the expression (3.9). Coordination arcs are by definition arcs in or outgoing to the time origin, and γ is a simple cycle. Therefore, there can be at most two coordination arcs in γ; in case there are exactly two, those must be consecutive. At least one coordination arc is contained in γ since it is a cycle of interest. Therefore, in any case, the set of all coordination arcs $\gamma \setminus \bar{\gamma}$ is a directed path; and the partition of γ into $\bar{\gamma}$ and $\gamma \setminus \bar{\gamma}$ is a partition of a cycle into two directed paths. We identify the start node of $\bar{\gamma}$ as $s(\bar{\gamma})$, its end node as $e(\bar{\gamma})$ and its length as $l_{\bar{\gamma}}$. Because $\bar{\gamma}$ is a directed path, the intermediate variables of the path cancel out in the left-hand side of (3.9) and the expression simplifies to

$$t_{e(\bar{\gamma})} - t_{s(\bar{\gamma})} \geq l_{\bar{\gamma}}. \tag{3.10}$$

In the inequality (3.10), the left-hand side are variables in $M_{\mathcal{S}}$ only, and the right-hand side $l_{\bar{\gamma}}$ is an instance-specific constant.

Since $A_{\mathcal{S}}$ may also contain arcs towards or from the time origin, γ may not necessarily contain two, but possibly only one coordination arc. In such case, either $t_{e(\bar{\gamma})}$ or $t_{s(\bar{\gamma})}$ is the time origin t_0, and (3.10) turns into an upper or lower bound on a variable in $M_{\mathcal{S}}$.

We define the logic Benders cut based on $\mathcal{I}^{\alpha}$ as the disjunction over all constraints of type (3.10) from all cycles of interest ($\gamma \neq \bar{\gamma}$) in $\mathcal{I}^{\alpha}$ as

$$\beta^{\alpha} = \bigvee_{\left\{\gamma \in \mathcal{I}^{\alpha} \,\middle|\, \bar{\gamma} \neq \gamma\right\}} \left(t_{e(\bar{\gamma})} - t_{s(\bar{\gamma})} \geq l_{\bar{\gamma}}\right). \tag{3.11}$$

The logic Benders cut (3.11) is satisfied if at least one precedence relation in the disjunction is satisfied. We prove the validity of the logic Benders cut (3.11) in Proposition 2. Then, we show that the logic Benders cut β^{α} (3.11) at least excludes $\mathcal{O}^{\alpha}_{\mathcal{M}}$ from the solution space of $\mathcal{M}^{\circ}$ to establish progress (Proposition 3). At last, we show that in case no infeasibility proof for $\mathcal{S}^{\alpha}$ exists, a feasible solution of $\mathcal{C}$ exists (Proposition 4).

Proposition 2. *The Benders cut β^{α} defined in (3.11) is satisfied by all feasible solutions of $\mathcal{C}$.*

Proof. Let $\mathcal{O}_{\mathcal{C}} = \{\bar{t}_i, i \in V\}$ be any feasible solution of $\mathcal{C}$. We consider the selection $\bar{\theta}_{\mathcal{C}} := \left\{(i,j) \in A_s | \bar{t}_j - \bar{t}_i \geq f_{ij}, \bar{t}_i \in \mathcal{O}_{\mathcal{C}}\right\}$ defined on the disjunctive graph of $\mathcal{C}$. We know from the proof of Proposition 1 that $\bar{\theta}_{\mathcal{C}}$ is complete and consistent. By the definition of our decomposition, we know that the selectable precedence relations and associated decisions of the subproblem $\mathcal{S}^{\alpha}$ are also selectable precedence relations and associated decisions of the

centralized problem $\mathcal{C}$, i.e., $A_{\mathcal{S},s} \subseteq A_s$ and $L_{\mathcal{S}} \subseteq L$. Thus, the selection $\bar{\theta}_{\mathcal{C}}$ restricted to the subproblem $\mathcal{S}^\alpha$, formally $\bar{\theta}_{\mathcal{S}} = \bar{\theta}_{\mathcal{C}} \cap A_{\mathcal{S},s}$, inherits from $\bar{\theta}_{\mathcal{C}}$ the property of completeness with respect to $L_{\mathcal{S}}$.

Given that $\bar{\theta}_{\mathcal{S}}$ is complete, there exists a positive length cycle $\gamma \in \mathcal{I}^\alpha$ where γ is fully contained in $G^\alpha(\bar{\theta}_{\mathcal{S}})$, or equivalently $\gamma \subseteq A^\alpha_{\mathcal{S},f} \cup \bar{\theta}_{\mathcal{S}}$ (see Definition 1 of the infeasibility proof). Moreover, γ contains at least one coordination arc and is therefore used to form a precedence relation defined by (3.10) in the Benders cut (3.11). In fact, if γ contains no coordination arc it would hold that $\gamma \subseteq A_{\mathcal{S},f} \cup \bar{\theta}_{\mathcal{S}}$. Consequentially, by the definition of our decomposition and the construction of $\bar{\theta}_{\mathcal{S}}$, we have that $\gamma \subseteq A_f \cup \bar{\theta}_{\mathcal{C}}$. This identifies that a positive length cycle exists in $G(\bar{\theta}_{\mathcal{C}})$, which contradicts our assumption that $\bar{\theta}_{\mathcal{C}}$ is consistent, i.e., that $\mathcal{O}_{\mathcal{C}}$ is a feasible solution.

If we now consider $\bar{\gamma}$ by excluding the coordination arcs from γ, we know that $\bar{\gamma}$ is a path for which it holds $\bar{\gamma} \subseteq A_{\mathcal{S},f} \cup \bar{\theta}_{\mathcal{S}}$, hence $\bar{\gamma} \subseteq A_{\mathcal{C},f} \cup \bar{\theta}_{\mathcal{C}}$. Thus, all precedence relations along $\bar{\gamma}$ are satisfied by $\mathcal{O}_{\mathcal{C}}$. This implies that for the $\bar{\gamma}$ considered, the precedence relation (3.10) is satisfied by $\mathcal{O}_{\mathcal{C}}$. Consequentially at least one term in the disjunction (3.11) is satisfied by $\mathcal{O}_{\mathcal{C}}$, hence β^α is satisfied by $\mathcal{O}_{\mathcal{C}}$. Since $\mathcal{O}_{\mathcal{C}}$ is an arbitrary solution of $\mathcal{C}$, we conclude that β^α is a valid cut, satisfied by all feasible solutions of $\mathcal{C}$. □

Proposition 3. *The Benders cut β^α (3.11) is violated by the master solution $\mathcal{O}^\alpha_{\mathcal{M}}$.*

Proof. Our proof relies on showing that $\mathcal{O}^\alpha_{\mathcal{M}}$ violates all precedence relations (3.10) that are part of the disjunction (3.11), i.e., β^α. To do so, consider any positive length cycle $\gamma \in \mathcal{I}^\alpha$, where $\gamma \neq \bar{\gamma}$. The positive length condition for such cycle can be written as:

$$l_{\bar{\gamma}} + l_{\gamma \setminus \bar{\gamma}}(\mathcal{O}^\alpha_{\mathcal{M}}) > 0. \tag{3.12}$$

where the notation $l_{\gamma \setminus \bar{\gamma}}(\mathcal{O}^\alpha_{\mathcal{M}})$ is to emphasize the fact that the length of the path $\gamma \setminus \bar{\gamma}$ depends on the master solution $\mathcal{O}^\alpha_{\mathcal{M}}$ whereas the length of $\bar{\gamma}$ does not.

In the precedence relation (3.10), we can reformulate the left-hand side based on the fact that $\bar{\gamma}$ and $\gamma \setminus \bar{\gamma}$ together form a cycle, such that start and end points of both paths are the same, but swapped:

$$t_{s(\gamma \setminus \bar{\gamma})} - t_{e(\gamma \setminus \bar{\gamma})} \geq l_{\bar{\gamma}}. \tag{3.13}$$

We know that for the directed path $\gamma \setminus \bar{\gamma}$, it must hold that $t_{e(\gamma \setminus \bar{\gamma})} - t_{s(\gamma \setminus \bar{\gamma})} \geq l_{\gamma \setminus \bar{\gamma}}(\mathcal{O}_{\mathcal{M}})$; therefore $-l_{\gamma \setminus \bar{\gamma}}(\mathcal{O}_{\mathcal{M}}) \geq t_{s(\gamma \setminus \bar{\gamma})} - t_{e(\gamma \setminus \bar{\gamma})}$. We can thus reformulate (3.13) into the following equivalent condition:

$$-l_{\gamma \setminus \bar{\gamma}}(\mathcal{O}_{\mathcal{M}}) \geq t_{s(\gamma \setminus \bar{\gamma})} - t_{e(\gamma \setminus \bar{\gamma})} \geq l_{\bar{\gamma}} \tag{3.14}$$

which is valid for any arbitrary master solution $\mathcal{O}_{\mathcal{M}}$.

Considering $\mathcal{O}^{\alpha}_{\mathcal{M}}$ in (3.14) and rearranging terms, it results that $l_{\bar{\gamma}} + l_{\gamma \setminus \bar{\gamma}}(\mathcal{O}^{\alpha}_{\mathcal{M}}) \leq 0$, which contradicts (3.12), stating that γ is a positive length cycle for $\mathcal{O}^{\alpha}_{\mathcal{M}}$. Therefore, (3.14) cannot be satisfied by $\mathcal{O}^{\alpha}_{\mathcal{M}}$, hence (3.10) cannot be satisfied by $\mathcal{O}^{\alpha}_{\mathcal{M}}$. Since this holds for any precedence relation included in the disjunction β^{α}, then β^{α} itself is violated by $\mathcal{O}^{\alpha}_{\mathcal{M}}$. □

Proposition 4. *If no infeasibility proof exists for subproblem $\mathcal{S}^{\alpha}$ in the decomposition scheme, the optimal master solution $\mathcal{O}^{\alpha}_{\mathcal{M}}$ can be extended to an optimal solution for $\mathcal{C}$.*

Proof. We denote the optimal solution of the master problem by $\mathcal{O}^{\alpha}_{\mathcal{M}}$. If no infeasibility proof for subproblem $\mathcal{S}^{\alpha}$ exists, then a feasible solution $\mathcal{O}^{\alpha}_{\mathcal{S}}$ to the subproblem $\mathcal{S}^{\alpha}$ (conditioned on $\mathcal{O}^{\alpha}_{\mathcal{M}}$) must exist. Merging these two solutions $\mathcal{O}^{\alpha} := \mathcal{O}^{\alpha}_{\mathcal{M}} \cup \mathcal{O}^{\alpha}_{\mathcal{S}}$ provides a solution to the entire $\mathcal{C}$.

By definition of our decomposition, it holds that $A = A_{\mathcal{M}} \cup A_{\mathcal{S}}$ and $L = L_{\mathcal{M}} \cup L_{\mathcal{S}}$. This implies that $\mathcal{O}^{\alpha}$ satisfies all constraints of $\mathcal{C}$ and is hence feasible. Moreover, since no variable appearing in S alone affects the objective function of $\mathcal{C}$, the optimality of $\mathcal{O}^{\alpha}$ follows directly from the optimality of $\mathcal{O}^{\alpha}_{\mathcal{M}}$. □

By construction, all theory shown in Section 3.5.2 and 3.5.3 also holds in case of K independent subproblems instead of a single subproblem.

3.6 IMPLEMENTATION

Given the theory from Section 3.5, we are now able to discuss the implementation of a Benders decomposition for MRTP. In the following, we will first introduce the specific decomposition of an instance of MRTP. Second, we will discuss an algorithm to generate either a feasible solution or an infeasibility proof in form of a set of cycles for a subproblem. We conclude the section with an overall discussion of the Benders decomposition procedure.

3.6.1 *A Decomposition of MRTP*

To decompose an instance of MRTP, we partition the constraints of the MRTP instance according to Section 3.5.1. We define $\mathcal{M}$ and $\mathcal{S}_k$ based on such partition of constraints. We must ensure that $\mathcal{M}$ and $\mathcal{S}_k$ satisfy condition (3.8) for our partition to achieve independence across subproblems.

When decomposing the MRTP problem we aim to find a decomposition where subproblems are substructures of the MRTP with a large solution space. In this way, the master problem has a considerably smaller solution space compared to the original problem (i.e., can be solved fast to optimality); and a subproblem is likely to impose few Benders cuts (i.e., a large solution space is likely to contain a feasible solution for a variety of master solutions to which no cut is necessary).

Our Benders decomposition considers a subproblem to be an instance of MRTP, therefore identical in type and structure of constraints to the master. We have thus considerable freedom in deciding the decomposition. We can allow any type of geographic decomposition, regardless of the amount of stations in the subproblems, different from existing approaches, e.g., [24]. In particular, we choose areas with dense infrastructure to create subproblems with a large solution space. For instance, the area around a station contains in general a dense railway infrastructure, which allows for a variety of possible routes for inbound and outbound trains and as such many different solutions. Due to our industrial setup, and for practical reasons, most of our subproblems are therefore including stations. In the test case we will use for the evaluation, we will show that subproblems can include no station, part of a station, one, or even multiple stations.

3.6.2 *Evaluating Feasibility of a Subproblem*

As discussed in Section 3.5.2, an algorithm to evaluate the feasibility of $\mathcal{O}^{\alpha}_{\mathcal{M}}$ on a subproblem $\mathcal{S}^{\alpha}_{k}$ must either return a feasible solution according to $\mathcal{O}^{\alpha}_{\mathcal{M}}$, or a set of cycles. This latter must be an infeasibility proof according to Definition 1 such that we can translate it into a Benders cut.

To evaluate the feasibility of a subproblem we use Satisfiability Modulo Theories (SMT), which is the combination of Satisfiability (SAT) solving [12] and a first-order logic [33]. The core algorithm to solve SAT, the DPLL [12] procedure is known to be particularly suitable to generate infeasibility proofs efficiently [1]. In MaxSat, that is the optimization over SAT, many algorithms [32] are based on this fact. We support our choice of SMT by a benchmark based on a commercial tool in Section 3.8.

Algorithm 1 illustrates our SMT procedure for the case of MRTP, following the terminology of the disjunctive graph. The algorithm is inspired by the ideas of [11]. The goal is to determine whether there exists a consistent complete selection θ_k on G^{α}_{k} and therefore whether $\mathcal{S}^{\alpha}_{k}$ has a feasible solution or not. We formulate the SAT problem, which we identify by its

Algorithm 1: SMT, A DPLL with Precedence Constraints.

input : $\mathcal{S}_k^\alpha$
output: $\mathcal{O}_k^\alpha$, $\mathcal{I}_k^\alpha$, β_k^α
init : $\Phi \leftarrow \mathcal{S}_k^\alpha$, $G_k^\alpha \leftarrow \mathcal{S}_k^\alpha$, $\theta_k = \emptyset$

```
1  while true do
2      confl ← UnitPropagation(Φ, θ_k)
3      if !confl then
4          confl ← Evaluate(G_k^α(θ_k))
5      if !confl then
6          if θ_k = complete then
7              O_k^α ← G_k^α(θ_k)
8              return (O_k^α, ∅, ∅)
9          θ_k ← θ_k ⋃ Decide()
10     else
11         if confl = Unsatisfiable then
12             I_k^α ← AnalyzeIP(confl)
13             β_k^α ← BendersCut(I_k^α)
14             return (∅, I_k^α, β_k^α)
15         else
16             Analyze(confl)
17             Backtrack(confl)
```

constraints Φ, to ensure that θ_k is complete. When DPLL searches for a complete selection, every partial selection encountered in the search procedure is evaluated for consistency. Consistency is determined by a cycle detection on $G_k^\alpha(\theta_k)$ in form of a longest path propagation. If an inconsistent (partial) selection is encountered, a constraint is added to Φ, to exclude such selection from the search of DPLL.

Algorithm 1 starts with an empty selection θ_k and continuously extends it by adding new choice sets W_c to it (line 9), till the selection is complete (line 6). `Decide` in line 9 uses SAT heuristics to pick the next choice set W_c to be added to θ_k. If a consistent and complete selection θ_k has been found, $\mathcal{O}_k^\alpha$ can be computed (see proof of Proposition 1) (line 7). In line 2, unit propagation [30] is performed, i.e., θ_k is propagated over the constraints Φ, to detect additional choice sets W_c to be added or excluded from θ_k. In case a constraint of Φ is violated by the resulting θ_k, the constraint is reported as a SAT conflict (`confl`). In line 4, in case no SAT conflict has been found by unit propagation,

θ_k is evaluated for consistency on $G_k^\alpha(\theta_k)$. In case $G_k^\alpha(\theta_k)$ contains a positive length cycle, `confl` is a new SAT constraint that prohibits θ_k to contain those W_c, which cause together the cycle on $G_k^\alpha(\theta_k)$. This SAT constraint is obviously violated by θ_k, as $G_k^\alpha(\theta_k)$ contains the cycle. In case the subproblem is determined to be infeasible because `confl` is an unsatisfiable constraint (line 11), Algorithm 1 terminates by analyzing the computations done so far, following the `AnalyzeIP` procedure of [1] (line 12). The analysis uses `confl` to determine the cycles on $G_k^\alpha(\theta_k)$, which are responsible for the infeasibility. We collect those cycles in $\mathcal{I}_k^\alpha$. Based on $\mathcal{I}_k^\alpha$, a Benders cut is generated (line 13). In case `confl` is possibly satisfiable, it is analyzed by standard SAT techniques (CDCL, [35]) to create a new constraint for Φ by strengthening `confl` (line 16). A backtracking (line 17) resets the DPLL search to a state where θ_k does not yet contain all W_c that together violate `confl`.

3.6.3 *Iterative Benders Decomposition*

The iterative scheme of Benders decomposition alternates between solving the master problem $\mathcal{M}$ and evaluating all subproblems $\mathcal{S}_k$ for feasibility. To solve $\mathcal{M}$ we may use any kind of mixed-integer solver. To evaluate each individual subproblem we can use any kind of algorithm that is able to compute an infeasibility proof for it. In particular, in Section 3.8 we compare the SMT algorithm of Section 3.6.2 with the commercial tool GUROBI-IIS to this purpose. In case the chosen algorithm identifies an infeasibility proof for the subproblem, we can translate it into a Benders cut for the master. Otherwise the algorithm must be able to determine a feasible solution.

In the Benders decomposition, before the evaluation of every subproblem, we may check if the subproblem has changed with respect to the previous iteration. If both, previous and current subproblem are identical, the evaluation of the current subproblem is redundant and it can be skipped. The iterative procedure terminates if for each subproblem $\mathcal{S}_k$ we find a feasible solution.

3.7 BENDERS CUT AGGREGATION

Until this point, at each iteration α of the Benders decomposition, first the master problem $\mathcal{M}^\alpha$ is solved and subsequently all subproblems $\mathcal{S}_k^\alpha$ are evaluated. The SMT algorithm of Section 3.6.2 for a subproblem either returns a feasible solution or a single Benders cut β_k^α. With the intention to decrease the number of iterations till a global solution is found, we intro-

Algorithm 2: SMT_{Agg}, Benders cut aggregation scheme for a subproblem.

input : $\mathcal{S}_k^\alpha$
output: $\left\{\beta_{k,1}^\alpha, \beta_{k,2}^\alpha, \cdots\right\}$
init : $i = 1$, $\mathcal{S}'^\alpha_k = \mathcal{S}_k^\alpha$

1 **while** $\mathcal{O}_k^\alpha = \emptyset$ **do**
2 $\quad \mathcal{O}_k^\alpha, \mathcal{I}_{k,i}^\alpha, \beta_{k,i}^\alpha \leftarrow$ `SMT(`$\mathcal{S}'^\alpha_k$`)`;
3 $\quad$ **if** $\mathcal{I}_{k,i}^\alpha \neq \emptyset$ **then**
4 $\quad\quad \mathcal{S}'^\alpha_k \leftarrow$ `Modify(`$\mathcal{S}'^\alpha_k, \mathcal{I}_{k,i}^\alpha$`)`;
5 $\quad i \leftarrow i + 1$;
6 **return** $\left\{\beta_{k,1}^\alpha, \beta_{k,2}^\alpha, \cdots\right\}$

duce a heuristic to compute further Benders cuts from a single subproblem $\mathcal{S}_k^\alpha$ within the same iteration α.

While the infeasibility proof $\mathcal{I}_k^\alpha$ for $\mathcal{S}_k^\alpha$ depends on $\mathcal{O}_\mathcal{M}^\alpha$, a Benders cut generated from $\mathcal{S}_k^\alpha$ does not. We exploit this property to perform an aggregation of Benders cuts in a single iteration of the Benders scheme. At each iteration, after generating a first cut with SMT, we modify $\mathcal{S}_k^\alpha$ and call SMT again on a modified subproblem $\mathcal{S}'^\alpha_k$ to possibly get another cut. We modify $\mathcal{S}_k^\alpha$ such that for the modified problem $\mathcal{S}'^\alpha_k$ the infeasibility proof we detected in the first place is no longer valid. Under this condition, when we call SMT on the modified subproblem, a new infeasibility proof is returned in case the subproblem is still infeasible.

We perform the above procedure iteratively as illustrated in Algorithm 2. For each new infeasibility proof we find (line 2), we modify $\mathcal{S}_k^\alpha$ (line 4) by a specific function `Modify`, and call SMT to see whether further cuts can be found. The modifications to $\mathcal{S}_k^\alpha$ must be such that after we modified, the set of cycles $\mathcal{I}_k^\alpha$ contains at least one non-positive cycle on the modified graph G'^α_k and is no longer an infeasibility proof by Definition 1. Another call of SMT must then either return a different proof, from which we can derive another Benders cut, or a feasible solution (line 3).

Concerning the modification operated, we could theoretically explore the subproblem to the fullest if for every $\mathcal{I}_k^\alpha$ we find, we would explore modifications to $\mathcal{S}_k^\alpha$ based on every cycle $\gamma \in \mathcal{I}_k^\alpha$ for which it holds $\gamma \neq \bar{\gamma}$. Practically this would lead to exponentially many calls of SMT, and instead

we opt for a heuristic procedure implementing a single modification per infeasibility proof found.

Modifications of $\mathcal{S}_k^\alpha$ are carried out on the corresponding graph G_k^α. In particular, we pick a cycle $\gamma \in \mathcal{I}_k^\alpha$, $\gamma \neq \bar{\gamma}$ and we modify the length of one of its arcs, such that the length of the cycle becomes non-positive. This mimics the effect of our logic Benders cut to $\mathcal{M}$.

The cycle to be modified must contain at least one coordination arc, i.e., $\gamma \neq \bar{\gamma}$. Otherwise, γ contains only constraints of the original problem; modifying them would mean to change the original problem. To make a cycle non-positive in length we relax a coordination constraint, i.e., change the length of a coordination arc in the cycle, which we assume equal to an increase in runtime delay. Thus, with our heuristic we choose to modify the cycle with the smallest length.

For the chosen cycle, we modify its length as we change the length of a coordination arc, such that the cycle becomes exactly zero length. The cycle can contain one or two coordination arcs. In case two are present, we prioritize changing the one outgoing from the time origin t_0, against the one ingoing to the time origin t_0. We motivate this choice as follows. Outgoing arcs from the time origin represent earliest bounds on nodes they point to. Decreasing the length of such an arc (to decrease the length of a cycle) means to allow scheduling an event earlier, i.e., let a train enter or exit the subnetwork associated with the subproblem earlier. Ingoing arcs to the time origin are latest bounds on nodes they originate from. Decreasing the length of such an arc, which is already negative (to decrease the length of a cycle), means to accept the delay of an event, i.e., a train may enter or exit the subnetwork associated with a subproblem later in time. In our aggregation, we would like to anticipate the effect of our Benders cut on solutions of the master problem. Scheduling a train earlier aligns more with the objective of the master, compared to delaying a train. We therefore assume that the master might favor this former case. When aggregation of cuts is considered, SMT_{Agg} (Algorithm 2) replaces the simple SMT algorithm of Section 3.6.2 to analyze all subproblems.

3.8 COMPUTATIONAL EXPERIMENTS

For a computational study of our logic-based Benders decomposition, a test case of SBB is used. The railway infrastructure of the test case represents the exact microscopic topology of the Swiss railway network in the related area. We analyze the performance of the decomposition with and without the

	Centralized	Master	Per Subproblem			Per Train		
Property	Total	Total	Avg.	Min.	Max.	Avg.	Min.	Max.
Trains	361	361	63.2	6	183	-	-	-
Areas	45	-	-	-	-	8.8	1	19
Time Windows [min]	46.7	47	46.6	45	128	46.5	46	153
Stops	3280	0	61.9	0	635	9.1	2	16
Transfer Connections	865	0	18.5	0	241	3	0	8
Blocks	955	313	15	1	110	124.3	21	270
Route Alternatives	5441	0	120.9	0	1124	15	0	45
Resource Conflicts	47588	10478	1265.2	0	12360	262.9	7	696
Max. Conflict Cmp.	361	248	49	0	183	89.1	8	234
Events (Nodes)	78758	38645	1551.6	27	13043	217	42	465
Relations (Edges)	411940	136588	7996.8	27	75993	-	-	-

TABLE 3.1: Overview on instance characteristics.

aggregation of Benders cuts and compare it to several benchmarks. All experiments are performed on a machine equipped with an Intel i7@2.60GHz and 32.0GB of RAM.

3.8.1 *Scenario*

The test case describes a selection of train services and related infrastructure from the original timetable of SBB over a planning horizon of roughly 9 hours. The test case spans the geographic triangle between the cities Zurich, Lucerne and Chur in Switzerland, which are around 150 km away from each other; and includes the bigger stations Zurich, Lucerne, Chur, Arth-Goldau and Zug. The test case is given as an instance of MRTP as described in Section 3.3. A solution of MRTP is a detailed schedule for all operations, that satisfies all operational and safety requirements.

Table 3.1 reports characteristics on the original test case, for the centralized problem; and decomposed, for the master, per subproblem (in minimum, maximum and average values), as well as per train (again, minimum, maximum and average values). Time windows reported in Table 3.1 are those restricting relevant events. The window size is determined by industrial input, to be roughly 45 minutes. In Table 3.1, we further report

Property	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Route Alternatives	488	947	1418	1898	2469	3100	3648	4193	4795	5441
Resource Conflicts	515	1874	3985	6753	10142	16010	21919	29296	37784	47588

TABLE 3.2: Overview on the centralized instance characteristics over different number of trains.

the size of the maximal conflict component (Max. Conflict Cmp.). We report it as a measure of complexity for the original test case, describing how much traffic is interacting. The value is computed by means of an undirected graph where each train service is represented by a node; two nodes are connected if the associated train services have at least one resource conflict. The maximal component is the largest connected component in such graph. From Table 3.1 we see that over 9 hours of planning horizon, the operations of all train services are in dependency to each other. The decomposition breaks some of these dependencies, as for the master but also for the avg. subproblem not all train services are conflicting with each other. We can identify situations of having zero stops (no stations) or more stops than trains (multiple stations) in a subproblem. Concluding, we have indeed subproblems that contain none, exactly one, or multiple stations.

To explore the behavior and performance of our decomposition on instances of different complexity, we derive multiple instances from the original test case by decreasing the number of trains to different percentages. 100% is the original test case. Trains are removed randomly but successively: e.g., the 20% instance contains all trains of the 10% instance. In Table 3.2 we report the number of route alternatives and resource conflicts for those instances which grow roughly quadratically. These mimic to some extent the complexity of those instances.

All instances are decomposed according to Section 3.6.1. In Figure 3.2 we illustrate the decomposition of the original test case. Each node in the graph represents one subproblem. The shape and shade of a node denotes the amount of blocks considered in a subproblem. Solid arcs indicate subproblems whose blocks are adjacent in the network. Dashed arcs indicate subproblems connected only by infrastructure of the master problem. The arc labels report the number of trains running between two connected subproblems within the period of planning.

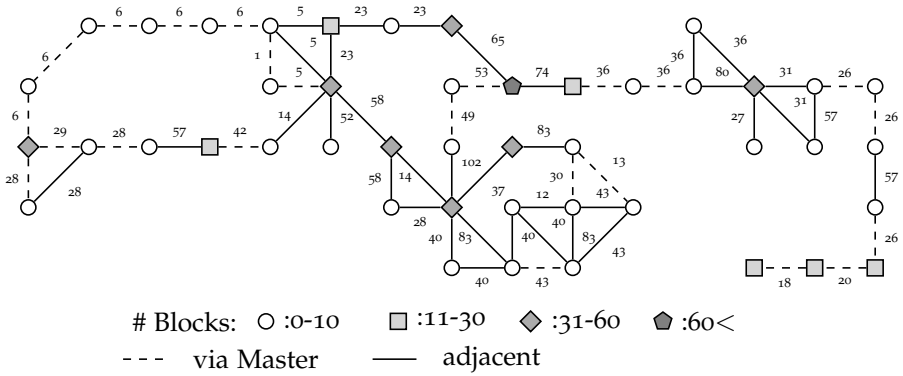

FIGURE 3.2: Connectivity and Usage of the Infrastructure.

3.8.2 *Performance Experiments*

In order to assess the performance of our Benders decomposition, all instances are solved by six different solution approaches. The centralized Problem (3.2) is solved with a constraint programming solver (IBM CPO (CPO), [21]) and a mixed-integer solver (GUROBI (GRB), [15]) to provide benchmarks. For CPO the disjunctive program (3.2) is implemented using the logic functionalities of the solver. For GRB a big-M formulation of (3.2) is used. For the decomposition, we evaluate versions without and with aggregation of Benders cuts; we refer to them as simple and aggregated decompositions respectively. In all decomposition approaches, the master problem is solved as a big-M formulation by GRB. For solving subproblems we use our proposed SMT approach and also the commercial tool GUROBI-IIS (GRB-IIS) that is part of GUROBI [15] as a benchmark on the subproblem evaluation. The tool computes an irreducible infeasible subset of constraints in a problem. We can translate such subset into a logic Benders cut. All experiments are computed with a time limit of 12 hours. For decomposed approaches we aborted runs where within 2 hours no Benders cut for a single subproblem could be found.

We report computation results in Table 3.3 for all approaches (columns) over all instances (rows). We indicate the fastest approach(es) per instance in bold. All solutions are optimal, otherwise we report the optimality gap if a feasible solution was found within 12 hours. Table 3.3 shows the original test case (100%) is not solvable to optimality by any of the approaches within 12 hours. In current practice of the industry, near optimal timetables

	Centralized		Aggregated				Simple			
	CPO	GRB	SMT		GRB-IIS		SMT		GRB-IIS	
Trains	Total [s]	Total [s]	Total [s]	Iter.	Total [s]	Iter.	Total [s]	Iter.	Total [s]	Iter.
10 %	286.7	3.7	**0.3**	2	4.5	2	**0.3**	3	1.4	2
20 %	5530.8	32.3	**0.9**	2	138.8	4	1.1	4	16.9	5
30 %	**8.7%*	135.5	**5.1**	5	996.2	4	7.0	12	122.5	9
40 %	-	2155.9	**23.9**	8	.	.	40.2	37	3214.0	45
50 %	-	**1.99%*	**31.5**	5	.	.	97.5	58	3766.9	61
60 %	-	**2.52%*	**78.6**	7	.	.	387.7	92	.	.
70 %	-	**5.97%*	**220.7**	11	.	.	1512.4	155	.	.
80 %	-	**15.64%*	**582.4**	11	.	.	4840.6	220	.	.
90 %	-	**101.82%*	->41		.	.	->260		.	.
100 %	-	-	->20		.	.	->228		.	.

[*] No optimality within 12h [*Opt. Gap*]. [-] No solution within 12h.
[.] No Infeasibility Proof within 2h.

TABLE 3.3: Computation for Centralized, Aggregated, and Simple Decomposition.

are designed by manually merging optimal timetables of smaller regions into one consistent network-wide timetable.

Concerning the centralized approaches, GUROBI (GRB) manages to solve instances to optimality up to 40% of trains and further provides feasible solutions till 90% of trains within 12 hours. IBM CPO (CPO) provides optimal solutions up to 20% and a feasible solutions for 30% of trains. Beyond this, no solutions can be found by CPO. In Section 3.8.4 we provide a deeper analysis why state-of-the-art solvers struggle with a centralized approach on MRTP.

Comparing centralized with decomposed approaches, a clear advantage of decomposed approaches is visible in Table 3.3. In the best case, the decomposed approaches allow to find a solution on instances of size twice as large, 40 times faster. The ratio between the computation time of the best centralized approach and the best decomposed approach grows with the increase of complexity (trains), showing the scalability potential of the proposed approach. Compared to centralized approaches, decomposed approaches are not able to return a feasible solution if not converged.

Trains		10%	20%	30%	40%	50%	60%	70%	80%
Simple	Iterations	3	4	12	37	58	92	155	220
	Total Cuts	4	8	46	129	200	342	548	716
	Avg Master [s]	0.04	0.13	0.39	0.83	1.34	3.69	8.98	20.72
Aggregated	Iterations	2	2	5	8	5	7	11	11
	Total Cuts	8	36	139	421	485	833	1255	1786
	Avg Master [s]	0.05	0.18	0.51	1.43	2.80	6.88	12.85	39.45

TABLE 3.4: Aggregation of Benders Cuts.

Comparing SMT with GUROBI-IIS, we see the advantage of SMT. We explain the difference in performances by the following two reasons. First, GUROBI-IIS computes minimal infeasibility proofs by reducing an initially large set of constraints. The reduction itself is computationally expensive as it must be carried out till a minimal proof has been found. The approach we use cannot be proven to find a minimal proof, and this results in a faster process. Second, GUROBI-IIS is designed to work for different types of problems and constraints, which requires generality of the method. Instead, our approach is specifically designed for SAT problems.

3.8.3 *Benders Cut Generation and Aggregation*

In this section we discuss the aggregation of cuts in the decomposed approaches of SMT. Approaches based on GUROBI-IIS are neglected due to the shown performance.

In Table 3.4 we discuss the performance of the simple and aggregated SMT. Columns identify the instances. Rows report (grouped by simple and aggregated) the amount of iterations till convergence; amount of total cuts generated by all subproblems over the entire search process; and the average of computation time of the master, over all iterations. The results clearly show the effectiveness of aggregation. The forty times faster computation times reported in Table 3.3 are due to much less iterations performed. On the other hand, each iteration itself is slightly slower. The number of cuts reported for the simple case represent a minimal amount of cuts necessary to converge. The aggregated case results in 2 to 3 times more cuts compared to that. This increases by a comparable factor the average time spent for computing the master solution. A different choice of `Modify` heuristic might

Trains		10%	20%	30%	40%	50%	60%	70%	80%
Simple	Relative Cuts	3	6	33	87	120	216	314	418
	Absolute Cuts	1	2	4	10	19	38	77	88
	Mixed Cuts	0	0	9	32	61	88	157	210
	Cut Size	1.50	1.63	1.50	1.99	2.07	2.07	2.25	2.31
Aggregated	Relative Cuts	7	32	94	343	357	603	863	1118
	Absolute Cuts	1	2	6	13	28	52	73	103
	Mixed Cuts	0	2	39	65	100	178	319	565
	Cut Size	1.38	1.61	2.02	2.53	2.52	2.52	2.80	3.54

TABLE 3.5: Classification of Benders Cuts.

result in less additional cuts and less iterations, for an even faster total computation time.

We finally study the Benders cuts, categorizing them by the type of precedence relations in the disjunction. As indicated in Section 3.5.3, the nodes in the linear precedence relation (3.10) might involve t_0, or not, depending on the number of coordination arcs in a cycle γ. We call the case of a precedence relation involving t_0, i.e., where there is one coordination arc in γ, an absolute relation. A relative relation is instead a precedence relation on two events of M_S, i.e., there are two coordination arcs in γ. A relative relation is the consequence of interactions among train services inside a subproblem, which is not yet represented in the constraints of the master. An absolute relation results from a train service originating or terminating in a subproblem. Accordingly, we define a relative Benders cut as a cut with terms in disjunction only associated with relative relations. An absolute cut is a cut with only associated absolute relations; a mixed cut is a cut, which includes terms associated with both types of precedence relations.

We report in Table 3.5 some statistics on the types of cuts observed. The rows describe the amount of relative, absolute, mixed cuts out of the amount of total cuts. Moreover, the average size of cuts is reported, i.e., number of terms in disjunction. Heavier traffic (i.e., a larger percentage of trains running) results on average in larger cuts as more train interactions exist. The aggregation procedure in general results in larger cuts. Exploring multiple conflicts beyond the smaller conflicts, which are in general those first detected by SMT, larger conflicts are found.

The amount of absolute cuts depends on the amount of train services starting or terminating in a subproblem, which is in general less than train services passing through a subproblem (which would result in relative cuts). The master has only limited influence on those train services as their starting or termination event is fixed inside the subproblem. As such most conflicts amongst these trains occur in any case and must be resolved by the master for convergence. This can be seen in a similar number of absolute cuts in the simple and aggregated approach. The same does not hold for relative and mixed cuts, where conflicts among train services in the subproblem strongly depend on the master solution. The consequence of the above is that the aggregation procedure proportionally and in total generates more mixed and relative cuts than absolute cuts.

3.8.4 *Impact of Routing Complexity*

In order to better understand the complexity of MRTP, we conducted the same experiments as in Table 3.3 but excluded routing decisions. To this purpose, we considered for all routing areas a single route alternative that is the default route alternative according to SBB. Table 3.6 reports computational results in the non-routing cases. Obviously, the non-routing problem results in larger runtime delays, growing for instance from 146 to 220 minutes and from 532 to 1883 minutes (respectively for 40%; for 80% of trains).

For the centralized approaches we see a striking improvement in the computation time to find the optimal solution. This shows how routing decisions are a significant factor in the complexity of MRTP. For the CPO solver, a detailed analysis revealed that in general good solutions are found quickly, roughly after 100 seconds of computation, but an optimal solution could not be determined within 12 hours for the more complex instances. This underlines the fact that constraint programming solvers are in general very efficient for problems of feasibility but not so much for optimization.

For decomposed approaches we see an opposite trend in Table 3.6. These approaches require more computation time compared to the experiments of Table 3.3, and compared to GRB on the same instances. In the original MRTP, routing decisions are one factor for the superior performance of decomposed approaches over centralized approaches. Our decomposition represents the solution space of subproblems in the master, by additional restrictions in form of Benders cuts. The larger solution space of a subproblem, when including routing decisions, is likely to impose less restrictions (i.e. less Benders cuts) onto the master. Fewer Benders cuts result in general

	Centralized		Aggregated				Simple			
	CPO	GRB	SMT		GRB-IIS		SMT		GRB-IIS	
Trains	Total [s]	Total [s]	Total [s]	Iter.	Total [s]	Iter.	Total [s]	Iter.	Total [s]	Iter.
10 %	1.2	**0.1**	0.3	2	1.3	2	0.3	2	1.2	2
20 %	210.2	**0.3**	0.9	2	42.9	3	1.0	3	14.9	5
30 %	1338	**1.0**	2.5	3	145.2	3	12.6	16	83.3	16
40 %	6292	**2.2**	16.5	6	1894	7	60.0	32	608.6	51
50 %	*15.4%	**3.3**	39.4	8	5317	8	299.7	75	1255	76
60 %	*20.3%	**6.2**	147.4	8	8024.3	7	1321.7	98	3737.5	128
70 %	-	**45.9**	1131.7	17	.	.	7432.8	196	25748.6	289
80 %	-	**135.6**	25138.1	15	.	.	-	-	.	.
90 %	-	**3411.3**	-	-	.	.	-	-	.	.
100 %	-	**6759.2**	-	-	.	.	-	-	.	.

[*] No optimality within 12h [*Opt. Gap*]. [-] No solution within 12h.
[.] No Infeasibility Proof within 2h.

TABLE 3.6: Computations for Centralized, Aggregated, and Simple Decomposition without Routing.

in a better performance of Benders decomposition. In the instances without routing, the amount of iterations is comparable with the routing instances, but the time spent per iteration is longer. Gurobi requires considerably more time to solve the more constrained master problems; determining the cuts takes slightly more time.

3.9 CONCLUSION

In this paper we introduce a logic-based Benders decomposition for the problem of microscopic railway timetable planning. Our decomposition differs from the literature as we consider subproblems that are equal, in structure of constraints, to the master (and the centralized problem). This results in larger degree of freedom in the decomposition. We limited our research in this paper to a decomposition where subproblems are feasibility problems only. For the decomposition we introduce a logic feasibility Benders cut for a class of disjunctive subproblems. We propose an efficient

algorithm to generate the logic Benders cut, based on techniques from the field of satisfiability solving. Numerical experiments consolidate our contribution in terms of performance compared to centralized approaches and confirm the efficiency of the proposed cut generation scheme compared to commercially available tools. We are able to solve realistic instances up to twice the size, 40 times faster than a centralized approach using a commercial solver. The decomposition works best when the subproblems have relatively large solution spaces, such that the master is relatively simpler, and the complexity of subproblems can be parallelized.

While we target the solution of microscopic railway timetable planning, our generalization of Benders decomposition is not limited to this application. In fact, the logic Benders cuts and the proposed algorithms to generate them can be used for any disjunctive program of the form of Problem (3.2). This is in fact a generalization of both job-shop scheduling as in [31] and multi-purpose machine job-shop scheduling as in [7], thus we believe applicability can reach out to the field of general scheduling.

Future research directions include the study of other aggregation heuristics. We identified how our aggregation creates redundant cuts, which increase the computational effort of the master. We also assume that a way to determine more minimal cuts could improve further the computational speed of our decomposition. In a different direction, studies on decomposing the railway timetable planning problem can provide more insights on how the decomposition of an instance affects the performance of the algorithm. Throughout this paper the instances used for our experiments were always decomposed in the same way, based on industrial inputs. It is left as an open question, whether a different geographical choice of subproblems could bring further benefits. Also, decompositions in time could be a potential alternative to a spatial decomposition.

ACKNOWLEDGMENTS

The authors thank the colleagues from SBB for the useful discussions and support during the implementation. The authors thank Alessio Trivella for helping with the manuscript. This project was supported by the SBB ETH Zürich Foundation.

REFERENCES

1. Asin, R., Nieuwenhuis, R., Oliveras, A. & Rodriguez-Carbonell, E. *Efficient generation of unsatisfiability proofs and cores in SAT* in *International Conference on Logic for Programming Artificial Intelligence and Reasoning* (2008), 16.
2. Balas, E. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics* **89**, 3 (1998).
3. Balas, E. Machine Sequencing via Disjunctive Graphs : An Implicit Enumeration Algorithm. *Operations Research* **17**, 941 (1969).
4. Benders, J. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* **4**, 238 (1962).
5. Borndörfer, R. & Schlechte, T. *Models for Railway Track Allocation* in *7th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS'07)* (2007).
6. Brännlund, U., Lindberg, P. O., Nõu, A. & Nilsson, J.-E. Railway Timetabling Using Lagrangian Relaxation. *Transportation Science* **32**, 358 (1998).
7. Brucker, P. & Schlie, R. Job-shop scheduling with multi-purpose machines. *Computing* **45**, 369 (1990).
8. Caprara, A., Fischetti, M. & Toth, P. Modeling and Solving the Train Timetabling Problem. *Operations Research* **50**, 851 (2002).
9. Codato, G. & Fischetti, M. Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. *Operations Research* **54**, 756 (2006).
10. Corman, F., D'Ariano, A., Pacciarelli, D. & Pranzo, M. Centralized versus distributed systems to reschedule trains in two dispatching areas. *Public Transport* **2**, 219 (2010).
11. Cotton, S. & Maler, O. *Fast and flexible difference constraint propagation for DPLL (T)* in *International Conference on Theory and Applications of Satisfiability Testing* (2006), 170.
12. Davis, M., Logemann, G. & Loveland, D. A Machine Program for Theorem-Proving. *Communications of the ACM* **5**, 394 (1962).
13. Fang, W., Yang, S. & Yao, X. A Survey on Problem Models and Solution Approaches to Rescheduling in Railway Networks. *IEEE Transactions on Intelligent Transportation Systems* **16**, 2997 (2015).

14. Geoffrion, A. Generalized Benders Decomposition. *Journal of Optimization Theory and Applications* **10**, 237 (1972).

15. Gurobi Optimization, L. *Gurobi Optimizer Reference Manual* 2021.

16. Harjunkoski, I. & Grossmann, I. E. Decomposition Techniques for Multistage Scheduling Problems using MIP and CP Methods. *Computers & Chemical Engineering*. **26**, 1533 (2002).

17. Harjunkoski, I., Jain, V. & Grossmann, I. E. Hybrid mixed-integer/constraint logic programming strategies for solving scheduling and combinatorial optimization problems. *Computers & Chemical Engineering*. **24**, 337 (2000).

18. Herrigel, S., Laumanns, M., Nash, A. & Weidmann, U. Hierarchical Decomposition Methods for Periodic Railway Timetabling Problems. *Transportation Research Record* **2374**, 73 (2013).

19. Hooker, J. N. Planning and Scheduling by Logic-Based Benders Decomposition. *Operations Research* **55**, 588 (2007).

20. Keita, K., Pellegrini, P. & Rodriguez, J. A three-step Benders decomposition for the real-time Railway Traffic Management Problem. *Journal of Rail Transport Planning & Management* **13**, 100170 (2020).

21. Laborie, P., Rogerie, J., Shaw, P. & Vilím, P. IBM ILOG CP optimizer for scheduling. *Constraints* **23**, 210 (2018).

22. Lamorgese, L. & Mannino, C. A non-compact formulation for job-shop scheduling problems in traffic management. *Operations Research* **67**, 1503 (2019).

23. Lamorgese, L. & Mannino, C. An Exact Decomposition Approach for the Real-Time Train Dispatching Problem. *Operations Research* **63**, 48 (2015).

24. Lamorgese, L., Mannino, C. & Piacentini, M. Optimal Train Dispatching by Benders'-Like Reformulation. *Transportation Science* **50**, 910 (2016).

25. Leutwiler, F. & Corman, F. A logic-based Benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research* **303**, 525 (2022).

26. Liu, L. & Dessouky, M. A decomposition based hybrid heuristic algorithm for the joint passenger and freight train scheduling problem. *Computers & Operations Research* **87**, 165 (2017).

27. Logic-based Benders decomposition. *Mathematical Programming* **96**, 33 (2003).

28. Luan, X., De Schutter, B., Meng, L. & Corman, F. Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. *Transportation Research Part B Methodological* **141**, 72 (2020).

29. Marcelli, E. & Pellegrini, P. Literature Review Toward Decentralized Railway Traffic Management. *IEEE Intelligent Transportation Systems Magazine* **13**, 234 (2021).

30. Marques-Silva, J. P. & Sakallah, K. A. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers* **48**, 506 (1999).

31. Mascis, A. & Pacciarelli, D. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operations Research* **143**, 498 (2002).

32. Morgado, A., Heras, F., Liffiton, M., Planes, J. & Marques-Silva, J. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints* **18**, 478 (2013).

33. Moura, L. d., Dutertre, B. & Shankar, N. *A tutorial on satisfiability modulo theories* in *International Conference on Computer Aided Verification* (2007), 20.

34. Wolsey, L. A. & Nemhauser, G. L. *Integer and combinatorial optimization* (John Wiley & Sons, 1999).

35. Zhang, L., Madigan, C., Moskewicz, M. & Malik, S. *Efficient conflict driven learning in a boolean satisfiability solver* in *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers* (2001), 279.

APPENDIX

3.10 AN EXAMPLE OF DECOMPOSITION

In the following, we give an example of the decomposition scheme considering a single subproblem. In Figure 3.3 the infrastructure of the example is depicted, identifying blocks by segments delimited by small vertical lines. We also report the three trains of the example and indication of the respective routes. The infrastructure (i.e., the blocks) belonging to the subproblem is indicated in green. Whenever two trains use the same infrastructure element on their route, there is a resource conflict to be resolved. The route of Train B goes through a routing area, as the train may use both platforms of the station. The trains do not stop at the station. Trains A, B, C enter the area under consideration (including the master) respectively at 0, 0, and 1 time units. The time windows for relevant events are large enough to be ignored in what follows. The example has three resource conflicts, two between Train A and B, one between Train B and C. This last conflict depends on the routing choice for Train B.

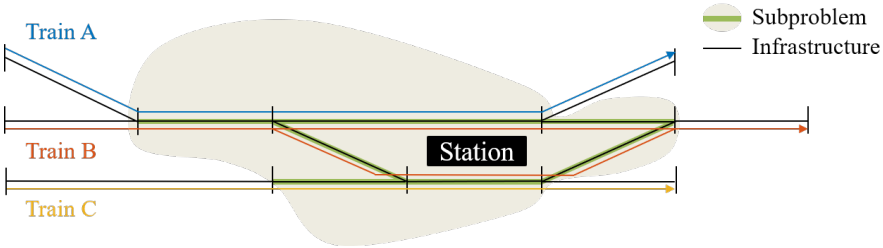

FIGURE 3.3: Railway Infrastructure of the example with 3 trains and indicated routes.

Figure 3.4 shows the disjunctive graph of the example illustrated in Figure 3.3, from a centralised point of view. The nodes represent start event times and the 0 node represents the time origin. Fixed precedence relations are illustrated as solid arcs and colored according to the train as in Figure 3.3. Black dashed arcs illustrate selectable precedence relations of resource conflicts. The route alternatives (choices) of Train B are illustrated as dashed and dotted arcs in color. Since the example contains a resource conflict on a route alternative, the auxiliary nodes $\{8_{in}, 9_{out}, 12_{in}, 13_{out}\}$ are introduced according to Section 3.4. For simplicity we assume black dashed arcs have zero length. Short (possibly diagonal) colored arcs have unitary length (1

min) and long colored arcs double the length (2 min). The colored arcs connected to the auxiliary nodes $\{8_{in}, 9_{out}, 12_{in}, 13_{out}\}$ have 0 length. The solid black arcs from the time origin have length as depicted. Arcs towards the time origin are neglected in the example, as we assume that all latest bounds of any time window to be tolerant enough, not to become restrictive for any intermediate or final optimal solution.

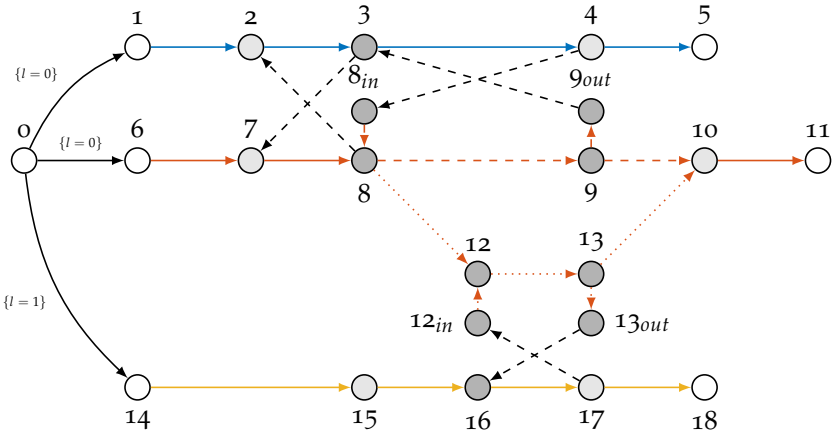

FIGURE 3.4: Disjunctive graph of the centralized problem $\mathcal{C}$ of the example with 3 trains.

The decomposition is illustrated in Figure 3.4 through the shading of nodes. All nodes shaded in gray are variables of $\mathcal{S}$. $M_{\mathcal{S}} = \{2, 4, 7, 10, 15, 17\}$ are variables of $\mathcal{S}$ and $\mathcal{M}$, and are shaded in light gray. All edges in-between (light or dark) shaded nodes belong to $\mathcal{S}$, all other edges are part of $\mathcal{M}$. For $\mathcal{M}$ we assume an objective minimizing the sum of t_5, t_{11} and t_{18}.

In the iterative process of decomposition, when solving the decomposed problem $\mathcal{S}^\alpha$, for any iteration α, the variables $M_{\mathcal{S}}$ are fixed to the values of the latest master solution by edges according to the coordination constraints (3.7). Figure 3.5 illustrates the disjunctive graph of $\mathcal{S}^\alpha$ at iteration α.

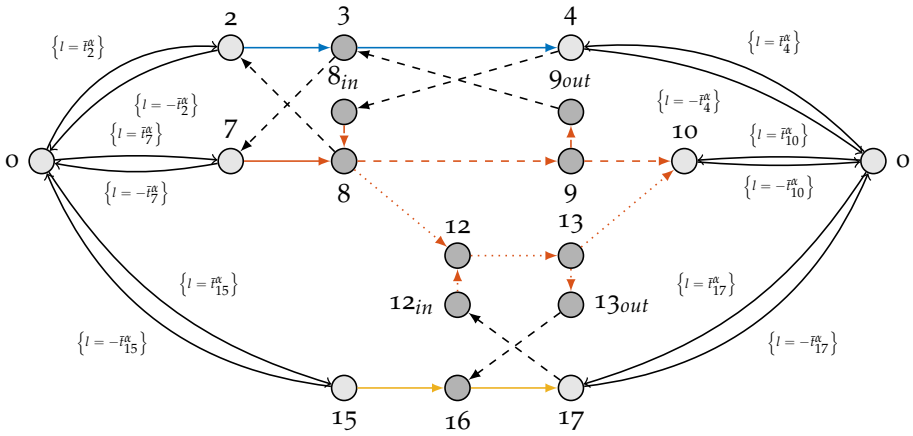

FIGURE 3.5: Disjunctive graph G^{α} of subproblem $\mathcal{S}^{\alpha}$.

$\mathcal{M}$ does not contain any decision or selectable precedence relations. All ordering and routing decisions belong to $\mathcal{S}$ and are the following:

$$
\begin{aligned}
D_1 &= \{W_1 = \{(8_{in},8),(8,9),(9,9_{out}),(9,10)\}; \\
&\qquad W_2 = \{(8,12),(12_{in},12),(12,13),(13,13_{out}),(13,10)\}\} \\
D_2 &= \{W_3 = \{(3,7)\}; W_4 = \{(8,2)\}\} \\
D_3 &= \{W_5 = \{(4,8_{in})\}; W_6 = \{(9_{out},3)\}\} \\
D_4 &= \{W_7 = \{(13_{out},16)\}; W_8 = \{(17,12_{in})\}\}.
\end{aligned}
$$

In the following we show an exemplary sequence of iterations till convergence. The iterations shown are one possibility to converge; the usage of different infeasibility proofs would have let to different Benders cuts and intermediate solutions in the scheme till the final solution. We will show the disjunctive graph of the master problem at every iteration to illustrate how the subproblem is adequately and increasingly represented in the master problem.

Iteration 1 At first $\mathcal{M}^1$ has to be solved. The disjunctive graph of $\mathcal{M}^1$ is illustrated in Figure 3.6. $\mathcal{M}^1$ only contains basic knowledge about $\mathcal{S}$ in form of minimal traveling times through the subproblem, without consideration of resource conflicts. We show these as arcs with depicted length in Figure 3.6.

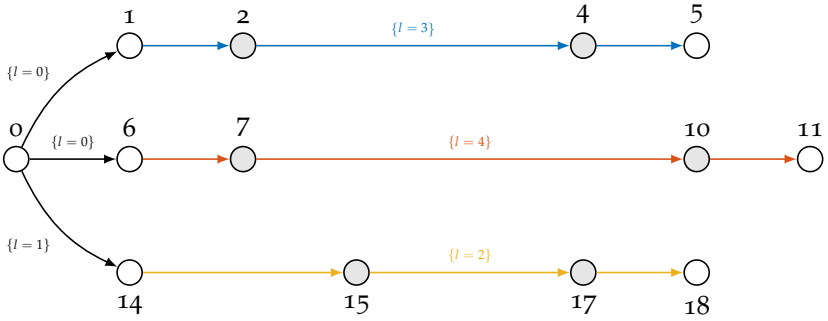

FIGURE 3.6: Disjunctive graph of $\mathcal{M}^1$.

Solving $\mathcal{M}^1$ yields the following optimal solution:

$$\mathcal{O}^1_{\mathcal{M}} = \{t_1 = 0, t_2 = 1, t_4 = 4, t_5 = 5, t_6 = 0, t_7 = 1, t_{10} = 5, t_{11} = 6, t_{14} = 1, \\ t_{15} = 3, t_{17} = 5, t_{18} = 6\}.$$

With $\mathcal{O}^1_{\mathcal{M}}$ we can find the following two cycles on the disjunctive graph G^1 of $\mathcal{S}^1$, which can be found in Figure 3.5,

$$\gamma_1 = \{(0,2),(2,3),(3,7),(7,0) \,|\, l = 1\}$$
$$\gamma_2 = \{(0,7),(7,8),(8,2),(2,0) \,|\, l = 1\}.$$

The set of cycles $\mathcal{I}^1 = \{\gamma_1, \gamma_2\}$ satisfies Definition 1 and is a valid infeasibility proof. The resulting Benders cut (a relative cut, as no term refers to t_0), which we will identify in color green as β^1, is:

$$\beta^1 = (t_7 - t_2 \geq 1) \bigvee (t_2 - t_7 \geq 1)$$

Iteration 2 Next $\mathcal{M}^2$ with β^1 has to be solved. We illustrate its (extended, compared to $\mathcal{M}^1$) generalized disjunctive graph in Figure 3.7.

$\mathcal{M}^2$ has two optimal solutions, we randomly choose the one where $(t_7 - t_2 \geq 1)$ in β^1 is satisfied,

$$\mathcal{O}^2_{\mathcal{M}} = \{t_1 = 0, t_2 = 1, t_4 = 4, t_5 = 5, t_6 = 0, t_7 = 2, t_{10} = 6, t_{11} = 7, t_{14} = 1, \\ t_{15} = 3, t_{17} = 5, t_{18} = 6\}.$$

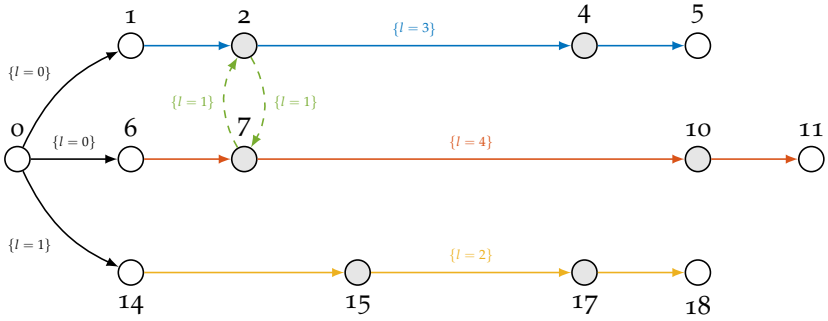

FIGURE 3.7: Disjunctive graph of $\mathcal{M}^2$.

With $\mathcal{O}^2_{\mathcal{M}}$ we can find the following four cycles on the disjunctive graph G^2 of Figure 3.5:

$$\gamma_3 = \{(0,2),(2,3),(3,4),(4,8_{in}),(8_{in},8),(8,9),(9,10),(10,0)\,|\,l=1\}$$
$$\gamma_4 = \{(0,7),(7,8),(8,9),(9,9_{out}),(9_{out},3),(3,4),(4,0)\,|\,l=3\}$$
$$\gamma_5 = \{(0,7),(7,8),(8,12),(12,13),(13,13_{out}),(13_{out},16),(16,17),(17,0)\,|\,l=1\}$$
$$\gamma_6 = \{(0,15),(15,16),(16,17),(17,12_{in}),(12_{in},12),(12,13),(13,10),(10,0)\,|\,l=1\}$$

The set of cycles $\mathcal{I}^2 = \{\gamma_3,\gamma_4,\gamma_5,\gamma_6\}$ satisfies Definition 1 and is a valid infeasibility proof. Here we like to point out that an alternative infeasibility proof could be $\mathcal{I}^2_{alt} = \{\gamma_2,\gamma_x,\gamma_3,\gamma_5,\gamma_6\}$ where $\gamma_x = \{(7,8),(8,9),(9,9_{out}),(9_{out},3),(3,7)\,|\,l=3\}$ is a cycle where $\gamma = \bar{\gamma}$, i.e., the length of the cycle does not depend on $\mathcal{O}_{\mathcal{M}}$ and would not contribute to the Benders cut, but instead γ_2 would appear again in the resulting Benders cut.

Considering $\mathcal{I}^2$, the resulting Benders cut (again, a relative cut) β^2 is:

$$\beta^2 = (t_{10} - t_2 \geq 6) \bigvee (t_4 - t_7 \geq 5) \bigvee (t_{17} - t_7 \geq 4) \bigvee (t_{10} - t_{15} \geq 4)$$

Iteration 3 Next $\mathcal{M}^3$ with β^1 and β^2 has to be solved. The extended disjunctive graph is illustrated in Figure 3.8. The optimal solution of $\mathcal{M}^3$ satisfies $(t_2 - t_7 \geq 1)$ in β^1 and $(t_{17} - t_7 \geq 4)$ in β^2:

$$\mathcal{O}^3_{\mathcal{M}} = \{t_1 = 0, t_2 = 2, t_4 = 5, t_5 = 6, t_6 = 0, t_7 = 1, t_{10} = 5, t_{11} = 6, t_{14} = 1, t_{15} = 3, t_{17} = 5, t_{18} = 6\}.$$

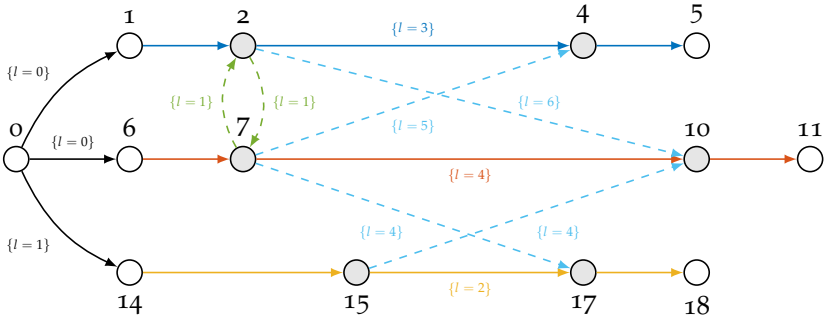

FIGURE 3.8: Disjunctive graph of $\mathcal{M}^3$.

With $\mathcal{O}^3_{\mathcal{M}}$ a feasible solution for $\mathcal{S}^3$ exists:

$$\mathcal{O}^3_{\mathcal{S}} = \{t_2 = 2, t_3 = 3, t_4 = 5, t_7 = 1, t_8 = 2, t_{12} = 3, t_{13} = 4, t_{10} = 5, t_{15} = 3, \\ t_{16} = 4, t_{17} = 5\},$$

and we have found an optimal solution for the original problem.

The solution satisfies constraints within the routing choice W_2 of routing decision D_1, i.e., train B travels via the dotted routing alternative (the lower one in Figure 3.4, via nodes 12 and 13). The constraints of W_4 of ordering decision D_2 are satisfied, i.e., Train B is scheduled before Train A; the ordering decision D_3 is irrelevant (due to the choice on D_1); constraints of W_7 of ordering decision D_4 are satisfied, i.e., Train B is scheduled before Train C.

3.11 LIST OF SYMBOLS

d	Train service
b	Railway infrastructure block
(d, b)	Operation related to the traversal of train d over block b.
$((d, b), (q, p))$	Precedence relation
$f_{db,qp}$	Amount of time separation in the precede relation $((d, b), (q, p))$.
t_{db}	Event time for the start of operation (d, b).
t_0	Time origin. $t_0 = 0$.
H	Relevant Events.
A	Set of all constraints (arcs) in an instance of MRTP.
A_f	Set of fixed constraints (arcs) in an instance of MRTP.
A_s	Set of selectable constraints (arcs) in an instance of MRTP.
W_c	Choice set. Set of selectable arcs.
D_l	Decision set. Set of choice sets.
G	Generalized disjunctive graph.
V	Nodes of G
A	Arcs of G
θ	Selection on a generalized disjunctive graph.
$\mathcal{C}$	Centralized Problem.
$\mathcal{M}$	Master Problem.
$\mathcal{O}_{\mathcal{M}}$	Master solution.
$\mathcal{S}$	Subproblem.
$\mathcal{O}_{\mathcal{S}}$	Subproblem solution.
$\bar{t}_i$	Value of time variable t_i in a solution.
M	Set of variables in $\mathcal{M}$.
$M_{\mathcal{S}}$	Set of variables optimized in $\mathcal{M}$ and fixed in $\mathcal{S}$.
S	Set of variables in $\mathcal{S}$.
α, δ	Iterations of the Benders scheme.
Ω_α	Set of iterations in a Benders scheme from 1 to $\alpha - 1$.
$\{.\}_{\mathcal{M}}$	Element related to master problem.
$\{.\}_{\mathcal{S}}$	Element related to subproblem.
$\{.\}^{\alpha}$	Element related to iteration α.
β	Benders cut.
$\mathcal{I}$	Infeasibility proof.
γ	Set of arcs as a directed cycle.
$\bar{\gamma}$	Set of arcs as a directed cycle, excluding coordination arcs.

4

ACCELERATING LOGIC-BASED BENDERS DECOMPOSITION FOR RAILWAY RESCHEDULING BY EXPLOITING SIMILARITIES IN DELAYS

Leutwiler, F. and Bonet Filella, G. and Corman, F. (2023). Accelerating logic-based Benders Decomposition for Railway Rescheduling by exploiting similarities in delays. Computers and Operations Research, 150:106075.

This is a post-print version of [24], differing from the published paper only in terms of layout and formatting.

Key findings:

- Statistical knowledge about delays in railway systems allows precomputing logic Benders cuts.
- A lazy-constraint scheme includes only useful cuts.
- With statistical measures most beneficial cuts for specific delay instances can be identified apriori.
- The reuse of logic Benders cuts accelerates the Benders decomposition scheme by 20% and can reach a potential speedup of 2.5 compared to centralized approaches.

Author contributions: Model and Decomposition: FL; Theory for Reuse: FL, GB; Implementation: FL, GB; Manuscript preparation and review: FL, FC.

ABSTRACT

The operation of a railway system is subject to unpredictable delays or disruptions. Operators control the railway system to minimize losses in performance. Real-time rescheduling is the adaptation of the schedule for a railway to any unforeseen delay or disturbance and recover an optimal system state. In this work we propose the extension of an existing Benders decomposition scheme used so far for timetabling, to the case of railway rescheduling. We show how to increase its computational speed by a factor 2, by considering libraries of Benders cuts computed for other instances, to be reused in the solution. We show how including extra cuts has to balance a speedup potential, with a general slowdown due to optimization problems of increased sizes. We show that, if input delays are in fact unknown, but come from a known statistical distribution, we can use a similarity measure to identify a-priori the most promising libraries of Benders cuts, which lead to speedups up to 20%.

4.1 INTRODUCTION

In the operation of a railway, unpredictable events and delays are unavoidable and continuously perturb the system. Railway systems are operated based on a timetable, that is carefully designed by solving a timetabling problem, and aims to maximize stability and performance of the railway system during the operation. Railway operators continuously monitor the railway system and identify deviations between plan and reality. In fact, in case an unforeseen event or delay perturbs the railway system, the initially designed timetable is often no longer a suitable plan of operation; degradation of performance, such as delay propagation, and, in worst cases, cancellations and short turning are the result.

The adaptation of the offline planned timetable to an ever-changing situation is termed rescheduling. Despite much rescheduling is still done by hand, automated tools have been developed. Such tools require the definition of a measure of system performance (related to delays), and suitable mathematical modelling of the constraints of the railway system. Typical actions to be considered in rescheduling include retiming, reordering and rerouting of trains over the infrastructure. The most advanced optimization algorithms can compute an optimal or near-optimal adaptation of the original timetable, considering the current perturbed state of the railway

system. While these approaches have proven their academic value in an increasing literature, real-life applications of such sort are in general limited [3]. The problems of timetabling and rescheduling (we refer to both of them as railway scheduling) are similar, and are in fact both NP-hard problems (e.g., [29]). At the same time the problem of rescheduling is faced during the real-time operation of the railway system, where the available time for computations is strongly limited. As a consequence, the size of possible optimization problems for rescheduling is bounded to local areas with rather limited size.

In this work, we propose a technique to find faster the solution to a rescheduling problem, by exploiting similarities of delay in instances of rescheduling, and precomputation.

We use a logic Benders decomposition for scheduling, that has been introduced in [25] and propose a way to improve it for rescheduling. We exploit the fact that in a daily repeating timetable, rescheduling problems for the same area and the same time of the day are all variations derived from the same timetable, and vary only by their input delays, i.e., the real-time delay of trains. Moreover, input delays are in general similar, i.e., they can be statistically characterized and described by a sufficiently large amount of samples. From the academic literature, we see that knowledge of the statistical distribution of delay, and similarity has not been extensively used.

We propose to precompute features on a set of delay instances, and use insight generated in this way during real-time use, to an unknown delay. We specifically use logic Benders cuts, generated in the solution process of Benders decomposition, as output of the precomputation, which can be included (reused) in the solution process of another instance. We assume that some logic Benders cuts that have been generated while solving one instance of rescheduling with a particular situation of input delay, are useful for the computation of a solution to a different instance, where input delays are different but similar to each other. The reused cuts decrease the amount of iterations and the amount of constraints in the master problem of a Benders decomposition; both reduce the total time spent solving the master problem, which is a substantial part of the computational burden in a Benders decomposition, resulting in an overall computational speedup. Logic Benders cuts for the reuse can be precomputed ahead of time and saved into libraries, such that we can reduce the amount of computation necessary at the moment of operation, when time is crucial. Two measures of similarity, applied to the situation of delay, are used to identify which libraries and cuts are actually useful for the reuse. We show that using

precomputed cuts from the most similar instance leads to a speedup up to ∼20%. In general, the potential of reuse is even larger, up to a factor of 2.5, compared to a decomposition scheme with no reuse of cuts; and even larger, when compared to a centralized solution by a commercial solver.

This paper is structured as following. We review related literature in Section 4.2 to highlight the contributions of this work. In Section 4.3, we introduce the railway rescheduling problem addressed. We extend a disjunctive formulation for our rescheduling problem, similar to [25], and apply the logic-based Benders decomposition of [25] in Section 4.4. We describe the details of the proposed approach in Section 4.5. Numerous experiments are provided on real-world examples in Section 4.7. We conclude in Section 4.8.

4.2 RELATED WORK

4.2.1 *Railway Rescheduling*

The literature on railway rescheduling shows a variety of scientific publications. Comprehensive overviews can be found in reviews such as [7] or [12].

From a high-level perspective we can categorize the literature in three aspects. One is granularity. In coarse granular models, i.e., macroscopic models, the network is abstracted into nodes and lines representing stations and connections between the stations respectively (e.g. [17, 35]). Instead, fine granular models, i.e., microscopic models, consider the network at the level of detail of a safety system (e.g., [9], [31], [33]). In those models, conflict free movements of trains over the network as well as routing of trains can be explicitly modelled. We further focus for our work on this second stream of works.

In the aspect of deviation from the planned operations, and information available in real-time about it, we can differentiate the literature in publications working on disturbances and those considering disruptions. Disturbances are usually considered as delays of trains in the magnitude of few minutes (e.g., [10]). Disturbances are common, and their empirical statistics can be collected and described by probability distributions. Disruptions are usually considered as events with more severe effects on the network, e.g., delays above 30 minutes [10] or the unavailability of parts of the network for several hours [5]. Disruptions are typically rare and large; and often dealt with as single cases, as scenarios (where the probability of occurrence cannot be estimated well).

In the aspect of solution approaches, the literature shows a comprehensive variety of methods. The variety ranges from Branch&Bound approaches (e.g., [14]), over Integer (e.g., [8]) and Mixed-Integer Programming (e.g., [31]) solved by commercial solvers, to a large variety of heuristics (e.g., [9]) or collaborative solutions (e.g, [13]), to name a few.

4.2.2 *Decomposition in Rescheduling*

As we use a decomposed approach, we quickly review a variety of decomposition approaches that have been proposed for railway scheduling (either timetabling or rescheduling). In general, we can classify these decompositions into hierarchical and decentralized structures.

In hierarchical decompositions (e.g., [23], [21], [20], [9], [25], [27], [6]), the original scheduling problem is separated into multiple partial optimization problems distributed over several hierarchical layers. In a hierarchical structure, solutions for the optimization on each layer are passed downwards on the hierarchical structure by additional constraints, penalties or by fixing some of the variables of the subordinate optimization problems. This coordination over the different layers happens top-down. Coordination in the opposite direction is usually achieved by means of additional constraints, penalties or heuristics (determining how to proceed, in case of conflicting partial solutions from the subordinate layers).

In decentralized decompositions (e.g., [32], [26], [4]), the original scheduling problem is separated into multiple partial and hierarchically equal optimization problems. The partial optimization problems are solved independently, and an iterative coordination steers them towards a global feasible solution.

4.2.3 *Precomputation and data-driven Approaches*

A few approaches proposed the precomputation in rescheduling, i.e., conduct computations in advance, well before the day of operation, save relevant result actions indexed in some feature space, and are very quick to apply the action for a situation with the same features. In [8], speed profiles for trains are precomputed, which are then, during the real-time application, selected and assembled to a schedule considering the current delays of the railway. In [34], a dynamic impact zone is created for delay situations in which rescheduling actions are necessary. The dynamic impact zone reduces the size of the rescheduling problem, increasing the compu-

tational speed of rescheduling. The dynamic impact zone is computed in real-time, based on precomputed scenarios of possible delay propagation and emerging resource conflicts. In [18], precomputation is done within a machine learning approach, where an algorithm of approximate dynamic programming uses learned costs of different decision. The approach is shown able to control traffic at a single junction.

4.2.4 *Contributions*

We consider railway rescheduling on a microscopic model, to handle small disturbances on a large and heavily utilised railway network. We formulate the rescheduling problem as a disjunctive program [2], similar to [25], encompassing the decisions of retiming, reordering and also rerouting. We extend the hierarchical logic-Benders decomposition of [25] to this case. The contributions of this work are:

1) We show that the reuse of logic Benders cuts is valid, when dealing with rescheduling instances that differ only by their input delay. We propose a modification for precomputed logic Benders cuts, which allows for a broader reuse of precomputed logic Benders cuts. Benders cuts for the reuse can be precomputed offline and stored in a library of cuts.

2) We propose a lazy-constraint approach for dynamically reusing logic Benders cuts, including cuts in the master problem only in case they are violated by the incumbent master solution. In this way, we can consider larger libraries of precomputed cuts, without an unnecessary slow down of the solution process (which would happen if too many cuts are directly included in the master problem). We show how the perfect reuse leads to solving an instance twice as fast as no-reuse.

3) We propose similarity measures to estimate a-priori, based on the input delays, the worth of a logic Benders cut in terms of expected computational speedup when reused. In this way we can reuse only the cuts that are most promising, avoiding to extend a problem by too many cuts. Overall, this achieves a significant speedup of the entire solution process; if a sufficiently similar instance exists in the training set, a 20% faster computation is achieved on average.

4.3 MICROSCOPIC RAILWAY RESCHEDULING

We consider the problem of railway rescheduling on a microscopic representation of the railway infrastructure. This computes an adaptation of

an existing (offline) timetable, which considers the real-time system state (i.e., actual delays of trains). The solution is a new schedule, which obeys the safety regulations of the railway, kinematics of trains and the railway infrastructure. A solution must be available within seconds or few minutes to enable real-time control of the railway system.

This problem has been addressed by many others, and our description of the problem is rather standard in this sense (e.g., [14, 31]). We optimize over times, routes and orders, to minimize total delays of trains, when some input delays arise in the network. We represent times by continuous variables, and routing and ordering decisions as discrete decision. As in [25], we generalize decisions to non-binary sets, and to larger sets, to properly model rerouting actions. Formally, the *discrete decision* is to select one *choice*, out of a finite set of possible choices, specific to the decision. Each choice results in a specific (set of) constraints to be satisfied by the solution.

We model the railway network at the level of blocks, i.e., sections of several hundred meters of tracks. A single operation is the passing of a train over a block, which is associated to an entry (*start*) event, and an exit (*end*) event. An instance is described by a list of trains, their planned timetable, the infrastructure, and some input delays.

The planned timetable results in temporal specifications on events in the problem that are identified as relevant by the operators of the railway (like arrivals/departures at/from stations). Such specifications prescribe a latest time for arrivals, an earliest time for departures, and possibly a minimal/maximal duration between two relevant events for passenger transfers. For an arrival of train after the latest time, there is an *arrival delay*.

The description of the infrastructure defines the dedicated infrastructure for the operation of trains, individually for each train. Dynamics of trains are abstracted into minimal traveling times for each individual block and train. The route of a train is the consecutive sequence of blocks from the train origin to its destination. In between origin and destination, the route may contain routing areas, where different alternative sequences of blocks are available to travel over the network. Each alternative sequence in a routing area provides a connection between the same starting and ending block. For each routing area of a train, a routing decision must be made to select one alternative sequence, that is used by the train in the final schedule. Routing decisions for all routing areas must be made to conclude with a final schedule.

Shared infrastructure gives rise to resource conflicts, i.e., the potential concurrent use of the same block by two trains. For each resource conflict amongst a pair of trains, caused by the respective temporal and infrastruc-

tural limitations, an ordering decision has to be made. For a final schedule, an order has to be decided for each resource conflict, i.e., each pair of trains in conflict.

We consider the real-time delay of trains as *input delay* in our problem. In the literature, delays are most commonly considered only on the entrance of a train into the network (e.g., [31], [14]). In our approach, we do not limit the occurrence of input delay to the first event of the train, i.e., the entrance into the network, but consider the possibility of an input delay at any relevant event of the train, which corresponds to a departure from a station. The input delay on a relevant event is a variation to the earliest departure time given by a temporal specification.

The solution of rescheduling is a schedule which, by suitable update of times, orders and routes for all running trains, minimizes the sum of all arrival delays over all relevant arrival events of the problem, given some statistically characterised input delays. We ignore early arrivals in this work.

4.4 A DECOMPOSED DISJUNCTIVE FORMULATION OF RESCHEDULING

4.4.1 *A Disjunctive Formulation of Rescheduling*

We formulate the microscopic railway rescheduling problem (MRR) as a disjunctive optimization problem [2], likewise to the disjunctive formulation of railway timetabling in [25]. A train d represents a sequence of operations (d, b), where b indicates the related block. We represent the start time of operation (d, b) by the continuous variable $t_{db} \in \mathbb{R}_+$. Time variables for the end of an operation are redundant as jobs cannot be paused or interrupted in MRR.

We use precedence relations to model the constraints of MRR. A precedence relation $((d, b), (q, p))$ is a linear inequality, constraining t_{qp} to occur at least $f_{db,qp}$ time units after t_{db}. Similar to [25] we can characterize precedence relations as *fixed* or *selectable*. The former are standard conjunctive constraints, which must be satisfied in any case by a solution of MRR; the latter model choices of reordering or rerouting.

We model minimal travel times and minimal/maximal durations between relevant events by precedence relations in a set A_f, which are fixed.

With a time origin $t_0 = 0$ we model absolute timing constraints as precedence relations. A set A_h contains (fixed) precedence relations representing earliest departure times $\tau_{lb,i}$ at relevant events, with respect to the origin of

times t_0, updated by (possibly null) input delay μ_i. In the set A_h we consider exactly one precedence relation for each relevant event that is a departure.

A set A_δ contains (fixed) precedence relations, which model the latest arrival time $\tau_{ub,i}$ at relevant events. We use those constraints to determine the arrival delay δ_{db} of a relevant event (d, b).

For the discrete decisions of MRR we use the modelling technique of [25]. The set A_s contains (selectable) precedence relations that are constraints on events, which must hold upon the choices for the discrete decisions. Selectable precedence relations A_s are grouped into choice sets W_c, where precedence relations of the same choice set can only be jointly selected. Choice sets represent the choices of a discrete decision. The choice sets again are grouped into a decision set D_l. The set D_l is the set of all choice sets related to the same decision l. Given D_l we can model a discrete decision l by the following disjunctive constraint,

$$\bigvee_{W_c \in D_l} \bigwedge_{((d,b),(q,p)) \in W_c} \left(t_{qp} - t_{db} \geq f_{db,qp} \right). \tag{4.1}$$

To model the fact that ordering decisions may depend on routing decisions, we use the technique of auxiliary variables as in [25]. The auxiliary variables replace the original variables in the precedence constraints of an ordering decision, which prescribe either order. Auxiliary variables are only constrained to be equal to the corresponding original variables, if the associated routing is chosen.

For simplicity we reduce the notion of an operation (d, b) of train d on block b, to i in the remainder of this paper. With L as the set of all decisions in an instance of MRR, we can formulate the MRR as the following disjunctive program,

$$\begin{aligned}
\min \quad & \sum_{(i,0) \in A_\delta} \delta_i \\
\text{s.t.} \quad & t_j - t_i \geq f_{ij} && (i,j) \in A_f \\
& t_i - t_0 \geq \tau_{lb,i} + \mu_i && (0,i) \in A_h \\
& t_0 - t_i \geq -\tau_{ub,i} - \delta_i && (i,0) \in A_\delta \\
& \bigvee_{W_c \in D_l} \bigwedge_{(i,j) \in W_c} \left(t_j - t_i \geq f_{ij} \right) && l \in L \\
& t_i \in \mathbb{R}_+ \, \forall \, t_i, \quad \delta_i \in \mathbb{R}_+ \forall \delta_i
\end{aligned} \tag{4.2}$$

where the objective is to minimize the sum of arrival delays δ_i at all arrival events in the problem. We can further simplify the constraints A_δ to $\delta_i - t_i \geq -\tau_{ub,i}$, using $t_0 = 0$.

Constraints of Problem (4.2) can be represented in the generalized disjunctive graph of [25]. We will later use the generalized disjunctive graph in Section 4.5 to modify precomputed logic Benders cuts for the reuse. The generalized disjunctive graph for Problem (4.2) is defined by the tuple $G = (V, A_f \cup A_h \cup A_\delta, A_s)$. V is the set of nodes, where each node represents a variable t_i or δ_i of Problem (4.2). A_f, A_h and A_δ are fixed arcs in the graph with length f_{ij}, $\tau_{lb,i} + \mu_i$ and $-\tau_{ub,i}$ respectively, representing the fixed precedence relations of Problem (4.2). A_s are selectable arcs with length f_{ij} representing the selectable precedence relations. Selectable arcs are grouped into the choice sets of Problem (4.2) and such choices sets are grouped into the decision sets of Problem (4.2). A selection $\theta \subseteq A_s$ on G is a set of selectable arcs such that $G(\theta) = (V, A_f \cup A_h \cup A_\delta \cup \theta)$ is a standard directed graph. The selection θ is *complete* if it contains for each decision $l \in L$ at least one choice set, i.e., $\forall l \in L, \exists W_c \in D_l$ s.t. $W_c \subseteq \theta$; else the selection is *partial*. For simplicity we further denote the generalized disjunctive graph simply as the disjunctive graph.

In Figure 4.1, we give an illustrative example of the generalized disjunctive graph (similar to example in [25]) for a scenario of two trains d and q, with train d passing a routing area with two routing alternatives; one of those alternatives leads to a resource conflict with train q. Nodes $(d,b)_{in}$ and $(d,c)_{out}$ represent auxiliary variables. We consider in the example (q,b) an arrival event with a latest arrival time $\tau_{ub,qb}$ and (q,r) a departure event with an earliest departure $\tau_{lb,qr}$ and an input delay of μ_{qr}

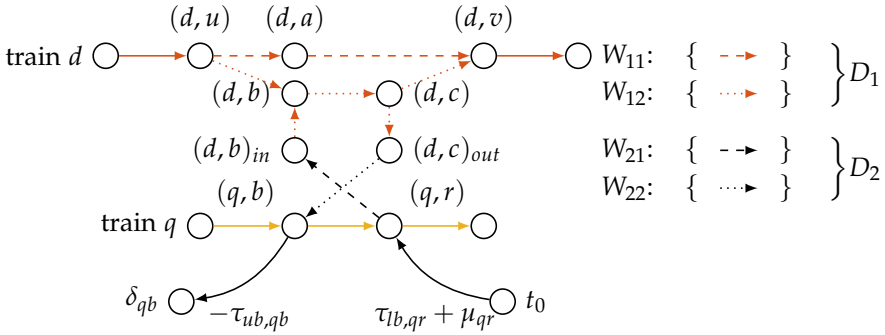

FIGURE 4.1: Example of the Generalized Disjunctive Graph.

4.4.2 *A Decomposition of Rescheduling*

We apply to Problem (4.2) the Benders decomposition of [25] to decompose the centralized problem $\mathcal{C}$, i.e., Problem (4.2), into a master problem $\mathcal{M}$ and a subproblem $\mathcal{S}$. Benders decomposition works iteratively, where at each iteration the master problem is first solved, and its solution is imposed to the subproblem. If the subproblem cannot find a feasible solution given the master solution, a proof of infeasibility can be generated. In this case, Benders feasibility cuts are generated from the proof of infeasibility and sent back to be included in the successive iterations of the master. The procedure continues until a feasible solution for the subproblem is found. The problem presented in this paper has some differences to [25] in the way it models delays. To be coherent with the decomposition of [25], the subproblem must be a problem of feasibility only. We partition the precedence relations of Problem (4.2) such that all A_δ and therefore all variables δ_i are considered as constraints and variables of the master problem, and the subproblem is a problem of feasibility only. According to [25], we decompose Problem (4.2) by partitioning A_f, A_h, A_δ and L into $A_{\mathcal{M},f}, A_{\mathcal{M},h}, A_\delta$ and $L_\mathcal{M}$ for the master; and $A_{\mathcal{S},f}, A_{\mathcal{S},h}$ and $L_\mathcal{S}$ for the subproblem. In this separation, we restate that $\mathcal{A}_\delta$ is only present in the master. Consequentially, $A_\mathcal{M} := A_{\mathcal{M},f} \cup A_{\mathcal{M},h}, \cup A_\delta \cup \left(\bigcup_{L_\mathcal{M}} \bigcup_{W_c}(i,j)\right)$ and $M := \left\{t_i, t_j \,|\, (i,j) \in A_\mathcal{M}\right\} \cup \left\{\delta_i \,|\, (i,0) \in A_\delta\right\}$ are the precedence relations and variables of the master; and $A_\mathcal{S} := A_{\mathcal{S},f} \cup A_{\mathcal{S},h} \cup \left(\bigcup_{L_\mathcal{S}} \bigcup_{W_c}(i,j)\right)$ and $S := \left\{t_i, t_j \,|\, (i,j) \in A_\mathcal{S}\right\}$ are the precedence relations and variables of the subproblem. With a set of Benders cuts B_R, which we additionally consider (reuse) in the master problem, we define the complete master problem of our Benders decomposition for Problem (4.2) at the iteration α of the

Benders decomposition scheme, i.e., $\mathcal{M}^\alpha(B_R)$, as the following disjunctive problem

$$
\begin{aligned}
\min \quad & \sum_{(i,0)\in A_\delta} \delta_i \\
\text{s.t.} \quad & t_j - t_i \geq f_{ij} && (i,j) \in A_{\mathcal{M},f} \\
& t_i - t_0 \geq \tau_{lb,i} + \mu_i && (0,i) \in A_{\mathcal{M},h} \\
& \delta_i - t_i \geq -\tau_{ub,i} && (i,0) \in A_\delta \\
& \bigvee_{W_c \in D_l} \bigwedge_{(i,j)\in W_c} \left(t_j - t_i \geq f_{ij}\right) && l \in L_{\mathcal{M}} \\
& \beta^r && r \in B^\alpha \cup B_R \\
& t_i \in \mathbb{R}_+ \,\forall\, t_i \in M, \quad \delta_i \in \mathbb{R}_+ \,\forall\, \delta_i \in M,
\end{aligned}
\tag{4.3}
$$

where β^r has the form of the logic Benders cut as introduced in [25], B^α is the set of all Benders cuts iteratively generated from analyzing the subproblem until iteration α. In general, we will select B_R, by a suitable procedure, from a larger set (a library) of Benders cuts $\mathcal{B}$. Once a Benders cut is included in the problem, i.e., either in B^α or B_R, it remains considered in the problem until the end.

We define the subproblem $\mathcal{S}^\alpha$ at iteration α of the decomposition scheme, given $\bar{t}_i^\alpha$ of master solution $\mathcal{O}^\alpha := \left\{\bar{t}_i^\alpha,\, i \in M\right\}$, to be,

$$
\begin{aligned}
\min \quad & 0 \\
\text{s.t.} \quad & t_i = \bar{t}_i^\alpha && \forall i \in M_{\mathcal{S}} \\
& t_j - t_i \geq f_{ij} && (i,j) \in A_{\mathcal{S},f} \\
& t_i - t_0 \geq \tau_{lb,i} + \mu_i && (0,i) \in A_{\mathcal{S},h} \\
& \bigvee_{W_c \in D_l} \bigwedge_{(i,j)\in W_c} \left(t_j - t_i \geq f_{ij}\right) && l \in L_{\mathcal{S}} \\
& t_i \in \mathbb{R}_+ \,\forall\, t_i \in S
\end{aligned}
\tag{4.4}
$$

where $M_{\mathcal{S}} := M \cap S$ are those variables appearing in $\mathcal{M}^\alpha$ and $\mathcal{S}^\alpha$ and that are fixed in the subproblem to the solution of the master problem $\mathcal{O}^\alpha$; in accordance to Benders decomposition.

In Figure 4.2, we illustrate a possible decomposition of the example given in Figure 4.1 by illustrating the disjunctive graphs of the master and subproblem. We define for the example $A_{\mathcal{M},f} = A_f \setminus \{((q,b),(q,r))\}$, $A_{\mathcal{M},h} = A_h$, $L_{\mathcal{M}} = \{\}$ and $A_{\mathcal{S},f} = \{((q,b),(q,r))\}$, $A_{\mathcal{S},h} = \{\}$, $L_{\mathcal{S}} = \{1,2\}$. Variables appearing in both, master and subproblem, i.e., variables $M_{\mathcal{S}}$, are shaded in gray in Figure 4.2.

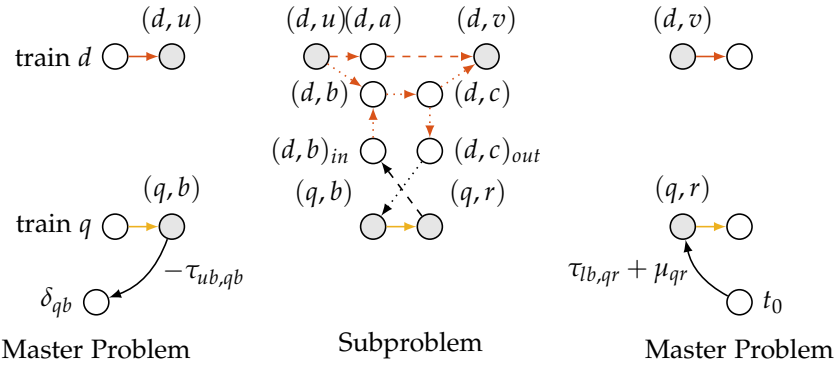

FIGURE 4.2: Example of a Decomposition on the Generalized Disjunctive Graph.

4.5 REUSING LOGIC BENDERS CUTS

4.5.1 *Preliminaries*

We reuse logic Benders cuts to accelerate the solution process of a rescheduling problem as we extend the master problem $\mathcal{M}$ of an instance $\mathcal{R}'$ by some logic Benders cuts B_R from a larger library $\mathcal{B}$ of precomputed Benders cuts. The library of cuts has been computed by solving another instance(s) $\mathcal{R}$ of MRR, which describes the same infrastructure, timetable and traffic as $\mathcal{R}'$, but under different input delays.

In case instances of MRR differ only by the input delay μ_i we show in Section 4.5.2 that optimality remains in case logic Benders cuts are reused among these instances, and the reuse of logic Benders cuts is a valid process.

Formally we define an instance $\mathcal{R}$ of MRR by the tuple $\mathcal{R} := (A_f, A_h, A_\delta, A_s, L)$. We denote as $\partial(\mathcal{R})$ the class of all instances which differ from an instance $\mathcal{R}$ of MRR only by input delay, and use the same decomposition. Those instances obviously contain the same number of events, precedence relations and decisions.

As discussed in Section 4.1, most railways are operated on a daily repeating schedule. Every day during the same time of the day, the same trains can be found on the railway network, and in fact the instances of MRR solved throughout the days are always considering the same trains, timetable and infrastructure. With our approach, we exploit this basic property of rescheduling problems. We are moreover able to quantify a similarity

of specific realizations of input delays (and therefore of instances), based on the fact that we can statistically characterize the input delays.

In Section 4.6.2, we will introduce a measure to identify, within this class of instances (and therefore class of cuts), instances which are more similar, and others which are less similar. The measure is based on the feature space of the respective sets A_h and A'_h, which describe the input delays μ_i and μ'_i. We will show that with a measure to quantify the similarity between instances, we can estimate the computational benefit of particular Benders cuts for their reuse.

4.5.2 *Modification of a logic Benders cut to preserve validity*

We consider the reuse of a logic Benders cut as valid, in case optimality is preserved. We can show that with an appropriate modification, the reuse of a cut between instances of MRR, which only differ in input delay, is guaranteed not to cut off any feasible solution, which implicitly preserves the optimality.

In Section 4.4.2, we applied the logic Benders decomposition [25] to the problem of rescheduling. In [25], a logic Benders cut has been introduced, which can be used for our decomposition of Problem (4.2) in Section 4.4.2. In principle, the logic Benders cut summarizes the constraints of the subproblem for the master problem, such that a solution of the master problem is consistent with the constraints of the subproblem and allows for a feasible solution in the subproblem.

Regarding the decomposition in Section 4.4.2, let us assume that p is a path (of fixed arcs) on the disjunctive graph of the subproblem $\mathcal{S}^\alpha$ and between two variables of $M_\mathcal{S}$ (one of them possibly t_0). Then, a solution of the master problem must satisfy the sum of all constrains (arcs) along such path, to make sure that a solution to the subproblem exists. In general, a path p on the disjunctive graph of the subproblem $\mathcal{S}^\alpha$ contains selectable arcs and the existence of the path depends upon the selection of particular arcs. In [25] it was shown that we can derive a set of paths (here denoted by $\mathcal{P}$) between variables of $M_\mathcal{S}$, for which we are guaranteed that any solution of the subproblem satisfies the constraints along at least one path $p \in \mathcal{P}$; the set is derived by a proof of infeasibility on the subproblem. As all paths in $\mathcal{P}$ are between variables of $M_\mathcal{S}$, a solution of the master problem must satisfy the constraints along at least one path in $\mathcal{P}$ as well. Otherwise, the subproblem has no feasible solution for the considered solution of the

master problem. From the set of paths $\mathcal{P}$, the following logic Benders cut has been introduced in [25],

$$\beta = \bigvee_{p \in \mathcal{P}} \left(t_{e(p)} - t_{s(p)} \geq l_p \right). \tag{4.5}$$

In the cut (4.5), $s(p)$ and $e(p)$ are the start and end node of path p respectively (both variables of $M_{\mathcal{S}}$) and l_p is the length of the path; the constraint $t_{e(p)} - t_{s(p)} \geq l_p$ is the result of summing up all left and right hand-sides of constraints along p and summarizes the constraints along p for variables of $M_{\mathcal{S}}$.

Between two instances of MRR $\mathcal{R}$ and $\mathcal{R}' \in \partial(\mathcal{R})$ only constraints of A_h change; it holds that $A_{\mathcal{S},s} = A'_{\mathcal{S},s}$ and $L_{\mathcal{S}} = L'_{\mathcal{S}}$, i.e., selectable arcs and decisions remain identical. Therefore, for any set of paths $\mathcal{P}$ on the disjunctive graph of $\mathcal{S}^\alpha$ (of $\mathcal{R}$), where any solution of $\mathcal{S}^\alpha$ satisfies the constraints along at least one path $p \in \mathcal{P}$, the same holds for the set $\mathcal{P}$ considered on the disjunctive graph of $\mathcal{S}'^\alpha$ (of $\mathcal{R}'$) and solutions of $\mathcal{S}'^\alpha$. Constraints A_h and A'_h of $\mathcal{R}$ and $\mathcal{R}'$ may differ because they show a different right hand-side μ_i (resp. μ'_i), such that a path p may have different length considered on the disjunctive graph of $\mathcal{S}^\alpha$ or $\mathcal{S}'^\alpha$. To reuse a logic Benders cut (in the form of Eq. 4.5) generated for $\mathcal{R}$ on $\mathcal{R}'$, we adjust all right hand-sides in the linear constraints of the logic Benders cut (4.5) according to μ'_i of $\mathcal{S}'^\alpha$ to

$$\beta' = \bigvee_{p \in \mathcal{P}} \left(t_{e(p)} - t_{s(p)} \geq l'_p \right) \tag{4.6}$$

where l'_p is the length of path p considered on the disjunctive graph of $\mathcal{S}'^\alpha$. With the modification of β to β' and the fact that selectable arcs and decisions remain identical between $\mathcal{R}$ and $\mathcal{R}'$, we are guaranteed that β' is a valid cut for $\mathcal{R}'$ and optimality remains under the reuse of cuts.

4.6 PROPOSED APPROACH

4.6.1 *A Lazy-Constraint Approach*

In this section we propose an implementation for the reuse of logic Benders cuts. We first propose a lazy-constraint approach for the iterative extension of the master problem in our decomposition by precomputed Benders cuts. The idea is to include only cuts, which are violated by the incumbent

Algorithm 3: Lazy-Constraint Reuse of logic Benders Cuts

input : Master $\mathcal{M}$, Subproblem $\mathcal{S}$, Library $\mathcal{B}$
output: $\mathcal{O}_\mathcal{C}$
init : $\alpha = 0$, $B^\alpha = \emptyset$, $\mathcal{O}_\mathcal{C} = \emptyset$, $B_R = \emptyset$,

while $\mathcal{O}_\mathcal{C} = \emptyset$ **do**
  do
    $\mathcal{O}^\alpha_\mathcal{M} \leftarrow$ `solve` $(\mathcal{M}^\alpha(B_R))$
    $B_V \leftarrow$ `getViolatedCuts`$(\mathcal{O}^\alpha_\mathcal{M}, \mathcal{B})$
    $B_R \leftarrow B_R \cup B_V$
  while $B_V \neq \emptyset$
  $\mathcal{O}^\alpha_\mathcal{S}, \mathcal{B}^\alpha_\mathcal{S} \leftarrow \mathtt{SMT}_{Agg}(\mathcal{S}^\alpha\ (\mathcal{O}^\alpha_\mathcal{M}))$
  if $\mathcal{O}^\alpha_\mathcal{S} = \emptyset$ **then**
    $B^{\alpha+1} \leftarrow B^\alpha \cup \mathcal{B}^\alpha_\mathcal{S}$
  else
    $\mathcal{O}_\mathcal{C} \leftarrow \mathcal{O}^\alpha_\mathcal{M} \cup \mathcal{O}^\alpha_\mathcal{S}$
  $\alpha \leftarrow \alpha + 1$

solution of the master problem and ignore the rest. This allows to keep the master problem smaller and with only cuts which potentially provide progress.

Given a library $\mathcal{B}$ of precomputed and modified logic Benders, which can be potentially reused, it is difficult to estimate in advance, which of these cuts will bring a computational benefit if reused. In case the master problem is extended by a precomputed cut, which does not affect any optimal solution or is dominated by those already in the master problem, the cut will add no computational benefit. Instead, the additional constraint will only decrease the computational performance of solving the master problem, as the master will increase in size. With the design of a lazy-constraint approach we aim to prevent such phenomena and exclude cuts with no computational benefit from the reuse.

We reuse logic Benders cuts in a lazy manner, to avoid including cuts with no computational benefit. That is, given a library of cuts for reuse, we evaluate in an iterative manner, which cuts are violated by the incumbent master solution and only add those cuts to the master problem. After every extension by violated cuts, we compute a new master solution and check again for further violated cuts. The lazy-constraint procedure is integrated in the entire process of the Benders decomposition as only after no fur-

ther violated cuts are found in the library, the subproblem is queried for feasibility, to possibly generate further, new logic Benders cuts.

The complete lazy-constraint approach is illustrated in Algorithm 3, where SMT_{Agg} (see Appendix 4.9) is the algorithm of [25] by which we address the subproblem. The expression $\mathcal{O}_{\mathcal{C}}$ denotes a solution to the centralized problem; as far as there is an infeasibility in the subproblem, a centralized solution cannot be found and the algorithm iteratively proceeds. In an iteration of Algorithm 3, Lines 3-5 adds violated constraints B_V from the library $\mathcal{B}$ in a lazy manner. The master problem is solved again in Line 3 with an iteratively growing set of cuts B_R considered, for every non-empty set of violated cuts B_V in Line 4, until no further violated logic Benders cut is found in the library $\mathcal{B}$. In Line 7 the subproblem is analyzed using the SMT_{Agg} algorithm of [25], which either returns a set of new Benders cuts to be added to the master problem, or a feasible solution for the subproblem. Lines 8-11 either add any new Benders cut to the master problem or generate a feasible solution for the centralized problem, if the subproblem is feasible.

With the lazy-constraint approach we define two types of logic Benders cuts in reuse:

Definition 2. *(Useless logic Benders Cut)*
We denote a logic Benders cut β as useless, if $\beta \in \mathcal{B}$ but $\beta \notin B_R$ after the termination of Algorithm 3. In other terms, in no iteration the cut was found to be violated by the incumbent master solution.

Definition 3. *(Useful logic Benders Cut)*
We denote a logic Benders cut β as useful, if $\beta \in \mathcal{B}$ and $\beta \in B_R$ after the termination of Algorithm 3. In other terms, in at least one iteration the cut was violated by the incumbent master solution.

Useless logic Benders cuts are cuts that provide no computational benefit when reused; if such cuts are not excluded from the reuse, these cuts can significantly decrease the computational speed of solving the master problem.

With the lazy-constraint approach we can further define a perfect set of logic Benders cuts as:

Definition 4. *(Perfect Set of logic Benders Cuts)*
We denote a set of logic Benders cuts $\mathcal{B}$ as perfect if $B^{\alpha} = \emptyset$ after the termination of Algorithm 3.

The perfect set of logic Benders cuts leads to a termination of Algorithm 3 after the first full iteration between master problem and subproblem. In case of a perfect set of cuts, no further cuts must be generated from the

subproblem to determine an optimal solution for the centralized problem. All necessary cuts are within the perfect set. From a computational point of view, this is the ideal situation.

4.6.2 *Identifying the Computational Benefit of reusable logic Benders Cuts*

With the lazy-constraint approach (Algorithm 3) we are able to exclude cuts with no computational benefit from inclusion in the master. Unfortunately, we are not guaranteed that useful cuts will actually lead to a computational speedup, in case they are reused. An excessive amount of cuts, despite being useful, actually decreases the performance of decomposition in case reused. In other terms, two processes are contrasting each other: the master getting larger and slower as more cuts are included, and the master getting faster (and the entire scheme having less iterations) in case the best cuts are added. In this case, a method is necessary to not only identify the useful cuts, but actually prioritize those few, which add a computational benefit to the entire solution process.

We assume that precomputed Benders cuts add the most computational benefit for reuse, in case the instance, which has been used to precompute cuts, shows similar input delays of trains as the instance for which cuts are being reused. As such we expect that for an instance $\mathcal{R}' \in \partial(\mathcal{R})$, cuts computed when solving $\mathcal{R}$ add the most computational benefit, if for each $(i,j) \in A'_h$ of $\mathcal{R}'$ there exists a $(i,j) \in A_h$ of $\mathcal{R}$ where $\mu_i \approx \mu'_i$.

We propose in this section two different measures to quantify the similarity in input delay μ. To this purpose, we summarize the situation of updated starting times of relevant events, including the input delays, of an instance $\mathcal{R}$ as we write all right hand-side $\tau_{lb,i} + \mu_i$, $(0,i) \in A_h$ in a vector $v_{\mathcal{R}}$. By definition of $\partial(\mathcal{R})$, all vectors $v_{\mathcal{R}}$ and $v_{\mathcal{R}'}$ we consider in reuse, have the same dimension. For a pair of instances $\mathcal{R}$ and $\mathcal{R}' \in \partial(\mathcal{R})$, we quantify similarity in input delay by the euclidean distance and the Sørensen-Dice coefficient [16] on the respective vectors $v_{\mathcal{R}}$ and $v_{\mathcal{R}'}$. The euclidean distance is defined as

$$e_{v_{\mathcal{R}},v_{\mathcal{R}'}} = ||v_{\mathcal{R}} - v_{\mathcal{R}'}||. \tag{4.7}$$

In case $e_{v_{\mathcal{R}},v_{\mathcal{R}'}} = 0$, $v_{\mathcal{R}}$ and $v_{\mathcal{R}'}$ are identical. The Sørensen-Dice coefficient is a measure to quantify the similarity of vectors and can be computed as,

$$s_{v_{\mathcal{R}},v_{\mathcal{R}'}} = \frac{2\,|v_{\mathcal{R}} \cdot v_{\mathcal{R}'}|}{|v_{\mathcal{R}}|^2 + |v_{\mathcal{R}'}|^2}. \tag{4.8}$$

In case $s_{v_{\mathcal{R}},v_{\mathcal{R}'}} = 1$, the vectors $v_{\mathcal{R}}$ and $v_{\mathcal{R}'}$ are identical. Values of $s_{v_{\mathcal{R}},v_{\mathcal{R}'}}$ between 0 and 1 indicate a measure of similarity between $v_{\mathcal{R}}$ and $v_{\mathcal{R}'}$.

4.7 COMPUTATIONAL EXPERIMENTS

In this section we conduct comprehensive experiments to study the reuse of logic Benders cuts. Compared to the theory above where only one subproblem is considered, we conduct our experiments on a decomposition with 40 different subproblems. The extension of the theory to the case of multiple subproblems is straightforward. The decomposition is chosen such that each subproblem can be treated individually without dependencies to other subproblems. The decomposition itself is a geographical decomposition, identical to the decomposition in [25]. The commercial solver Gurobi [19] is used to solve the master problem.

All experiments were run on the Euler cluster of ETH Zurich on a Intel(R) Xeon(R) CPU E3-1585L v5 @ 3.00GHz processor with 4GB of RAM.

4.7.1 *Instances and Delay Scenarios*

We consider for our computational experiments the railway traffic within the triangle of the cities Zurich, Luzern and Chur in Switzerland between 08:00 and 09:00 in the morning; Table 4.1 provides an overview in numbers.

For our experiments we consider a typical Monte Carlo scheme, where instances $\mathcal{R} = (A_f, A_h, A_\delta,$
$A_s, L)$ differ only by the input delay μ considered in the constraints in A_h. The input delay is generated with the goal of having sufficiently different patterns of input delay to understand the potential and limit of the approach. The procedure goes as follows.

First, we randomly determine the trains which are experiencing an input delay. Instead of using a probability for the likelihood of a train being delayed, we first decide, based on an uniform random distribution, how many out of all trains experience a delay (ranging from none, to all). We then randomly select such number of trains out of all trains. In this way, we achieve an equal probability for the number of delayed trains over our generated MRR instances. Given the trains experiencing a delay, we determine for each such train a single relevant event that is a departure, where a delay should occur. The event is randomly selected out of all departure events related to the train. At last, we define the value of input delay μ_i for each selected relevant event i. We sample the value of the input delay from

Time Horizon	Subproblems	Trains	Blocks	Routing Alternatives	Resource Conflicts
08:00-09:00	40	80	626	790	3240

TABLE 4.1: Overview of Original Timetabling Instance.

a Weibull distribution according to [11] (see parameters in Table 4.2). We do not consider negative input delays in this work, such that we resample from the Weibull distribution as long as we receive a negative delay value.

For the experiments, we randomly generate two sets of instances, with the same procedure and same statistical properties. A first set of 100 instances is the *test set* and is used to evaluate the performance of our algorithm, with/without reuse of cuts. A second set of 1000 instances is the *training set* and is solved in advance by the normal Benders decomposition (i.e. without considering any library of cuts for reuse) to generate a large library $\bar{\mathcal{B}}$ of logic Benders cuts. From this latter large library, we identify the actual library $\mathcal{B}$ available for reuse in Algorithm 3.

4.7.2 *Computational Benefit of Benders Cuts*

In a series of experiments we show the computational benefit of cuts by taking a perfect set of cuts and changing the proportion of perfect cuts against non-perfect (random) cuts in such set. We designed two experiments, one with a library of fixed size (i.e., when adding a non-perfect cut, we remove one perfect cut), and one with a library of increasing size (i.e. the perfect cuts are always considered, but increasingly many non-perfect cuts are included).

We create for each of our test instances a perfect set (library) $\mathcal{B}^*$ of logic Benders cuts for the reuse. We create such sets by applying, for each test instance, Algorithm 3 multiple times and iteratively collecting and reapplying the generated cuts, until a perfect set has been found. We have

Scale	Shape	Shift
395	2.5	-315

TABLE 4.2: Weibull Parameters [11].

to reapply generated cuts because Algorithm 3 is rather sensitive to $\mathcal{B}$. Therefore, for different cuts considered in input $\mathcal{B}$, different cuts might be generated as B^α from the subproblem analysis. To generate $\mathcal{B}^*$, we start for each instance with an empty set of generated cuts $\mathcal{B}_{gen} = \emptyset$, which we use as library $\mathcal{B}$ for Algorithm 3. After the first termination of Algorithm 3, we can retrieve from the algorithm the set B^α, i.e., the set of all Benders cuts generated by the subproblems during this first run of the algorithm. Such set B^α is not necessarily perfect yet. In case B^α is reused, it can occur that other cuts might be needed as the master problem results in different intermediate optimal solutions, due to the reused cuts. To create a perfect set, we set $\mathcal{B}_{gen} = \mathcal{B}_{gen} \cup B^\alpha$, i.e., we incrementally extend the set of generated cuts by B^α and consider this extended set $\mathcal{B}_{gen}$ as input $\mathcal{B}$ for Algorithm 3. This will consider all previously generated cuts when we run Algorithm 3 again.

We repeat this process until no further cuts are generated from any subproblem during Algorithm 3, i.e., $B^\alpha = \emptyset$. Finally, when $B^\alpha = \emptyset$, the set of cuts $\mathcal{B}_{gen}$ is a perfect set of cuts, but it might contain useless cuts. We avoid useless cuts in the perfect library $\mathcal{B}^*$ by taking only the useful cuts from $\mathcal{B}_{gen}$. We can get the useful cuts which are in $\mathcal{B}_{gen}$, by considering the latest run of Algorithm 3 where $B^\alpha = \emptyset$. In this run of Algorithm 3, after termination, the set $\mathcal{B}_R$ corresponds exactly to those cuts of $\mathcal{B}_{gen}$ that are useful, i.e., that have been violated at some point during Algorithm 3.

A perfect sets of cuts generated by this procedure is not necessarily minimal, but as we use B_R at the end of the procedure, $\mathcal{B}^*$ is guaranteed not contain any useless cuts.

We consider in this subsection a series of libraries for each instance of the testing set, all of the size of the corresponding perfect library $|\mathcal{B}^*|$, where an increasing amount of cuts from the original perfect library $\mathcal{B}^*$ are substituted with other (random) cuts from other delay cases, randomly chosen from $\bar{\mathcal{B}}$. We assume those latter have on average low computational benefits.

In Figure 4.3 we illustrate the performance on average over all test instances for an increasing proportion of random cuts in a library $\mathcal{B}$ of fixed size. The three figures report respectively from top to bottom: the computational time, normalized by the case with no reuse, i.e., $\mathcal{B} = \emptyset$; the amount of iterations until the solution is found; and the amount of Benders cuts $|B_R|$ actually included in the master from the library $\mathcal{B}$. The top two plots report the lazy-constraint approach (red), the direct reuse (orange) and the case of no reuse (black). The bottom plot reports in solid lines the total amount of cuts reused from the library, i.e., $|B_R|$, and in dotted lines the

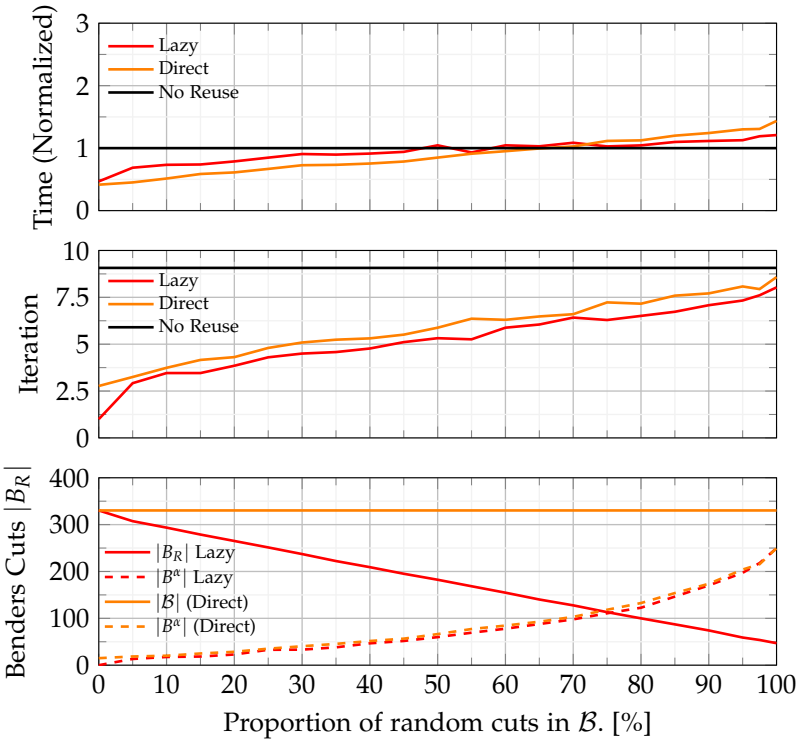

FIGURE 4.3: Time, Iteration and Cuts for Direct Reuse, Lazy-Constraint vs. No Reuse.

total amount of cuts newly generated in the iterative process. In red is the lazy-constraint approach, in orange the direct reuse.

In the top plot of Figure 4.3, we can see that both, the lazy-constraint approach as well as the direct reuse, show a speedup around 2 for the perfect cuts (the left end of the plot). With an increasing substitution of perfect cuts by random cuts, the computational performance decreases. Above 65% of random cuts, the reuse performs worse than the normal decomposition where no reuse is considered. An explanation for the decrease in performance is given in the middle and bottom plot of Figure 4.3. We can see that with an increasing proportion of random cuts, the necessary iterations till convergence increase, almost up to the number of the decomposition with no reuse, i.e., when $\mathcal{B} = \emptyset$. The increasing amount of iterations is caused by the lack of useful cuts in the library $\mathcal{B}$, due to the increasing number

of random cuts. In other words, the random cuts are not resulting in less iterations, and instead only increase the size of the master problem. The lack of useful cuts is show in the bottom plot of Figure 4.3, where for the lazy-constraint approach the number of cuts reused (B_R) from the library $\mathcal{B}$ continuously decreases for higher proportions of random cuts, while the number of newly generated cuts B^α (dashed line) increases.

Furthermore, at 100% random cuts, lazy-constraint still finds that some of the random cuts are useful, see bottom plot, and includes these in the master problem. The computational performance in such case lies above 1, i.e., it is slower than no reuse. We thus empirically find that even cuts determined as useful can result in a slow down of the decomposition scheme.

A further interesting point in Figure 4.3 is the fact that our perfect sets of Benders cuts are perfect when using the lazy-constraint approach, but not perfect under the direct reuse. We see this empirically as in the middle plot of Figure 4.3, the average number of iterations for direct reuse is around 2.5 instead of 1, as in the lazy-constraint approach. This is caused by the fact that Gurobi, which is used to solve the master problem in both cases, computes different optimal solutions depending on whether cuts are added iteratively, i.e., in a lazy manner, or all together. Therefore it is possible that in the direct reuse a minimal amount of further iterations and logic Benders cuts are necessary to converge.

4.7.3 *Issues from excessive amount of Benders Cuts reused*

With a second series of experiments we aim to study the decreasing performance in experiments of Section 4.7.2, for large proportions of random cuts. We want to estimate whether this is caused by the lack of computationally beneficial cuts or the presence of random but useful cuts. Random cuts might distract the decomposition scheme from converging quickly. If the latter case applies, we clearly must take care in generating the library $\mathcal{B}$ for actual reuse.

In this second series of experiments we consider libraries, where on top of the complete perfect set of Benders cuts $\mathcal{B}^*$ for our test instances (which are definitely useful), an increasing amount of additional random cuts is added. Compared to the experiments of Section 4.7.2, the libraries considered increase in size from $|\mathcal{B}^*|$ to $\sim 40|\mathcal{B}^*|$; and are always a superset of the perfect sets $\mathcal{B}^*$ for the individual test instances.

Figure 4.4 reports the computational results for variable library sizes, in an analogous manner to Figure 4.3. Starting from the top plot, we can see

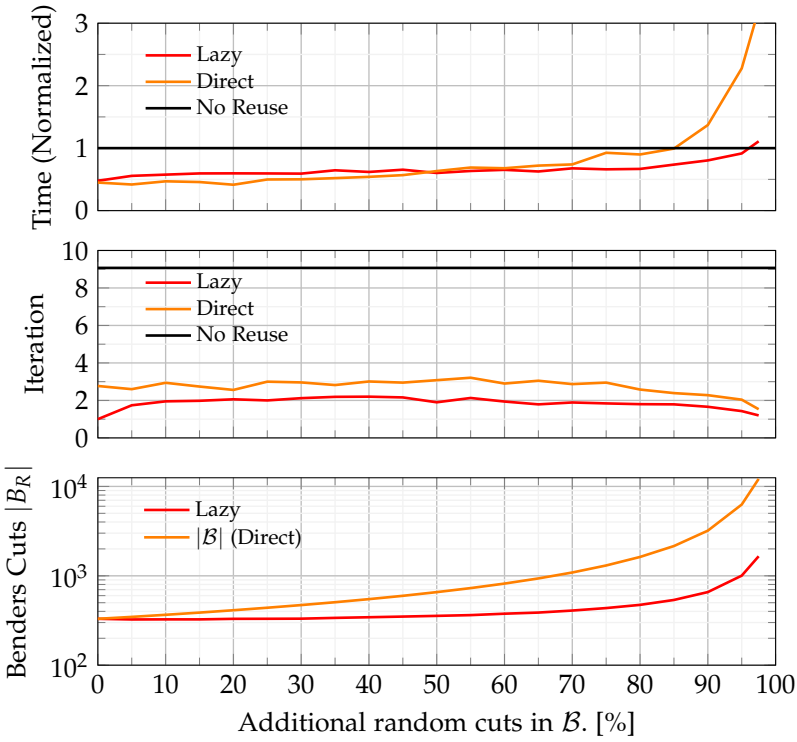

FIGURE 4.4: Time, Iteration and Cuts for Direct Reuse, Lazy-Constraint vs. No Reuse.

again the performance increase of a factor 2 for the reuse of just the perfect set of cuts (i.e., the left end of the plot is at 0.5). We can further see, that with an increasing amount of additional random cuts in the library, the computational performance of both, direct and lazy-constraint reuse, decreases. The bottom plot of Figure 4.4 (with logarithmic y-axis), explains such decrease, due to a significant increase in the size of the master problem. The plots in Figure 4.4 further illustrate well, the two main effects of excluding useless cuts from reuse as in the lazy-constraint approach. On the one hand, the absence of useless cuts leads to a smaller master as shown in the bottom plot of Figure 4.4. On the other hand, as the middle plot shows, the absence of useless cuts leads to a reduced number of iterations in the decomposition scheme. Both lead to a significant computational speedup, as shown in the top plot. Also, including useless cuts in the library does not have a

systematic effect on the amount of iterations required until convergence of the scheme (the lines in the middle plot are mostly horizontal).

Overall, we empirically conclude from Figure 4.4 that reusing an excessive amount of Benders cuts, even if useful, negates the overall benefits of reusing computationally beneficial logic Benders cuts, such as the perfect cuts $\mathcal{B}^*$. Furthermore, we can see in Figure 4.4 that also the lazy-constraint approach, which considers only useful cuts, shows a slow down in computational performance. This is due to random, but useful cuts, which are included, and increase the size of the master without causing a desired decrease in iterations of the scheme. As a result, they slow down the computational performance of the solution process.

The above clearly confirms the importance of carefully selecting the cuts in the library for the reuse, beyond their usefulness, but based on (some measure of) their computational benefit to the entire solution process, which is the topic of the next subsection.

4.7.4 *Reuse of cuts from libraries of similar instances*

In this section we analyze the reuse of logic Benders cuts under real-world conditions. In this case, it is unlikely that a perfect set of precomputed cuts is available. We thus create a series of experiments where we pair the libraries computed each by a single training instance (out of the 1000 training instances), with our 100 test instances. This results in 100'000 pairs, matching a single training library (i.e., the cuts computed on a single training instance) and a single test instance (to apply the library). Which such "Training Library - Test Instance" pairs, we want to understand under which conditions logic Benders cuts result in a computational benefit. The experiments in this section are exclusively computed using our lazy-constraint approach.

In Table 4.3 we report computational results for all pairs of training library - test instance, as well as the ideal performance using the perfect library; and two further benchmarks for a general comparison in performance. The columns of the Table 4.3 are as follows.

The Perfect Library repeats the ideal performance from the previous sections, in case the perfect set of Benders cuts is reused. The Best Library considers, for each test instance, the computational performance using the "best" library, i.e., the one, out of the 1000 libraries computed by the 1000 instances of the training set, which results in the smallest computational time. The Average (Avg.) Library reports the computational performance on average over all pairs of training library - test instance. This would

Approach	Perfect Lib.	Best Lib.	Avg. Lib.	No Lib.	Gurobi
Time [s]	32.30	33.19	94.31	81.86	201.24
Normalized Time	0.45	0.46	1.16	1	3.92
Iterations [-]	1 (5.85*)	4.04 (8.01*)	7.60 (12.96*)	9.07	-
Total Cuts [-]	345.21	307.64	359.01	382.40	-

Lib.: Library; Avg.: Average.
* Total Master Solves, including Lazy-Constraint.

TABLE 4.3: Potential when reusing Cuts from Libraries of logic Benders Cuts.

correspond of taking a random training instance and reusing its cuts on a random test instance. As benchmarks, we report in Table 4.3 also the computational results of solving the test instances with the normal logic Benders decomposition of [25] as No Library, and the solution of the centralized Problem (4.2), directly computed by the commercial solver Gurobi [19]. All approaches in Table 4.3 report an optimal solution within a tolerated optimality gap of 1% (on solutions of the master problem), which is considered sufficient for practical applications.

Overall, Table 4.3 clearly shows the advantage of decomposition over the centralized benchmark of Gurobi. Further, we can see that with the reuse, in case of the best library, we can achieve a speedup of a factor similar to the perfect library, empirically proving the potential of reusing logic Benders cuts also under real-world conditions. Table 4.3 indicates that a speedup by the reuse of precomputed logic Benders cuts is not guaranteed in case of an arbitrary reuse of cuts, i.e., for the average library it is 16% slower. This restates the importance of selecting carefully the cuts for reuse.

The performances in different cases of decomposition in Table 4.3 can be explained by the number of iterations, in particular solves of the master problem, and total amount of Benders cuts in the master problem. Due to the lazy-constraint approach, the number of times the master problem is solved is higher than number of iterations done by the decomposition scheme in cases of reuse. In Table 4.3, the perfect library and the best library show significantly less iterations (and slightly less master solves) than the no library case, which explains the observed speedup in both cases. Also, in case of the lazy-constraint, most of the master solves are generally performed in the first iteration of the Benders scheme. In the first iteration, the master is usually small in terms of additional Benders cuts and thus rather

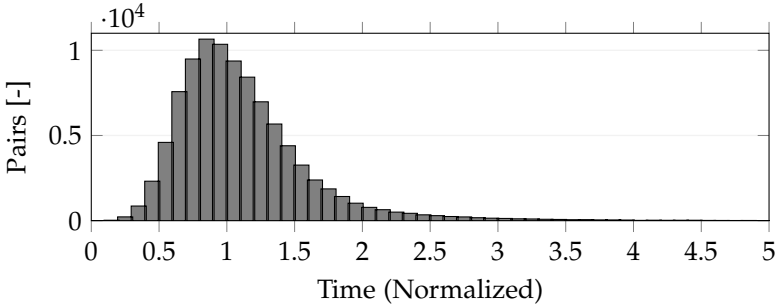

FIGURE 4.5: Histogram of Computational Times for Pairs of Training Library - Test Instance.

efficient to be solved, compared to a master problem after several iterations. In other terms, master solves in the No Library case are in general more complex. The reason for the rather unsatisfying performance of the average pairs in Table 4.3 has been already commented in Section 4.7.2. Random cuts, which most of training cuts seem to be, are more harmful than useful, and tend to slow down the solution procedure.

In Figure 4.5 we report a histogram of the computational performance (x-axis, normalized by no reuse), for all pairs of training library - test instance. This describes the empirical probability for the performance of pairs. The expected value of this histogram is the performance of the average library, i.e., 1.16. In the histogram of Figure 4.5 the total probability for a pair to result in a computational benefit is 48.2%, while the probability of a computational degradation is 51.2%. Therefore, a random cut is slightly more likely to result in a slow down, than a speedup. In the histogram of Figure 4.5, 26.2% of the pairs show a computational time of 0.8 or less; we will consider a normalized computation time of 0.8, which corresponds to 20% faster computation, to be a significant enough speedup for practical considerations.

The variability in Figure 4.5 illustrates the existence of cuts with high and low computational benefit in the reuse. Clearly, the pairs of best libraries reported in Table 4.3 relate to pairs that are on the left in Figure 4.5, i.e., with a normalized computational time < 1. These are the pairs, i.e., the libraries, we wish to identify efficiently in a real-life application to improve computational speed of the decomposition process.

4.7.5 *A Measure to estimate Speedup resulting from a Library*

In the previous two sections, we analyzed the computational benefits of libraries of logic Benders cuts, and the impact of the excessive reuse of cuts. We concluded that for a real-world application, an estimation of the computational benefit of (a library of) logic Benders cuts is necessary, to limit the reuse of logic Benders cuts to those with a high computational benefit only. In Section 4.6.2, we proposed two measures to estimate such computational benefit, at the level of libraries generated from a single instance. Those measures describe the similarity of input delay between the instance determining the library, and the target instance we want to solve. The ultimate goal is to identify a-priori the libraries, which lead to a performance as good as the best library, which is reported in Table 4.3; and in general, the libraries which approximate a performance comparable to using the perfect set of logic Benders cuts.

To estimate the accuracy of our similarity measures in indicating the computational benefit of a library of logic Benders cuts, we analyze the euclidean distance and the Sørensen-Dice coefficients on the pairs of training library - test instance from Section 4.7.4. In particular, we analyze the relation between the speedup (i.e., how smaller the computational time is; normalized by the computational time when no cut is reused) achieved by a particular training library - test instance, and the similarity measures. We evaluate our measures for each pair using the training instance, from which the library has been generated; and the test instance, on which the library has been reused. We consider libraries generated by only a single instance.

In Figure 4.6 we report a histogram of values on the Sørensen-Dice and euclidean measures, over the 100'000 pairs of Section 4.7.4. In Figure 4.6, we overlay in red the average normalized computational times, which are computed as average over the pairs in the respective individual bin. The top plot in Figure 4.6 reports on the Sørensen-Dice measure. The plot empirically confirms the ability of the measure to discriminate between libraries which have different computational benefit. Specifically, while the majority of pairs in Figure 4.6 show no computational benefits, above a Sørensen-Dice value of 0.9998 the libraries being reused clearly bring a computational benefit. The red line decreases to the right, below 0.8, for a maximum speedup of 0.5 (twice as fast) for Sørensen-Dice values close to 1. Such an observed speedup matches with the reported speedup in case of a set of perfect cuts (0.46) in Section 4.6.1, i.e., cuts generated from an instance with a Sørensen-Dice value of exactly 1. The bottom plot of Figure 4.6 reports the case of the

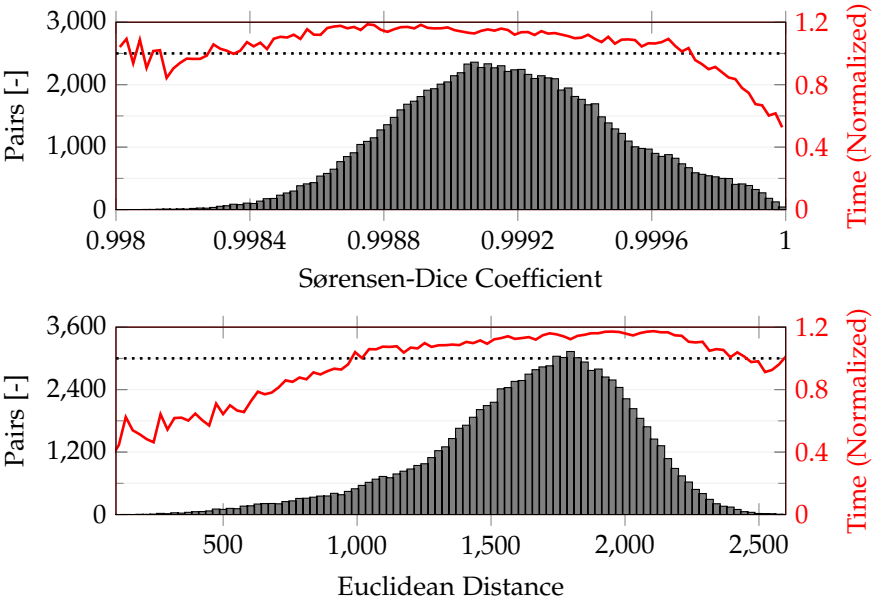

FIGURE 4.6: Histogram of Similarity Measures over Pairs of Training Library - Test Instance, overlaid with Computational Time (red).

euclidean distance, and in general confirms a similar ability to discriminate libraries with high computational benefit from those of no benefit. In this case, values of the euclidean distance below 700 are connected with a decrease in normalized computational time below 0.8. In both plots of Figure 4.6, fluctuations in the computational times are caused by sample approximations, and low numbers of samples for the extreme cases.

The Sørensen-Dice measure shows a slightly better discriminating power than euclidean. In case of Sørensen-Dice, 1.38% of all pairs are categorized into bins with an average normalized computational time below 0.8. In comparison, for the euclidean distance, only 1.17% of all pairs were categorized into bins with a computational time below 0.8. Note as a reference that up to 26.2% of the pairs have a speedup of 0.8 or better (see Figure 4.5).

The histograms reported in Figure 4.6 give further an estimate on the statistics of the similarity measures over instances. From our experiments, reported in Figure 4.6, out of the 100 test instances, only 23 instances have a matching library, inside the training set of 1000 instances, where the training instances have a measure above 0.9998 in Sørensen-Dice and below 700 in

euclidean. This calls for a more diversified set of training instances, either achieved by an even larger training set, or by a different sampling of the domain of input delays. For instance, one can target coverage of the delay domain, by Sobol sampling; and not match the density of the probability, like the simple Monte Carlo scheme we used. For an actual real-life application, thus, we would expect to increase the set of instances for the precomputation of logic Benders cuts. This would increase the probability of finding good cuts for a wide range of instances, ahead of time, as it is required in reality.

Finally, we analyze the benefits of the similarity measure for computational benefit of cuts under real-world conditions. For this analysis we first focus on 23 testing instances, for which we have available highly similar training instances (libraries), i.e., those with at least one library showing a measure above 0.9998 in Sørensen-Dice or below 700 in the euclidean distance. We report in Figure 4.7 a similar analysis as in Figure 4.4, with the top plot showing the normalized computational time, the middle plot the iterations, and the bottom plot the amount of cuts considered. Instead of reporting the performance of the best library (in terms of speedup) over the entire training set, which in reality would be a-priori unknown, we report the average performance over the increasing set of the most similar libraries.

Specifically, we rank the instances by the similarity measure, and we run the algorithms with each library from the best k ranked ones. We then take the average performance over this set. We consider an increasing k, and analyse how the performance changes. The x-axes of those three plots are the same, and describe a logarithmically increasing set of training library - test instance pairs k considered from 1 (i.e., only the best ranked), up to 1000 (i.e. all libraries available).

We rank the libraries separately for Sørensen-Dice (decreasing), reported in all plots as orange solid line; and euclidean (increasing), reported in all plots as red, dashed line. There are actually very little differences between the two measures. We also report a random selection of libraries, i.e., without a similarity measure (reported in black).

If our similarity measures are able to identify the most valuable libraries, we expect for a small set of training library - test instance pairs considered, the best performance. Increasing the amount of pairs, we expect the performance benefits to decrease to a larger average normalized computational time. This is due to the fact that the increasingly added instances have a smaller similarity and therefore result in a decreasing performance on average.

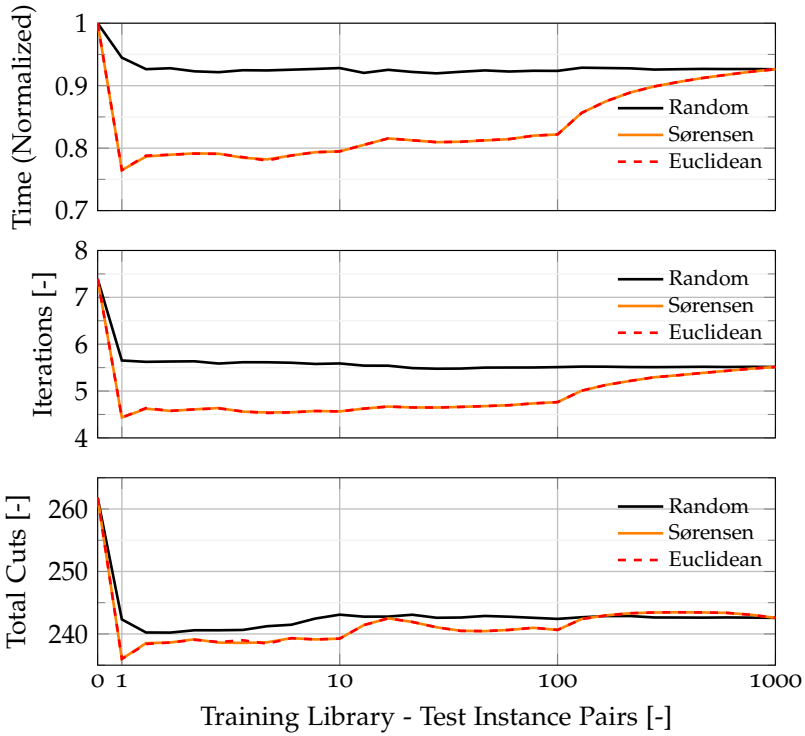

FIGURE 4.7: Computational performance on an increasing selection of Training Library- Test Instance Pairs considered. Reported on test instances with pairs of high similarity measures only.

In Figure 4.7 we can indeed see such effect, of sharp decrease (i.e. the most similar is actually good), and rebound (i.e. the less similar are actually not good, going towards the right along the x-axis) for testing instances with similar libraries. In the plots describing computational time and the iterations, this phenomenon is particularly visible: the improvement in performance considering the most similar library achieves a value well below 0.8, on average. We conclude that our measures are able to indicate libraries with high computational value.

We extend the analysis of computational benefits for the reuse under real-world conditions to all testing instances in Table 4.4. We exclusively focus on the Sørensen-Dice measure as the euclidean has in general shown identical results. In Table 4.4 we report the average normalized computational time

	Number of Libraries				
Sørensen	1	10	100	500	1000
$1 \geq$ ≥ 0.9998	0.8091 (3%)	0.7484 (23%)	0.7505 (23%)	0.7379 (23%)	0.7391 (23%)
$0.9998 >$ ≥ 0.9989	1.1265 (73%)	1.2364 (76%)	1.2102 (76%)	1.1942 (76%)	1.2310 (77%)
$0.9989 >$ ≥ 0.998	1.0995 (24%)	0.6583 (1%)	1.3006 (1%)	0.9582 (1%)	-

(*): Percentage share of test instances, with the best library in that category.

TABLE 4.4: Computational Time (Normalized) over an increasing Number of Libraries.

over testing instances. We consider a reduced set of libraries (with increasing size of 1, 10, 100, 500 and 1000) from the training instances, which are randomly selected from the full set of 1000 libraries available. The idea is to understand the benefit of a larger set of libraries. We consider for each testing instance and each reduced set of libraries the performance of the most similar library in the reduced set of libraries, with respect to the Sørensen-Dice measure. In other terms, we take the single most similar out of 1, 10, 100, 500 and 1000, and we use that as library of reference. We report computational time (normalized by no reuse), further differentiated by the Sørensen-Dice measure of the most similar library (to distinguish cases where the most similar library is indeed highly similar, or less). In brackets, we report the percentage of test instances and corresponding most similar library into the three similarity classes. For the case of library with a unitary size, we repeat the test 10 times and take the average.

In Table 4.4 we can see that, if only one library is considered per test instance, only few test instances show high similarity to that library (i.e., $\geq$ 0.9998); 73% of the instances show medium similarity and 24% of instances show a low similarity. The medium and low similarity of libraries reflects in the average computational times, which are for medium similarity and low similarity clearly > 1. When considering 10 or more libraries per testing instance, we can find (as discussed earlier) 23% of the instances with a highly similar library, and for almost all remaining test instances a library with medium similarity. The test instances with highly similar libraries

show a computational speedup of $\sim 25\%$, empirically showing the ability of Sørensen-Dice to identify beneficial libraries.

Further, we can see in Table 4.4 that little computational improvements are made by increasing the set of libraries considered from 10 to 1000. We explain such behavior by the fact that Sørensen-Dice is an imperfect estimator; and seems not able to differentiate the very best, when sufficiently many libraries of similar quality are available. Moreover, the results in Table 4.4 can be explained by the delay in the different test instances. The 23% of instances with a highly similar library are instances with very common delays, which drastically increases the probability of a highly similar library. The remaining instances are instances with uncommon, and large, input delay, drastically reducing the probability of highly similar libraries. In that case, even 1000 training instances are insufficient to produce a highly similar library.

4.8 CONCLUSION

In this paper, we introduce an approach to enable a faster solution process for railway rescheduling, considering precomputation and statistical measures of similarity for the input instances. We start from adapting the logic Benders decomposition for timetabling of [25] to the purpose of real-time rescheduling, where computational time is strongly limited. We propose the reuse of precomputed logic Benders cuts to accelerate the iterative decomposition process, exploring the similarity between rescheduling problems dealing with the same railway network, traffic and timetable, and varying input delay. With a modification on precomputed Benders cuts we are able to reuse any precomputed cut, as far as the instance (i.e. trains, infrastructure, planned timetable) is the same. For the reuse of cuts, we propose a lazy-constraint approach, which only includes in the master those Benders cuts, that provide progress in the iterative decomposition process. We propose two measures that are able to identify the potential computational benefit from a library of precomputed Benders cuts, based on the similarity of the input delay of the training (which generated the library) and test instance. This allows to avoid an excessive amount of logic Benders cuts considered for reuse, and prevents large amounts of constraints being included in the master problem of the Benders decomposition.

With the lazy-constraint approach we are able to exclude useless precomputed cuts from the reuse and avoid to add those to the master problem. This prevents an exponential growth of computational time (twice as slow,

when 95% additional cuts are considered) due an increasing size of the master problem, when direct reuse would be considered. The reuse of the best logic Benders cuts under a lazy-constraint scheme can find a solution twice as fast as the normal Benders decomposition without precomputation and reuse of cuts. Compared to a benchmark with a commercial solver on a centralized approach, we achieve an overall speedup up to a factor of 5.

With a further analysis, we are able to show that similarity in input delay is a promising estimator for the computational benefit of reusing cuts from a library. Both proposed similarity measures, for the considered delay statistics, are able to identify the 1% of instances which result in a speedup. The reuse of Benders cuts from libraries from instances with similar input delay achieved a comparable acceleration as in the ideal case of reuse with the perfect logic Benders cuts. This underlines the value of reusing logic Benders cuts in a real-world application.

The computational results underline the practical benefits of findings in this paper. In reality, rescheduling actions from railway dispatchers are expected between within 3 [14] up to 10 [22] minutes. With the novel methodology of this paper, we are able to solve instances previously only solvable in over 3 minutes, in around 1 minute of computational time; this emphasizes the practical benefits of the proposed methodology.

Future research should clearly include additional studies on identifying the potential of precomputed cuts and larger libraries. In our case, if libraries with a sufficiently high similarity exist, the speedup is perceivable. Thus, a detailed sampling of the actual delay domain (which depends on the instance and operations, in general) has to cover the possible delays with a sufficiently fine-grained detail. It is advisable for a real-world application, where more sophisticated data structures and very large libraries are acceptable, to increase the number of training instances, and to use more sophisticated sampling schemes. While we were able to propose two valuable measures for determining the similarity of instances, it remains an open question for future research, whether other features can determine the similarity of instances towards usability of Benders cuts, for a higher computational speedup.

ACKNOWLEDGMENTS

The authors thank the colleagues from SBB for the useful discussions and support during the implementation. This project was supported by the ETH Zürich Foundation.

REFERENCES

1. Asin, R., Nieuwenhuis, R., Oliveras, A. & Rodriguez-Carbonell, E. *Efficient generation of unsatisfiability proofs and cores in SAT* in *International Conference on Logic for Programming Artificial Intelligence and Reasoning* (2008), 16.
2. Balas, E. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics* **89**, 3 (1998).
3. Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M. & Schlechte, T. Recent success stories on integrated optimization of railway systems. *Transportation Research Part C Emerging Technologies* **74**, 196 (2017).
4. Bretas, A., Mendes, A., Chalup, S., Jackson, M., Clement, R. & Sanhueza, C. *Modelling railway traffic management through multi-agent systems and reinforcement learning* in *In Elsawah, S.(ed.) MODSIM2019, 23rd International Congress on Modelling and Simulation* (2019), 291.
5. Cacchiani, V., Caprara, A., Galli, L., Kroon, L., Maróti, G. & Toth, P. Railway rolling stock planning: Robustness against large disruptions. *Transportation Science* **46**, 217 (2012).
6. Cacchiani, V., Caprara, A. & Toth, P. A column generation approach to train timetabling on a corridor. *4OR* **6**, 125 (2008).
7. Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L. & Wagenaar, J. An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B Methodological* **63**, 15 (2014).
8. Caimi, G., Fuchsberger, M., Laumanns, M. & Lüthi, M. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. *Computers & Operations Research* **39**, 2578 (2012).
9. Corman, F., D'Ariano, A., Pacciarelli, D. & Pranzo, M. Dispatching and coordination in multi-area railway traffic management. *Computers & Operations Research* **44**, 146 (2014).
10. Corman, F., D'Ariano, A., Pacciarelli, D. & Pranzo, M. Optimal inter-area coordination of train rescheduling decisions. *Transportation Research Part E Logistics and Transportation Review* **48**, 71 (2012).

11. Corman, F., D'Ariano, A., Pranzo, M. & Hansen, I. A. Effectiveness of dynamic reordering and rerouting of trains in a complicated and densely occupied station area. *Transportation Planning and Technology* **34**, 341 (2011).

12. Corman, F. & Meng, L. A Review of Online Dynamic Models and Algorithms for Railway Traffic Management. *IEEE Transactions on Intelligent Transportation Systems* **9**, 1 (2014).

13. D'Ariano, A. & Hemelrijk, R. Designing a multi-agent system for cooperative train dispatching. *IFAC Proceedings Volumes* **12** (2006).

14. D'Ariano, A., Pacciarelli, D. & Pranzo, M. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operations Research* **183**, 643 (2007).

15. Davis, M., Logemann, G. & Loveland, D. A Machine Program for Theorem-Proving. *Communications of the ACM* **5**, 394 (1962).

16. Dice, L. R. Measures of the Amount of Ecologic Association Between Species. *Ecology* **26**, 297 (1945).

17. Dollevoet, T., Huisman, D., Kroon, L. G., Veelenturf, L. P. & Wagenaar, J. C. Application of an iterative framework for real-time railway rescheduling. *Computers & Operations Research* **78**, 203 (2017).

18. Ghasempour, T. & Heydecker, B. Adaptive railway traffic control using approximate dynamic programming. *Transportation Research Part C Emerging Technologies* **113**, 91 (2020).

19. Gurobi Optimization, L. *Gurobi Optimizer Reference Manual* 2021.

20. Keita, K., Pellegrini, P. & Rodriguez, J. A three-step Benders decomposition for the real-time Railway Traffic Management Problem. *Journal of Rail Transport Planning & Management* **13**, 100170 (2020).

21. Lamorgese, L. & Mannino, C. A non-compact formulation for job-shop scheduling problems in traffic management. *Operations Research* **67**, 1503 (2019).

22. Lamorgese, L. & Mannino, C. An Exact Decomposition Approach for the Real-Time Train Dispatching Problem. *Operations Research* **63**, 48 (2015).

23. Lamorgese, L., Mannino, C. & Piacentini, M. Optimal Train Dispatching by Benders'-Like Reformulation. *Transportation Science* **50**, 910 (2016).

24. Leutwiler, F., Bonet Filella, G. & Corman, F. Accelerating logic-based Benders Decomposition for Railway Rescheduling by exploiting similarities in delays. *Computers and Operations Research* **150**, 106075 (2023).

25. Leutwiler, F. & Corman, F. A logic-based Benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research* **303**, 525 (2022).

26. Liu, J., Chen, L., Roberts, C., Li, Z. & Wen, T. *A Multi-agent Based Approach for Railway Traffic Management Problems* in *2018 International Conference on Intelligent Rail Transportation (ICIRT)* (2018), 1.

27. Luan, X., De Schutter, B., Meng, L. & Corman, F. Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. *Transportation Research Part B Methodological* **141**, 72 (2020).

28. Marques-Silva, J. P. & Sakallah, K. A. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers* **48**, 506 (1999).

29. Mascis, A. & Pacciarelli, D. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operations Research* **143**, 498 (2002).

30. Moura, L. d., Dutertre, B. & Shankar, N. *A tutorial on satisfiability modulo theories* in *International Conference on Computer Aided Verification* (2007), 20.

31. Pellegrini, P., Marlière, G., Pesenti, R. & Rodriguez, J. RECIFE-MILP: An Effective MILP-Based Heuristic for the Real-Time Railway Traffic Management Problem. *IEEE Transactions on Intelligent Transportation Systems* **16**, 2609 (2015).

32. Perrachon, Q., Chevrier, R. & Pellegrini, P. Experimental study on the viability of decentralized railway traffic management. *WIT Transactions on the Built Environment* **199**, 337 (2020).

33. Samà, M., D'Ariano, A., Corman, F. & Pacciarelli, D. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers & Operations Research* **78**, 480 (2017).

34. Van Thielen, S., Corman, F. & Vansteenwegen, P. Considering a dynamic impact zone for real-time railway traffic management. *Transportation Research Part B Methodological* **111**, 39 (2018).

35. Veelenturf, L. P., Kidd, M. P., Cacchiani, V., Kroon, L. G. & Toth, P. A railway timetable rescheduling approach for handling large-scale disruptions. *Transportation Science* **50**, 841 (2016).

APPENDIX

4.9 SMT$_{Agg}$ - GENERATION AND AGGREGATION OF LOGIC BENDERS CUTS

In this appendix we repeat details of Algorithm SMT$_{Agg}$ in [25]. We use SMT$_{Agg}$ in this paper to evaluate the feasibility of the subproblem $\mathcal{S}^\alpha$ (Problem 4.4) in the decomposition of Section 4.4.2. In case of infeasibility, Algorithm SMT$_{Agg}$ creates a logic Benders cut as in [25]. Algorithm SMT$_{Agg}$ is based on Satisfiability Modulo Theories (SMT), which is a combination of Satisfiability (SAT) solving [15] and a first-order logic [30]. Furthermore, concepts of [1] are used in SMT$_{Agg}$ for the discovery of infeasibility proofs.

Algorithm SMT$_{Agg}$ is an aggregation of logic Benders cuts, such that if SMT$_{Agg}$ is invoked on a subproblem, the algorithm computes not a single, but multiple logic Benders cuts. By the aggregation, the total number of iterations till convergence in the Benders scheme is reduced. Inside

Algorithm 4: SMT

input : $\mathcal{S}^\alpha$
output: $\mathcal{O}^\alpha$, $\mathcal{I}^\alpha$, β^α
init : $\Phi \leftarrow \mathcal{S}^\alpha$, $G^\alpha \leftarrow \mathcal{S}^\alpha$, $\theta = \emptyset$

```
1  while true do
2      confl ← UnitPropagation(Φ, θ)
3      if !confl then
4          confl ← Evaluate(G^α(θ))
5      if !confl then
6          if θ = complete then
7              O^α ← G^α(θ)
8              return (O^α, ∅, ∅)
9          θ ← θ ⋃ Decide()
10     else
11         if confl = Unsatisfiable then
12             I^α ← AnalyzeIP(confl)
13             β^α ← BendersCut(I^α)
14             return (∅, I^α, β^α)
15         else
16             Analyze(confl)
17             Backtrack(confl)
```

SMT_{Agg}, Algorithm SMT (Algorithm 4) of [25] is invoked multiple times, with different initial conditions to discover multiple logic Benders cuts.

In Algorithm SMT (Algorithm 4), Φ are constraints of SAT used to model the decision and choice sets of $\mathcal{S}^\alpha$. $\mathcal{I}^\alpha$ is an infeasibility proof for $\mathcal{S}^\alpha$ used to derive the logic Benders cut β^α. Algorithm 4 proceeds by extending iteratively an initially empty selection θ through new choice sets W_c (line 9), until the selection is complete (line 6) or an unsatisfiable constraint has been found (line 11). `Decide` in line 9 selects, by some SAT heuristics, new choice sets W_c to extend θ. If θ is a complete selection (line 6), the algorithm returns a feasible solution of the subproblem, derived by the disjunctive graph of the subproblem (line 7). After every extension of θ in line 9, unit propagation [28] is performed (line 2) and θ is extended by additional choice sets, that are implied through the constraints in Φ. If after such extension of θ, a violated constraint (`confl`) is found in Φ, such constraint is either generally unsatisfiable (line 11) and an infeasibility proof, together with a logic Benders cut can be derived (line 12 and 13); or the constraint can be satisfied by a different θ and selection must be adjusted by first analyzing the violated constraints (line 16) and then removing appropriate choice sets from θ (line 17).

SMT_{Agg} (Algorithm 5) invokes Algorithm 4 multiple times with different master solutions imposed to the subproblem $\mathcal{S}^\alpha$ to generate multiple logic Benders cut in a single iteration of the Benders scheme. Algorithm 5 start by appliying Algorithm 4 on the subproblem $\mathcal{S}^\alpha$, which has imposed the latest master solution $\mathcal{O}^\alpha_\mathcal{M}$ (line 3). In case Algorithm 4 discovers an infeasibility proof and a logic Benders cut, $\mathcal{S}^\alpha$ is modified; the master problem is extended by the cut. This means that the next iterations will result in a solution $\mathcal{O}^\alpha_\mathcal{M}$ which will satisfy the latest discovered Benders cut β^α_i (line 4). Algo-

Algorithm 5: SMT_{Agg}, Benders cut aggregation scheme for a subproblem.

input : $\mathcal{S}^\alpha$
output: $\{\beta^\alpha_1, \beta^\alpha_2, \cdots\}$
init : $i = 1$, $\mathcal{S}'^\alpha = \mathcal{S}^\alpha$

1 **while** $\mathcal{O}^\alpha = \emptyset$ **do**
2 $\quad \mathcal{O}^\alpha, \mathcal{I}^\alpha_i, \beta^\alpha_i \leftarrow$ `SMT(`$\mathcal{S}'^\alpha$`)`
3 $\quad$ **if** $\mathcal{I}^\alpha_i \neq \emptyset$ **then**
4 $\quad\quad \mathcal{S}'^\alpha \leftarrow$ `Modify(`$\mathcal{S}'^\alpha, \mathcal{I}^\alpha_i$`)`
5 $\quad i \leftarrow i + 1$
6 **return** $\{\beta^\alpha_1, \beta^\alpha_2, \cdots\}$

rithm 4 is reapplied to the modified version of the subproblem $\mathcal{S}'^{\alpha}$ to discover further Benders cuts. Eventually, Algorithm 4 in line 4 will return a feasible solution and Algorithm 5 terminates and returns all aggregated cuts.

5

HEURISTICS OF SET COVERING IN A BENDERS DECOMPOSITION FOR RAILWAY TIMETABLING

Leutwiler, F. and Corman, F. (submitted). Heuristics of Set Covering in a Benders Decomposition for Railway Timetabling. Manuscript submitted to Computers and Operations Research on 09.2022.

This is a pre-print (revised) version, which is currently under revision in Computers and Operations Research.

Key findings:

- A logic Benders decomposition for railway timetabling to which combinatorial Benders cuts can be applied and the master problem is a set covering problem.
- Incremental heuristics are an efficient method to address the problem of set covering in the Benders decomposition scheme.
- With incremental heuristics on set covering the problem of timetabling can be solved ~ 20 times faster with only an average loss of quality (additional minutes delay) of ~ 7.5.

Author contributions: Model and Decomposition: FL; Design of Heuristics: FL; Implementation: FL; Manuscript preparation and review: FL, FC.

ABSTRACT

Railway timetabling is a major challenge in the operation of railway. The timetable of a railway determines times, orders and routes of trains on the network and thereby defines the performance of the entire railway system. Railway operators are keen to maximize the economic performance of their railway system, such that high expectations are given towards the design of a timetable. With a Benders decomposition we propose an approach on the problem of timetabling, where we consider optimality of a timetable separately from the feasibility of a timetable. In the decomposition, optimality is determined in a problem of set covering, while feasibility is evaluated on the constraints of timetabling. With efficient heuristics on the problem or set covering in our decomposition, high quality solution for the problem of timetabling are provided in short time. The novel approach provides heuristic solutions up to $\sim$ 20 times faster than standard approaches by commercial solvers, with an average gap of $\sim$ 7.5% in the optimality of solutions. Extensive experiments empirically confirm the benefits of the new approach.

5.1 INTRODUCTION

Railway timetabling is the problem of designing a timetable for the operation of a railway system. The timetable for a railway system defines departure, passing and arrival times for all trains on all points in the network and thereby sets the performance of the entire system; the importance of a good design for a timetable is crucial. With a well designed timetable, railway operators are able to explore the economic potential of available resources in the railway system and guarantee the profitability the railway company.

Railway operators are keen to automate the process of railway timetabling. With the automation, a more global and comprehensive view on the problem of timetabling is possible, in comparison to the human planners of nowadays, such that novel timetables further increase the performance of railways. Recent advances in academia [3] show promising results, leading to railway timetables of higher performance. (e.g. 6% more punctual trains and 9% less trains with delays over 15min compared to current practices in [25] for automated timetabling in real-time rescheduling).

Still, a need to solve bigger instances exists (e.g., [24]) The problem of timetabling is known to be NP-Hard [30], which makes it difficult for algorithms used in automation, to scale towards bigger instances and achieve

optimal solutions for timetabling on a large scale, e.g., an entire railway network. Often automated approaches are applied only locally and used as supporting tools for a human planner.

In this paper we address the problem of railway timetabling. This is the determination of times of arrivals, departure and passing of railway services along a railway line. We consider a high level of detail (microscopic) by considering constraints at the level of each infrastructure element, as well as routing flexibility. We decompose the formulation of timetabling by a Benders decomposition [19] using the combinatorial Benders cuts of [8]. In our Benders decomposition, the master problem is a problem of set covering and the subproblem is a problem of timetabling, where feasibility must be evaluated. We propose multiple near-optimal heuristic approaches for the master problem, i.e., a problem of set covering, and particularly exploit the fact that in the scheme of Benders decomposition the master is growing incrementally in constraints, over the iterations of the Benders scheme. With our novel approach, we propose an efficient alternative to existing approaches with improved scalability, contributing towards the gap between academia and large scale practical applications.

This paper is structured as following. In Section 5.2 we review related literature and state the contribution of this work. In Section 5.3 we state the problem of timetabling and provide a disjunctive formulation in Section 5.4. We propose a Benders decomposition for the problem of timetabling in Section 5.5. In Section 5.6 we introduce multiple approaches to the problem of set covering including a novel set of heuristics, designed for a problem of set covering incrementally growing in constraints. Exhaustive experiments in Section 5.6 empirically confirm the strength of our novel heuristic approach 5.7 compared to several existing benchmarks. We conclude in Section 5.8.

5.2 RELATED WORK

In this paper, we deal with a microscopic variant of the timetabling problem; this has much similarity with railway timetabling problems of the literature, but also with railway rescheduling, i.e., the real-time adjustment of railway timetables to a delayed situation. We comprehensively name those two problems as railway scheduling. In this section we provide a brief overview of the literature on the topics of scheduling and decomposition in railways to position our work in the existing literature. We conclude this section with the contributions of this work.

5.2.1 *Railway Scheduling*

In railway scheduling, we may classify models by four major aspects: granularity of infrastructure, representation of time, inclusion of routing and periodicity.

In the literature, we find two main classes of infrastructure models. Macroscopic models, i.e., coarse granular models, abstract the railway network into nodes and lines (e.g., [16, 37]). Microscopic models, i.e., fine granular models, consider the infrastructure at the level of the safety systems, divided into blocks of railway track, few hundred meters long (e.g., [9], [32], [36]). Only microscopic models can represent conflict free movements and routing of trains over the network. Those have been mostly applied for rescheduling.

The times of operations, which are to be scheduled, can either be modeled in discrete form, i.e., by discrete variables (e.g., [5]), or in continuous form, i.e., by continuous variables (e.g., [13]).

In microscopic models, routing of trains may be considered. The routing has a strong influence on the complexity of scheduling problems, such that problems including routing decisions for trains are in general much more complex. Models of railway scheduling in general consider routing decisions in additional variables of the problem (e.g., [32]); models excluding routing decisions in general consider the routes of trains as a given input to the model (e.g., [13]).

Scheduling can be performed including constraints of periodicity (e.g., [31]). Including such constraints, the result is a periodic schedule, which can repetitively be applied to the railway systems. In this case the planning horizon of the scheduling problem can be reduced to a single period and then rolled out over multiple periods to create a timetable, e.g., for an entire day. While decreasing the planning horizon, constraints of periodicity notably increase the complexity of the scheduling problem.

The different models of railway scheduling are addressed by many different methods throughout the literature. Extensive overviews of methods are provided in [4] or [17]. In general approaches can be differentiated by whether an optimal solution is computed or a heuristic is used to find near-optimal solutions. Heuristic approaches are in general a trade off between closeness to optimality of a solution and computational time. A group of heuristics in railway scheduling is based on rules, specific to the application of railways, where it is iteratively decided on the variables of the scheduling problem (e.g., First-Come-First-Served [17], Arc-Greedy Heuristics [34]). Other heuristics in railway scheduling are applications of heuristics in gen-

eral mathematical programming (e.g., Variable Neighbourhood Search [35], Tabu-Search [10] or Genetic-Algorithms [17]). Further heuristics address elements of the scheduling problem in different stages, e.g., trains in the order of priority [21, 27], or ordering decisions before routing decisions [12].

5.2.2 *Decomposition*

In the literature of railway scheduling numerous decomposition approaches can be found. Decomposition approaches often show better scalability than centralized (undecomposed) approaches, which makes them an interesting class of approaches to tackle large-scale problems of railway scheduling. A group of approaches (e.g., [9], [25]) propose geographic decompositions, where the scheduling problem is decomposed based on the geographic position of railway infrastructure. Other groups of approaches propose temporal (e.g., [28]), entity (resource) based (e.g., [6]) or generic decompositions based on properties of the underlying optimization problem (e.g., [23], [22]).

Generic decompositions (e.g., [22, 23] or [12]) can be considered as applications of the general decomposition techniques from mixed-integer programming as discussed, e.g., in [38]. In these decompositions, variables in an optimization problem are optimized in different groups, where groups are within a hierarchical structure. Depending on the grouping of variables, particular groups of variables result in optimization problems of different structure and class, e.g., linear programming [23], or mixed-integer programming [12, 22]. We consider the decomposition of this work as a generic decomposition.

5.2.3 *Benders Decomposition*

Benders decomposition [2] is a hierarchical decomposition procedure for mathematical programming. The decomposition is designed for problems of mathematical optimization for which it is possible to identify complicating and non-complicating variables. Complicating variables are the main cause of the complexity in the problem; if complicating variables are fixed to constant value, the remaining problem is significantly easier to solve. Variables that are not complicating variables are denoted non-complicating variables. The Benders decomposition scheme is an iterative optimization of the complicating variables (denoted as the master problem) and an optimization of the non-complicating variables (denoted as the subproblem). In the subproblem, in which complicating variables are considered constant,

constraints (Benders cuts) are determined, which are then added to the master problem. By the Benders cuts, eventually an optimal master solution will be found, for which a feasible subproblem solution exists, leading to a global optimal solution.

In the standard application of Benders decomposition to mixed-integer linear programming, integer variables are complicating and continuous variables are non-complicating. In this case, the subproblem is a problem of linear programming and standard Benders cuts [19] can be used.

In [8] the authors show that in the special case, where integer variables only appear together with continuous variables in constraints of big-M, and where the objective is independent of the continuous variables, the standard Benders cut can be strengthened to the combinatorial Benders cut. In this special case of mixed-integer programming, the solution of the master problem is a solution over binaries of big-M constraints and implies a set of (big-M) constraints to the subproblem. If the subproblem, including such implied constraints, is feasible, a global optimal solution has been found. In case the subproblem is infeasible, there exists an infeasible subset of constrains within all the constraints of the subproblem, that is the reason for the infeasibility. The combinatorial Benders cut is a constraint cutting off solutions from the solution space of the master problem, which would lead to the particular infeasible subset of constraints in the subproblem, which has been used to derive the cut.

5.2.4 *Contribution*

We consider in this work the problem of railway timetabling through a microscopic, non-periodic model including the routing of trains. Our model of timetabling extends the model of [26]. The objective of the timetabling problem is the delay of events, with regards to the latest preferred time given. We discretize this objective and propose a generic decomposition of the problem by a logic Benders decomposition. Despite it is built on the same mathematical model, this decomposition is fundamentally different from the decomposition of [26]. In the new decomposition, thanks to the discretization of the objective, we can apply the combinatorial Benders cuts of [8]. In our decomposition, the master problem is a problem of set covering; to solve it, we introduce various heuristic solution approaches. With the efficient heuristics we can solve much faster and/or with a better objective, the microscopic timetabling problem addressed, compared to a series of benchmarks. The particular contributions of this work are:

1) We propose a Benders decomposition on a disjunctive formulation of railway timetabling, where the master problem is problem of set covering and the subproblem is a problem of timetabling, for which only feasibility must be evaluated. In the Benders decomposition we can apply the combinatorial Benders cut of [8].

2) We introduce several heuristic approaches to solve the problem of set covering in the proposed Benders decomposition. We propose heuristic approaches exploring particularly the incremental growth in constraints of the set covering problem, inside the scheme of Benders decomposition.

3) In an exhaustive series of experiments, we provide empirical evidence to quantify the performance of our novel approach. We put our approach into perspective with an existing general approach to timetabling [18] using the two commercial solvers Gurobi [20] and Z3 [15]. In comparison to the general approach, we can solve instances of timetabling up to ~ 20 times faster with only an average gap of $\sim 7.5\%$ to an optimal solution.

5.3 PROBLEM DESCRIPTION

The problem of railway timetabling is to design a schedule (the timetable) for all operations performed by trains on the railway network. The following description of railway timetabling extends the description of [26].

We consider microscopic timetabling; that is the railway infrastructure is represented with a microscopic detail, where a single operation of a train corresponds to the train passing a single block. Such operation is scheduled by defining a continuous time for the start and end *event* of the operation, i.e., the entry and exit of the train to the block. The solution of a timetabling problem is a timetable, which determines, for all trains in the network, an exit and an entry time for each block that is passed by the train.

With service requirements for timetabling, railway operators define desired regularities, frequency of services and passenger transfers which should be established in the resulting timetable. We consider service requirements in form of temporal limitations, i.e., earliest and latest times for arrivals and departures, as well as required minimal (maximal) duration between an arrival and a departure of two trains to assure enough time for passenger transfers.

Differently to [26], we consider in this work *planning deviations* for timetabling. We do this to enlarge the feasible solution space, and consider the possibility that an arrival or departure event is scheduled actually shortly *after* its given latest time from the service requirements. In this case,

we consider the planning deviation for such event to be the difference in time between the latest time given for the event and the actual time the event is scheduled in the timetable. We consider furthermore planning deviation to be limited to a maximal allowed deviation, that is the same for all events and given by the railway operators.

With *discrete decisions* in railway timetabling we determine the order or route of trains. A discrete decision is to select a single *choice*, out of a finite set of choices, where each choice has consequences (in form of constraints) for the events of the timetabling problem.

Safety regulations require that no concurrent occupation of any block in the network by two or more trains, i.e., no resource conflict, occurs in the network. We consider an *ordering decision* for any pair of trains with a possible resource conflict to avoid any such conflicts in the final timetable. The ordering decision consists of two choices, that are the two possible orderings for a pair of trains, where the constraints related to each choice prevent the occurrence of the conflict.

We consider the infrastructure available to a train, restricted to a limited set of routing alternatives, where each *routing alternative* is a sequence of blocks in the network. The limited infrastructure is defined by railway operators and includes only the few most plausible routing alternatives according to the type of service of the train. We consider a *routing decision* to select a routing alternative (choice) to be used by a train. Routing alternatives are geographically grouped into *routing areas*. In each routing area, multiple routing alternatives are in parallel between two unique points in the network, that are the entry and exit point to the routing area. To select one routing alternative in a routing area corresponds to a routing decision. A single train may cross multiple routing areas on its path from the origin of the train to the destination and thus may require multiple routing decisions. The *route* of a train is a unique sequence of blocks from origin to destination, determined by the choices on all routing decisions of the train.

In summary, in the problem of microscopic railway timetabling, we optimize over event times, routing and ordering decisions to compute a conflict-free timetable, which minimizes the sum of all planning deviations over all arrival and departure events.

5.4 A MODEL FOR RAILWAY TIMETABLING

We use a disjunctive formulation [1] to model the problem of railway timetabling. In particular, we use the formulation of [26] and adapt it to

the minimization of sum of total planning deviation. The following section summarizes the disjunctive model from [26].

For a train d, the operation related to block b is indicated (d, b) and the related time for the start event of such operation (d, b) by the continuous variable $t_{db} \in \mathbb{R}_+$. Variables for the end event of an operation are redundant as a sequence of operations cannot be paused in railway timetabling.

The precedence relation $((d, b), (q, p))$ is to model a temporal dependency between events. The precedence relation is a linear inequality constraint to assure that t_{qp} is at least $f_{db,qp}$ time units scheduled after t_{db}. Precedence relations in timetabling as of [26] are either *fixed* or *selectable*. Precedence relations that are fixed must hold in any solution of railway timetabling. Selectable precedence relations must hold upon selection. Precedence relations are selected by the discrete decisions of railway timetabling. Selectable precedence relations are grouped into *choice sets* W_c, and choices sets into *decisions sets* D_l, to model the discrete decisions. A choice s imposes (selects) the precedence relations in W_c jointly, to be satisfied by the events of the timetable. A discrete decision l is modeled as the disjunctive constraint

$$\bigvee_{W_c \in D_l} \bigwedge_{((d,b),(q,p)) \in W_c} \left(t_{qp} - t_{db} \geq f_{db,qp} \right). \tag{5.1}$$

With a set of fixed precedence relations A_f, the minimal travel times of trains over blocks outside of routing areas and further all earliest times as well as minimal (and maximal) transfer times are modeled. A time origin $t_0 = 0$ is used to model earliest times as precedence relations.

With a set of selectable precedence relations A_s, minimal travel times on blocks inside of routing areas, and constraints for either order on a pair of trains are modeled. Selectable precedence relations of minimal travel times are grouped into choice sets W_c as the related blocks are in routing alternatives; all choice sets of all routing alternatives in the same routing area build a decision set D_l for the routing decision. The constraint for one order on a pair of trains is a single precedence relation, such that the decision set of an ordering decision contains two choice sets, each with a single precedence relation to impose either order of the pair of trains.

Auxiliary variables are used to model the dependency of ordering decisions and routing decision. An ordering decision is necessary if infrastructure is shared among a pair of trains; whether infrastructure is shared depends on the chosen routing alternatives. Auxiliary variables are additional (artificial) events, used in the precedence relations of ordering decision to model the dependency between ordering and routing decisions.

Auxiliary variables duplicate events in routing alternatives and are constrained to equality with the original variables if and only if the appropriate routing alternative is chosen.

5.4.1 *Minimizing Planning Deviation*

We model the planning deviation in timetabling by a second set of fixed precedence relations A_δ. Precedence relations in A_δ model latest times of events and identify the planning deviation δ_{db} of the arrival or departure event (d, b). The planning deviation δ_{db} is defined as the difference in time between the actual time of the event t_{db} and the latest time f_{db} of the event, given as an input to the problem. We do not consider negative planning deviations, such that $\delta_{db} \geq 0$. We can identify the planning deviation δ_{db} of an event (d, b) by the following precedence relation

$$\delta_{db} - t_{db} \geq -f_{db}. \tag{5.2}$$

Different from A_f, precedence relations in A_δ are defined on only one event, e.g., event (d, b).

If we denote by L the set of all discrete decisions of timetabling and further reduce the notion of an operation (d, b) to i, we can write the problem of railway timetabling as the disjunctive program,

$$\begin{aligned}
\min \quad & \sum_{(i) \in A_\delta} \delta_i & & \\
\text{s.t.} \quad & \delta_i - t_i \geq -f_i & & (i) \in A_\delta \\
& t_j - t_i \geq f_{ij} & & (i,j) \in A_f \\
& \bigvee_{W_c \in D_l} \bigwedge_{(i,j) \in W_c} \left(t_j - t_i \geq f_{ij}\right) & & l \in L \\
& \delta_i \in \mathbb{R}_+,\ \delta_i \leq \delta_{max}\ \forall\, \delta_i, \quad t_i \in \mathbb{R}_+\ \forall\, t_i. & &
\end{aligned} \tag{5.3}$$

The constant δ_{max} is the maximal allowed deviation, that is the same for all events and given by the railway operators. The objective of Problem (5.3) is to minimize the sum of all planning deviations over all events given a latest time f_i.

5.4.2 *A discretization of Planning Deviation*

In Problem (5.3) planning deviations are represented by the continuous variables δ_i. To apply our approach and decompose Problem (5.3) into a

problem of set covering and a problem of feasibility, we need to discretize the planning deviation of arrival and departure events to a finite set of possible values; such values are later the selectable elements (sets) in the problem of set covering. We discretize the planning deviation of an arrival or departure event δ_i, originally in the interval $[0, \delta_{max}]$, to $K+1$ discrete values, i.e., $\left\{\bar{\delta}_{i,0}, \cdots, \bar{\delta}_{i,K}\right\}$, where $\bar{\delta}_{i,k} = k\frac{\delta_{max}}{K}$.

We then replace each precedence relation in A_δ of Problem (5.3) by a series of $K+1$ big-M constraints for all possible discrete values of δ_i, i.e.,

$$\bar{\delta}_{i,k} - t_i \geq -f_i - Mx_{i,k}, \quad k \in \{0, \cdots, K\}, \tag{5.4}$$

where $x_{i,k} \in \{0,1\}$. In the series of constraints (5.4), it must hold that $x_{i,K} = 0$, to always enforce the constraint of maximal allowed deviation. We keep the constraint $k = K$ in the series (5.4) for the simplicity of notation; in later experiments we will omit the big-M term of such constraint.

In the series of constraints (5.4), in case a binary $x_{i,k} = 1$, the related constraint of k is relaxed, i.e., trivially satisfied. Naturally, for the series of constraints (5.4), it holds that if there exists any assignment on the binaries $x_{i,k}$ satisfying the series of constraints (5.4) for a given set of values t_i, there exists a possibly different assignment on the binaries $x_{i,k}$ satisfying the series of constraints (5.4) for the same values of t_i, where it holds $x_{i,k+1} \leq x_{i,k}$. In other words if constraint (5.4) for k is satisfied by t_i (i.e., $x_{i,k} = 0$) any constraint (5.4) for p, where $p > k$ is trivially also satisfied. We can interpret this assignment on the binaries, where it holds $0 = x_{i,K} = x_{i,p+1} < x_{i,p} = x_{i,0} = 1$, as an allowed deviation for the event i of at most $\bar{\delta}_{i,p+1}$; all constraints up to (and including) p in the series (5.4) are relaxed as the corresponding $x_{i,k} = 1$. In case the constraints in the series (5.4) are relaxed up to constraint p, exactly $p+1$ constraints, i.e., $k \in \{0, \cdots, p\}$, are relaxed and the planning deviation δ_i of event i is limited to,

$$\frac{\delta_{max}}{K} \sum_{k \in \{0, \cdots, p\}} x_{i,k} \geq \delta_i. \tag{5.5}$$

In Equation 5.5 we can extend the sum to all binaries, i.e., $\{0, \cdots, K\}$, as binaries $x_{i,k} = 0$ for $k > p$.

In the above, we chose big-M constraints in (5.4) with binary variables $x_{k,i}$ instead of integer (discrete) variables for the discretization of planning deviation δ_i. In this manner, we achieve an optimization problem identical in constraints to the problem of [8] and can use their combinatorial Benders cuts in our decomposition of the timetabling problem.

With the discretization of constraints in A_δ by the series of constraints (5.5), we can translate Problem (5.3) into a problem of microscopic timetabling with discrete planning deviation,

$$
\begin{aligned}
\min \quad & \frac{\delta_{max}}{K} \sum_{(i) \in A_\delta} \left(\sum_{k \in \{0,\cdots,K\}} x_{i,k} \right) & & \\
\text{s.t.} \quad & \bar{\delta}_{i,k} - t_i \geq f_i - M x_{i,k} & & k \in \{0,\cdots,K\},\ (i) \in A_\delta \\
& t_j - t_i \geq f_{ij} & & (i,j) \in A_f \\
& \bigvee_{W_c \in D_l} \bigwedge_{(i,j) \in W_c} \left(t_j - t_i \geq f_{ij} \right) & & l \in L \\
& x_{i,K} = 0 & & \\
& t_i \in \mathbb{R}_+ \ \forall\, t_i, \quad x_{i,k} \in \{0,1\} \ \forall\, x_{i,k} & &
\end{aligned}
\tag{5.6}
$$

where the objective is now to minimize the total sum discretized planning deviations. If we consider continuous planning deviations in Problem (5.3), it holds for every optimal solution, that for each δ_i either the related constraint in A_δ is tight, i.e., $\delta_i = t_i - f_i$, or $\delta_i = 0$. That is, either there is no planning deviation or exactly as much deviation considered in the objective of Problem (5.3) as the event is actually scheduled after its latest time.

For Problem (5.6), instead, planning deviations are discrete. In this case, it is possible that $\bar{\delta}_{i,k} > t_i - f_i > \bar{\delta}_{i,k-1}$ and there is more planning deviation considered in the objective of Problem (5.6) than the event is actually scheduled after its latest time. This is simply due to the discretization of planning deviation.

5.5 SET COVERING IN RAILWAY TIMETABLING

We propose in this work a novel approach for railway timetabling where we decompose Problem (5.6) by a Benders decomposition into a problem of set covering as the master problem, and a problem of timetabling, where only feasibility is to be verified, as the subproblem. By the discretization of planning deviations in Section 5.4.2 we are able to apply the combinatorial Benders cut of [8] to our decomposition.

5.5.1 *A combinatorial Benders Decomposition*

We decompose our *centralized* problem $\mathcal{C}$ (Problem 5.6) into a *master* problem $\mathcal{M}$ and a single *subproblem* $\mathcal{S}$ according to standard Benders decomposition

[19]. We define the master problem to be the optimization over all binary variables $x_{i,k}$ of the Problem (5.6). We optimize all remaining variables in the subproblem. All constraints of Problem (5.6) in such decomposition are constraints in $\mathcal{S}$. The master problem $\mathcal{M}$ at the iteration α, i.e., $\mathcal{M}^\alpha$ in decomposition scheme of Benders can be written as

$$\begin{aligned} \min \quad & \sum_{(i)\in A_\delta} \left(\sum_{k\in\{0,\cdots,K\}} x_{i,k} \right) \\ \text{s.t.} \quad & \sum_{(i,k)\,\in\,\mathcal{I}^\alpha} x_{i,k} \geq 1 \qquad \forall \mathcal{I}^\alpha \\ & x_{i,k} \in \{0,1\} \ \forall x_{i,k} \end{aligned} \tag{5.7}$$

where β is a Benders cut and $\mathcal{B}^\alpha$ the set of all cuts aggregated till iteration α in the Benders scheme. In Section 5.5.2 we will show that β has the shape of the combinatorial Benders cut [8]. We further omit the factor $\frac{\delta_{max}}{K}$ from the objective of Problem (5.6) in $\mathcal{M}^\alpha$ as it does not change the optimality of a solution.

The subproblem $\mathcal{S}$ in our decomposition is the optimization over all variables of the centralized problem $\mathcal{C}$ except variables $x_{i,k}$, together with all constraints of $\mathcal{C}$. In our decomposition $\mathcal{S}$ is only a problem of feasibility; no variables of $\mathcal{S}$ appear in the objective of $\mathcal{C}$. The subproblem $\mathcal{S}^\alpha$ of our decomposition at iteration α can be written as,

$$\begin{aligned} \min \quad & 0 \\ \text{s.t.} \quad & \bar{\delta}_{i,k} - t_i \geq -f_i && (i,k) \in \bar{A}^\alpha_\delta \\ & t_j - t_i \geq f_{ij} && (i,j) \in A_f \\ & \bigvee_{W_c \in D_l} \ \bigwedge_{(i,j)\,\in\,W_c} \left(t_j - t_i \geq f_{ij}\right) && l \in L \\ & t_i \in \mathbb{R}_+ \ \forall t_i \end{aligned} \tag{5.8}$$

where $\bar{A}^\alpha_\delta := \left\{(i,k) \in A_\delta \times \{0,\cdots,K\} \mid \bar{x}^\alpha_{i,k} = 0\right\}$ is the set of all constraints in the series (5.4) that must hold in the subproblem, due to the incumbent master solution $\mathcal{O}^\alpha$ at iteration α in case $\bar{x}^\alpha_{i,k} = 0$ in $\mathcal{O}^\alpha$.

We can interpret Subproblem (5.8) as a problem to determine whether there exists a feasible timetable for a particular amount of discrete planning deviation; the particular planning deviation is defined by the solution of $\mathcal{M}^\alpha$.

5.5.2 *A Combinatorial Benders Cut for Railway Scheduling*

In [8] the authors introduce the combinatorial Benders cuts for a Benders decomposition that is structurally identical to our decomposition in Section 5.5.1. That is, the solution of the master problem $\mathcal{O}^{\alpha}_{\mathcal{M}}$ is a solution over binaries, which imply (in our case by $\bar{A}^{\alpha}_{\delta}$) a set of constraints onto the subproblem. We thus make use of the combinatorial Benders cut as introduced in [8] for our Benders decomposition of Section 5.5.1.

The combinatorial Benders cut is generated from an infeasible subset of constraints in the subproblem; we denote such subset further by $\mathcal{I}^{\alpha}$. Different from [8] our subproblem, i.e., Problem (5.8), is a problem of mixed-integer linear programming. We therefore cannot use the techniques of [8] to determine $\mathcal{I}^{\alpha}$ as they rely on strong duality in linear programming. Instead, we propose in Section 5.6.1 to use an existing algorithm, that we designed in a previous work for an geographic logic-based Benders decomposition [26].

Given an infeasible subset of constraints $\mathcal{I}^{\alpha}$, in such subset only those constraints are of importance to the combinatorial Benders cut, which are imposed by the incumbent master solution; in our case the constraints in $\bar{A}^{\alpha}_{\delta}$. We denote these constrains by $\mathcal{I}^{\alpha}_{\delta} := \mathcal{I}^{\alpha} \cap \bar{A}^{\alpha}_{\delta}$. For each constraint $(i,k) \in \mathcal{I}^{\alpha}_{\delta}$, there exists an associated binary variable $x_{i,k} \in M$ and we can write the combinatorial Benders cut β of [8] for our decomposition of Section 5.5.1 as

$$\beta^{\alpha} := \sum_{(i,k) \in \mathcal{I}^{\alpha}_{\delta}} x_{i,k} \geq 1. \tag{5.9}$$

The Benders cut (5.9) is clearly a constraint of set covering and thus $\mathcal{M}^{\alpha}$ a problem of set covering.

5.6 IMPLEMENTATION

Algorithm 6 shows the iterative scheme of Benders decomposition. In every iteration of the scheme, the master problem is solved (Line 2) and if necessary extended by an additional Benders cut (Line 5), which is generated from the analysis of the subproblem (Line 3), see Section 5.6.1. With the decomposition of our timetabling problem and the combinatorial Benders cut from Section 5.5, we discuss in this Section first an algorithm from one of our previous works [26], that can be used to analyze the subproblem; and later propose several approaches in Section 5.6.2 and Section 5.6.3 to address the master problem in our decomposition.

Algorithm 6: Benders Decomposition Scheme

input : $\mathcal{M}$, $\mathcal{S}$
output: $\mathcal{O}_\mathcal{C}$
init : $\alpha = 0$, $\mathcal{O}_\mathcal{C} = \emptyset$, $\mathcal{M}^\alpha = \mathcal{M}(\mathcal{B}^\alpha = \emptyset)$

while $\mathcal{O}_\mathcal{C} = \emptyset$ **do**
  $\mathcal{O}^\alpha_\mathcal{M} \leftarrow$ `Solve(`$\mathcal{M}^\alpha$`);`
  $\mathcal{O}^\alpha_\mathcal{S}, \mathcal{I}^\alpha \leftarrow$ `Analyze(`$\mathcal{S}^\alpha$`);`
  if $\mathcal{I}^\alpha_k \neq \emptyset$ **then**
    $\mathcal{B}^{\alpha+1} \leftarrow \mathcal{B}^\alpha \cup \beta^\alpha(\mathcal{I}^\alpha)$;
  else
    $\mathcal{O}_\mathcal{C} \leftarrow \mathcal{O}^\alpha_\mathcal{M} \cup \mathcal{O}^\alpha_\mathcal{S}$
  $\alpha \leftarrow \alpha + 1$;
return $\mathcal{O}_\mathcal{C}$

5.6.1 *An Infeasible Subset of Constraints in the Subproblem*

In this paper, we identify an infeasible subset of constraints $\mathcal{I}^\alpha$ in the subproblem $\mathcal{S}^\alpha$ of our decomposition, using the algorithm SMT (Algorithm 7) of [26]. Algorithm 7 computes an infeasibility proof for a problem that is identical by the type of constraints to our Subproblem (5.8). The infeasibility proof returned by Algorithm 7 is a set of cycles on a graphical representation G^α of $\mathcal{S}^\alpha$. The set arcs of all cycles in $\mathcal{I}^\alpha$ is an infeasible subset of constraints $\mathcal{I}^\alpha$ from which we can extract $\mathcal{I}^\alpha_\delta$ and use this latter to build the combinatorial Benders cut (5.9). In case Algorithm 7 cannot find a proof of infeasibility, $\mathcal{S}^\alpha$ is feasible and a corresponding solution $\mathcal{O}^\alpha_\mathcal{S}$ is returned. In the following we provide a brief summary of Algorithm SMT (Algorithm 7) of [26]:

Algorithm 7 is a combination of Boolean Satisfiability Solving (SAT) [14] with the logic of difference constraints from [11]; difference constraints are mathematically identical to the precedence relations of timetabling. The algorithm searches iteratively for a set of selectable precedence relations $\theta \subseteq A_s$, which satisfies all disjunctions $l \in L$ of $\mathcal{S}^\alpha$, i.e., $\forall l \in L$, $\exists W_c \in D_l$ s.t. $W_c \subseteq \theta$, and for which there exists an assignment for the variables t_i of $\mathcal{S}^\alpha$ satisfying the precedence relations $A_f \cup \bar{A}^\alpha_\delta \cup \theta$. If such θ can be found, feasibility of $\mathcal{S}^\alpha$ is proven. In [26], an assignment for t_i is proven to exist for the constraints $A_f \cup \bar{A}^\alpha_\delta \cup \theta$ if a graph $G^\alpha(\theta)$, where each of these constraints is a directed arc, is free of any positive length cycle.

Algorithm 7: SMT, A DPLL Algorithm with Precedence Constraints.

input : $\mathcal{S}^\alpha$
output: $\mathcal{O}^\alpha_\mathcal{S}$, $\mathcal{I}^\alpha$
init : $\Phi \leftarrow \mathcal{S}^\alpha$, $G^\alpha \leftarrow \mathcal{S}^\alpha$, $\theta = \emptyset$

```
1  while true do
2      confl ← UnitPropagation(Φ, θ)
3      if !confl then
4          confl ← Evaluate(G^α(θ))
5      if !confl then
6          if θ = Complete then
7              O^α ← G^α(θ)
8              return (O^α, ∅)
9          θ ← θ ⋃ Decide()
10     else
11         if confl = Unsatisfiable then
12             I^α ← AnalyzeIP(confl)
13             return (∅, I^α)
14         else
15             Analyze(confl)
16             Backtrack(confl)
```

In Algorithm 7 an initially empty set θ is iteratively extended by choice sets W_c from the decisions of $\mathcal{S}^\alpha$ until θ satisfies all disjunctions (decisions) of $\mathcal{S}^\alpha$. The set of constraints Φ in Algorithm 7 models the disjunctions of $\mathcal{S}^\alpha$ in terms of Boolean satisfiability constraints to assure that Algorithm 7 indeed computes a set θ, which satisfies all disjunctions of $\mathcal{S}^\alpha$. In Line 9, `Decide` selects choice sets W_c by heuristics of SAT, to extend θ. After every extension of θ, `UnitPropagation` in Line 2 propagates implications given by the constraints Φ and the newly selected choice set W_c. If no Boolean satisfiability constraint is violated by any implication (i.e., `!confl`), Line 4 evaluates if a feasible assignment for variables t_i exists with respect to the constraints $A_f \cup \bar{A}^\alpha_\delta \cup \theta$; for evaluation, the graphical representation $G^\alpha(\theta)$ is checked for positive length cycles. If no feasible assignment exists, Line 4 returns a new, violated Boolean satisfiability constraint computed from $G^\alpha(\theta)$. If Line 2 or Line 4 returns a violated constraint, either such constraint can be satisfied by a different θ or is generally unsatisfiable (`confl` = *Unsatisfiable*). If satisfiable by a different θ, Line 15 and 16 are

to determine the necessary changes in θ by analyzing first the violated constraint and then removing choices sets W_c leading to the violation of the constraint, from θ in a backtracking procedure. If the violated constraint cannot be satisfied by any other θ, Line 12 computes an infeasibility proof for $\mathcal{S}^\alpha$ based on the violated constraint. Finally in case neither Line 2 or 4 returns a violated constraint and θ satisfies all disjunctions of $\mathcal{S}^\alpha$, Line 7 computes a feasible assignment for the variables t_i, which satisfies the constraints $A_f \cup \bar{A}^\alpha_\delta \cup \theta$; such assignment is a feasible solution for $\mathcal{S}^\alpha$.

5.6.2 *Classical Approaches for Set Covering in the Master Problem*

In standard Benders decomposition [19], the master problem is solved to optimality in every iteration. We propose in this paper, among others, a decomposition approach, where the master is solved to optimality in every iteration using the commercial solver Gurobi [20]. In this case, i.e., if the master problem is solved to optimality, Algorithm 6 returns an optimal solution for the centralized problem, once no further Benders cut is generated (Line 7).

Different from standard Benders decomposition, in more recent literature, the master problem in a Benders decompositions is no longer solved to optimality. The validity of the Benders decomposition scheme, in particular the validity of Benders cut does not depend on the optimality or feasibility of a master solution, such that heuristic master solutions [33] or solutions of a relaxed master problem [29] are used in decompositions of the literature. In the same manner, we can address the master problem (Problem 5.7) in our decomposition by a heuristic approach. In the particular case of this paper, the master problem is a problem of set covering and we propose a decomposition approach, where we use the standard heuristic of Chvatal [7] for problems of set covering, to solve our master problem. In this approach, only a near optimal, but feasible solution of the master is computed in Line 2 of Algorithm 6. The usage of a heuristic is intended to reduce the computational time of a single iteration in the Benders scheme and reduce the total time till convergence of Algorithm 6.

In experiments of Section 5.7 we provide experimental results on both, the approach where the master problem is solved to optimality in each iteration by the commercial solver Gurobi, and the approach where the master problem is solved by the heuristic of Chvatal.

5.6.3 *Incremental Approaches for Set Covering in the Master Problem*

In alternative to classical approaches, where the master problem is solved from scratch in every iteration of the Benders scheme, without considering any information from previous master solutions, we propose in this section incremental heuristic approaches to solve the master problem. In this case, the incumbent solution of the master problem is based on the solution from the previous iteration. In particular, the solution of the previous iteration in the Benders scheme is adapted to be conform with the latest Benders cut. With such heuristics, we intend to minimize the computational effort inside a single iteration of the Benders scheme and reduce significantly the total time till convergence.

We adapt a master solution $\mathcal{O}^{\alpha-1}$ from the previous iteration $\alpha-1$ to the latest Benders cut β^{α} of the current iteration α by changing at least one binary variable $x_{i,k}$, $(i,k) \in \mathcal{I}^{\alpha}$ from value 0 to 1, from the previous solution $\mathcal{O}^{\alpha-1}$ to the incumbent solution $\mathcal{O}^{\alpha}$. The change of a binary variable $x_{i,k}$ from value 0 to 1 corresponds to an increase of planning deviation at event i, from $\bar{\delta}_{i,k}$ to $\bar{\delta}_{i,k+1}$. Variables $x_{i,k}$, $(i,k) \in \mathcal{I}^{\alpha}$ have altogether value 0 in $\mathcal{O}^{\alpha-1}$; otherwise related constraints would not appear in $\mathcal{I}^{\alpha}$ as they would not be imposed to $\mathcal{S}^{\alpha}$. We propose in the following four different heuristics to determine which binary(ies) $x_{i,k}$, $(i,k) \in \mathcal{I}^{\alpha}$ must change in values from $\mathcal{O}^{\alpha-1}$ to $\mathcal{O}^{\alpha}$:

Definition. *(Min Appearance)*
In the heuristic of minimal appearance, we select out of the binaries $x_{i,k}$, $(i,k) \in \mathcal{I}^{\alpha}$, the single binary variable, which appears the least number of times in the already computed constraints of set covering (Benders cuts) in $\mathcal{B}^{\alpha}$. Such heuristic is intended to minimize the number of redundant relaxed set covering constraints in $\mathcal{B}^{\alpha}$.

Definition. *(Max Appearance)*
In the heuristic of maximal appearance, we select out of the binaries $x_{i,k}$, $(i,0,k) \in \mathcal{I}^{\alpha}$, the single binary variable, which appears the most number of times in the already computed constraints of set covering (Benders cuts) in $\mathcal{B}^{\alpha}$. Opposed to Min Appearance, such heuristic is intended to maximize the number of constraints in $\mathcal{B}^{\alpha}$ relaxed by a single binary variable.

Definition. *(Complete Satisfaction)*
In the heuristic of complete satisfaction, all binaries $x_{i,k}$, $(i,0,k) \in \mathcal{I}^{\alpha}$ are set equal to 1. Such heuristic is to relax (remove) the most constraints from the subproblem in a single iteration of the Benders scheme to reduce the amount of total iterations in the Benders scheme until a feasible subproblem solution is found and convergence achieved.

Definition. *(Random)*
In the heuristic of random selection, a single binary is uniformly random selected from the binaries $x_{i,k}$, $(i, 0, k) \in \mathcal{I}^{\alpha}$ *and set equal to* 1. *Such heuristic shall provide a benchmark for the numerical experiments in Section 5.7.*

In Section 5.7 we provide exhaustive experiment on all incremental heuristics of this section and put those in relation to classical approaches in Benders decomposition (Section 5.6.2) and further existing approaches for railway timetabling.

5.7 COMPUTATIONAL EXPERIMENTS

With a series of comprehensive experiments we analyze the performance of heuristic and optimal approaches proposed in this paper and compare results with standard benchmarks. All experiments reported in the following sections are computed on the Euler Cluster of ETH Zurich, on a single AMD EPYC 7763 processor with 120 GB of RAM and a time limit of 24 hours per experiment.

5.7.1 *Instances*

For our experiments we use 14 original scenarios of timetabling. These scenarios are geographic and temporal excerpts from current timetable of the Swiss Federal Railways in Switzerland and include all microscopic details of the actual infrastructure. All excerpts have a time horizon of planning between 2 and 6 hours. We name the scenarios of timetabling according to the biggest cities within the corresponding geographic excerpt; names of cities are decoded in Table 5.1.

We analyze the scalability of our approaches of set covering as we scale the 14 scenarios of timetabling to different instances of timetabling with a reduced number of trains. We consider the 14 scenarios of timetabling

ZUE	Zurich	HE	Herisau	CH	Chur	ZAS	Zurich Altstetten
BN	Bern	YV	Yverdon	LZ	Luzern	RH	Romanshorn
AA	Aarau	BEL	Bellinzona	BDF	Burgdorf	GD	Arth-Goldau
BS	Basel	SO	Solothurn	SG	St. Gallen	OTH	Othmarsingen

TABLE 5.1: Decoding of Scenario Names to the Cities of Switzerland.

as instances with 100% of trains and derive from each scenario 9 further instances with a reduced percentage of trains (i.e., 90%, 80%, ... of trains). We reduce the number of trains as we randomly but continuously remove trains such that, e.g., all trains of an 80% instance are within the 90% instance from the same original scenario. In total we have generated 140 instances of timetabling derived from the 14 different original scenarios of timetabling.

In Table 5.2 in the Appendix we provide detailed numbers on all instances of timetabling used in this work. We report the number of trains, stops performed by trains, transfers between trains, ordering and routing choices, as well as the maximal allowed planning deviation. Figure 5.1 gives an overview over the most important characteristics of the 14 original scenarios of timetabling. In the upper plot of Figure 5.1, we report the number of trains in each scenario; in the lower plot we report the number of choices in all discrete decisions of the scenario, broken down into routing and ordering choices. From Figure 5.1 we can see that especially the first three scenarios, which are three scenarios of the railway traffic between Zurich (the busiest railway station in Switzerland) and Chur, contain a large number of ordering and routing choices for a rather lower amount of trains. This high number of choices is the result of a complex railway infrastructure around the railway station of Zurich. A high number of choices is often a good indicator for a high computational complexity of an scenario. We can also see in Figure 5.1 that some scenarios contain a large amount of trains, but at the same time a rather average number of choices. This occurs often in cases of timetabling, for areas where the railway traffic in opposite direction is routed over separate tracks, and only little interactions are taking place between trains.

5.7.2 *Benchmark Approaches for Timetabling*

We put the approaches of this work in comparison with a standard (centralized) approach of railway timetabling to show the benefits and drawbacks of our heuristic approaches. Our benchmark is based on the methodologies of [18]. In [18], the authors propose several improvements in the formulation of railway scheduling to increase the computational performance of a commercial solver applied to a problem of railway rescheduling. We adopt from this work particularly the tightening of temporal bounds for events of the scheduling problem, based on the maximal allowed planning deviation. Based on these tightened bounds, according to [18], we can fix entire decisions or exclude particular choices from decisions in the timetabling problem. We omit for this work the heuristic fixation of a maximal planning

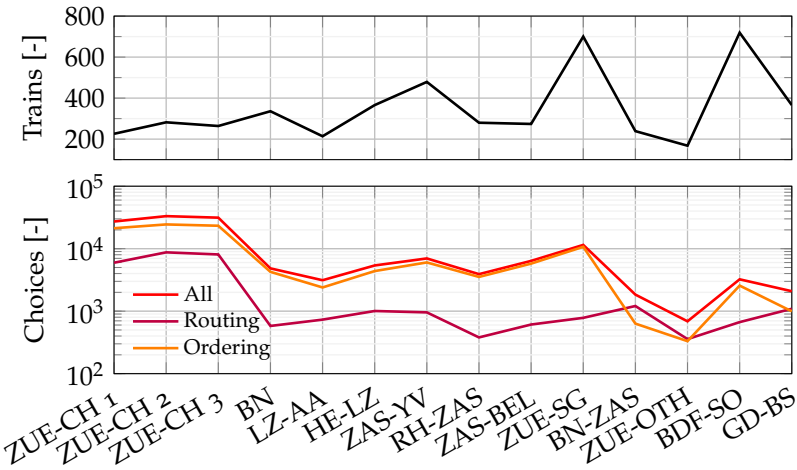

FIGURE 5.1: Trains and Choices in Scenarios of Timetabling.

deviation as done in [18]. In our case we are given a fixed maximal allowed planning deviation by the operators of the railway.

We apply our benchmark approach to the problem of railway timetabling with a continuous objective (Problem 5.3) as well as with a discrete objective (Problem 5.6). We use for the benchmarks the commercial mixed-integer solver Gurobi [20] and the Boolean satisfiability solver Z3 [15]. Z3, likewise to our Algorithm 7, is a SAT solver extended to various types of constraints (including precedence relations).

5.7.3 *Set Covering Heuristics*

In a first series of experiments we study the performance of our classical (Section 5.6.2) and incremental (Section 5.6.3) set covering approaches. We apply all approaches to all 140 instances of timetabling. All numbers reported on experiments in this section are an average over five computationally identical runs.

As stated in the beginning of this section, all experiments are given a time limit of 24 hours for computations. In consequence, the classical set covering approaches, i.e., the Chvatal heuristic and the Optimal approach of set covering did not provide solutions within the time limit for all instances. We report in Figure 5.2 the percentage of trains up to which the individual

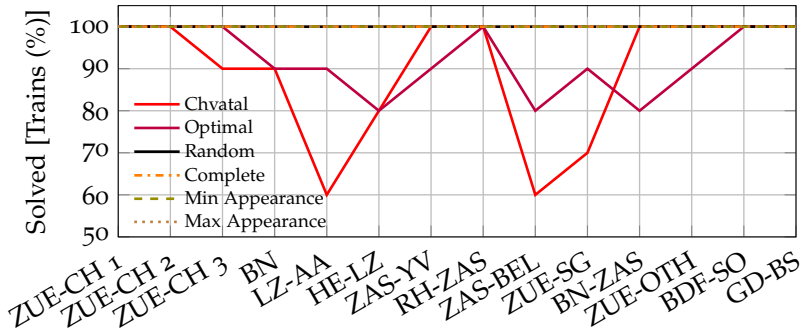

FIGURE 5.2: Instances Solved by Set Covering Approaches.

approaches were able to provide a solutions within the limit of 24 hours. In Figure 5.2 we see that the Chvatal heuristic and the Optimal approach fail to provide solutions mainly on the same instances, indicating the computational complexity of solving these instances to (near) optimality by classical Benders decomposition approaches. In comparison, the Chvatal heuristic mostly provides fever solutions than the Optimal approach, reaching a lower level of percentage of trains, to which a solution could be found. This lower performance might be caused due to higher fluctuations in heuristics (non optimal) incumbent master solutions, which themselves lead to more iterations in the Benders scheme till convergence. We can see the higher number of iterations for Chvatal in Figure 5.5 as we report the total number of constraints (equal to number of iterations) for different approaches. The incremental set covering heuristics provide solutions for all 140 instances within 24 hours. We provide a plausible comparison between all (classical and incremental) approaches of set covering as we further report results only on those instances, which were solved by all approaches based on set covering.

In Figure 5.3 we report the computational time of different set covering approaches. In the top plot of Figure 5.3 we report the absolute computational time, on average over all 14 scenarios of timetabling for different instances with different percentages of trains; the bottom plot reports the average computational time scaled by the computational time of the Random heuristic of set covering. We use the Random heuristic as a benchmark in set covering approaches. We report the computational time in both plots on a logarithmic axis. In Figure 5.3 the Min Appearance and the Chvatal heuristic show significantly higher computational times than the remaining set covering approaches. We will see later, when looking at the statistics of

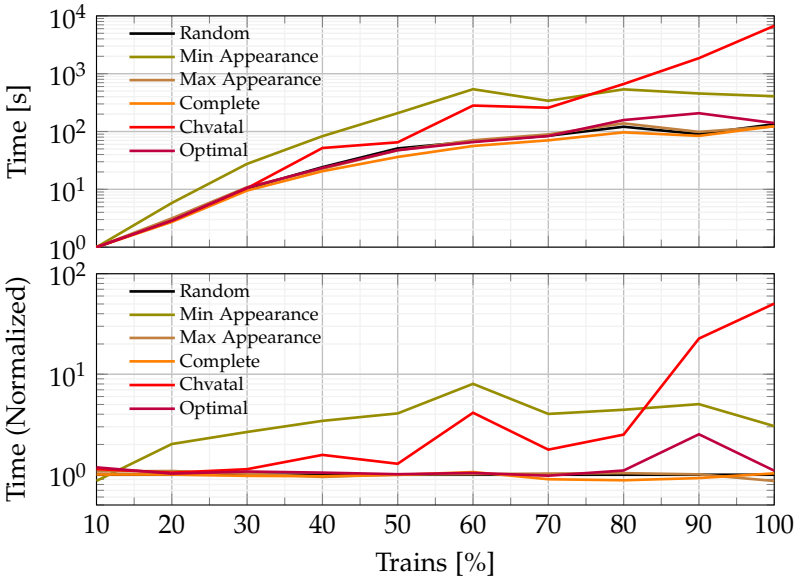

FIGURE 5.3: Average Computation Time of Set Covering over 14 Scenarios.

set covering constraints in the Benders decomposition scheme (see Figure 5.5), that in particular these two approaches show the largest amount of constraints generated, and thus the most iterations performed till convergence. Regarding scalability, we see a similar increase of computational time over an increasing percentage of trains for almost all set covering approaches. The exception is the Chvatal heuristic, which shows a linear increase on the logarithmic scale of Figure 5.3 and thus a strong exponential increase of computational time for higher percentages of trains. In Figure 5.3, the Optimal set covering approach scales similar to the other set covering heuristics, which is to some extent a falsified image of the reality. Figure 5.3 reports only instances that are solved by all set covering heuristics; as shown in Figure 5.2 this excludes in total 11 of the 140 generated instances of timetabling, that could not be solved by the Optimal set covering approach. Accounting for these would clearly reveal a worse scalability of the Optimal approach than actually reported in Figure 5.3 and worse than the incremental set covering approaches.

In set covering approaches we would expect the best performance from the Complete Satisfaction heuristic. Such heuristic relaxes the highest number of constraints in the subproblem in every iteration, such that we expect

a quick convergence to a feasible solution. Different from our expectations, Figure 5.3 does not report a particular superior performance of the Complete Satisfaction heuristic in comparison to other heuristics; only in Figure 5.5 we can see a minor decrease in iterations (constraints) for the Complete Satisfaction heuristic. We can conclude from such result that the majority of the infeasible subsets of constraints detected by Algorithm 7 must be non-overlapping or only marginally overlapping, in terms of containing the same constraints. Only if infeasible subsets of constraints would overlap significantly, removing all such constraints from the subproblem in a single iteration, as done by the Complete Satisfaction heuristic, would bring a computational benefit; in this case the Complete Satisfaction heuristic would avoid all overlapping infeasible subsets in the subproblem in a single iteration. Else the removal of a single constraint from the subproblem for each infeasible subset is sufficient to avoid the infeasible subset in the subproblem and is thus as effective as removing all of them.

In Figure 5.4 we report the objectives of timetables computed by different approaches of set covering. The objectives reported in the figure are the (continuous) planning deviation of a timetable, as in Problem 5.3; we thus report the (continuous) planning deviation also for timetables computed based on a discrete objective, to provide a plausible comparison. In the style of Figure 5.3, we report in Figure 5.4 in the top plot the absolute objective values (total minutes of planning deviation as reals); in the bottom plot we report the gap of the objective (planning deviation) of approaches with respect to (and normalized by) the objective of our benchmark heuristic, the Random heuristic. The gap of the objective is reported on average over all 14 scenarios of timetabling for different percentages of trains (instances) to see the scaling behaviour.

Most striking in Figure 5.4 is the performance of the Min Appearance heuristic. The objective of such heuristic is a factor higher (worse) than for all other approaches. The other approaches perform all rather similar to each other, with the Chvatal heuristic and the Optimal approach performing slightly better for high percentages of trains. Figure 5.4 indicates that avoiding redundant relaxations as done by the Min Appearance heuristic is likely to result in timetables of poor quality. The opposite thinking, i.e., if we exploit the overlap of infeasible sets of constraints and focus on removing constraints from the subproblem, which appear in multiple such sets, seems to have rather little effect on the quality of timetables. The Max Appearance heuristic, which exploits exactly such overlap shows a rather similar quality of timetables as the Random and also the Complete heuristic. These results

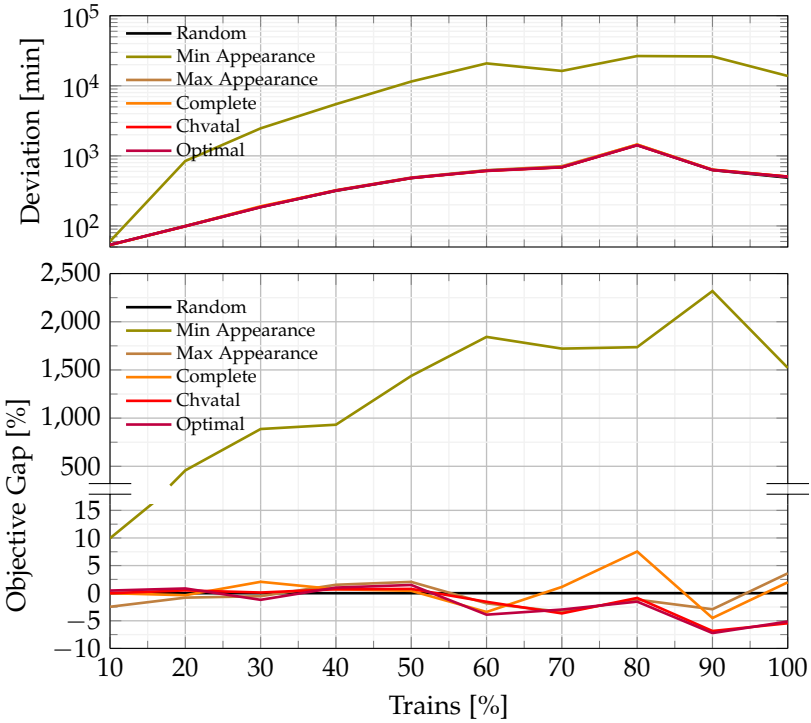

FIGURE 5.4: Average Objective of Set Covering over 14 Scenarios.

match with the earlier conclusion, that infeasible subsets of constraints overlap only minor; in such case, the Max Appearance heuristic performs algorithmically almost identical as the Random heuristic. In summary, apart from the Min Appearance, the performance of incremental set covering approaches shows rather independent from the particular heuristic used. This indicates that the performance of incremental set covering approaches is more dominated by the infeasible subsets of constraints discovered, than by the particular incremental heuristic, used to adapt the master solution.

We further see in Figure 5.4 a rather expected increase of planning deviation for an increasing amount of trains. The decrease of total planning deviation after 80% of trains for most of the approaches is caused by neglecting more and more unsolved instances for 90% and 100% in the figure; neglected instances show on average a high planning deviation. The bottom plot of Figure 5.4 shows that the Chvatal heuristic performs very similar to

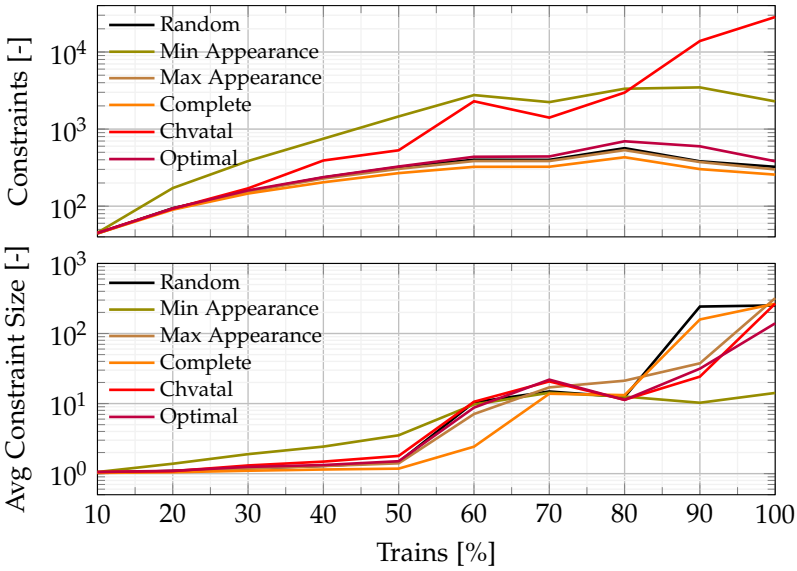

FIGURE 5.5: Average Problem Size in Set Covering over 14 Scenarios.

the Optimal approach with respect to planning deviation, providing thus timetables of almost optimal quality. The incremental heuristics all perform rater similar to each other, with the Complete heuristic showing some more fluctuations in quality. With respect to classical approaches, the incremental heuristics perform with a gap of around $\sim 7\%$, minimum $\sim 5\%$. Overall Figure 5.4 empirically underlines, that we are able to provide solutions of good quality, i.e., gaps to an optimal solution of $\sim 7\%$, in short time, with heuristic approaches of set covering.

Finally, we report in Figure 5.5 statistics on the set covering problem in solves of set covering approaches. In the top plot of Figure 5.5, we report the number of Benders cuts, i.e., set covering constraints, generated in the Benders scheme till convergence; in the bottom plot we report the average size of set covering constraints, i.e., number of binary variables in the sum of constraint 5.9. The number of constraints generated is equal to the number of iterations done by the Benders scheme as a single constraint is generated in each iteration. Figure 5.5 provides an empirical explanation to the computational performance of the set covering approaches as reported in Figures 5.2, 5.3 and 5.4. In the top plot of Figure 5.5, we see a tendency of the Chvatal heuristic to generate an over-proportional amount of constraints

(iterations), which explains the failure of such heuristic to solve all 140 instances of timetabling. For the Optimal approach, we see an average performance in terms of constraints and constraint size. We thus argue the reason for unsolved instances in Figure 5.2 to be the computational burden of solving the master to optimality in every iteration of the Benders scheme. We saw in Figure 5.2 that the Min Appearance heuristic, different from the Chvatal heuristic was able to solve all 140 instances. We explain such result by Figure 5.5, as we see a less drastic increase of constraints (iterations) for higher percentages of trains for the Min Appearance heuristic, than for the Chvatal heuristic.

Regarding the constraint size, the Min Appearance produces for low percentages slightly bigger constraints, but then for high percentages of trains (above 80%) significantly smaller constraints, in comparison to all other approaches of set covering. In combination with the total number of constraints generated, the Min Appearance heuristic seems to produce a large number of constraints with small size, in terms of binaries in the sum. In theory, as discussed by [8], a combinatorial Benders cut of small size is mathematically stronger and thus likely to be more useful for the computation of a solution. In the application of our heuristics, such effect seems to be rather negligible. We can see in Figure 5.5 that for heuristic approaches, not the strength of constraints seems to be the dominant factor for performance, but rather the number of constraints (and iterations) necessary to convergence; iterations seem to be less for constraints of bigger size. Furthermore, we also conclude from Figure 5.4 and Figure 5.5 that a larger number of constraints leads to more planning deviation in a timetable (see Min Appearance). We explain such behavior, as in the case where more iterations are necessary to converge, more constraints are removed from the subproblem and thus more planning deviation is possible in a solution of the subproblem.

In Figure 5.5 we see approaches generating constraints of size 1. Such constraints can occur in case a train cannot be scheduled without a planning deviation exclusively due to constraints, which must hold in any case (fixed precedence relations), i.e., minimal travel time and transfer constraints. In this case, the infeasible subset of constraints is a set of only one constraint (the constraint that must be relaxed to allow for a planning deviation of the train) and consequentially set covering constraint has size 1. In case a discrete decision (in particular an ordering decision) with its selectable precedence relations, is part of an infeasible subset of constraints, the related set covering constraints has size 2 or bigger. In this case the infeasible

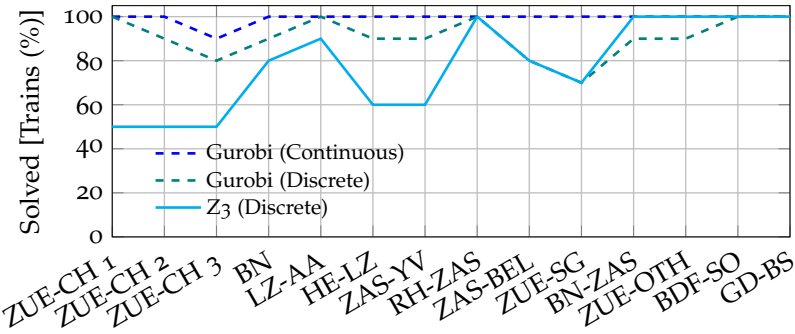

FIGURE 5.6: Instances Solved to Optimality (Gurobi) or Feasibility (Z3) by Benchmark Approaches.

subset will contain constraints of planning deviation (5.4) related to more than one train, in particular all trains involved in the decision.

5.7.4 *Comparison of Benchmarks and Set Covering*

In a second series of experiments we compare the results of our set covering approaches with the benchmark approaches of Section 5.7.2.

Likewise to approaches in set covering, due to the limitation of computational time to 24 hours, benchmark approaches could not provide results for all instances within the given time. In the style of Figure 5.2 for set covering, we report in Figure 5.6 the highest percentages of trains up to which individual benchmark approaches provide solutions within the time limit. For benchmarks computed by Gurobi, we report the number of optimal solutions as in all cases at least a feasible solution was provided. We report the number of optimal solutions by a dashed line. The Z3 solver was in general unable to provide any solution for the problem of timetabling with a continuous objective; we do not report such benchmark in Figure 5.6. Also for the timetabling with discrete objective Z3 provides only few solutions, disclosing the struggle of Z3 with the problem of timetabling of this work. The low performance of Z3 can be explained by the design and tuning of Z3 mainly for problems of feasibility and not optimality as it is the case for our problem of timetabling.

We can see in Figure 5.6 that the commercial solver Gurobi, which is designed for optimization over continuous variables, is able to solve more instances to optimality with the continuous objective than with the discrete

objective. Problem (5.6) with a discrete objective contains a higher number of big-M constraints. These often decrease the performance of mixed-integer solvers such as Gurobi, as big-M constraints have a poor linear relaxation and such relaxations are heavily used in mixed-integer optimization.

In the style of Figure 5.3, we report in Figure 5.7 the computational time of benchmark approaches together with the Random heuristic (as the set covering benchmark) and the Optimal approach of set covering (as the only optimal approach based on set covering). The top plot in Figure 5.7 reports the absolute computation time; the bottom plot reports the normalized computational time with respect to the Random heuristic of set covering. With dashed lines in Figure 5.7 we report numbers that are computed as averages over less than the 14 scenarios of timetabling: some instances could not be solved by the respective approach for the percentage of trains reported on the x-axis. Solid lines report an average over all 14 scenarios of timetabling. We can see in Figure 5.7 that Gurobi shows a better performance for the continuous objective than for the discrete objective, as discussed earlier, due to a higher numbers of big-M constraints in the discrete case. Z3 shows a significant slower performance than Gurobi for the discrete approach, as discussed, likely due to the design of Z3 for problems of feasibility, and thus showing difficulties in optimization.

Comparing benchmarks to set covering approaches, we see a major advantage of incremental set covering heuristics regarding computational speed. At 100% of trains the Random heuristic shows a computational speedup of roughly a factor $\sim$ 390 compared to the fastest benchmark, i.e., Gurobi (Continuous). Such high computational speedup is mainly due to the three instances ZUE-CH 1, ZUE-CH 2 and ZUE-CH 3, especially ZUE-CH 3, where no optimal solution could be found by Gurobi. These three instances propose a significant challenge for Gurobi due to their high amount of routing alternatives, while the instances are solved rather efficiently by the set covering heuristics. When considering the median of the computational performance to account for outliers, the computational speedup between Gurobi and the Random heuristic reduces to a factor $\sim$ 20.

We can further see in Figure 5.7 that the Optimal set covering approach, while initially performing as good as the Random heuristic, approaches for percentages of trains above 80%, a performance similar to Gurobi, such that we can empirically see no major advantage in solving a problem of timetabling to optimality by our decomposition (set covering) approach, instead of directly applying a commercial solver. The bottom plot of Figure 5.7 indicates a benefit of incremental heuristics over benchmarks regarding

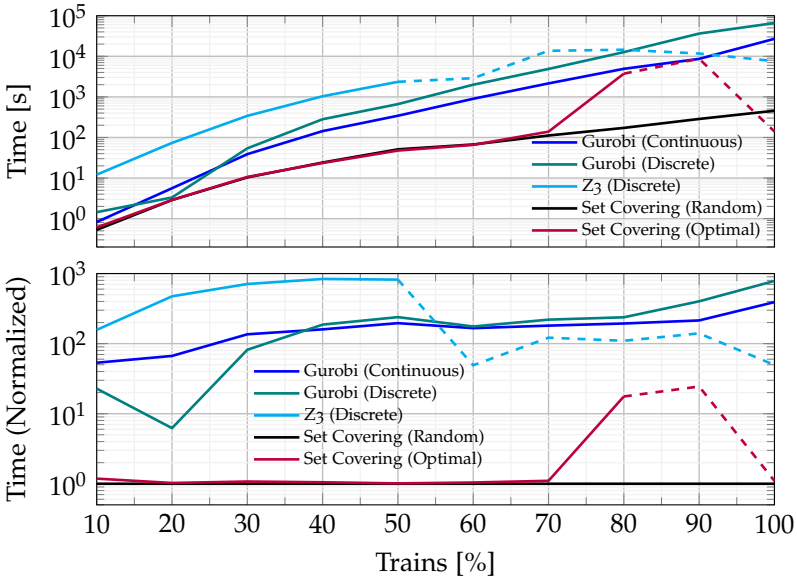

FIGURE 5.7: Average Computation Time of Benchmarks over 14 Scenarios.

scalability. The normalized time of benchmarks (normalized by the Random heuristic) notably increases for high percentages of trains (above 70%), empirically underlining the advantage of incremental set covering heuristics regarding the computational scalability.

While in speed, incremental set covering heuristics proved superior, it remains to analyze the quality (planning deviation) of timetables provided by benchmarks, to empirically estimate the sacrifice in quality made by set covering heuristics in favor of computational speed. With Figure 5.8 we report in the style of Figure 5.4 the performance of benchmarks in terms of planning deviation in the computed timetables. We provide a comparison to set covering approaches as we report the Random heuristic and the Optimal set covering approach in the same plots. In the top plot of Figure 5.8, we report the absolute planning deviation of the timetable solutions of different approaches. In the bottom plot, we report the gap in the objective (planning deviation) of benchmarks with respect to (and normalized by) the Random set covering heuristic. The gap is reported on average over all 14 scenarios of timetabling for different percentages of trains. As for Figure 5.7, dashed lines indicate a reporting over an average over less than 14 scenarios.

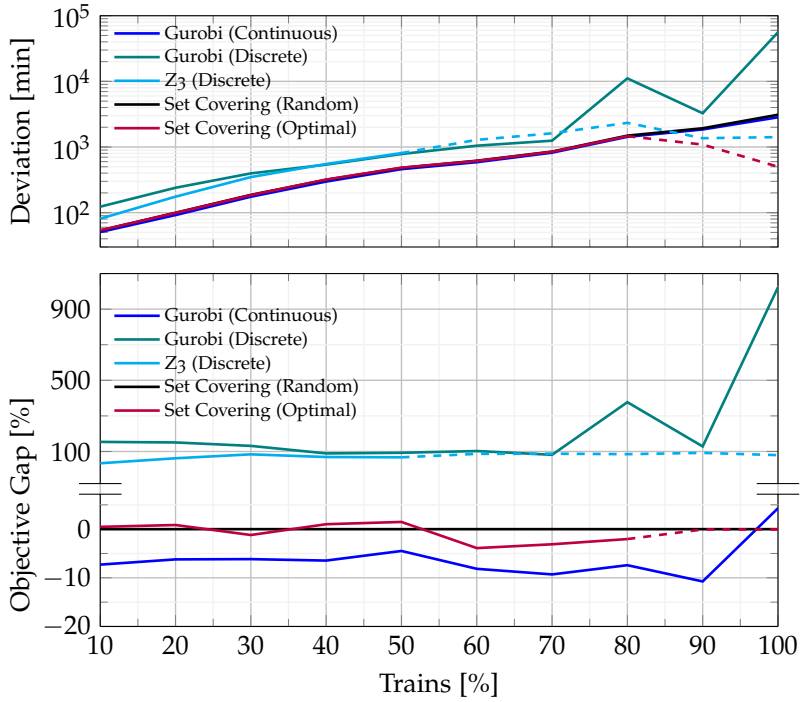

FIGURE 5.8: Average Objective of Benchmarks over 14 Scenarios.

In Figure 5.8, we see that on average, in case of a discrete objective, the benchmark approaches Gurobi (blue) and Z3 (cyan) perform significantly worse in terms of total planning deviation than set covering approaches. We explain the superior performance of set covering heuristics by Algorithm 7. In Algorithm 7, in case the master solution allows for a feasible solution in the subproblem, such solution is defined by a choice for each discrete decision and a time for each event. Given the choices, the times for the events are computed through a longest path propagation over a directed graph, whose arcs represent fixed and selected (chosen) precedence relations in the subproblem. Event times found in this manner minimize the total sum of all event times. This is the key difference between approaches of set covering and benchmark approaches with a discrete objective. In case of benchmarks, event times are arbitrarily chosen within their feasible range (according to the choices of timetabling) and not minimized in their total

sum; only discrete planning deviation is minimized. Therefore, benchmarks show more (continuous) planning deviation than set covering approaches.

In general, we see in Figure 5.8 that Gurobi with a continuous objective in timetabling performing best; this approach provides an optimal solution to the continuous formulation of railway timetabling given in this paper. In the bottom plot of Figure 5.8, we see that Gurobi (Continuous) computes on average timetable with $\sim$ 7.5% less planning deviation than the Random heuristic of set covering. Also, we can see that due to the discretization, the Optimal set covering approach is unable to achieve a performance as good as Gurobi with a continuous objective. Finally, we see that for 100% of trains Gurobi (Continuous) computes timetables with an average of $\sim$ 4% more planning deviation than the Random heuristic. The on average higher planning deviation is caused by few instances where Gurobi was unable to provide an optimal solution within the time limit (see Figure 5.6). If such instances are ignored, Gurobi (Continuous) shows an average performance of $\sim$ 10% less planning deviation than the Random set covering heuristic.

5.8 CONCLUSION

In this paper we introduce multiple approaches for solving the problem of microscopic railway timetabling, based on a novel reformulation of the problem for which the solution process becomes a set covering problem. With a discretization of the objective in timetabling we can decompose the timetabling problem using a Benders decomposition, into a problem of set covering as the master problem and a problem of timetabling, where only feasibility must be evaluated, as the subproblem. In the set covering problem, the sets to be selected correspond to planning deviations of trains, such that the selection of a set corresponds to an additional amount of planning deviation for a train. The Benders decompositions separates the question of optimality (i.e., deviation) in the master from the question of feasibility in the subproblem (i.e., routing, ordering and timing). The Benders decomposition we propose is designed to exploit the combinatorial Benders cuts introduced in [8]. In this paper, we propose multiple approaches to address the set covering problem that is the master problem in the proposed decomposition. Along with standard approaches of solving the master problem to optimality in every iteration of the Benders scheme, we propose 4 different incremental heuristics to address the master problem. We propose incremental heuristics, which exploit the incremental growth of the master problem

in the Benders scheme and maintain a master solution throughout the iterations of the scheme by incremental adaptations of an existing solution.

In two exhaustive series of experiments we analyze the performance of all of our set covering approaches and compare such results with multiple benchmarks. In benchmarks, we address the problem of timetabling by commercial solvers, in particular Gurobi and Z3. Both solvers are applied to the microscopic timetabling problem with both continuous and the discrete objective. Experiments show that with set covering heuristics we can solve instances of timetabling up to a factor of ~ 20 faster, while maintaining a quality of timetables with on average 7.5% more planning deviation compared to an optimal solution computed by Gurobi; heuristics never exceeding more than 10% of additional planning deviation compared to an optimal solution.

Further research in the approach of set covering for railway timetabling should clearly include the design and analysis of further incremental heuristics. Apart from the Min Appearance heuristic, all other heuristics show a very similar performance, including the Random heuristics. This indicates that heuristics proposed in this paper do not yet fully exploit the potential of our set covering approach, as no heuristic shows a clear superiority over the Random heuristic and further research is advisable. Also a mix of heuristics and an optimal approach in set covering should be investigated. In the current setup, incremental heuristics stop as soon as a feasible solution is found. A different branch of research should investigate possible improvements in quality (planning deviation) of timetables, when given additional time for further computations beyond a first feasible solution.

ACKNOWLEDGMENTS

The authors thank the colleagues from SBB for the useful discussions. This project was supported by the ETH Zürich Foundation.

REFERENCES

1. Balas, E. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics* **89**, 3 (1998).
2. Benders, J. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* **4**, 238 (1962).
3. Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M. & Schlechte, T. Recent success stories on integrated optimization of

railway systems. *Transportation Research Part C Emerging Technologies* **74**, 196 (2017).

4. Cacchiani, V., Caprara, A. & Fischetti, M. A lagrangian heuristic for robustness, with an application to train timetabling. *Transportation Science* **46**, 124 (2012).
5. Caimi, G., Fuchsberger, M., Laumanns, M. & Lüthi, M. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. *Computers & Operations Research* **39**, 2578 (2012).
6. Caprara, A., Monaci, M., Toth, P. & Guida, P. L. A Lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics* **154**, 738 (2006).
7. Chvatal, V. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research* **4**, 233 (1979).
8. Codato, G. & Fischetti, M. Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. *Operations Research* **54**, 756 (2006).
9. Corman, F., D'Ariano, A., Pacciarelli, D. & Pranzo, M. Dispatching and coordination in multi-area railway traffic management. *Computers & Operations Research* **44**, 146 (2014).
10. Corman, F., D'Ariano, A., Pacciarelli, D. & Pranzo, M. A tabu search algorithm for rerouting trains during rail operations. *Transportation Research Part B Methodological* **44**, 175 (2010).
11. Cotton, S. & Maler, O. *Fast and flexible difference constraint propagation for DPLL (T)* in *International Conference on Theory and Applications of Satisfiability Testing* (2006), 170.
12. D'Ariano, A., Corman, F., Pacciarelli, D. & Pranzo, M. Reordering and local rerouting strategies to manage train traffic in real time. *Transportation Science* **42**, 405 (2008).
13. D'Ariano, A., Pacciarelli, D. & Pranzo, M. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operations Research* **183**, 643 (2007).
14. Davis, M., Logemann, G. & Loveland, D. A Machine Program for Theorem-Proving. *Communications of the ACM* **5**, 394 (1962).
15. De Moura, L. & Bjørner, N. *Z3: An Efficient SMT Solver* in *Tools and Algorithms for the Construction and Analysis of Systems* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008), 337.

16. Dollevoet, T., Huisman, D., Kroon, L. G., Veelenturf, L. P. & Wagenaar, J. C. Application of an iterative framework for real-time railway rescheduling. *Computers & Operations Research* **78**, 203 (2017).

17. Fan, B., Roberts, C. & Weston, P. A comparison of algorithms for minimising delay costs in disturbed railway traffic scenarios. *Journal of Rail Transport Planning & Management* **2**, 23 (2012).

18. Fischetti, M. & Monaci, M. Using a general-purpose Mixed-Integer Linear Programming solver for the practical solution of real-time train rescheduling. *European Journal of Operations Research* **263**, 258 (2017).

19. Geoffrion, A. Generalized Benders Decomposition. *Journal of Optimization Theory and Applications* **10**, 237 (1972).

20. Gurobi Optimization, L. *Gurobi Optimizer Reference Manual* 2021.

21. Herrigel, S., Laumanns, M., Nash, A. & Weidmann, U. Hierarchical Decomposition Methods for Periodic Railway Timetabling Problems. *Transportation Research Record* **2374**, 73 (2013).

22. Keita, K., Pellegrini, P. & Rodriguez, J. A three-step Benders decomposition for the real-time Railway Traffic Management Problem. *Journal of Rail Transport Planning & Management* **13**, 100170 (2020).

23. Lamorgese, L. & Mannino, C. A non-compact formulation for job-shop scheduling problems in traffic management. *Operations Research* **67**, 1503 (2019).

24. Lamorgese, L., Mannino, C. & Natvig, E. An exact micro–macro approach to cyclic and non-cyclic train timetabling. *Omega (United Kingdom)* **72**, 59 (2017).

25. Lamorgese, L., Mannino, C. & Piacentini, M. Optimal Train Dispatching by Benders'-Like Reformulation. *Transportation Science* **50**, 910 (2016).

26. Leutwiler, F. & Corman, F. A logic-based Benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research* **303**, 525 (2022).

27. Liu, L. & Dessouky, M. A decomposition based hybrid heuristic algorithm for the joint passenger and freight train scheduling problem. *Computers & Operations Research* **87**, 165 (2017).

28. Luan, X., Schutter, B. D., van den Boom, T., Corman, F. & Lodewijks, G. Distributed optimization for real-time railway traffic management. *IFAC-PapersOnLine* **51**, 106 (2018).

29. Maher, S. J. Implementing the branch-and-cut approach for a general purpose Benders' decomposition framework. *European Journal of Operations Research* **290**, 479 (2021).

30. Mascis, A. & Pacciarelli, D. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operations Research* **143**, 498 (2002).

31. Odijk, M. A. A constraint generation algorithm for the construction of periodic railway timetables. *Transportation Research Part B Methodological* **30**, 455 (1996).

32. Pellegrini, P., Marlière, G., Pesenti, R. & Rodriguez, J. RECIFE-MILP: An Effective MILP-Based Heuristic for the Real-Time Railway Traffic Management Problem. *IEEE Transactions on Intelligent Transportation Systems* **16**, 2609 (2015).

33. Poojari, C. A. & Beasley, J. E. Improving benders decomposition using a genetic algorithm. *European Journal of Operations Research* **199**, 89 (2009).

34. Pranzo, M., Meloni, C. & Pacciarelli, D. *A new class of greedy heuristics for job shop scheduling problems* in *International Workshop on Experimental and Efficient Algorithms* (2003), 223.

35. Sama, M., D'Ariano, A., Corman, F. & Pacciarelli, D. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers & Operations Research* **78**, 480 (2017).

36. Samà, M., D'Ariano, A., Corman, F. & Pacciarelli, D. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers & Operations Research* **78**, 480 (2017).

37. Veelenturf, L. P., Kidd, M. P., Cacchiani, V., Kroon, L. G. & Toth, P. A railway timetable rescheduling approach for handling large-scale disruptions. *Transportation Science* **50**, 841 (2016).

38. Wolsey, L. A. & Vanderbeck, F. in *50 Years Integer Program. 1958-2008* 431 (Springer, 2010).

APPENDIX

5.9 INSTANCE CHARACTERISTICS

Scenario	Max Deviation	Instance:	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
ZUE-CH 1	30 min	Trains	22	45	67	90	113	135	158	180	203	226
		Stops	155	309	449	606	781	923	1074	1213	1349	1507
		Transfers	2	10	22	30	40	57	82	116	140	170
		Ordering	98	846	1927	3515	5389	7774	10395	13439	17142	21261
		Routing	433	1183	1893	2539	3107	3739	4297	4925	5420	5962
ZUE-CH 2	30 min	Trains	28	56	84	112	141	169	197	225	253	282
		Stops	205	365	549	718	894	1093	1287	1469	1609	1777
		Transfers	2	8	18	30	44	62	102	133	164	196
		Ordering	267	947	2297	4012	5983	9036	12006	15161	18974	24447
		Routing	965	1978	2987	3832	4622	5410	6499	7062	7909	8739
ZUE-CH 3	30 min	Trains	26	52	79	105	132	158	184	211	237	264
		Stops	146	308	524	693	907	1062	1238	1399	1562	1746
		Transfers	0	2	11	21	32	58	78	109	140	188
		Ordering	262	953	2296	4172	6132	8455	11592	14812	18962	23294
		Routing	783	1510	2227	3187	4154	4733	5518	6400	7296	8085
BN	30 min	Trains	33	67	100	134	168	201	235	268	302	336
		Stops	329	668	980	1265	1598	1920	2283	2597	2875	3143
		Transfers	2	7	12	27	45	70	97	133	173	212
		Ordering	36	198	348	578	929	1267	2143	2703	3358	4258
		Routing	49	97	137	189	241	286	430	482	526	580
LZ-AA	30 min	Trains	21	42	64	85	107	128	149	171	192	214
		Stops	232	431	610	780	951	1150	1319	1503	1606	1789
		Transfers	0	10	19	22	32	46	68	84	113	134
		Ordering	22	61	176	295	487	780	1101	1526	1977	2396
		Routing	50	97	185	233	311	433	513	599	650	732
HE-LZ	30 min	Trains	36	73	109	146	183	219	256	292	329	366
		Stops	399	767	1091	1490	1885	2292	2770	3048	3406	3751
		Transfers	3	17	33	53	75	101	135	179	223	278
		Ordering	56	175	359	675	1124	1852	2381	2949	3729	4384
		Routing	110	170	258	371	483	696	777	853	947	1006
ZAS-YV	30 min	Trains	47	95	143	191	239	287	335	383	431	479
		Stops	402	899	1426	1863	2376	2781	3240	3660	4159	4639
		Transfers	2	10	18	26	50	79	110	142	184	225
		Ordering	69	266	584	966	1493	2183	2779	3646	4843	6025
		Routing	70	164	254	312	418	514	579	667	842	956

TABLE 5.2: Instance Characteristics, Part I.

Scenario	Max Deviation	Instance:	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
RH-ZAS	30 min	Trains	28	56	84	112	140	168	196	224	252	280
		Stops	338	628	926	1343	1689	1931	2324	2662	2935	3260
		Transfers	2	2	12	26	41	51	69	81	116	143
		Ordering	20	90	234	456	718	1079	1647	2111	2869	3533
		Routing	21	52	84	120	143	199	223	260	348	380
ZAS-BEL	1 h	Trains	27	54	82	109	137	164	191	219	246	274
		Stops	244	552	815	1085	1387	1612	1865	2207	2467	2787
		Transfers	1	6	7	15	19	29	44	61	70	85
		Ordering	32	181	507	966	1512	1911	2517	3403	4612	5767
		Routing	32	78	132	180	230	256	298	354	568	611
ZUE-SG	30 min	Trains	70	140	210	280	350	420	489	560	630	700
		Stops	776	1651	2314	3001	3724	4433	5183	6067	6868	7752
		Transfers	3	19	43	71	123	180	245	327	413	503
		Ordering	112	354	977	1941	3068	4182	5480	7147	8718	10709
		Routing	86	113	314	395	486	531	580	674	715	782
BN-ZAS	30 min	Trains	23	47	71	95	119	143	167	191	215	239
		Stops	247	496	741	1016	1294	1539	1845	2101	2325	2640
		Transfers	1	1	3	8	14	21	30	39	51	64
		Ordering	0	0	0	0	0	406	442	514	557	632
		Routing	62	102	162	200	245	1033	1061	1101	1145	1209
ZUE-OTH	30 min	Trains	16	33	50	67	84	100	117	134	151	168
		Stops	220	393	596	745	880	1052	1170	1328	1496	1640
		Transfers	0	2	4	5	11	14	19	26	37	44
		Ordering	29	62	90	106	146	156	199	212	269	331
		Routing	271	279	289	295	307	307	323	327	337	357
BDF-SO	1 h	Trains	71	143	215	287	359	431	503	575	647	719
		Stops	858	1709	2542	3418	4201	4950	5856	6674	7516	8273
		Transfers	6	11	35	81	138	203	260	329	426	514
		Ordering	0	0	0	0	236	653	907	1176	1993	2566
		Routing	18	42	74	90	168	272	376	462	564	668
GD-BS	30 min	Trains	36	73	109	146	183	219	256	292	329	366
		Stops	317	700	936	1282	1542	1859	2129	2392	2718	3150
		Transfers	0	5	16	28	43	64	86	107	141	180
		Ordering	0	0	0	443	533	659	728	800	924	994
		Routing	46	94	128	741	859	927	971	993	1064	1092

TABLE 5.3: Instance Characteristics, Part II.

6

CONCLUSIONS

6.1 MAIN FINDINGS

The present thesis is a contribution towards an automation of railway scheduling with novel innovative algorithmic solutions. To discuss the findings of this thesis we recall the research questions of Section 1.3:

How can the scalability and efficiency of algorithms in railway scheduling be improved?

With the research conduction in the individual chapters of this thesis we can answer the main research question of this thesis as follows:

The problem of railway scheduling can be decomposed in different domains and by different principles of decomposition. Individual decompositions have different advantages and disadvantages for different types of problems in railway scheduling. E.g., some decompositions are suitable for problems concerning large infrastructure others are suitable for problems containing extremely dense railway traffic. Individual advantages and disadvantages are the result of mainly three factors. With the domain of decomposition it is defined which physical or mathematical aspect of the scheduling problem is to be exposed for decomposition (e.g., geography or time). By the principle of decomposition it is defined, how the particular a aspect (defined by the domain) is exposed for decomposition and finally the solution method defines how a decomposed formulation of the scheduling problem is addressed, to retrieve a solution for the original problem of scheduling.

Base on these findings, we propose in this thesis a geographically inspired, logic-based Benders decomposition for the problem of railway timetabling, to create an approach with improved scalability behavior. With the development of novel theory it is possible to expose a structure in the problem of railway timetabling, different from the existing literature. Empirical results on the proposed decomposition underline the benefits of our proposed approach. With the novel decomposition an approach is developed, which shows an improved scalability, compared to other existing approaches in the literature.

With a study on recurrent situation in the problem of railway rescheduling, we were able to further enhance the decomposition approach from previous studies of this thesis. It showed possible to learn patterns of recurrent situations using statistical knowledge on the problem of rescheduling. Learnt patters can be reused in the decomposition approach of previous studies to decrease the overall computational effort and improve the performance of the algorithm, for the time-critical application of real-time railway rescheduling. With a series of experiments the benefits of statistical learning and the exposure of recurrent situations could clearly be shown.

In an alternative direction, applications of infeasibilities in the problem railway timetabling were analyzed. Using proofs of infeasibility on the problem of railway timetabling, it was possible to design a series of highly efficient incremental algorithms for timetabling. Incremental approaches sacrifice the optimality of a solution in timetabling, in favor of computational speed. Therefore, these approaches provide solution in very short time.

Overall, the thesis shows how approaches of decomposition can be used to achieve better scalability in algorithms for railway timetabling and how recurrent situations can be exposed to improve such performance even further. Infeasibilities are shown to provide valuable information on the problem of timetabling, leading to highly efficient methods of timetabling. The thesis builds a basis for future development of scalable and efficient algorithms on problems of railway rescheduling.

Q1: *How can problems of railway scheduling be decomposed?*

In Chapter 2, we report on current practices of decomposition in the literature of railway scheduling. We identified in the literature different domains, principles of decompositions as well as a variety of solution methods. In particular the geographic, temporal, entity (resource) and generic domain have been identified. Two principles of decomposition are used to expose structures related to individual domains in a problem of scheduling. The principle of complicating variables exposes structures in a problem of scheduling as variables are separated into multiple groups, each group optimized in an individual optimization problem. The principle of complicating constraints exposes structures in a problem of scheduling as constraints are separated into multiple groups, each group optimized in an individual optimization problem. Solution methods are considered either hierarchical or decentralized. In hierarchical methods, some problems of a decomposition are responsible for the coordination of other problems.

Hierarchical methods are not fully parallelizable and different groups of problems must be addresses sequential. In advantage, hierarchical methods, through their scheme of coordination provide often near optimal or optimal solutions. In decentralized methods, all problems of a decomposition are self-responsible for a coordination with other problems. While decentralized methods are in general heuristics, those can be fully parallelized and each problem can be addressed independently of the others. Research gaps have been identified in the study of the literature, including the application of Benders decomposition, which motivated our research in Chapter 3.

Q2: *Can decomposition improve the scalability of algorithms for railway timetabling?*

In Chapter 3 a logic-based Benders decomposition has been developed, as a possible algorithm with improved scalability for the problem of railway timetabling. In Chapter 3, we propose a model for railway timetabling very similar to the literature, where we include the often excluded task of routing trains on the railway network. We provide a novel way to model dependencies of routing decisions and ordering decisions. The major contribution of Chapter 3 is a novel logic-based Benders decomposition for a generic formulation of a scheduling problem. In comparison to the existing literature, the propose logic-based decomposition shows strong generality in the way how problems of railway timetabling can be decomposed. In the existing literature, decomposition approaches often rely on the fact that problems of the decomposition are of a particular type (e.g., shortest-path or linear and continuous). Differently, in the proposed decomposition, all problems of the decomposition are considered as generic problems of railway timetabling. This generality proposes a major freedom in the exposure of structures and decomposition of a problem in railway scheduling, and is a major advantage of the proposed decomposition approach. An implementation of the novel methodology in Chapter 3 lead to Python-Smarties (Appendix A), that is a highly efficient tool for the discovery of infeasibility proofs in problems of railway scheduling. We propose in Chapter 3 an application of the novel methodology for a geographic decomposition of the problem of railway timetabling. A series of experiments, based on real-life examples of timetabling provided by SBB, empirically proves the value of our proposed methodology. With the decomposition of the timetabling problem, it was possible to solve instances of twice the size, up to 40 times faster than with a

conventional approach applied to an undecomposed formulation of the problem. Experiments also show a limited performance of the decomposition, in case the structure to be exposed by the decomposition is only marginally present or completely absent. In conclusion, we were able to develop, based on the methodology of decomposition, a novel algorithm for railway timetabling with an improved scalability compared to existing approaches.

Q3: *Can statistical learning of recurrent situations improve existing approaches of OR for the time-critical problem of real-time railway rescheduling?*

In Chapter 4 an approach of statistical learning has been proposed on the problem of railway rescheduling to expose recurrent situation in railway rescheduling. Statistical knowledge about recurrent situations in railway rescheduling has been used to learn patterns in such problems, which have been included in the decomposition approach of Chapter 3, to improve the overall computational performance. We were able provide theory, showing that patters in rescheduling can be learned from solving large numbers of rescheduling problems (generated from statistical knowledge) and reused on new and different problems of rescheduling on the same infrastructure. Experiments on real-life data indicate a strong potential, of reusing learnt patterns to improve the Benders decomposition of Chapter 3. The computational performance can potentially be improved up to a factor of 2 in terms computational speed using methods of statistical learning and learnt patterns. Practically, limitations of the statistical learning approach have been experienced as learnt patterns do not always result in a computational speedup and filtering on patterns is necessary. A first design of a filter was able to achieve a practical speedup of a factor 1.2 compared to the decomposition of Chapter 3. In summary, with the research of Chapter 4 we were able to show a great potential in the exposure of patterns in recurrent situation of rescheduling to enhance existing methods and provide approaches, efficient enough for time-critical applications, such as real-time railway rescheduling.

Q4: *How can proofs of infeasibility be used to design efficient algorithms for railway timetabling?*

In Chapter 5 we develop an efficient algorithm for the problem of railway timetabling, exploiting the computational insights gained by infeasibilities

inside a problem railway timetabling. We propose in Chapter 5 an approach for railway timetabling, which is an alternative application of Benders decomposition compared to Chapter 3. The decomposition exploits infeasibilities in railway timetabling, which allows us to address the problem of finding an optimal timetable by solving a set covering problem. The solution to the set covering problem defines the amount of planning deviation in a timetable, such that in a second problem of the decomposition, we evaluate the existence of a timetable for the given amount of planning deviation. In such decomposition, we address the problem of set covering with highly efficient incremental heuristics, which allows us to provide in short-time, solutions to the problem of timetabling with reasonable high quality. The proposed approach makes use of Python-Smarties to efficiently determine the feasibility or infeasibility of a timetabling problem for a fixed amount of planning deviation. Experiments on real-life data show that with incremental heuristics we can achieve significant computational speedups in comparison to conventional approaches. With the research of Chapter 5 we showed that we can exploit infeasibilities in a problem of railway timetabling to create incremental heuristics, which are extremely efficient algorithms for the problems of railway scheduling.

6.2 IMPLICATIONS

The findings of this thesis have an impact on practitioners in the railway industry and researches in academia studying the problems of railway traffic management and other problems of scheduling in operations research. The thesis has the most impact on the following three fields of practice and academia:

Railway Timetabling

Foremost, the main contribution of this thesis is a versatile decomposition approach for the problem of railway timetabling in Chapter 3. With a geographic decomposition of the railway timetabling problem an algorithm has been developed, which shows improved scalability against existing approaches. The result are significant computational speedups, which allows to solve larger instances of railway timetabling to an optimal solution. The novel algorithm with an improved scalability has several implications on the industry and academia.

With the novel geographic decomposition approach, developed in this thesis, we combine complex theoretical concepts with a geographic sep-

aration of the timetabling problem. The theoretical contributions for the geographic decomposition are based on concepts from mathematical optimization, which clearly require a certain prior knowledge in mathematics. In contrast, the geographic decomposition of railway timetabling itself has a rather natural interpretation. We can understand each subproblem in such decomposition as a geographically local timetabling problem, and the master problem as a timetabling problem, coordinating railway traffic amongst geographic areas. This simple interpretation of our decomposition helps railway practitioners to understand the results of our algorithm and provides great practical value. In fact, for experiments on our geographic decomposition, the railway system in our real-life instances has been geographically separated into local areas by railway practitioners. This underlines how the geographic decomposition, despite including complex theoretical concepts, can easily be understood by a broad audience of practitioners in the railway industry. With the geographic decomposition approach we provide a basis for the design of highly scalable and reliable algorithms for scheduling large-scale problems of railway timetabling. The decomposition allows for partition of the railway scheduling problem, where a significant amount of computational complexity is distributed onto several partial problems, each with substantial computational complexity. Partial problems of the decomposition can addressed by parallel computations, possibly distributed over multiple physical machines. With the proposed decomposition approach, we see a basis given for scheduling algorithms able to handle cases of scheduling for railway networks in the size of entire countries.

In an alternative branch of research, we developed a different decomposition-based approach for the problem of railway timetabling, where infeasibilities in railway timetabling are exposed to design highly efficient heuristics. With the development of an efficient algorithm for railway timetabling, we see the basis given for a tool, that can be used by railway practitioners to find quickly, local adaptations of timetables, e.g., when planning extra trains. In practice, when adapting timetables locally for maintenance, extra trains, or construction, practitioners usually aim to try out several alternative solutions, such that solutions should be provided within few minutes to avoid long waiting times. With the above discussed research, we provide an algorithmic basis for planning tools used in the daily adaptations of timetables due all sorts of special occasions.

Overall, the findings of this thesis are a step towards automated railway timetabling in large-scale. With more capability in solving larger instances of timetabling, less divisions of the network-wide timetabling problems

must be made, and global solutions nearer to optimality can be achieved. These timetables will clearly increase the utilization of existing resources. Thereby, the total capacity of the railway system can be increased, without or with only minor acquisitions of new infrastructure, which is clearly favorable as acquisitions of infrastructure are usually cost-intensive, time consuming and increase the overall maintenance in a railway system. The result is a more economical railway system, which is desired by railway operators and will also reflect beneficially to railway customers in lower ticket prices.

Railway Traffic Management

In this thesis, we propose to expose recurrent situations in railway rescheduling to improve the computational performance of our geographic decomposition of Chapter 3. Statistical knowledge about the recurrent situations allow to learn patters, which, if reused, accelerate the performance of the geographic decomposition.

The increase in computational speed is of great value for the time-critical application of real-time railway rescheduling, where an existing timetable must be adapted in short time to the current state of the railway system. More efficient algorithms allow railway operators to better react on disturbances of the railway system during the operation. With more computational performance, larger geographic areas and more traffic can be considered in adjustment of the timetable. When considering larger geographic areas, adjusted timetables incorporate a more global view, preventing better the spread of delay in the railway network. Overall, with an increase of computational performance the system recovery can be improved, leading to an increase in production, reliability and consequentially customer service. An efficient algorithm for rescheduling as proposed in Chapter 4 brings great value to railway operators.

Scheduling in Operations Research

With the present thesis several contributions on the problem of scheduling have been made. While applications in this thesis are exclusively on railways, we pointed out that theoretical findings in this thesis are not limited to the railway sector. All theoretical contributions are made on a rather generic disjunctive mathematical problem, such that the findings of Chapter 3, Chapter 4 and Chapter 5 can be applied upon any problem of the industry or academia, which can be written in the same form (e.g.,

problems of job-shop scheduling [7]). Theoretical contribution of this thesis can provide a basis to other researchers working on different scheduling problems. Algorithms of this thesis can be used by practitioners in other branches of the industry. In general, scheduling problems are present in many academic and industrial problems (e.g., [2]), such that we see great potential for further use of the results in this thesis.

6.3 OUTLOOK

Railway timetabling, railway rescheduling and more general railway traffic management are still active fields of research, where gaps between academia and practice still exists and problems of practice are in many cases still overwhelmingly complex for modern approaches of academia. The findings of this thesis are a contribution towards closing this gap, but clearly a gap remains also after the presented research of this thesis, leaving a variety of future research possibilities. Here below we discuss overarching possibilities of future research. More detailed information on future research is given at the end of individual chapters in the thesis.

Domains of Decomposition

Chapter 3 presents a generic tool for the decomposition of a railway scheduling problem. In the experiments of Chapter 3 a geographic decomposition has been studied, where subproblems correspond to areas of high railway track density. Differently, a study of the literature revealed decomposition approaches in railway scheduling spreading over four different domains: geographic, temporal, entity and generic. In the scope of this thesis only decompositions in the geographic domains have been studied and it remains a open topic for future research to study decompositions also in different domains, using the theoretical and methodological findings of Chapter 3. As we discovered different advantages of different domains in an analysis of the literature, we expect different performances, in case our generic decomposition of Chapter 3 is applied to different domains. Different domains may show particularly valuable for specific instances of railway scheduling.

In the temporal domain, where the time period of a scheduling problem is partitioned into smaller periods of time for scheduling, a single publication of a temporal decomposition with independent subproblems has been found, i.e., [5]. Our theoretical findings of Chapter 3 clearly can be extended to a decomposition in the temporal domain, which would

lead, in contrast to [5], to a temporal decomposition based on complicating variables rather than complicating constraints. We believe future research should study an application of Chapter 3 in the temporal domain, as we expect similar or better computational improvements as experienced for the geographic domain. Computational improvements could exceed those of a geographic decomposition as in Chapter 2, we have seen that the number of complicating elements in a temporal decomposition can be lower than for a geographic decomposition, leading to a smaller master problem and a more balanced decomposition.

We also see possible applications of findings in Chapter 3 in the generic domain. A particular example of such a decomposition has already been given within the scope of this thesis, by the decomposition proposed in Chapter 5. In case of Chapter 5, variables related to the objective are identified as complicating. Together with an appropriate formulation of the scheduling problem, the logic-based Benders cut introduced in Chapter 3 reduced to the combinatorial Benders cut of [1]. We see great potential for more scalable and more efficient algorithms, by decompositions in the generic domain, likewise to Chapter 5 exploring other, different definitions of complicating variables. Recent publications, besides the work of this thesis in Chapter 5, e.g., [3, 4] similarly motivate a decomposition in the generic domain.

Configurations of a Decomposition

While experiments on real-life data in Chapter 3 showed already impressive improvements, no study on the configurations of a decomposition has been made within the scope of this thesis. The decompositions used for the experiments of Chapter 3, but also Chapter 4 are designed by human planners at SBB, based on practical understanding and experience on the problem. It remains for further research to understand the effects of different configurations for a decomposition, especially on the computational performance. With the configuration of a decomposition in the geographic domain we define size, shape and content of areas in a railway network which decompose the problem. Other studies on configurations in decomposition (e.g., [6]) show a clear dependency between the configuration and computational performance.

We expect different configurations to show varying performance also for the decompositions of this thesis. The higher the number of subproblems, the more significant we expect the coordination effort and thus the complexity of the master problem; at the same time a high number of sub-

problems allows for more efficient parallelization. Differently, larger sized subproblems are expected to result in less coordination effort, compared to small but numerous subproblems, due to the larger solution space of an individual subproblem.

Also, experiments of Chapter 3 show, that even a decomposition showing in general good computational results, may show bad computational performance simply due to an inappropriate instance of timetabling (see experiments where routing decisions have been excluded). The performance of a decomposition thus not only depends on the type of problem at hand, but further on the particular instance of scheduling at hand. We see it as a relevant direction for further studies, not only to study different configurations of decompositions in general, but possibly extending such study to configurations, which are adapted to individual instances of timetabling, rescheduling or other problems.

All above future research can also be extended to decompositions outside of the geographic domain. As discussed above, other domains show great potential and a combination of research on domains and configurations is strongly advisable.

REFERENCES

1. Codato, G. & Fischetti, M. Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. *Operations Research* **54**, 756 (2006).
2. Kallrath, J. Planning and scheduling in the process industry. *OR spectrum* **24**, 219 (2002).
3. Keita, K., Pellegrini, P. & Rodriguez, J. A three-step Benders decomposition for the real-time Railway Traffic Management Problem. *Journal of Rail Transport Planning & Management* **13**, 100170 (2020).
4. Lamorgese, L. & Mannino, C. A non-compact formulation for job-shop scheduling problems in traffic management. *Operations Research* **67**, 1503 (2019).
5. Luan, X., De Schutter, B., Meng, L. & Corman, F. Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. *Transportation Research Part B Methodological* **141**, 72 (2020).

6. Luan, X., De Schutter, B., Meng, L. & Corman, F. Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. *Transportation Research Part B Methodological* **141**, 72 (2020).

7. Zhang, J., Ding, G., Zou, Y., Qin, S. & Fu, J. Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing* **30**, 1809 (2019).

A

PYTHON-SMARTIES

In the course of this thesis a Boolean satisfiability solver (Python-Smarties) has been developed. The solver is designed to address problems of railway scheduling; in particular to determine the feasibility of a scheduling problem in an efficient manner. In case a scheduling problem is infeasible, the solver identifies a possible cause for the infeasibility and provides a corresponding proof of infeasibility. Throughout the entire thesis, proofs of infeasibility are used in different approaches, with Python-Smarties as the algorithmic backbone.

This appendix is a summary of theoretical concepts from different literature in Satisfiability solving and graph theory. Concepts and algorithms discusses below are combined within the scope of this thesis, into the implementation of Python-Smarties. With Python-Smarties a novel, fast and efficient tool for the computation of infeasibility proofs in problems of railway scheduling has been created.

A.1 SAT - BOOLEAN SATISFIABILITY SOLVING

The problem of Boolean satisfiability solving (SAT) is a problem of feasibility over a set of Boolean variables $\{x_1, x_2 \cdots x_n\}$ [10]. A literal in SAT is a representation of the value of a Boolean variable and is either the variable, e.g., x_1, representing $x_1 = \mathit{True}$, or its negation, e.g., $\neg x_1$ representing $x_1 = \mathit{False}$. An assignment is a set of literals, which contains exactly one literal for every Boolean variable of the problem. An assignment is partial if for some variables, the set does not contain a corresponding literal. The problem SAT is to find an assignment such that all constraints of the problem are satisfied. The constraints of SAT are known as clauses. A single clause is the disjunction (logic OR) over literals, e.g., $\{x_1 \vee \neg x_3 \vee \neg x_{10} \vee \cdots\}$, and satisfied by an (partial) assignment, if at least one literal from the disjunction of the clause is contained in the assignment. Problems of SAT are in general written in conjunctive normal form (CNF), which is a conjunction (logic AND) of multiple clauses. A model is a feasible solution for a problem of SAT and an assignment, which satisfies all clauses of the problem.

A.2 DPLL FOR SAT

The majority of state-of-the-art solvers for problems of SAT are based on the DPLL (Davis-Putnam-Logemann-Loveland) scheme [5, 6]. DPLL is a branching scheme over the Boolean variables of a SAT problem, including the concept of unit propagation. At every incumbent node of the branching tree in DPLL, a (partial) assignment on the Boolean variables is given. Variables with no corresponding literal in the partial assignment are considered unassigned at the incumbent node.

For a partial assignment (at a node of the branching tree), a *unit clause* is a clause, which is not satisfied by the partial assignment, but the assignment contains the negation of all but a single literal from the disjunction that is the clause (hence the name "unit" clause). If for a partial assignment, there exists a unit clause in a problem of SAT, such partial assignment can only be extended to a model, i.e., a feasible solution, if the single literal in the unit clause, whose negation is not yet in the partial assignment, is added to such assignment; otherwise the unit clause will inevitably be violated. The extension of a partial assignment by a further literal due to unit clauses is referred to as unit propagation [9].

In DPLL different heuristics are used at nodes of the branching tree to decide on the next variable to branch on, i.e., the next literal to be added to the partial assignment. The most common heuristic in SAT is the Variable State Independet Decay Sum (VSIDS [11]), which keeps track of the occurrences of variables in clauses of a SAT problem.

A.3 CONFLICT-DRIVEN CLAUSE LEARNING

State-of-the-art SAT solvers make use of conflict-driven clause learning (CDCL) [9] inside DPLL to strengthen the initial set of clauses by additional clauses, likewise to additional cuts in Branch-and-Cut schemes, e.g., for mixed-integer linear programming. The scheme of CDCL is to learn a new clause in case of a conflict in DPLL. A *conflict* in DPLL is a clause, which is violated by the partial assignment at a node in the branching tree, due to implied literals from unit propagation. In CDCL, a new clause is generated from a conflict as the violated clause is combined with related consequences of unit propagation. The new (learnt) clause is a disjunction over those literals, which together with unit propagation, lead to the conflict. Such clause becomes unit for any partial assignment, where one (wrong) additional literal would cause again the conflict at hand. Therefore, the conflict

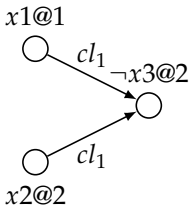

FIGURE A.1: Implication graph with a single clause cl_1.

is avoided in the further search process of DPLL through unit propagation on the learnt clause.

Clauses in CDCL are learnt as literals responsible for the conflict are identified, based on an implication graph (see [9]). The implication graph is a directed graph to represent implications (unit propagations) between literals at an incumbent node of the branching tree in DPLL. Nodes in the implication graph represent literals and arcs represent (unit) clauses, i.e., implications. A node in the implication graph has zero ingoing arcs in case it represents a selected literal, i.e., a literal not implied by any unit clause, but branched-on in the branching tree of DPLL. In case a node is implied by a unit clause, the node has as many ingoing arcs as the implying unit clause has literals; not counting the literal of the node itself, which is also part of the clause. Figure A.1 shows an example of an implication graph with a single clause $cl_1 = \{\neg x_1 \vee \neg x_2 \vee \neg x_3\}$. The @ written to each literal indicates the depth in the branching tree of DPLL, at which the literal is either branched on, or implied by unit propagation. In Figure A.1 variable $x1$ is branched to the literal $x1$ (*True*) at the first level of the tree, which results in no implication through unit propagation. At the second level $x2$ is branched to the literal $x2$ (*True*), which together with literal $x1$ of the previous level leads to the implication $\neg x3$ at the same level (level 2), due to the (now unit) clause cl_1.

We extend the example of Figure A.1 by three further clauses, i.e., $cl_2 = \{x3 \vee \neg x_4 \vee \neg x5\}$, $cl_3 = \{x3 \vee x6\}$, $cl_4 = \{x5 \vee \neg x6\}$ to illustrate the occurrence of a conflict and the resulting clause learning. In Figure A.2, we illustrate the implication graph of all four clauses and the occurrence of a conflict κ. The conflict is given at the third level of branching in DPLL, when variable $x2$ is branched to literal $x2@3$, i.e., $x2 = \mathit{True}$. In this case, unit propagation of cl_1 leads to $\neg x_3@3$, which then, by unit propagation of cl_2, leads to $\neg x5@3$. Finally, the clauses cl_3 and cl_4 become unit due to $\neg x_3$ and $\neg x5$, and imply contradicting literals for the variable x_6, i.e., x_6 and $\neg x_6$ respectively, proposing a conflict κ, i.e., either cl_3 or cl_4 is the violated

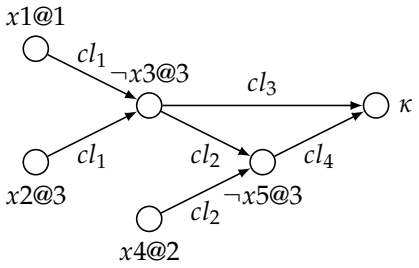

FIGURE A.2: Implication graph with a conflict κ.

clause depending on which unit propagation, i.e., unit propagation of cl_3 or cl_4, is processed first.

Based on the implication graph of Figure A.2 different learning schemes generate different learnt clauses. The simplest learned clause from the conflict κ in Figure A.2 would be a conjunction over the negation of all selected (i.e., not implied) literals that are the roots of the conflict node in the implication graph, i.e., $cl_{learned} = \{\neg x_1 \vee \neg x_2 \vee \neg x_4\}$. More advanced schemes, such as the scheme of an unique implication point [9, 13] aim to generate learnt clauses of smaller size, i.e., fewer literals, known to be stronger constraints in the problem of SAT. In Figure A.2 the node of $\neg x_3$ is a *unique implication point* (bottleneck), such that, independent of the roots of node $\neg x_3$, in case $x3 = False$, the conflict in Figure A.2 can occur. In the unique implication point scheme, the learnt clause is the disjunction over the negation of all unique implication points in the implication graph, e.g., $cl_{learnt} = \{x_3 \vee \neg x_4\}$. In the graph of Figure A.2 $x4$ and $\neg x5$ are equally unique implication points.

If a conflict has been detected in DPLL, the resulting learnt clause is by design, violated by the incumbent partial assignment at the node in the branching tree of DPLL, at which the conflict has been detected. A non-chronological backtracking is performed, based on the learnt clause, where a jump is made in the branching tree up to a earlier node, where the learnt clause is not violated, but a unit clause due to the incumbent partial assignment at such node.

A.4 DIFFERENCE CONSTRAINTS IN SAT

In the literature, Boolean Satisfiability solvers have been extended by various kinds of constraints other than clauses; these extensions are in general

referred to as satisfiability modulo theories (SMT). SMT is the problem of determining whether a first-order formula [12], i.e., the modulo theory, is satisfiable with respect to some logic (Boolean) theory, i.e., a problem of SAT.

One such modulo theory are difference constraints, i.e., linear inequality constraints such as $y - x \geq k$ for two variables y, x and a constant k. Difference constraints are of particular interest for this thesis as all kinds of scheduling problems contain precedence relations, which are mathematically the same as difference constraints.

Difference constraints are coupled with the logic of a SAT problem, as values of Boolean variables define whether a particular set of difference constraints is to be satisfied or not. The problem of SMT considering difference constraints is to find an assignment for Boolean variables, such that a set of the difference constraints, implied by such assignment, is satisfiable. A set of difference constraints is satisfiable if for variables in difference constraints, e.g., y, x, values can be found, such that all difference constraints of the set are satisfied.

Difference constraints are integrated into DPLL through the processes of validation and propagation [4]. Validation is the lazy evaluation of an existing (partial) assignment on Boolean variables for feasibility on difference constraints. Propagation is the proactive extension of an existing partial assignment on Boolean variables by further literals to guarantee satisfiability of difference constraints, likewise to unit propagation. (In the context of this thesis, for Python-Smarties, only the validation of difference constraints is implemented.)

To evaluate the satisfiability of a set difference constraints, an efficient approach has been proposed in [4]. A set of difference constraints can be validated for satisfiability using a directed graph, where each node represents a variable and each arc represents a single difference constraint. Such graph is free of any positive length cycle if and only if the set of difference constraints that are the arcs of the graph, is satisfiable. Positive length cycles in a graph can be detected, e.g., by the Bellman-Ford algorithm [2], with a runtime of $O(n * m)$. In the application of difference constraints in SAT, i.e., in DPLL, the graph representing the difference constraints changes only in few arcs between adjacent nodes in the branching tree. The authors of [4] show, that in this case it possible to use the computationally less expensive algorithm of Dijkstra [7] with a runtime of $O(n * log(n) + m)$ to evaluate the graph for positive length cycles. The result is a significant reduction in the computational effort to validate a set of difference constraints for satisfiability.

In case a positive cycle is detected during the validation of difference constraints, the assignment on Boolean variables implies an unsatisfiable set of difference constraints. All literals of the assignment, which impose at least one arc in the detected cycle are responsible for the non-satisfiability of the difference constraints. These literals can be treated likewise to literals of unique implication points in CDCL, and a new clause can be generated (learnt). From the learnt clause a non-chronological backtracking can be made.

A.5 INFEASIBILITY IN SAT

SAT solvers have widely been used to compute proofs of infeasibility (e.g., in MaxSAT [10]). The most common approach to compute a proof of infeasibility with a SAT solver is the usage of marker literals [1]. With marker literals every clause in an original SAT problem is marked (extended) by an additional (marker) literal. When solving a problem of SAT to compute a proof of infeasibility an initial partial assignment is considered, which contains the negation of all marker literals. With such initial assignment all clauses maintain their original logical meaning. An initial (fixed) assignment is often referred to as assumptions in SAT solving. When using maker literals on an infeasible problem of SAT, the DPLL scheme will eventually discover a conflict, that is learnt clause only containing marker literals. The marker literals in such conflicting clause indicate original clauses of the problem that build together a proof of infeasibility. Marker literals impose a rather minor overhead in implementation, but are generally known to decrease performance of SAT solvers [1] and propose significant performance issues for large-scale SAT problems.

In alternative to marker literals an ancestor list over the learnt clauses can be kept [1]. If a SAT solver is applied to an infeasible problem without marker literals, the DPLL will eventually discover a conflict, that is a learnt and empty clause, i.e., a clause with no literal. The empty clause is a constraint, which cannot be satisfied in any case, proving the infeasibility of the problem. With an ancestor list, those clauses of the original problem can be identified, from which the empty clause has been deduced. The identified clauses of the original problem together, are a proof of infeasibility. The ancestor list can be derived using the implication tree. Ancestors of a learnt clause are all clauses that are arcs in the implication graph between the unique implication points used as literals in the learnt clause, and the conflict node. In the example of Figure A.2 for $cl_{learnt} = \{x_3 \vee \neg x_4\}$ the

ancestors are cl_2, cl_3 and cl_4. Differently if $cl_{learnt} = \{x_3 \vee x_5\}$ the ancestors would only be cl_3 and cl_4. An ancestor list can be stored in two different structures, i.e., a flat structure or a tree structure [1]. In a flat structure, only clauses of the original problem are enlisted as ancestors of learnt clauses. That is, whenever a previously learnt clause is one of the ancestors of a new learnt clause, such clause is resolved into clauses of the original problem using the ancestor list. Flat ancestor lists suffer in general from large numbers of ancestors and thus using a significant amount of memory. In alternative to a flat structure, the ancestor list can be kept in a tree structure. In this case, also learnt clauses can be enlisted as ancestors and each entry in the ancestor list reports only immediate ancestors. In this way the entries of the ancestor list are kept significantly smaller than in a flat structure. In disadvantage to a flat structure, in an ancestor list of tree structure, the tree must be resolved after an empty clause has been discovered, to get to the clauses of the original problem, responsible for the infeasibility of a problem.

A.6 PYTHON-SMARTIES

In the course of this thesis a SAT solver has been developed for the efficient detection of infeasibility proofs in the problem of Boolean Satisfiability extended by difference constraints. The SAT solver for this thesis has been developed based on the open source SAT Solver MiniSat 2.2 [8]. MiniSat 2.2. is an highly optimized implementation of the CDCL scheme in C++. Python-Smarties has been implemented in C++ and has been encapsulated into a Python package for an easy application.

For the precedence relations (difference constraints) of scheduling problems, the open source code of MiniSat 2.2 has been extended by the Dijkstra-based algorithm of [3], discussed in Section A.4. The implementation of difference constraints in Python-Smarties does only support the validation of an assignment on Boolean variables and a proactive propagation has not been implemented.

In Python-Smarties proofs of infeasibility are traced by a tree structured ancestor list as proposed by [1]. Not only allows such ancestor list to trace infeasibilities with minor computational overhead, but also allows for further features in the solver. In particular, the solver can be warm started, i.e., learnt clauses can be kept in the solver, even after the removal of clauses. In normal SAT solvers, in case a clause is removed, all learnt clauses have to be removed as dependencies between learnt clauses and clauses of the original problem are unknown. Differently, using an ancestor list, all such

dependencies are known and in case an original clause is removed, only dependent learnt clauses can be removed; the solver keeps all other learnt clauses for warm starting. This feature becomes particularly handy in applications, where the SAT solver is queried multiple times on problems only differing by few clauses.

All implementations above of features for MiniSat 2.2 are optimized to reduce the amount of queries to the RAM, general memory consumption and continuous storage usage in the RAM.

REFERENCES

1. Asin, R., Nieuwenhuis, R., Oliveras, A. & Rodriguez-Carbonell, E. *Efficient generation of unsatisfiability proofs and cores in SAT* in *International Conference on Logic for Programming Artificial Intelligence and Reasoning* (2008), 16.
2. Bellman, R. On a routing problem. *Quarterly of Applied Mathematics* **16**, 87 (1958).
3. Codato, G. & Fischetti, M. Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. *Operations Research* **54**, 756 (2006).
4. Cotton, S. & Maler, O. *Fast and flexible difference constraint propagation for DPLL (T)* in *International Conference on Theory and Applications of Satisfiability Testing* (2006), 170.
5. Davis, M., Logemann, G. & Loveland, D. A Machine Program for Theorem-Proving. *Communications of the ACM* **5**, 394 (1962).
6. Davis, M. & Putnam, H. A Computing Procedure for Quantification Theory. *Journal of the ACM* **7**, 201 (1960).
7. Dijkstra, E. W. A Note on Two Problem in Connexion with Graphs. *Numerische Mathematik* **1**, 269 (1959).
8. Eén, N. & Sörensson, N. *An extensible SAT-solver* in *International Conference on Theory and Applications of Satisfiability Testing* (2003), 502.
9. Marques-Silva, J. P. & Sakallah, K. A. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers* **48**, 506 (1999).
10. Morgado, A., Heras, F., Liffiton, M., Planes, J. & Marques-Silva, J. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints* **18**, 478 (2013).

11. Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L. & Malik, S. *Chaff: Engineering an efficient SAT solver* in *Proceedings of the 38th annual Design Automation Conference* (2001), 530.

12. Nieuwenhuis, R., Oliveras, A. & Tinelli, C. Solving SAT and SAT modulo theories: From an abstract davis - putnam - logemann - loveland procedure to DPLL(T). *Journal of the ACM* **53**, 937 (2006).

13. Zhang, L., Madigan, C., Moskewicz, M. & Malik, S. *Efficient conflict driven learning in a boolean satisfiability solver* in *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers* (2001), 279.

PUBLICATIONS

Articles in peer-reviewed journals:

1. Leutwiler, F., Bonet Filella, G. & Corman, F. Accelerating logic-based Benders Decomposition for Railway Rescheduling by exploiting similarities in delays. *Computers and Operations Research* **150**, 106075 (2023).
2. Leutwiler, F. & Corman, F. A logic-based Benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research* **303**, 525 (2022).
3. Leutwiler, F. & Corman, F. A Review of Principles and Methods to decompose Large-Scale Railway Scheduling Problems. *Under Revision in European Journal of Transportation and Logistics*.
4. Leutwiler, F. & Corman, F. Heuristics of Set Covering in a Benders Decomposition for Railway Timetabling. *Under Revision in Computers and Operations Research*.