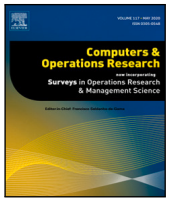
Set covering heuristics in a Benders decomposition for railway timetabling

Computers & Operations Research 159 (2023) 106339

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor



Set covering heuristics in a Benders decomposition for railway timetabling

Florin Leutwiler, Francesco Corman *

Institute for Transport Planning and Systems, ETH Zürich, 8092 Zürich, Switzerland

ARTICLE INFO

Keywords:
Timetabling
Combinatorial benders cut
Set covering
Railway

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 timetables should be designed taking into account service requirements that result in a performant railway system In this work, we address a specific subdomain of timetabling, focusing on short-term tactical changes for already existing timetables, modeled with microscopic detail. With a Benders decomposition we propose an approach on this specific microscopic timetabling problem. In the decomposition, we consider quality and optimality of a timetable separately from the feasibility of a timetable. Quality is determined in a set covering problem and feasibility in a mixed-integer scheduling problem. With efficient heuristics on the problem of set covering in our decomposition, high quality solution for the resulting timetables are provided in short time, which enables an interactive design of adapted timetables. The novel approach provides heuristic solutions up to ~ 20 times faster than standard approaches by commercial solvers, with an average gap of ~ 7.5% in the optimality of solutions. Extensive experiments empirically confirm the benefits of the new approach.

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 of 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 timetabling problem is possible, in comparison to the human planners of nowadays, such that novel timetables further increase the performance of railways. Recent advances in academia (Borndörfer et al., 2017) show promising results, leading to railway timetables of higher performance (e.g., 6% more punctual trains and 9% less trains with delays over 15 min compared to current practices in Lamorgese et al. (2016) for automated timetabling in real-time rescheduling).

In this paper we address a specific variant of the problem of railway timetabling, where short-term tactical changes are required, starting from an already established and planned timetable. Such short-term changes may be, e.g., extra passenger trains for large public events or freight trains for a temporary increase of freight capacity. When automating the solution of this tactical, short term problem, computational time should be within few minutes. Such a fast solution time enables a human-interactive design of the final timetable.

In the timetabling problem addressed in this paper, precise times of arrivals, departures and passings for railway services along multiple railway lines are to be determined. Differently from many approaches, we consider a tactical scale, by which an established reference timetable already exists, and a solution has to be found relatively quickly.

The input to our timetabling problem is thus a mix of information about planned trains in an existent reference timetable; and information about new trains to be additionally included in the actual timetable. The incremental scheduling problem is sometimes solved with the existing traffic being totally fixed (Cacchiani et al., 2010). Instead, our industrial partner considers the already scheduled traffic available for minor deviations, as follows. The input to our timetabling problem is just a rough timetable for the entire set of trains (existing and additional). The events related to service quality, e.g., arrival and departure times or connections, are specified with a limited specified amount of freedom, i.e., a limiting time window.

In such way and with the objective to minimize deviations from these limiting time windows we compute an extended timetable, similar to the original reference timetable, but with enough flexibility to

* Corresponding author.
E-mail address: francesco.corman@ivt.baug.ethz.ch (F. Corman).

https://doi.org/10.1016/j.cor.2023.106339
Received 20 September 2022; Received in revised form 19 June 2023; Accepted 24 June 2023
Available online 3 July 2023

efficiently incorporate any additional required services. A solution to our timetabling problem has microscopic detail and provides exact information on infrastructure usage (usage of routes, conflicts for limited infrastructure resources). Due to the tactical scope, a solution should be found relatively fast, because many other operational processes are directly depending on the result, which might be implemented just a few hours later. The result is a detailed timetable that is operationally feasible.

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 (Geoffrion, 1972) using the combinatorial Benders cuts of Codato and Fischetti (2006). In our Benders decomposition, the master problem is a set covering problem and the subproblem is a timetabling problem, where feasibility must be evaluated. We propose multiple near-optimal heuristic approaches for the master problem, i.e., a set covering problem, 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. Our novel approach proves to be an efficient complement and alternative to existing approaches, for the specific variant of the timetabling problem, computation requirements, and the instances considered. The key added value of the novel approach is the improved efficiency and thus scalability, contributing towards the gap between academia and large scale practical applications. Moreover, due to its fast computing time, the approach can effectively enable an interactive design of adapted timetables for planners in the railway industry.

This paper is structured as following. In Section 2 we review related literature and state the contribution of this work. In Section 3 we describe the specific timetabling problem we are dealing with, and we model it by a disjunctive formulation in Section 4. We propose a Benders decomposition for the timetabling problem in Section 5. In Section 6 we introduce multiple approaches to the set covering problem including a novel set of heuristics, designed for a set covering problem that is incrementally growing in constraints. Exhaustive experiments in Section 7 empirically confirm the strength of our novel heuristic approach compared to several existing benchmarks. We conclude in Section 8.

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.

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., Veelenturf et al. (2016) and Dollevoet et al. (2017)). 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., Corman et al. (2014), Pellegrini et al. (2015) and Samà et al. (2017)). 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., Caimi et al. (2012)), or in continuous form, i.e., by continuous variables (e.g., D'Ariano et al. (2007)).

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 via additional variables of the problem (e.g., Pellegrini et al. (2015)); models excluding routing decisions in general consider the routes of trains as a given input to the model (e.g., D'Ariano et al. (2007)).

Scheduling can be performed including constraints of periodicity (e.g., Odijk (1996)). 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 Cacchiani et al. (2012) or Fan et al. (2012). 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 Fan et al. (2012), Arc-Greedy Heuristics Pranzo et al. (2003)). Other heuristics in railway scheduling are applications of heuristics in general mathematical programming (e.g., Variable Neighborhood Search Samà et al., 2017, Tabu-Search Corman et al., 2010 or Genetic-Algorithms Fan et al., 2012). Further heuristics address elements of the scheduling problem in different stages, e.g., trains in the order of priority (Herrigel et al., 2013; Liu and Dessouky, 2017), or ordering decisions before routing decisions (D'Ariano and Pranzo, 2008).

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., Corman et al. (2014) and Lamorgese et al. (2016)) 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., Luan et al. (2018)), entity (resource) based (e.g., Caprara et al. (2006)) or generic decompositions based on properties of the underlying optimization problem (e.g., Lamorgese and Mannino (2019) and Keita et al. (2020)).

Generic decompositions (e.g., Lamorgese and Mannino (2019) and Keita et al. (2020) or D'Ariano and Pranzo (2008)) can be considered as applications of the general decomposition techniques from mixed-integer programming as discussed, e.g., in Wolsey and Vanderbeck (2010). 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 (Lamorgese and Mannino, 2019), or mixed-integer programming (Keita et al., 2020; D'Ariano and Pranzo, 2008). We consider the decomposition of this work as a generic decomposition.

2.3. Benders decomposition

Benders decomposition (Benders, 1962) 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 linear programming problem, and standard Benders cuts (Geoffrion, 1972) can be used.

In Codato and Fischetti (2006) 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.

2.4. Contribution

We consider in this work the railway timetabling problem through a microscopic, non-periodic model including the routing of trains, targeting the usage in a tactical stage. Our model of timetabling is relatively non-standard in the timetabling literature and is based on the model of Leutwiler and Corman (2022). The objective of the considered timetabling problem is the minimization of the delay of events, with regards to a latest preferred time given. We discretize this objective, in this work, 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 Leutwiler and Corman (2022). In the new decomposition, thanks to the discretization of the objective, we can apply the combinatorial Benders cut of Codato and Fischetti (2006). In our decomposition, the master problem is a set covering problem; 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 heuristic and optimal 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 a set covering problem and the subproblem is a timetabling problem, for which only feasibility must be evaluated. In the Benders decomposition we can apply the combinatorial Benders cut of Codato and Fischetti (2006).

(2) We introduce several heuristic approaches to solve the set covering problem 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. Instances of timetabling used for the experiments contain between 168 and 719 trains, and between ∼3000 and ∼31 000 discrete scheduling decisions. We put our approach into perspective with an existing general approach to timetabling (Fischetti and Monaci, 2017) using the two commercial solvers Gurobi (Gurobi Optimization, 2021) and Z3 (de Moura and Bjørner, 2008). For medium and small instances, the performance is comparable to the one of commercial solvers. For larger instances, we report a good scalability as we can solve instances of timetabling up to ∼20 times faster with only an average gap of ∼7.5% to an optimal solution.

3. Problem description

We address a problem of microscopic, non-periodic, railway timetabling to design a schedule (the timetable) for all operations performed by trains on the railway network. The problem is addressed at a tactical stage, where it is to update an existing reference timetable according to short-term demand changes, only few day before operation. We describe in detail the specific aspects of our problem, which extend the description of Leutwiler and Corman (2022); to which the reader is referred for more details. By microscopic detail, we refer to a description of railway services by set of operations, where each operation of a train corresponds to the train passing a single block section, i.e., block, in the railway network. In a solution of a timetabling problem, i.e., a timetable, each operation is scheduled by associating a time (in general, a continuous value) for the start and end *event* of the operation, i.e., the entry and exit of the train to the block.

The input of the timetabling problem specifies a set of tentative train services. While other approaches (see Cacchiani et al. (2010)) consider the existing traffic (i.e., traffic from the reference timetable) as fixed, or movable in terms of time, but fixed in its orders, we only care that the traffic already existing keeps the original functions. Those are encoded by the planners into a set of service requirements. The process by which those service requirements are generated goes beyond what we can share in a research paper, but includes the service contracts with the transport authorities, constraints due to technology, resources, infrastructures, considerations of travel chains, etc. In any case, the starting assumption is that the update of the reference timetable needs to be fulfilling those service requirements, and in case impossible, it can relax them.

The input considered is thus a set of services requirements, several for each train service, that are the results of political decisions and expected demand on the railway network. A service requirement manifests either as a limiting time window for a relevant event of the train service, e.g., the arrival or departure of a train, or as a constraint among two relevant events. In their sum, service requirements can describe a desired frequency for each service, assure synchronization of services for smooth passenger transfers and establish regularity (i.e., homogeneous spreading of services over time) in the timetable.

With service requirements that are time windows we can establish frequencies and regularity in the timetable. Time windows have usually sizes around ∼5 to ∼15 min. With service requirements that are constraints among two relevant events we can establish a required minimal (maximal) duration between an arrival of a feeder train and the departure of a connected train to assure enough time for passenger transfer; a minimum time between operations of trains on the same block; and minimum time for each operation (e.g., to allow alighting and boarding of passengers at stations).

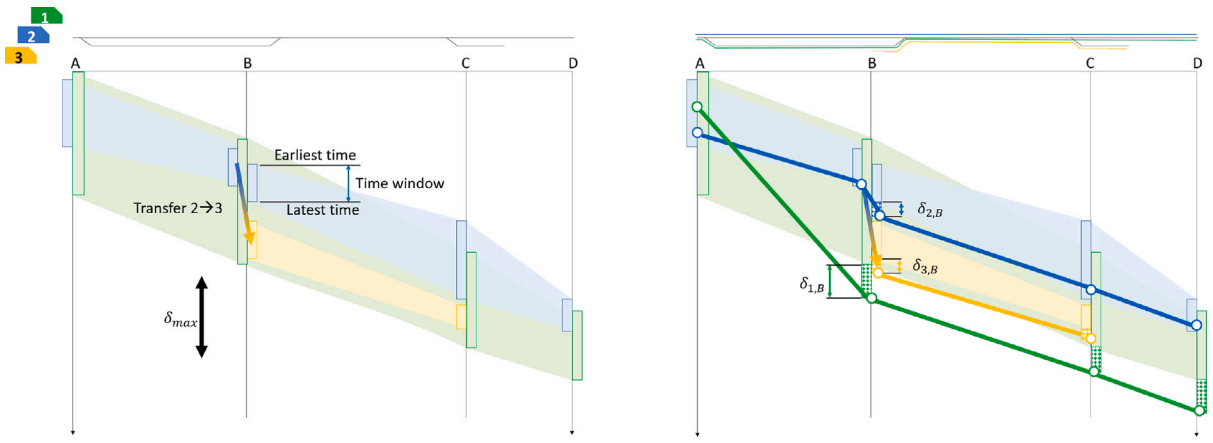


Fig. 1. Input (left) and Output (right) of the timetabling problem considered.

The solution of a timetabling problem, i.e., a timetable, is the determination, for all trains in the network, of an entry and exit for each block that is passed by the train; such times must satisfy all constraints of the problem. In the specific variant of timetabling we are dealing with in this paper (and differently from Leutwiler and Corman (2022)), we consider *planning deviations* when timetabling. By considering planning deviations, we enlarge the feasible solution space, and consider the possibility that an arrival, passing or departure event is scheduled actually shortly *after* the latest time from the time window of the service requirements. In this case, the planning deviation for such event is the difference in time between the latest time given for the event (i.e. upper bound of the time window), and the actual time the event is scheduled in the timetable. We consider planning deviation to be limited to a maximal allowed deviation δ_{max}, that is the same for all events and given by the railway operators. An introduction of planning deviation in a timetable might lead to a shift of services in time, which might lead to an effective shift in service frequencies from some moment in time, to some other moment. The amount of transfers to be offered by a timetable is not affected by those shifts as transfers are enforced by different constraints. Thus, even a timetable with planning deviations provides a sufficiently high quality of service to customers.

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 Fig. 1 (left) we depict graphically the input to our problem. Three trains (1, 2, 3) identified by the color green, blue and yellow respectively need to be scheduled, with their service requirements reported in a time (vertical, increasing downwards) and space (horizontal, from station A to B, C, and D) diagram. Train 1 goes from A to D without any planned stop; Train 2 goes from A to D with a stop in B (described by the two vertical boxes, the left for the arrival, the right for the departure). Train 3 runs only between B and C, and has a connection from train 2 at station B (reported as an arrow with color varying from blue to yellow). The infrastructure is shown on top of the plot, with a single track between station B and C. The time windows are reported at the 4 stations as long vertical boxes; and the region of time and space connecting them is shaded in the color of the train. The diagram reports also a δ_{max}, which is the maximum allowed deviation beyond the time windows. The figure shows how time windows may have varying size at different events of the same train, and may overlap between different trains.

A possible solution to this problem is shown in Fig. 1 (right). For each train a specific time is found, depicted by the colored node at each station, connected by a thick line. For each train, also a detailed route is found, depicted on the very top along the infrastructure. In the solution reported, train 1 (green) is overtaken by train 2 (blue) between station A and B; this is possible given the route of the two trains. Train 3 (yellow) departs from station B shortly after train 2; and train 1 runs between station B, C and D after all other trains. Many events are scheduled outside of the specific time window. Those are reported as long vertical boxes filled with a diamond pattern. At station B, all three trains are scheduled outside of their time windows, and the value for those three deviations is reported in Figure as $\delta_{2,B}$, $\delta_{3,B}$, $\delta_{1,B}$ respectively. At station C only train 1 and 3 have a planning deviation; and at station D, only train 1.

The planners might accept this solution as train 1 leaves station A very early, and leaves it available for other services; or not, because it has a relatively long travel time. In fact, many aspects (infrastructure capacity at tracks; at stations, availability of vehicle; inclusion of multiple travel chains of passengers) are at stake, and are modeled either explicitly in the objective function and constraints; or are left to human decisions and interactive solutions.

Whenever a timetable satisfies all given service requirements, we are assured such timetable provides a minimal required quality of service, sufficient for a practical application.

In summary, in the microscopic railway timetabling problem, 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.

4. A model for railway timetabling

We use a disjunctive formulation (Balas, 1998) to model our railway timetabling problem. In particular, we use the formulation of

Leutwiler and Corman (2022) and adapt it to the minimization of sum of total planning deviation. The following section summarizes the disjunctive model from Leutwiler and Corman (2022).

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 Leutwiler and Corman (2022) 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

$$\bigvee_{W_c \in D_l} \bigwedge_{((d,b),(q,p)) \in W_c} \left(t_{qp} - t_{db} \geqslant f_{db,qp}\right). \tag{1}$$

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.

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 $\delta_{db} \geqslant 0$. We can identify the planning deviation δ_{db} of an event (d,b) by the following precedence relation

$$\delta_{db} - t_{db} \geqslant -f_{db,.}. \tag{2}$$

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 railway timetabling problem as the disjunctive program,

$$\begin{aligned}
\min \quad & \sum_{(i) \in A_\delta} \delta_i \\
\text{s.t.} \quad & \delta_i - t_i \geqslant -f_{i,.} && (i) \in A_\delta \\
& t_j - t_i \geqslant f_{i,j} && (i,j) \in A_f \\
& \bigvee_{W_c \in D_l} \bigwedge_{(i,j) \in W_c} \left(t_j - t_i \geqslant f_{i,j}\right) && l \in L \\
& \delta_i \in \mathbb{R}_+,\ \delta_i \leqslant \delta_{max}\ \forall\, \delta_i, \quad t_i \in \mathbb{R}_+\ \forall\, t_i.
\end{aligned} \tag{3}$$

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 (3) is to minimize the sum of all planning deviations over all events given a latest time $f_{i,.}$.

4.2. A discretization of planning deviation

In Problem (3) planning deviations are represented by the continuous variables δ_i. To apply our approach and decompose Problem (3) into a set covering problem 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 set covering problem. 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., $\left\{\bar{\delta}_{i,0}, \dots, \bar{\delta}_{i,K}\right\}$, where $\bar{\delta}_{i,k} = k\frac{\delta_{max}}{K}$.

We then replace each precedence relation in A_δ of Problem (3) by a series of $K+1$ big-M constraints for all possible discrete values of δ_i, i.e.,

$$\bar{\delta}_{i,k} - t_i \geqslant -f_{i,.} - M x_{i,k}, \quad k \in \{0, \dots, K\}, \tag{4}$$

where $x_{i,k} \in \{0,1\}$. In the series of constraints (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 (4) for the simplicity of notation; in later experiments we will omit the big-M term of such constraint.

In the series of constraints (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 (4), it holds that if there exists any assignment on the binaries $x_{i,k}$ satisfying the series of constraints (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 (4) for the same values of t_i, where it holds $x_{i,k+1} \leqslant x_{i,k}\ \forall k \in \{0, \dots, K-1\}$. In other words if constraint (4) for k is satisfied by t_i (i.e., $x_{i,k} = 0$) any constraint (4) for p, where $p > k$ is trivially also satisfied, as the constraint of p relates to more deviation than the constraint of k. 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 (4) are relaxed as the corresponding $x_{i,k} = 1$. In case the constraints in the series (4) are relaxed up to constraint p, exactly $p+1$ constraints, i.e., $k \in \{0, \dots, p\}$, are relaxed and the planning deviation δ_i of event i is limited to,

$$\frac{\delta_{max}}{K} \sum_{k \in \{0,\dots,p\}} x_{i,k} \geqslant \delta_i. \tag{5}$$

In Eq. (5) we can extend the sum to all binaries, i.e., $\{0, \dots, K\}$, as binaries $x_{i,k} = 0$ for $k > p$.

In the above, we chose big-M constraints in (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 Codato and Fischetti (2006) 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), we can translate Problem (3) into a microscopic timetabling problem with discrete planning deviation,

$$\begin{aligned}
\min \quad & \frac{\delta_{max}}{K} \sum_{(i)\in A_\delta} \left(\sum_{k\in\{0,\dots,K\}} x_{i,k} \right) \\
\text{s.t.} \quad & \bar{\delta}_{i,k} - t_i \geqslant -f_{i,\cdot} - M x_{i,k} && k \in \{0,\dots,K\},\ (i)\in A_\delta \\
& t_j - t_i \geqslant f_{i,j} && (i,j)\in A_f \\
& \bigvee_{W_c\in D_l} \bigwedge_{(i,j)\in W_c} \left(t_j - t_i \geqslant f_{i,j}\right) && l \in L \\
& t_i \in \mathbb{R}_+ \,\forall t_i, \quad x_{i,k} \in \{0,1\}\ \forall x_{i,k}, \quad x_{i,K} = 0
\end{aligned} \tag{6}$$

where the objective is now to minimize the total sum of discretized planning deviations. If we consider continuous planning deviations in Problem (3), it holds for every optimal solution, that for each δ_i either the related constraint in A_δ is tight, i.e., $\delta_i = t_i - f_{i,\cdot}$, or $\delta_i = 0$. That is, either there is no planning deviation or exactly as much deviation considered in the objective of Problem (3) as the event is actually scheduled after its latest time.

For Problem (6), instead, planning deviations are discrete. In this case, it is possible that $\bar{\delta}_{i,k} > t_i - f_{i,\cdot} > \bar{\delta}_{i,k-1}$ and there is more planning deviation considered in the objective of Problem (6) than the event is actually scheduled after its latest time. This is simply due to the discretization of planning deviation.

5. Set covering in railway timetabling

We propose in this work a novel approach for railway timetabling where we decompose Problem (6) by a Benders decomposition into a set covering problem as the master problem, and a timetabling problem, where only feasibility is to be verified, as the subproblem. By the discretization of planning deviations in Section 4.2 we are able to apply the combinatorial Benders cut of Codato and Fischetti (2006) to our decomposition.

5.1. A combinatorial benders decomposition

We decompose our *centralized* problem $\mathcal{C}$ (Problem (6)) into a *master* problem $\mathcal{M}$ and a single *subproblem* $\mathcal{S}$ according to standard Benders decomposition (Geoffrion, 1972). We define the master problem to be the optimization over all binary variables $x_{i,k}$ of the Problem (6). We optimize all remaining variables in the subproblem. All constraints of Problem (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

$$\begin{aligned}
\min \quad & \sum_{(i)\in A_\delta} \left(\sum_{k\in\{0,\dots,K\}} x_{i,k} \right) \\
\text{s.t.} \quad & \beta && \beta \in \mathcal{B}^\alpha \\
& x_{i,k} \in \{0,1\}\ \forall x_{i,k}
\end{aligned} \tag{7}$$

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 5.2 we will show that β has the shape of the combinatorial Benders cut (Codato and Fischetti, 2006). We further omit the factor $\frac{\delta_{max}}{K}$ from the objective of Problem (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,

$$\begin{aligned}
\min \quad & 0 \\
\text{s.t.} \quad & \bar{\delta}_{i,k} - t_i \geqslant -f_{i,\cdot} && (i,k)\in \bar{A}^\alpha_\delta \\
& t_j - t_i \geqslant f_{i,j} && (i,j)\in A_f \\
& \bigvee_{W_c\in D_l} \bigwedge_{(i,j)\in W_c} \left(t_j - t_i \geqslant f_{i,j}\right) && l \in L \\
& t_i \in \mathbb{R}_+ \,\forall t_i
\end{aligned} \tag{8}$$

where $\bar{A}^\alpha_\delta := \left\{(i,k)\in A_\delta \times \{0,\dots,K\} \mid \bar{x}^\alpha_{i,k} = 0\right\}$ is the set of all constraints in the series (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 (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 $\mathcal{M}^\alpha$.

5.2. A combinatorial benders cut for railway scheduling

In Codato and Fischetti (2006) the authors introduce the combinatorial Benders cuts for a Benders decomposition that is structurally identical to our decomposition in Section 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 Codato and Fischetti (2006) for our Benders decomposition of Section 5.1.

The combinatorial Benders cut is generated from a set of constraints that together determines the infeasibility of the subproblem (we call it for simplicity an *infeasible subset of constraints*). We denote such subset further by $\mathcal{I}^\alpha$. Different from Codato and Fischetti (2006) our subproblem, i.e., Problem (8), is a mixed-integer linear programming problem. We therefore cannot use the techniques of Codato and Fischetti (2006) to determine $\mathcal{I}^\alpha$ as they rely on strong duality in linear programming. Instead, we propose in Section 6.1 to use an existing algorithm, that we designed in a previous work for a geographic logic-based Benders decomposition (Leutwiler and Corman, 2022).

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 $x_{i,k}$ and we can write the combinatorial Benders cut β of Codato and Fischetti (2006) for our decomposition of Section 5.1 as

$$\beta^\alpha := \sum_{(i,k)\in\mathcal{I}^\alpha_\delta} x_{i,k} \geqslant 1. \tag{9}$$

The Benders cut (9) is clearly a constraint of set covering and thus $\mathcal{M}^\alpha$ a set covering problem.

6. Implementation

Algorithm 1 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 6.1. With the decomposition of our timetabling problem and the combinatorial Benders cut from Section 5, we discuss in this Section first an algorithm from one of our previous works (Leutwiler and Corman, 2022), that can be used to analyze the subproblem; and later propose several approaches in Sections 6.2 and 6.3 to address the master problem in our decomposition.

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 1: Benders Decomposition Scheme

input : $\mathcal{M}$, S
output: $\mathcal{O}_C$
init : $\alpha = 0$, $\mathcal{O}_C = \emptyset$, $\mathcal{M}^\alpha = \mathcal{M}(\mathcal{B}^\alpha = \emptyset)$

1 **while** $\mathcal{O}_C = \emptyset$ **do**
2 $\quad \mathcal{O}^\alpha_{\mathcal{M}} \leftarrow$ `Solve`$(\mathcal{M}^\alpha)$;
3 $\quad \mathcal{O}^\alpha_{S}, \mathcal{I}^\alpha \leftarrow$ `Analyze`(S^α);
4 $\quad$ **if** $\mathcal{I}^\alpha_k \neq \emptyset$ **then**
5 $\quad\quad \mathcal{B}^{\alpha+1} \leftarrow \mathcal{B}^\alpha \cup \beta^\alpha(\mathcal{I}^\alpha)$;
6 $\quad$ **else**
7 $\quad\quad \mathcal{O}_C \leftarrow \mathcal{O}^\alpha_{\mathcal{M}} \cup \mathcal{O}^\alpha_{S}$
8 $\quad \alpha \leftarrow \alpha + 1$;
9 **return** $\mathcal{O}_C$

Algorithm 2: SMT, A DPLL Algorithm with Precedence Constraints.

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

1 **while** true **do**
2 $\quad$ confl $\leftarrow$ `UnitPropagation`(Φ, θ)
3 $\quad$ **if** !confl **then**
4 $\quad\quad$ confl $\leftarrow$ `Evaluate`$(G^\alpha(\theta))$
5 $\quad$ **if** !confl **then**
6 $\quad\quad$ **if** $\theta =$ *Complete* **then**
7 $\quad\quad\quad \mathcal{O}^\alpha \leftarrow G^\alpha(\theta)$
8 $\quad\quad\quad$ **return** $(\mathcal{O}^\alpha, \emptyset)$
9 $\quad\quad \theta \leftarrow \theta \bigcup$ `Decide()`
10 $\quad$ **else**
11 $\quad\quad$ **if** confl $=$ *Unsatisfiable* **then**
12 $\quad\quad\quad \mathcal{I}^\alpha \leftarrow$ `AnalyzeIP`$($confl$)$
13 $\quad\quad\quad$ **return** $(\emptyset, \mathcal{I}^\alpha)$
14 $\quad\quad$ **else**
15 $\quad\quad\quad$ `Analyze`$($confl$)$
16 $\quad\quad\quad$ `Backtrack`$($confl$)$

(Algorithm 2) of Leutwiler and Corman (2022). Algorithm 2 computes an infeasibility proof for a problem that is identical by the type of constraints to our Subproblem (8). The infeasibility proof returned by Algorithm 2 is a set of cycles on a graphical representation G^α of S^α. The set of 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 (9). In case Algorithm 2 cannot find a proof of infeasibility, S^α is feasible and a corresponding solution $\mathcal{O}^\alpha_S$ is returned. In the following we provide a brief summary of Algorithm SMT (Algorithm 2) of Leutwiler and Corman (2022):

Algorithm 2 is a combination of Boolean Satisfiability Solving (SAT) (Davis et al., 1962) with the logic of difference constraints from Cotton and Maler (2006); 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 S^α, i.e., $\exists W_c \in D_l$ s.t. $W_c \subseteq \theta$ $\forall l \in L$ and for which there exists an assignment for the variables t_i of S^α satisfying the precedence relations $A_f \cup \bar{A}^\alpha_\delta \cup \theta$. If such θ can be found, feasibility of S^α is proven. In Leutwiler and Corman (2022), 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.

In Algorithm 2 an initially empty set θ is iteratively extended by choice sets W_c from the decisions of S^α until θ satisfies all disjunctions (decisions) of S^α. The set of constraints Φ in Algorithm 2 models the disjunctions of S^α in terms of Boolean satisfiability constraints to assure that Algorithm 2 indeed computes a set θ, which satisfies all disjunctions of S^α. 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 S^α based on the violated constraint. Finally in case neither Line 2 or 4 returns a violated constraint and θ satisfies all disjunctions of S^α, 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 S^α.

6.2. Classical approaches for set covering in the master problem

In standard Benders decomposition (Geoffrion, 1972), 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 (Gurobi Optimization, 2021). In this case, i.e., if the master problem is solved to optimality, Algorithm 1 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 (Poojari and Beasley, 2009) or solutions of a relaxed master problem (Maher, 2021) are used in decompositions of the literature. In the same manner, we can address the master problem (Problem (7)) in our decomposition by a heuristic approach. In the particular case of this paper, the master problem is a set covering problem and we propose a decomposition approach, where we use the known best possible heuristic for problems of set covering, i.e., the heuristic of Chvatal (Chvatal, 1979), 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 1. 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 1.

In experiments of Section 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.

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 $x_{i,k}$, $(i,k) \in \mathcal{I}_{\delta}^{\alpha}$ 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 $x_{i,k}$, $(i,k) \in \mathcal{I}_{\delta}^{\alpha}$ have altogether value 0 in $\mathcal{O}^{\alpha-1}$; otherwise related constraints would not appear in $\mathcal{I}_{\delta}^{\alpha}$ as they would not be imposed to $\mathcal{S}^{\alpha}$. We propose in the following four different heuristics to determine which binary(ies) $x_{i,k}$, $(i,k) \in \mathcal{I}_{\delta}^{\alpha}$ 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 $x_{i,k}$, $(i,k) \in \mathcal{I}_{\delta}^{\alpha}$, 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 $x_{i,k}$, $(i,k) \in \mathcal{I}_{\delta}^{\alpha}$, 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 $x_{i,k}$, $(i,k) \in \mathcal{I}_{\delta}^{\alpha}$ 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 $x_{i,k}$, $(i,k) \in \mathcal{I}_{\delta}^{\alpha}$ and set equal to 1. Such heuristic shall provide a benchmark for the numerical experiments in Section 7.

In Section 7 we provide exhaustive experiment on all incremental heuristics of this section and put those in relation to classical approaches in Benders decomposition (Section 6.2) and further existing approaches for railway timetabling.

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 h per experiment.

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 h. We name the scenarios of timetabling according to the biggest cities within the corresponding geographic excerpt; names of cities are decoded in Table 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 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.

Table 1
Decoding of scenario names to the cities of Switzerland.

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

In Table 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. Fig. 2 gives an overview over the most important characteristics of the 14 original scenarios of timetabling. In the upper plot of Fig. 2, 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 Fig. 2 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 low 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 effort, necessary to solve the scenario. We can also see in Fig. 2 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.

All results considered in this work are validated for practical applicability through a tool that was provided by the Swiss Federal Railways. All reported results (timetables) are guaranteed to satisfy all service requirements and be of value for a practical application.

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 Fischetti and Monaci (2017). In Fischetti and Monaci (2017), the authors propose several improvements in the formulation of railway scheduling to increase the computational performance of a commercial solver applied to a railway rescheduling problem. 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 Fischetti and Monaci (2017), 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 deviation as done in Fischetti and Monaci (2017). 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 railway timetabling problem with a continuous objective (Problem (3)) as well as with a discrete objective (Problem (6)). We use for the benchmarks the commercial mixed-integer solver Gurobi (Gurobi Optimization, 2021) and the Boolean satisfiability solver Z3 (de Moura and Bjørner, 2008). Z3, likewise to our Algorithm 2, is a SAT solver extended to various types of constraints (including precedence relations).

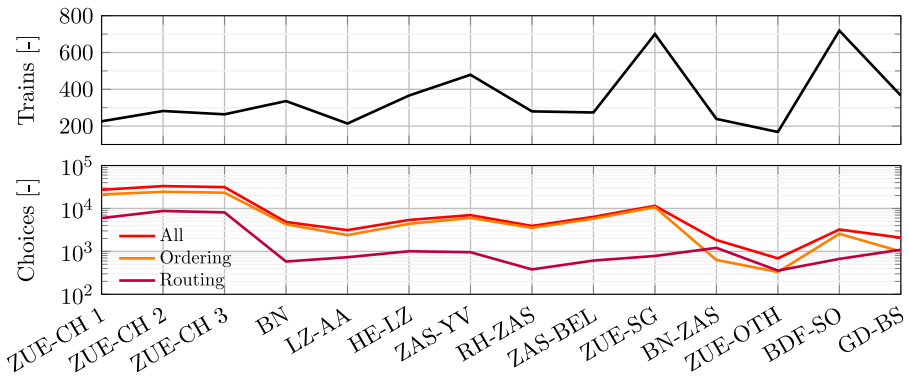


Fig. 2. Trains and choices in scenarios of timetabling.

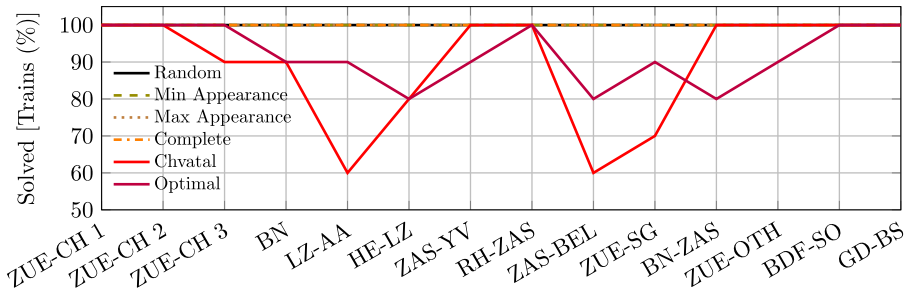


Fig. 3. Instances solved by set covering approaches.

7.3. Set covering heuristics

In a first series of experiments we study the performance of our classical (Section 6.2) and incremental (Section 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 h 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 Fig. 3 the percentage of trains up to which the individual approaches were able to provide a solutions within the limit of 24 h. In Fig. 3 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 fewer 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 Fig. 6 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 h. 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 Fig. 4 we report the computational time of different set covering approaches. In the top plot of Fig. 4 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 Fig. 4 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 set covering constraints in the Benders decomposition scheme (see Fig. 6), 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 Fig. 4 and thus a strong exponential increase of computational time for higher percentages of trains. In Fig. 4, the Optimal set covering approach scales similar to the other set covering heuristics, which is to some extent a falsified image of the reality. Fig. 4 reports only instances that are solved by all set covering heuristics; as shown in Fig. 3 this excludes in total 15 of the 140 generated instances of timetabling. For 11 of those 15 instances, also the Optimal set covering approach reached the time limit. Accounting for these would clearly reveal a worse scalability of the Optimal approach than actually reported in Fig. 4 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, Fig. 4 does not report a particular superior performance of the Complete Satisfaction heuristic in comparison to other heuristics; only in Fig. 6 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 2 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

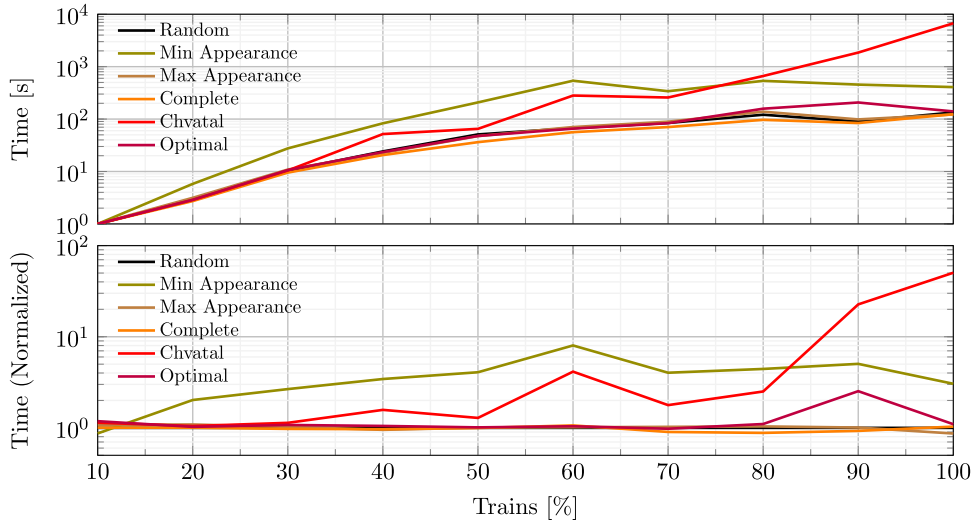


Fig. 4. Average computation time of set covering over 14 scenarios.

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 Fig. 5 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 (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 Fig. 4, we report in Fig. 5 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 behavior.

Most striking in Fig. 5 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. Fig. 5 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 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 Fig. 5 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 Fig. 5 shows that the Chvatal heuristic performs very similar to 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 ∼7%, minimum ∼5%. Overall Fig. 5 empirically underlines, that we are able to provide solutions of good quality, i.e., gaps to an optimal solution of ∼7%, in short time, with heuristic approaches of set covering.

Finally, we report in Fig. 6 statistics on the set covering problem in solves of set covering approaches. In the top plot of Fig. 6, 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 (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. Fig. 6 provides an empirical explanation to the computational performance of the set covering approaches as reported in Figs. 3–5. In the top plot of Fig. 6, 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 within 24 h. 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 Fig. 3 to be the computational burden of solving the master to optimality in every iteration of the Benders scheme. We saw in Fig. 3 that incremental heuristics, different than the Chvatal heuristic, where able to solve all 140 instances. We explain such result by Fig. 6, as we see a less drastic increase of constraints (iterations) on higher percentages of trains for incremental heuristics 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 Codato and Fischetti (2006), 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 Fig. 6 that for heuristic approaches, not the strength of constraints seems to be the dominant factor for

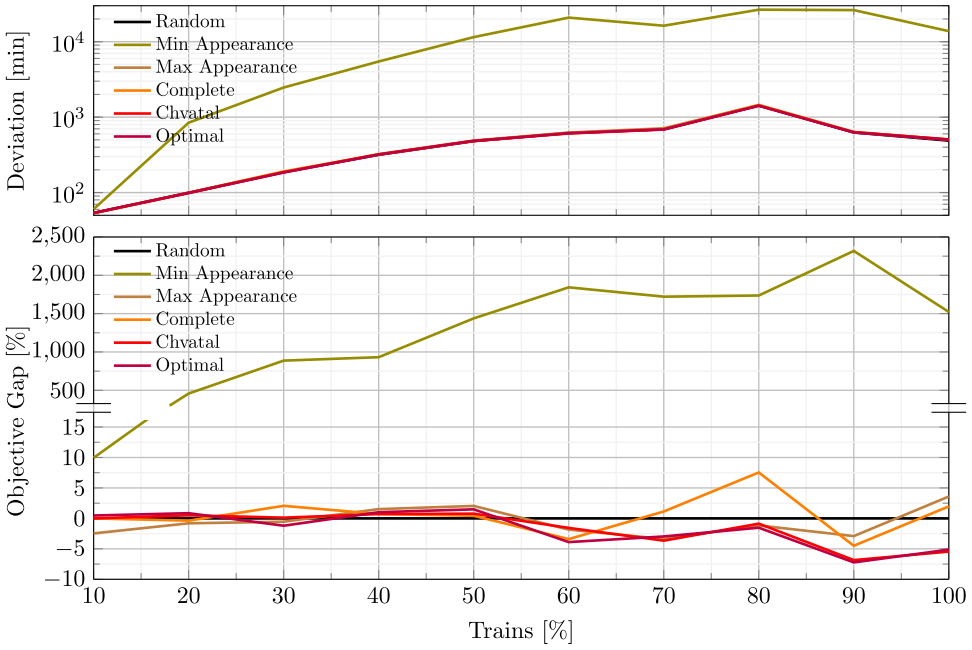


Fig. 5. Average objective of set covering over 14 scenarios.

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 Figs. 5 and 6 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 Fig. 6 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 subset will contain constraints of planning deviation (4) related to more than one train, in particular all trains involved in the decision.

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 7.2.

Likewise to approaches in set covering, due to the limitation of computational time to 24 h, benchmark approaches could not provide results for all instances within the given time. In the style of Fig. 3 for set covering, we report in Fig. 7 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 timetabling problem with a continuous objective; we do not report such benchmark in Fig. 7. Also for the timetabling with discrete objective Z3 provides only few solutions, disclosing the struggle of Z3 with the timetabling problem 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 timetabling problem.

We can see in Fig. 7 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 (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 Fig. 4, we report in Fig. 8 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 Fig. 8 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 Fig. 8 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 Fig. 8 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 ∼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, at 100% of trains, where no optimal solution could be provided by Gurobi. These three instances propose a significant challenge for Gurobi due to their high amount of routing

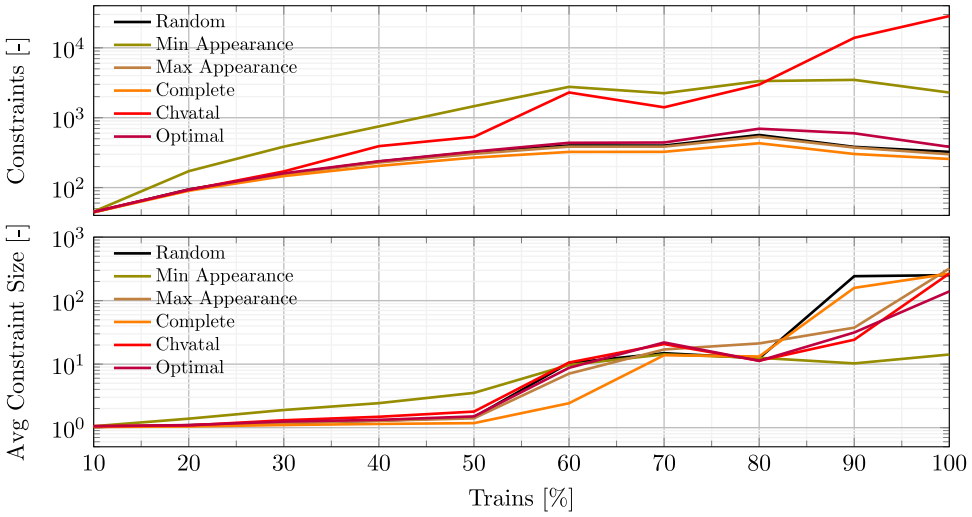


Fig. 6. Average problem size in set covering over 14 scenarios.

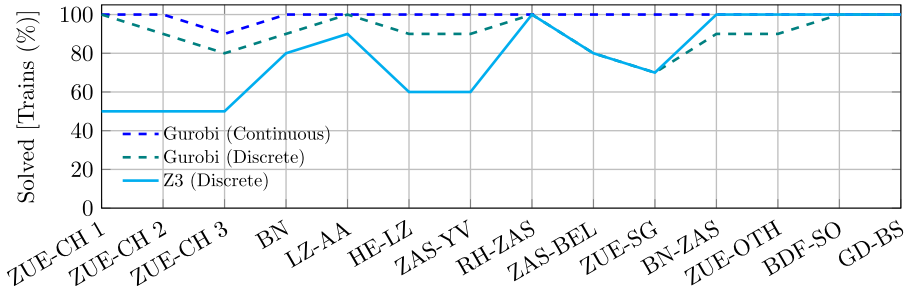


Fig. 7. Instances solved to optimality (Gurobi) or feasibility (Z3) by benchmark approaches.

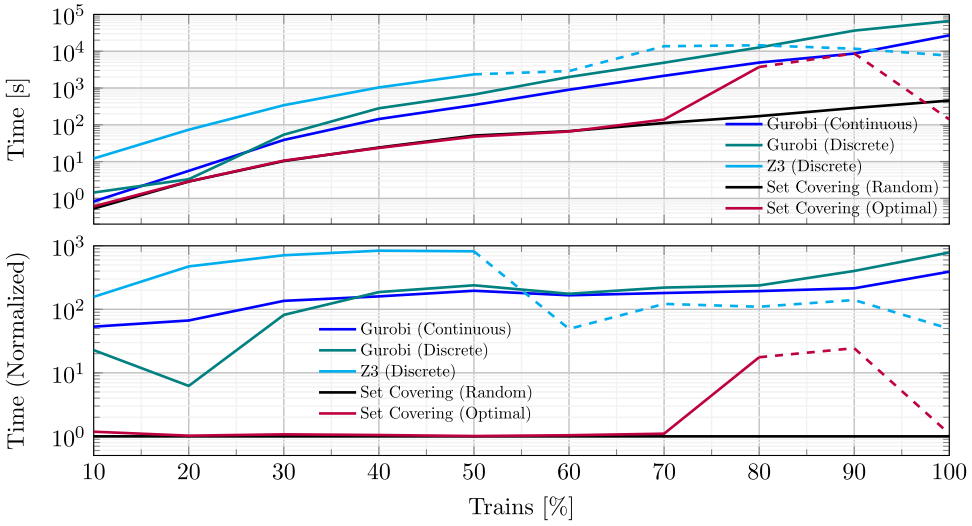


Fig. 8. Average computation time of benchmarks over 14 scenarios.

alternatives, while the instances are solved rather efficiently by the set covering heuristics. Ignoring completely these three instances, the average computational speedup for the 100% instances is still well above 50.

We can further see in Fig. 8 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 timetabling problem to optimality by our decomposition (set covering) approach, instead of directly applying a commercial solver. The bottom plot of Fig. 8 indicates a benefit of incremental heuristics over benchmarks regarding scalability. The normalized time of benchmarks (normalized by the Random heuristic) notably increases for high percentages of trains (above 70%), empirically underlining

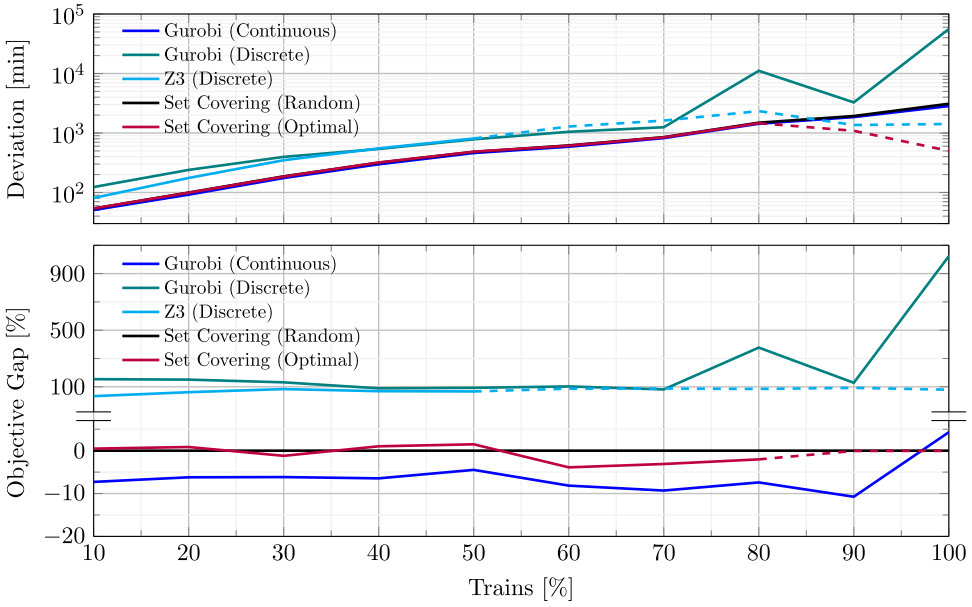


Fig. 9. Average objective of benchmarks over 14 scenarios.

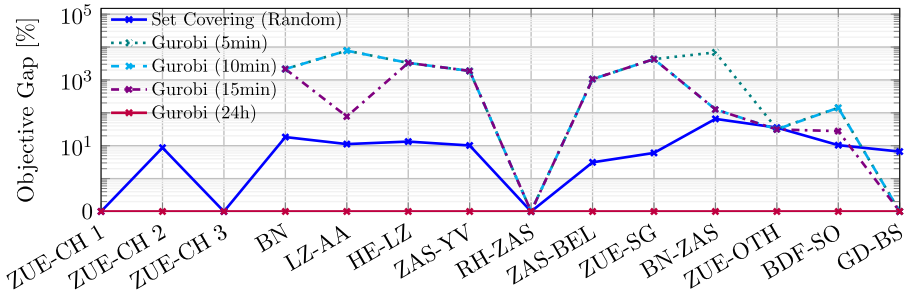


Fig. 10. Objective gaps for different time limitations on instances with 100% of trains.

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 Fig. 9 we report in the style of Fig. 5 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 Fig. 9, 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 Fig. 8, dashed lines indicate a reporting over an average over less than 14 scenarios. A negative objective gap in the bottom plot of Fig. 9 indicates a performance better than the Random heuristic in terms of objective value.

In Fig. 9, 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 2. In Algorithm 2, 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 Fig. 9 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 Fig. 9, we see that Gurobi (Continuous) computes on average timetable with ∼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 ∼4% more planning deviation than the Random heuristic. The on average higher planning deviation is caused by one instance where Gurobi was unable to provide an optimal solution within the time limit (see Fig. 7). If such instance is ignored, Gurobi (Continuous) shows an average better performance of ∼10% less planning deviation than the Random set covering heuristic. In terms of

Table 2
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	10 395	13 439	17 142	21 261
		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	12 006	15 161	18 974	24 447
		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	11 592	14 812	18 962	23 294
		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
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	10 709
		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

absolute planning deviation, 10% correspond to 310 additional minutes of planning deviation, which, considering an average of 350 train per instance, results in 0.9 min of additional planning deviation per train.

Finally, we study the performance of Gurobi in a setting where computational time is strictly limited; this would for example correspond to a human-interactive usage of Gurobi. We report in Fig. 10 the objective gap of Gurobi under different computational time limits, i.e., 5, 10 and 15 min, with respect to Gurobi at 24 h of computational time. We report further the same objective gap also for the Random set covering heuristic in Fig. 10. Experiments in Fig. 10 report only on instances with 100% of trains. In the earlier Fig. 9, we saw that given enough computational time, Gurobi was outperforming our heuristic in terms of objective value.

Fig. 10 clearly shows the opposite case, that the heuristics finds better solution than Gurobi, when this latter is subject to the strict time limitations matching the speed of the heuristic. Fig. 10 shows that for the first three instances, Gurobi was unable to provide any solution, within 15 min. For the remaining instances, the Random heuristic mostly provides better solutions, except for: instance RH-ZAS, where all approaches result in the same result; and instance GD-BS where all Gurobi approaches lead to the same result, that is ∼8% better than the results of the Random heuristic. Overall Fig. 10 underlines that our heuristics is able to find quickly good solutions, for the instances and time limit considered in the considered variant of the timetabling problem. The solutions found are much closer to the optimal solutions (computed without time limitations) than a time limited standard approach like Gurobi. The heuristic solutions are good enough and fast enough to allow implementation, and support a human-interactive application for timetabling.

8. Conclusion

In this paper, we introduce multiple approaches for solving a variant of the microscopic railway timetabling problem, which is solved to determine detailed solutions, starting from a given reference timetable, in a tactical scope. Based on a novel reformulation of the problem, 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 set covering problem as the master problem and a timetabling problem, 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 Codato and Fischetti (2006). 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 timetabling problem 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.

CRediT authorship contribution statement

Florin Leutwiler: Idea, Conceptualization, Implementation, Data analysis, Writing, Revision. **Francesco Corman:** Writing, Revision.

Data availability

The authors do not have permission to share data.

Acknowledgment

This work was supported by the SBB-ETH Mobility initiative of the ETH Zürich Foundation.

Appendix

See Table 2.

References

Balas, E., 1998. Disjunctive programming: Properties of the convex hull of feasible points. Discrete Appl. Math. 89 (1–3), 3–44.
Benders, J., 1962. Partitioning procedures for solving mixed-variables programming problems. Numer. Math. 4, 238–252.
Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M., Schlechte, T., 2017. Recent success stories on integrated optimization of railway systems. Transp. Res. C Emerg. Technol. 74, 196–211.
Cacchiani, V., Caprara, A., Fischetti, M., 2012. A lagrangian heuristic for robustness, with an application to train timetabling. Transp. Sci. 46 (1), 124–133.
Cacchiani, V., Caprara, A., Toth, P., 2010. Scheduling extra freight trains on railway networks. Transp. Res. B 44 (2), 215–231.
Caimi, G., Fuchsberger, M., Laumanns, M., Lüthi, M., 2012. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. Comput. Oper. Res. 39 (11), 2578–2593.
Caprara, A., Monaci, M., Toth, P., Guida, P.L., 2006. A Lagrangian heuristic algorithm for a real-world train timetabling problem. Discrete Appl. Math. 154 (5 SPEC. ISS.), 738–753.
Chvatal, V., 1979. A greedy heuristic for the set-covering problem. Math. Oper. Res. 4 (3), 233–235.
Codato, G., Fischetti, M., 2006. Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. Oper. Res. 54 (4), 756–766.
Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2010. A tabu search algorithm for rerouting trains during rail operations. Transp. Res. B Methodol. 44 (1), 175–192.
Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2014. Dispatching and coordination in multi-area railway traffic management. Comput. Oper. Res. 44, 146–160.
Cotton, S., Maler, O., 2006. Fast and flexible difference constraint propagation for DPLL(T). Lecture Notes in Comput. Sci. 4121 LNCS, 170–183.
D'Ariano, A., Pacciarelli, D., Pranzo, M., 2007. A branch and bound algorithm for scheduling trains in a railway network. European J. Oper. Res. 183 (2), 643–657.
D'Ariano, A., Pranzo, M., 2008. An Advanced Real-Time Train Dispatching System for Minimizing the Propagation of Delays in a Dispatching Area Under Severe Disturbances. Netw. Spat. Econ. 31 (6), 715–720.
Davis, M., Logemann, G., Loveland, D., 1962. A Machine Program for Theorem-Proving. Commun. ACM 5, 394–397.

Dollevoet, T., Huisman, D., Kroon, L.G., Veelenturf, L.P., Wagenaar, J.C., 2017. Application of an iterative framework for real-time railway rescheduling. Comput. Oper. Res. 78, 203–217.

Fan, B., Roberts, C., Weston, P., 2012. A comparison of algorithms for minimising delay costs in disturbed railway traffic scenarios. J. Rail Transp. Plan. Manag. 2 (1–2), 23–33.

Fischetti, M., Monaci, M., 2017. Using a general-purpose Mixed-Integer Linear Programming solver for the practical solution of real-time train rescheduling. European J. Oper. Res. 263 (1), 258–264.

Geoffrion, A., 1972. Generalized Benders Decomposition. J. Optim. Theory Appl. 10 (4), 237–260.

Gurobi Optimization, L., 2021. Gurobi optimizer reference manual.

Herrigel, S., Laumanns, M., Nash, A., Weidmann, U., 2013. Hierarchical Decomposition Methods for Periodic Railway Timetabling Problems. Transp. Res. Rec. J. Transp. Res. Board 2374 (1), 73–82.

Keita, K., Pellegrini, P., Rodriguez, J., 2020. A three-step Benders decomposition for the real-time Railway Traffic Management Problem. J. Rail Transp. Plan. Manag. 13 (July 2019), 100170.

Lamorgese, L., Mannino, C., 2019. A non-compact formulation for job-shop scheduling problems in traffic management.. Oper. Res. 67 (6), 1503–1782.

Lamorgese, L., Mannino, C., Piacentini, M., 2016. Optimal Train Dispatching by Benders'-Like Reformulation. Transp. Sci. 50 (3), 910–925.

Leutwiler, F., Corman, F., 2022. A logic-based benders decomposition for microscopic railway timetable planning. European J. Oper. Res..

Liu, L., Dessouky, M., 2017. A decomposition based hybrid heuristic algorithm for the joint passenger and freight train scheduling problem. Comput. Oper. Res. 87, 165–182.

Luan, X., Schutter, B.D., van den Boom, T., Corman, F., Lodewijks, G., 2018. Distributed optimization for real-time railway traffic management. IFAC-PapersOnLine 51 (9), 106–111.

Maher, S.J., 2021. Implementing the branch-and-cut approach for a general purpose Benders' decomposition framework. European J. Oper. Res. 290 (2), 479–498.

de Moura, L., Bjørner, N., 2008. Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 337–340.

Odijk, M.A., 1996. A constraint generation algorithm for the construction of periodic railway timetables. Transp. Res. B Methodol. 30 (6), 455–464.

Pellegrini, P., Marlière, G., Pesenti, R., Rodriguez, J., 2015. RECIFE-MILP: An Effective MILP-Based Heuristic for the Real-Time Railway Traffic Management Problem. IEEE Trans. Intell. Transp. Syst. 16 (5), 2609–2619.

Poojari, C.A., Beasley, J.E., 2009. Improving benders decomposition using a genetic algorithm. European J. Oper. Res. 199 (1), 89–97.

Pranzo, M., Meloni, C., Pacciarelli, D., 2003. A new class of greedy heuristics for job shop scheduling problems. Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 2647, 223–236.

Samà, M., D'Ariano, A., Corman, F., Pacciarelli, D., 2017. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. Comput. Oper. Res. 78, 480–499.

Veelenturf, L.P., Kidd, M.P., Cacchiani, V., Kroon, L.G., Toth, P., 2016. A railway timetable rescheduling approach for handling large-scale disruptions. Transp. Sci. 50 (3), 841–862.

Wolsey, L.A., Vanderbeck, F., 2010. Reformulation and decomposition of integer programs. In: 50 Years Integer Program. 1958-2008. Springer, pp. 431–502.