A decomposition approach for integrated locomotive scheduling and driver assignment in rail freight transport

Andreas Bärmann, Alexander Martin*, and Jonasz Staszek*

*Analytics and Optimization Lab

Technische Universität Nürnberg

Ulmenstr. 52, 90544 Nürnberg, Germany

{alexander.martin,jonasz.staszek}@utn.de, andreas.baermann@fau.de

July 2, 2024

Abstract

In this work, we consider the integrated problem of locomotive scheduling and driver assignment in rail freight companies. Our aim is to compute an optimal simultaneous assignment of locomotives and drivers to the trains listed in a given order-book. Mathematically, this leads to the combination of a set-packing problem with compatibility constraints and a multi-commodity-flow problem. We develop a binary-programming formulation to model the given task and improve it by performing a clique-based tightening of the original set-packing inequalities. The objective function of this model makes sure that as many trains as possible are running. To handle the computational complexity of the problem, we introduce a novel decomposition approach which decomposes the problem into a master locomotive scheduling problem and a subproblem for driver assignment. It exploits the fact that the master problem is empirically much easier to solve than the subproblem. For any fixed solution of the master problem, we can use the subproblem to either confirm feasibility of the master solution or to derive valid inequalities from various constraint classes to cut the infeasible master solution off and reiterate. To further improve solution times, we also develop a presolve heuristic. We demonstrate the potential of the presented method by solving a large-scale real-world problem instance provided by our industry partner DB Cargo Polska S.A., as well as a set of derived realistic instances.

Keywords: Integrated Locomotive Scheduling and Driver Assignment, Railway Transport, Integer Programming, Decomposition, Cutting Planes

Mathematics Subject Classification: 90C90 - 90C10 - 49M27 - 90C57

1 Introduction

Over the last few decades, the efficiency of optimization algorithms and the available computing power have grown so much that many real-world industrial planning problems which were once considered computationally intractable can now be solved very fast in practice. More and more often, this allows for the solution of large-scale resource planning tasks which previously had to be dealt with sequentially due to their complexity in an integrated fashion. Naturally, this leads to a much better exploitation of synergies between the different planning steps, as was asserted early on in the field (cf. e.g. Raff (1983)). This applies in particular to the planning of railway systems, where the intermediate planning phases are highly interdependent. For example, the mutual compatibilities of vehicles and drivers play a significant role for finding feasible assignments of staff and rolling stock to the trains in the schedule. The use of optimization techniques is critical to ensure the best possible utilization of the scarce resources (such as locomotives or the working time of drivers). While integrated planning of vehicle and crew has long been studied and successfully implemented in bus transportation and airways (see e.g. Huisman (2004) or Weide et al. (2010)), these results cannot be easily transferred to railway transportation due to its significantly more complicated nature – both legal and organizational. Hence, only a few first attempts have been made so far in assigning locomotives and their drivers jointly (cf. Dauzère-Pérès et al. (2015) and Aksoy and Altan (2013)).

In the present work, we develop a binary-programming formulation for the integrated locomotive scheduling and driver assignment problem. The objective function in this formulation ensures that as many trains as possible are running. We then derive a formulation strengthening technique which consists in performing a clique-based tightening of the original set-packing inequalities. To deal with the computational complexity of the problem, we decompose the problem into a master locomotive assignment problem and a subproblem for driver assignment. This is achieved by relaxing the constraints which ensure the mutual compatibility of drivers and locomotives. We exploit the fact that the master problem is empirically much easier to solve than the subproblem. Further, we introduce various classes of valid inequalities which can be seen as projections from the integrated problem onto the space of locomotive variables. For any fixed solution of the locomotive master problem, the driver subproblem either confirms its feasibility or computes such a valid inequality to cut the infeasible master solution off and reiterate. We also develop a presolve heuristic to reduce the solution times of the subproblem. Finally, we show how to extend our algorithm to an exact, globally optimal method by incorporating combinatorial Benders cuts.

This study is – to the best of our knowledge – the first one to consider locomotive scheduling and driver assignment in an integrated fashion. Concerning staff planning, we directly focus on driver assignment – we skip the intermediate step of driver scheduling considered by other authors. This means that instead of first coming up with a set of feasible driver shifts, the computed solution directly delivers an individual plan for each driver.

We also discuss the characteristics of both the rail freight transport and the problem we study which justify such an approach. We evaluate our method in a detailed real-world case study. As we will see, the efficiency of the chosen modelling and solution approach allows to solve the problem on a country-wide scale, computing a globally optimal monthly plan for a major player in Poland, whereas other studies typically consider shorter time spans or smaller geographical regions. The assignments generated could then be used as a "core" plan, which may be considered to be a starting point for the creation of a full plan in the company. Moreover, the algorithm delivers a monthly plan for both locomotives and drivers – covering nearly all of the trains – in less than two hours, which is more than sufficient for the use of our methods in practice. In addition, we also perform a number of additional experiments to "stress-test" the method at the hand of more demanding settings with scarce resources and show that it still delivers high-quality solution in comparatively short time.

This article consists of six sections. After this introduction, we discuss the relevant literature from the field of integrated vehicle and crew scheduling in Section 2, paying special attention to the works in the field of railway planning. Next, in Section 3 we introduce the underlying mathematical model (formulated as a combination of multi-commodity flow and set covering with compatibility, conflict and multiple-choice constraints) as well as the formulation-strengthening approach. Further, in Section 4 we present the decomposition-based solution approach described above. In Section 5, we show its effectiveness on problem instances provided by our industrial partner, DB Cargo Polska, including "stress-tests" in situations where resources are scarce. Finally, we conclude in Section 6, and give some directions for further research.

2 Literature overview

The topic of joint vehicle and crew scheduling in general has been treated quite extensively in the literature. However, the primary focus of the research works in this field lies on urban transportation and airlines. Only a few works considering joint vehicle and crew scheduling in the railway context can be found, although there is a long history of vehicle scheduling and crew scheduling approaches studied individually here. For an overview of the literature on railway vehicle scheduling, we refer the reader to Cordeau et al. (1998) and Piu and Speranza (2014). A very thorough study of the literature on railway crew scheduling has been written by Heil et al. (2020). In the following, we begin by discussing some approaches to duty scheduling, both as a stand-alone problem and in conjunction with driver scheduling. We also discuss the reasons which allow us to skip this planning step. Then we outline the literature on integrated vehicle and crew scheduling (which sometimes also includes crew assignment).

Duty scheduling approaches The duty scheduling is a complex step in the planning process in the railway industry, especially for rail passenger traffic. Unlike passenger trains, rail freight transportation is – to a significant extent – a last-minute business (see Jütte and Thonemann (2012)). The existing literature suggests that constantly changing order-books are normal to all rail freight carriers. For example, Heil et al. (2020) mentions a case of a large European freight railway carrier, for whom at most 80% of all trains in the order-book are regular trains, with the remaining 20% being added, cancelled or changed at short notice. Similar proportions are reported by our industrial partner. In some cases, the order-books are completely irregular – see Kumar et al. (2009). As a result, the standard operational planning horizon for rail freight carriers ranges between a day and a month. Overall, the discussed phenomenon of frequent changes in the order-book limits the usability of pre-planned rosters in the planning practice of rail freight carriers.

Nonetheless, a considerable amount of work which is significant to this article has been done in the field of duty scheduling. Some authors (Caprara et al. (2001); Goumopoulos and Housos (2004); Koniorczyk et al. (2015)) treat the duty generation step as a standalone problem. Others integrate it with crew rostering – for instance, see Borndörfer et al. (2017b). For approaches in freight transportation, we refer the reader to Dalal and Jensen (2001); Ernst et al. (2001); Guttkuhn et al. (2003); Khmeleva et al. (2014, 2018); Vaidyanathan et al. (2007); Vaidyanathan and Ahuja (2015).

Taking into account the highly unstable nature of order-books, the approach we introduce skips the step of duty generation. This way, the steps of duty scheduling and crew scheduling are merged into the crew assignment step we consider in this work. To incorporate the working time constraints, which are usually dealt with in the stage of duty generation, we consider the most popular types of such constraints (i.e. "short" breaks after each shift, "long" breaks once a week, non-working days on weekends etc.) in the step of driver assignment. Hence, the omission of duty generation causes no harm to the quality of driver rosters – they are compliant with the key working time regulations. We would also like to point out that there exist some complex rules (i.e. intra-shift break) which we leave of out the scope of our analysis, but which could – at least partly – be incorporated in our approach (a more detailed discussion will follow in Section 3).

Integrated vehicle and crew scheduling First, it can be noted that a vast majority of the works in this field studies bus or urban transportation systems. An excellent overview in this context was published by Ibarra-Rojas et al. (2015). Exemplarily, we mention the works of Haase et al. (2001), Freling et al. (2003), Huisman et al. (2005) Huisman (2004), Huisman and Wagelmans (2006), Mesquita and Paias (2008), Steinzen et al. (2007), Borndörfer et al. (2008), Laurent and Hao (2008), Amberg et al. (2011), Boyer et al. (2018), Amberg et al. (2019), Perumal et al. (2021)). Some works consider planning in airlines, see for example Cordeau et al. (2001b), Mercier et al. (2005), Mercier and Soumis (2007), Weide et al. (2010), Díaz-Ramírez et al. (2014), Dunbar et al. (2014). Other areas for which integrated vehicle and crew scheduling models were developed include postal delivery (Hollis et al. (2006)) and road transportation (Drexl et al. (2013)).

Following Ibarra-Rojas et al. (2015), we can divide the existing approaches to integrated vehicle and crew scheduling problem into exact and heuristic ones.

Exact approaches Most of the exact solution methods are based on column generation. It is frequently used together with Lagrangian heuristics to solve set partitioning/covering formulations of the problem, see e.g. Haase et al. (2001), Freling et al. (2003) or Huisman et al. (2005)). These approaches are able to supply good-quality solutions to relatively small instances within a three-hour period on a personal computer. Column generation is also used with a different, quasi-assignment model by Gaffi and Nonato (1999). A different approach was suggested by Borndörfer et al. (2008), who developed a Lagrangian relaxation method to solve the problem. Based on these works, Mesquita and Paias (2008) developed a solution incorporating column generation for a partitioning/covering model and dedicated branching rules. This allows them to solve larger instances and to reduce computation time compared to e.g. Huisman et al. (2005) or Borndörfer et al. (2008).

Heuristic approaches An important contribution to heuristic approaches to the integrated vehicle and driver scheduling problem was made by Huisman (2004). He proposes a kind of moving-horizon modelling which is solved via Lagrangian-relaxation-based heuristics, column generation as well branch and bound algorithms. Other heuristic approaches in the literature include Greedy Randomized Adaptive Search Procedures (GRASP) (Laurent and Hao (2008), De Leone et al. (2011)), evolutionary algorithms (Steinzen et al. (2007)) as well as local search algorithms (Valouxis and Housos (2002)).

We note that a considerable number of studies considers crews to be uniform or capable to work with any vehicle (e.g. Gaffi and Nonato (1999), Freling et al. (2003), Laurent and Hao (2008) or Perumal et al. (2021)), whereas the studies considering a limited crew-vehicle comparability are not as numerous (see e.g. Huisman (2004), Hollis et al. (2006) or Boyer et al. (2018)). Similarly, while some works focus on only one type of vehicles (such as Haase et al. (2001) or Mesquita and Paias (2008)), others include multiple types (see Gaffi and Nonato (1999), Cordeau et al. (2001a), Borndörfer et al. (2008) or Amberg et al. (2019)).

Robust approaches In the context of integrated vehicle and driver scheduling for airlines, robust optimization approaches play a significant role. For example, the works of Weide et al. (2010), Dück et al. (2012), Petersen et al. (2012) and Dunbar et al. (2014) present models which attain robustness against delay propagation in the schedule. While the methods developed in Dück et al. (2012), Petersen et al. (2012) and Dunbar et al. (2014) minimize the sum of propagated delays, Weide et al. (2010) focuses on cost minimization.

Approaches to integrated vehicle scheduling, crew scheduling and rostering There are also some studies which combine vehicle scheduling, driver scheduling and driver rostering. In Mesquita et al. (2013), the authors propose a formulation combining set covering, multi-commodity flow and covering-assignment, which is then solved by Benders decomposition. There are also some heuristic approaches to the combined problem, such as Shen and Xia (2009) based on local search or Mesquita et al. (2011) based on an iterative MIP (mixed-integer programming) heuristic. For the sake of completeness, we also mention that some works consider vehicle and crew routing as well as scheduling jointly – on top of assigning crews and vehicles, they also decide on the departure and arrival times of individual connections (see e.g. Lam et al. (2020)). We also mention the work of Borndörfer et al. (2017a), who show an example of planning mobile tours of toll inspectors on German motorways, for which they present an integrated vehicle routing and crew rostering model as well as a real-world case study.

Integrated vehicle and crew scheduling in railway planning In a thorough study of the available literature, we only found two works which attempt to solve the integrated vehicle and crew scheduling problem in the context of railway transport. To the best of our knowledge, the first work in this field is due to Aksoy and Altan (2013). Using a multi-commodity flow formulation with node demands, they optimize the flows of locomotives and crews between yards (to which trains are assigned). Their objective is then to ensure the required number of locomotives and crews are available at a yard at the beginning of a given day of a week while minimizing the costs of moving each without a train. However, their model is a very coarse representation of the necessities of the real-world planning process. It mainly captures the flow balance for drivers and locomotives between yards in order to perform a capacity planning matching the number of required drivers and locomotives on each day of the week. Their model does not include an assignment of concrete drivers and locomotives to the trains to be staffed. Further, it does not include many of the critical constraints for ensuring the feasibility of such an assignment, like the working time constraints of the drivers. The computational results they show only feature a trivial instance with 3 yards, with no mentioning where the data came from.

Compared to Aksoy and Altan (2013), the model we present includes a detailed representation of the standard operational requirements, including various locomotive types and non-uniform skills of the drivers. Moreover, we derive a novel, efficient algorithmic solution approach based on problem decomposition, cutting planes and a dedicated preprocessing. The quality of the resulting model and the performance of the dedicated solution algorithm are demonstrated at the hand of real-world, country-scale problem instances provided by DB Cargo Polska.

Our work is further related to that of Dauzère-Pérès et al. (2015), who focus on vehicle scheduling and crew scheduling in railway passenger traffic. Similar to us, they start from the relaxation of the coupling constraints between driver and locomotive assignment. Then they use a Lagrangian relaxation heuristic to obtain high-quality solutions. They also describe their computational and implementation experience gathered in collaboration with their industrial partner, the French national railway carrier SNCF. In comparison to Dauzère-Pérès et al. (2015), our case study is significantly more comprehensive in that it is based on country-wide data for a whole month, whereas they focus on a single region and a time horizon of one week.

Our contribution Most notably, we consider the *integrated vehicle scheduling and driver assignment problem* in the railway industry. This means, unlike the other two works discussed above, our model includes assigning concrete drivers to concrete trains to be staffed, including detailed working time requirements – which are not explicitly stated in Dauzère-Pérès et al. (2015). To achieve this goal, we propose an efficient decomposition-based solution approach, which takes advantage of custom valid inequalities (implemented as cutting planes) and a dedicated preprocessing. Further, our approach can solve instances which are about five times larger than those presented in Dauzère-Pérès et al. (2015).

The performance of our solution algorithm is demonstrated at the hand of real-world, country-scale problem instances provided by DB Cargo Polska.

3 Problem modelling

In the following, we will present our mathematical model for integrated locomotive scheduling and driver rostering. We start by introducing the modelling requirements and restrictions we took into account to construct a first, basic version of the model. Then a technique for improving the model formulation – clique tightening – will be discussed. We emphasize that the presented model accurately represents the real-world planning challenge as defined by our industrial partner, a major provider of rail freight traffic in Poland. At the same time, our modelling allows for the application of the approach in many different countries, not only Poland, via parameter choices or straightforward model extensions.

3.1 Overview of modelling requirements and assumptions

A key question for optimizing the use of locomotives and drivers is: how many of the planned trains in the order-book can be carried by compatible locomotives and served by drivers with appropriate licenses given the scarce resources at hand at a railway company? The model developed here is thus intended to find such an assignment of drivers and locomotives to the pre-scheduled trains, with the aim that as many of them as possible can run.

Our model considers three distinct blocks of requirements and restrictions; they are related to (i) drivers, (ii) locomotives and (iii) mutual compatibilities between locomotives, drivers and the trains. All necessary information about the trains to be performed is given in the so-called *order-book* – a data set containing details about the origin and destination station of each train, its planned departure and arrival times as well as locomotive and driver requirements. In our modelling, we include numerous classical working time constraints, encountered across numerous geographies. This includes maximum shift length, minimum break between shifts, a longer break once a week and a specified day off every few weeks. We also account for the fact that the drivers are usually assigned to a planned region and that some shifts need to start or end in that region. Additionally, to distinguish between individual weeks in longer time horizons, we will use the notion of *calculation weeks* – which are subsequent, non-overlapping planning periods, spanning seven days each. In this work, we will consider time frames spanning from one week to one month. More details about the individual instances considered are included in Section 5. Given the above information, we can use an indirect modelling of time and space to keep the size of the resulting model small. In particular, we will model time and space via mutual compatibilities and conflicts of individual trains.

In the next subsections, we will introduce the modelling requirements and restrictions we took into account when deriving the model presented later. They pertain to all the three constraint groups mentioned above. We will also discuss the modelling requirements and restrictions taken into account when deriving the objective function. The requirements stem from physical and legal conditions and hence must not be violated. The restrictions reflect the planning practice of our industrial partner; therefore, we provide a short rationale for each of them and also mention in which way they could be extended to consider other planning practices. Thus, although our model strongly reflects the tasks that are faced by planners in Poland, it is still a general approach which can be adapted to other regions via modifications of the assumptions stated below.

3.1.1 Driver restrictions and requirements

This subsection presents restrictions and requirements related to the planning of drivers. They comprise the standard types of driver-related limitations (e.g. working time, assignment to regions etc.) as well as the planning practice of the industrial partner.

Requirements One of the commonly encountered requirements which we include in our work prescribes that drivers work for some maximal number of hours. We model this maximal length of a shift with a parameter c^{shift}. It is also a standard requirement that the driver has to rest for at least a stipulated minimal time after each shift. Such a minimal rest time will be defined by a parameter c^{short}. We will call a break following each shift a *short break*. Yet another standard operational limitation consists in ensuring that each driver has at least one *long break*, defined as a longer period of uninterrupted rest, per calculation week. The duration of the long break is modelled by the parameter c^{long}. We also assume that the entire duration of the long break must be incorporated in full within one week. Another constraint which is frequently seen in the literature requires the driver to have a specific day of week off once every few weeks. In our approach, we assume that the day off is Sunday, although it could well be any other day of week. Moreover, the frequency in which the day off shall be granted will be modelled by the parameter c^{sunday}, which denotes the maximal number of working Sundays per month. Another common planning practise is to assume that every driver has a set of "licensed" train routes (e.g. origin-destination pairs). We may assign the driver only to trains traversing these routes. We also need to consider the fact that each driver is licensed to a limited number of locomotive classes, and only these locomotive classes can be assigned to the considered driver. We require that this be unconditionally respected. Table 1 presents a summary of the parameters used for modelling the working time constraints.

Name	Description
c^{shift}	Maximal duration of shift in hours
c^{short}	Minimal duration of short break in hours
c^{long}	Minimal duration of long break in hours
c^{sunday}	Maximal count of working Sundays per driver in a month

Table 1: Parameters for the working time constraints

Restrictions Our industrial partner assigns drivers to one of three planning regions, corresponding to a geographic subdivision of Poland into three regions – roughly south, west and east. We assume that we are free to schedule drivers to any train they are licensed to drive. However, per company directive a driver's first job in the planning period must start in the home region of the driver. Similarly, the drivers' last job in the planning period has to end in the respective home region to which they are assigned. As a consequence, we are free to let drivers rest in hotels if they end their work away from their home region. Based on real-world practice, we take it that sometimes drivers are brought to the first train in the shift by car. We also assume that drivers rest in the location of the destination station of the last train of their shift. In our modelling, we grant the drivers a short break plus transportation time if a transport is required. We could easily extend the model to accommodate to other rules regarding work location by adapting the availability of drivers to drive trains far away from their planning region.

As per the planning practise of our industrial partner, we do not consider intra-shift breaks in the following. We would like to point out that such breaks are required, among others, in Germany, Denmark and Sweden (cf. Bach et al. (2016)). Indeed, our approach can easily be generalized to account for at least some of such rules – for example by enforcing a break of fixed length after each served train. This could be achieved by a slight redefinition of one of the sets required for constraint construction (discussed later in this section).

It should also be mentioned that the constraints on the maximal working time per a given time unit are not included in our modelling. Such constraints are frequently seen in many EU countries. Although this is the case for Poland too, the maximal working time is calculated on a quarterly basis though rather than monthly or weekly. Since the maximal size of our instances is one month, as per the industrial partner's directive we decided against the inclusion of these considerations in our model. We would also like to point out to the fact, that such constraints could easily be integrated into our modelling.

We further do not consider the possibility to perform training rides (aiming at maintaining and extending the set of routes to which a driver is licensed) as this would result in an overmodelling – since the validity of a route in the set of "licensed" routes in Poland is 180 days, such decisions need to be made on a tactical or even strategic level, depending on the planned order portfolio. They could easily be integrated, however, by excluding the drivers from serving any trains during their training periods, and by directing appropriate locomotives to the corresponding stations (by fixing the relevant variables). A similar argument holds for extending the drivers' certification to a new locomotive type – given the price of new locomotives, such decisions are usually made on a strategic level, whereas we focus on the operational activities.

We assume that at most one driver can be staffed to a train. Unlike public transport and airline industry, this is standard operational practice at our industrial partner. Nevertheless, our model could be extended to allow for multiple drivers in one train by changing one constraint group from set packing to set cover. While we would gain another degree of freedom to transport crew via deadheading, this would result in a different problem structure and would require a separate study of polyhedral properties.

3.1.2 Locomotive restrictions and requirements

Next, we will discuss the restrictions and requirements related to the planning of locomotives. We include many of the constraints frequently encountered in this field as well as the particular planning practice of our industrial partner.

Requirements Concerning the locomotives, the basic requirements we consider comprise their tractive power and their source of energy (electricity or diesel). A locomotive needs to have sufficient horsepower to carry a given train, and the availability of catenaries needs to be taken into account. Further, unlike drivers, locomotives may only pick up trains in the location where their previous train has ended.

Restrictions For the sake of simplicity, we assume that we are free to select the starting location of each locomotive. It is, however, straightforward to consider the actual starting locations of the locomotives by restricting the respective set of initial trains they can cover. Additionally, we assume that locomotives only move when carrying a train. Note here that we do not plan any so-called *empty runs* of locomotives in our model (also known as *deadheading*), i.e. trips of locomotives without a train in order to place them where they are needed next. We instead assume that empty runs are pre-planned and thus already part of the order-book. We decided for such an approach to the inclusion of empty runs in our modelling in order to better reflect the planning practise of the industrial partner, even if this sometimes results in a slight less-than-100% coverage of the trains scheduled therein. While differentiating between "normal" trains and empty runs would be interesting, for the ease of exposition, we chose not to incorporate this distinction, as this would have required additional notation. On the other hand, it is straightforward to include it in the model. Additionally, as empty run generation would require us to add new connections to the existing order-book, the problem studied would fall into the realm of train scheduling problems. It would also considerably increase the scope of the planning challenge considered in the work. Hence we decided to remove it from the scope of the integrated locomotive scheduling and driver assignment problem studied here.

We further take it that only one locomotive is required to carry each train. This limitation is again drawn from the operational practice of our industrial partner and could be at least partially lifted by a slight change in the modelling approach. Further, since some trains in the instances supplied are carried by locomotives which are not the property of our industrial partner, we require that for each of these trains a "virtual" locomotive exists which is capable of carrying only that one train. For such trains, we only need to find a suitable driver.

Finally, it should be mentioned that the maintenance needs of locomotives are not included directly in our model, since planning the time and location of maintenance periods is largely a different planning problem. We should also note that maintenance constraints constitute a critical part of the planning chain, both for freight and passenger transportation, where the vehicles frequently do not return to the their depots for several weeks. However, we could easily accommodate for at least the basic, daily maintenance schedules by making sure that the locomotive stays at the maintenance station for a sufficiently long time between carrying trains.

3.1.3 Compatibilities of locomotives, drivers and trains

Further, we summarize the requirements and the restrictions imposed by our industry partner which are related to the mutual compatibilities between locomotives, drivers and trains.

Requirements We require that each train be served by a driver who is licensed to the route of the train and to the employed locomotive type as described in Subsection 3.1.1. We also enforce that a locomotive have enough horsepower and that it be operated via an appropriate source of energy, see Subsection 3.1.2.

Restrictions Normally, the planners make an indication of the locomotive type which shall serve each train already at the stage of constructing the order-book. We relax this assumption and allow that a train is carried by any locomotive having sufficient power and an appropriate energy source. Therefore, in many cases we can benefit from the ability to choose between more locomotive types than just the stipulated one.

3.1.4 Restrictions and requirements pertaining~~pertainting~~ to the objective function

Last, we summarize the requirements and the restrictions related to the formulation of the objective function.

Requirements The basic requirement in the objective function is that as many trains as possible are running. A train is running if and only if both a driver and a locomotive was assigned to it.

Restrictions In this work, we measure the quality of solutions only by the number of trains running. Such a decision is due to the unstable nature of rail freight, manifesting itself in the great variability of the order-book in time (see Section 2 for a more detailed discussion). This innate instability in the order-book, together with the shortage of both locomotives and train drivers in the entire European market, justifies the choice of the largest number of trains running as the measure of quality of the solutions returned by our algorithm. An interesting further research avenue would be to also include cost aspects, as well as driver satisfaction measures in the objective function, as many of these could easily be integrated into our modelling.

3.2 The optimization model

We now model the optimization problem described above as a combination of a set-packing problem with compatibility, conflict, and multiple-choice constraints and a multi-commodity-flow problem. Our objective is to maximize the number of trains performed, i.e. the number of trains for which both a locomotive and a driver can be found. The inputs to the model are a set T of trains to be performed, a set of locomotives $\mathscr{L}$ and a set of drivers D. Moreover, let us define the set S of stations which contains the origins and destinations of all trains. For each train $t \in T$, we consider its origin $o(t) \in S$, its arrival station $a(t) \in S$ as well as its departure time $s(t) \in \mathbb{R}$ and arrival time $e(t) \in \mathbb{R}$. Additionally, for each driver $d \in D$ let $H(d) \subset S$ denote the stations which belong to the home region of that driver. To denote the subsets of locomotives compatible with a driver $d \in D$ or a train $t \in T$, we use $\mathscr{L}^d$ and $\mathscr{L}^t$ respectively. Let D^l and D^t represent the sets of drivers compatible with a locomotive $l \in \mathscr{L}$ or a train $t \in T$ respectively. We also define T^l and T^d as the subsets of trains compatible with a locomotive $l \in \mathscr{L}$ or a driver $d \in D$ respectively. We notice that some of the locomotives $l \in \mathscr{L}$ have exactly the same sets of compatible trains T^l and compatible drivers D^l as other locomotives in $\mathscr{L}$. In order to make the locomotive part more compact, we partition $\mathscr{L}$ into subsets L in which all locomotives $l \in L$ have the same set of compatible trains T^l. We call this partition $\mathcal{L}$, and a subset $L \in \mathcal{L}$ is called a *locomotive class*. Further, we define $T^L \subseteq T$ as the set of trains compatible with a locomotive class $L \in \mathcal{L}$ and denote by $\mathcal{L}^t \subseteq \mathcal{L}$ the set of locomotive classes compatible with a given train $t \in T$. Additionally, $\mathcal{L}^d \subseteq \mathcal{L}$ shall denote the subset of locomotive classes compatible with a given driver $d \in D$, and D^L shall denote the set of drivers compatible with a locomotive class L. Finally, we use $W \subset \mathbb{N}$ to denote the set of calculation weeks.

Before building the model, we also perform an initial preprocessing of the drivers, locomotives and trains sets. If, for a train $t \in T$ and for a locomotive class $L \in \mathcal{L}^t$ no suitable driver can be found (or $D^t \cap D^L = \emptyset$), we remove the locomotive class L from $\mathcal{L}^t$. We also remove the train t from the set T^L. Similarly, if for a train $t \in T$ and for a driver $d \in D^t$, no suitable locomotive can be found (or $\mathcal{L}^t \cap \mathcal{L}^d = \emptyset$), we remove the driver d from D^t. We also remove the train t from the set T^d.

We also make a number of modeling assumptions, required for the construction of the optimization model. It is apparent that every driver shift is characterized by a distinct first and last train (job). Hence, as we assume that we may plan at most one driver to drive each train, any train $t \in T^d$ can be a first job or a last job in a shift for at most one driver $d \in D$. This assumption does not apply to the jobs which we later denote as the ones, whose arrival time indicates the beginning of the long break. For these we allow that the arrival time of one train determines the beginning time of the long break for an arbitrary number of drivers. Such a deviation is due to a different nature of the long break – rather than at the end of each shift, it needs to be fully included in the respective calculation week. Thanks to this assumption, we ensure that a long break can be planned for all the drivers $d \in D$. This assumption is not equivalent to allowing multiple drivers to drive the same train – in our model, trains (and in particular their arrival times) serve as a proxy for modeling time directly, and nothing precludes multiple drivers from starting their long breaks at the same time. Technically, deciding when to start a driver's long break is modeled by a different set of variables than assigning a driver to drive a train. Moreover, we do not allow the long break to take place between two calculation weeks. We also do not admit moving the long break to another calculation week – in other words, the long break must be fully included in the calculation week it pertains to. Hence, only a subset of trains within the calculation week may serve as an indicator of the beginning of the long break. For extreme cases (e.g. few trains over a long time period), this might result in infeasible assignments – namely because there are not enough "long-break beginnings" for all drivers. However, this usually does not occur in practice. We also allow the long break to actually be longer than c^{long} hours to the benefit of the drivers.

To facilitate the understanding of this long-break requirement, let us consider an example. Assume that each calculation week starts on a Monday at 0:00 and ends on the subsequent Sunday at 23:59. We also assume that the long break needs to be 48 hours long (or $c^{\text{long}} = 48$). Then a train t_1 departing on Monday at 13:00 and arriving at 15:00 is a feasible beginning of the long break, as the break would last from Monday 15:00 to Wednesday 15:00 and so would be fully included in the calculation week. A different train, t_2, which departs on Friday at 23:30 and arrives on Saturday at 1:30, is not a feasible long break beginning, since the corresponding long break would not be fully included in the same calculation week. In particular, it would end on the subsequent Monday at 1:30 – a time-point which belongs to the subsequent calculation week.

Sets and parameters required for constraint construction We also introduce a number of sets required to build the constraints of the model. They represent the relationships between trains to incorporate the assumptions and requirements discussed in Subsection 3.1. For example, they list the trains which run simultaneously to another one, or trains which could be served successively. In Table 1, we had already introduced parameters which allow for the modification of sets and constraints pertaining to the working time regulations. This way, our approach is general and hence it may easily be applied for various companies in different countries.

Table 2 presents a summary of the sets required for constraint building. Their exact definitions can be found in Appendix A. For ease of notation, we write $t_1 \leq t_2$ for two trains $t_1, t_2 \in T$ if t_2 departs at the same point in time or later than t_1.

Name	Description
$T^{\text{B+}}_{t,d}$	Trains unassignable to driver d if train t is the last job in a shift
$T^{\text{LB+}}_{t,d}$	Trains unassignable to driver d if t is their last job before a long break
$T^{\text{B-}}_{t,d}$	Trains unassignable to driver d if t is their first job in a shift
$T^{\text{week\_assignment}}_{w,d}$	Trains which belong to calculation week w for a driver d
$T^{\text{week}}_{w,d}$	Trains which belong to calculation week w and are suitable as the last ones before a long break for a driver d
$T^{\text{sunday}}_{w,d}$	Trains which belong to the Sunday in calculation week w for a driver d
$T^{\text{shift\_beginning}}_{t,d}$	Preceding trains assignable to driver d if he is assigned to train t
$T^{\text{shift\_end}}_{t,d}$	Future trains assignable to driver d if he is assigned to train t
$T^{\text{time}}_{t,d}$	Trains which are feasible for driver d and in time conflict with train t
$T^{\text{time\_global}}_{t}$	Trains which are in time conflict with the train $t \in T$
$T^{\text{after\_break}}_{t,d}$	Trains which can be the first job of the next shift of driver d after his short break if t is the last train in his previous shift
$T^{\text{before\_break}}_{t,d}$	Trains which can be the last job of the previous shift of driver d before the short break if t is the first train in his next shift
$T^{\text{next}}_{t,L}$	Future trains assignable to locomotive class L if it is assigned to train t

Table 2: Descriptions of the sets required for constraint construction

Multi-commodity flow part of the model – the locomotive assignment We consider the set $\mathcal{L}$ of locomotive classes as commodities which need to be "routed" through a directed graph $\mathscr{G} = (\mathscr{V}, \mathscr{A})$, defined via

$$\mathscr{V} := T \cup \{\Sigma, \Theta\}$$

and

$$\mathscr{A} := \{(t_1, t_2) \mid t_1 \in T^L \quad \wedge \quad t_2 \in T^{\text{next}}_{t_1,L} \quad \forall L \in \mathcal{L}\}.$$

with $\Sigma, \Theta \in \mathscr{V}$ being the source and sink nodes of $\mathscr{G}$ respectively. They are artificial nodes and do not represent any actual trains. In our model, we will represent each individual locomotive class as a separate commodity. The choice of an arc $a = (t_1, t_2) \in \mathscr{A}$ by a locomotive class means that a locomotive of this class first serves train t_1 and directly afterwards train t_2. As an abbreviation for the outgoing and the incoming arcs of a node $t \in \mathscr{V}$ in the graph $\mathscr{G}$, we use $\delta^+(t)$ and $\delta^-(t)$ respectively. Further, for each $L \in \mathcal{L}$ we denote with $\mathscr{A}^L \in \mathscr{A}$ the subset of arcs which are compatible with the locomotive class L.

Per definition of the set $\mathscr{A}$, a locomotive class may be chosen for a given arc if it is compatible with both the first and the second corresponding train. Further, each arc shall have unit capacity, i.e. it can be chosen by at most one locomotive class.

After solving the model, assigning a specific, individual locomotive to arcs and nodes in the flow network

$\mathscr{G}$ becomes a straightforward task. We obtain a solution in the form of directed paths through the network, connecting the source node Σ to the sink node Θ for all the locomotive classes $L \in \mathcal{L}$. To make the locomotive assignment concrete, we only need to assign each of the paths to an actual locomotive in the class $L \in \mathcal{L}$.

Decision variables In our model, we need to make sure that each train $t \in T$ is staffed with exactly one suitable driver $d \in D^t$ and one locomotive of suitable class $L \in \mathcal{L}^t$ if it shall run. The decision to assign driver $d \in D^t$ to a train $t \in T$ is modelled by the binary variables x_d^t. The binary variable $f_L^{t_1,t_2}$ models the decision to have locomotive class $L \in \mathcal{L}^{t_1} \cap \mathcal{L}^{t_2}$ carry the two trains $t_1, t_2 \in T$ in direct succession.

To comply with the working time requirements, we need to consider the first and the last job in the shift of a driver explicitly. The binary variable y_d^t asks if train $t \in T^d$ is the first job of a driver $d \in D$ on the respective shift. Similarly, the binary variable v_d^t models the choice of the last job in a shift before a short break. The binary variable z_d^t asks if the arrival time of train t indicates the beginning of the long break for driver d. We also need to know whether a driver $d \in D$ works on the Sunday of week w. This is determined by the binary variable h_d^w.

Finally, for modelling purposes we also need to know which trains $t \in T^d$ are the first and the last job for driver $d \in D$ in the planning period. We do that with the help of the binary variables α_d^t and ω_d^t, respectively. While several of the choices are similar in structure, we believe that this level of detail in the presentation is important to precisely convey the information about the individual working time constraints. All in all, the variables are summarized in Table 3.

Name	Description	Type
$f_L^{u,v}$	Trains u, v are served by locomotive class L in direct succession	binary
x_d^t	Train t is served by driver d	binary
y_d^t	Train t is the first job of driver d in the respective shift	binary
v_d^t	Train t is the last job of driver d before a short break	binary
z_d^t	The arrival time of train t indicates the beginning of the long break for driver d	binary
α_d^t	Train t is the first train of driver d in the planning period	binary
ω_d^t	Train t is the last train of driver d in the planning period	binary
h_d^w	Driver d works on the Sunday of calculation week w	binary

Table 3: Summary of decision variables used in the model

Model formulation We can now state a full formulation of the joint locomotive scheduling and driver assignment problem we consider in this work as a binary optimization problem:

$$\max \quad \sum_{t \in T} \sum_{d \in D^t} x_d^t \tag{1.1}$$

$$\text{s.t.} \quad x_d^{t_1} \leq \sum_{\substack{L \in \mathcal{L}^{t_1} \cap \mathcal{L}^d, \\ t_2 : (t_1, t_2) \in \mathscr{A}^L}} f_L^{t_1, t_2} \qquad (\forall t_1 \in T) \quad (\forall d \in D^{t_1}) \tag{1.2}$$

$$\sum_{t_2 : (t_1, t_2) \in \mathscr{A}^L} f_L^{t_1, t_2} \leq \sum_{d \in D^L \cap D^{t_1}} x_d^{t_1} \qquad (\forall t_1 \in T) \quad (\forall L \in \mathcal{L}^{t_1}) \tag{1.3}$$

$$\sum_{d \in D^t} x_d^t \leq 1 \qquad (\forall t \in T) \tag{1.4}$$

$$\sum_{t \in T^d} \alpha_d^t \leq 1 \qquad (\forall d \in D) \tag{1.5}$$

$$\sum_{t \in T^d} \omega_d^t \leq 1 \qquad (\forall d \in D) \tag{1.6}$$

$$x_d^t + x_d^{t_1} \leq 1 \qquad (\forall d \in D) \quad (\forall t \in T^d) \quad (\forall t_1 \in T_{t,d}^{\text{time}}) \tag{1.7}$$

$$y_d^t + x_d^{t_1} \leq 1 \qquad (\forall d \in D) \quad (\forall t \in T^d) \quad (\forall t_1 \in T_{t,d}^{\text{B-}}) \tag{1.8}$$

$$v_d^t + x_d^{t_1} \leq 1 \qquad (\forall d \in D) \quad (\forall t \in T^d) \quad (\forall t_1 \in T_{t,d}^{\text{B+}}) \tag{1.9}$$

$$z_d^t + x_d^{t_1} \leq 1 \qquad (\forall d \in D) \quad (\forall t \in T^d) \quad (\forall t_1 \in T_{t,d}^{\text{LB+}}) \tag{1.10}$$

$$v_d^t \leq \omega_d^t + \sum_{t_1 \in T_{t,d}^{\text{after\_break}}} y_d^{t_1} \qquad (\forall d \in D) \quad (\forall t \in T^d) \tag{1.11}$$

$$y_d^t \leq \alpha_d^t + \sum_{t_1 \in T_{t,d}^{\text{before\_break}}} v_d^{t_1} \qquad (\forall d \in D) \quad (\forall t \in T^d) \tag{1.12}$$

$$x_d^t \leq \sum_{t_1 \in T_{t,d}^{\text{shift\_beginning}}} y_d^{t_1} \qquad (\forall d \in D) \quad (\forall t \in T^d) \tag{1.13}$$

$$x_d^t \leq \sum_{t_1 \in T_{t,d}^{\text{shift\_end}}} v_d^{t_1} \qquad (\forall d \in D) \quad (\forall t \in T^d) \tag{1.14}$$

$$x_d^t \leq \sum_{t_1 \in T_{w,d}^{\text{week}}} z_d^{t_1} \quad (\forall w \in W)\ (\forall d \in D)\ (\forall t \in T_{w,d}^{\text{week\_assignment}}) \tag{1.15}$$

$$\alpha_d^{t_1} \leq \sum_{t_2 \in T^d : t_2 \geq t_1} \omega_d^{t_2} \quad (\forall d \in D)\ (\forall t_1 \in T^d) \tag{1.16}$$

$$x_d^t \leq h_d^w \quad (\forall d \in D)\ (\forall w \in W)\ (\forall t \in T_{w,d}^{\text{sunday}}) \tag{1.17}$$

$$\sum_{w \in W} h_d^w \leq c^{\text{sunday}} \quad (\forall d \in D) \tag{1.18}$$

$$y_d^t \leq x_d^t \quad (\forall d \in D)\ (\forall t \in T^d) \tag{1.19}$$

$$v_d^t \leq x_d^t \quad (\forall d \in D)\ (\forall t \in T^d) \tag{1.20}$$

$$\alpha_d^t \leq x_d^t \quad (\forall d \in D)\ (\forall t \in T^d) \tag{1.21}$$

$$\omega_d^t \leq x_d^t \quad (\forall d \in D)\ (\forall t \in T^d) \tag{1.22}$$

$$\sum_{t_0 : (t_0, t_1) \in \mathscr{A}^L} f_L^{t_0, t_1} - \sum_{t_2 : (t_1, t_2) \in \mathscr{A}^L} f_L^{t_1, t_2} = 0 \quad (\forall L \in \mathcal{L})\ (\forall t_1 \in T^L) \tag{1.23}$$

$$\sum_{L \in \mathcal{L}^{t_2}} \sum_{t_1 : (t_1, t_2) \in \mathscr{A}^L} f_L^{t_1, t_2} \leq 1 \quad (\forall t_2 \in T) \tag{1.24}$$

$$\sum_{t \in T^L} f_L^{\Sigma, t} \leq |L| \quad (\forall L \in \mathcal{L}) \tag{1.25}$$

$$\sum_{t \in T^L} f_L^{\Sigma, t} - \sum_{t \in T^L} f_L^{t, \Theta} = 0 \quad (\forall L \in \mathcal{L}) \tag{1.26}$$

$$x_d^t \in \{0, 1\} \quad (\forall t \in T)\ (\forall d \in D^t) \tag{1.27}$$

$$y_d^t \in \{0, 1\} \quad (\forall t \in T)\ (\forall d \in D^t) \tag{1.28}$$

$$v_d^t \in \{0, 1\} \quad (\forall t \in T)\ (\forall d \in D^t) \tag{1.29}$$

$$z_d^t \in \{0,1\} \qquad (\forall t \in T) \quad (\forall d \in D^t) \tag{1.30}$$

$$\alpha_d^t \in \{0,1\} \qquad (\forall t \in T) \quad (\forall d \in D^t) \tag{1.31}$$

$$\omega_d^t \in \{0,1\} \qquad (\forall t \in T) \quad (\forall d \in D^t) \tag{1.32}$$

$$h_d^w \in \{0,1\} \qquad (\forall d \in D) \quad (\forall w \in W) \tag{1.33}$$

$$f_L^{t_1,t_2} \in \{0,1\} \qquad \forall L \in \mathcal{L} \quad \forall (t_1,t_2) \in \mathscr{A}^L. \tag{1.34}$$

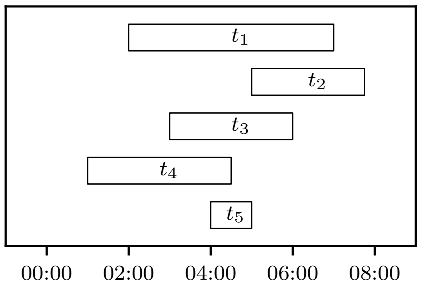
With objective function (1.1), we maximize the number of trains running. Constraints (1.2) and (1.3) make sure that either both a locomotive and a driver are assigned to a train or none of them; they also take care that driver and locomotive are mutually compatible. With (1.4), we ensure that at most one driver is assigned to a train. Constraints (1.5) and (1.6) enforce that each driver has a unique first and last job respectively. Using constraint (1.7), we ensure that no two trains which run simultaneously are assigned to the same driver. With (1.8) and (1.9), we model that the minimal length of a short break shall not be violated. Additionally, with (1.10) we require the integrity of the long break. Using (1.11), we make sure that each last job t in a shift is succeeded by a first job of the next shift, or that the job t is the last one assigned to driver d in the plan. Similarly, with (1.12) we ensure that each first job t in a shift is predecessed by a last job of the previous shift, or that the job t is the first one assigned to driver d in the plan. Constraints (1.13) and (1.14) enforce the selection of variables denoting the beginning and the end of a shift. Via (1.15), we make sure that at least one long break per week is assigned to each driver in each week in the planning period. With the help of (1.16), we achieve that the last job of a driver in the plan is either the same or a later one as the first job in the plan. Using constraints (1.17) and (1.18), we guarantee that a driver works on at most c^{sunday}-many Sundays in a given planning period. Constraints (1.19), (1.20), (1.21) and (1.22) tie each shift variable to the corresponding staffing variable.

For the locomotive part of the model, (1.23) ensures that a locomotive that serves a train t arrives at its origin station and in the due time, and that similarly it later departs from the arrival station of train t. With constraint (1.24), we make sure that at most one locomotive class $L \in \mathcal{L}^{t_2}$ is selected to serve a train $t_2 \in T$. Constraint (1.25) ensures that the number of locomotives of a given class used in the plan does not exceed the cardinality of the locomotive class. Via (1.26), we assure the integrity of the locomotive schedule. Finally, constraints (1.27) – (1.34) require all decision variables to be binary.

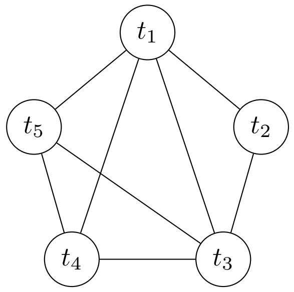
3.3 Model preprocessing – clique tightening

Model (1) is very complex, mostly due to its size. We will now consider a way to reduce the number of constraints needed, which usually greatly benefits the computations times. Namely, we will use *clique tightening* to improve the formulation of the conflict constraints.

Consider the conflict constraints (1.7), (1.8), (1.9) and (1.10). They all refer to a situation in which at most one of the considered trains can be assigned to the same driver. With a growing number of trains, the number of such constraints will grow cubically. For all drivers $d \in D$, we construct a graph $\mathcal{G}^d_{\text{time}} = (\mathcal{V}^d_{\text{time}}, \mathcal{E}^d_{\text{time}})$, where $\mathcal{V}^d_{\text{time}} \coloneqq T^d$ and $\mathcal{E}^d_{\text{time}} \coloneqq \{(t_1, t_2) : t_1 \in T^d \wedge t_2 \in T^{\text{time}}_{t_1,d}\}$. For the exemplary case depicted in Figure 1a, the resulting conflict graph $\mathcal{G}^d_{\text{time}}$ is presented in Figure 1b.



(a) Train schedule



(b) Resulting conflict graph

Figure 1: Example of a time conflict graph

We notice that a grouping of the nodes in this graph to cliques can be employed to come up with stronger constraints. Obviously, each edge of the graph itself induces a 2-clique in the graph. We are now interested in finding fewer, but larger cliques in order to express an equivalent, smaller set of constraints. In order to do so, we need to make sure that each edge is covered by at least one of the cliques. This can be achieved by searching for a (minimal) clique edge cover of the graph $\mathcal{G}^d_{\text{time}}$.

Proposition 3.1. *Let $\mathcal{C}^d_{\text{time}}$ be a clique edge cover of the graph $\mathcal{G}^d_{\text{time}}$. Then constraint* (1.7) *is equivalent to the following constraint:*

$$\sum_{t \in C} x^t_d \quad \leq \quad 1 \qquad (\forall d \in D) \;\; (\forall C \in \mathcal{C}^d_{\text{time}}). \tag{1.7b}$$

In the example from Figure 1b, we can use the two cliques $C_1 \coloneqq \{t_1, t_3, t_4, t_5\}$ and $C_2 \coloneqq \{t_1, t_2, t_3\}$ to construct two conflict constraints which are equivalent to the eight constraints induced by the individual edges. This reformulation also results in a tighter description of the underlying convex hull of feasible solutions (see e.g. Brito and Santos (2021)).

For the remaining constraints (1.8), (1.9) and (1.10), we can further extend this concept by a slight redefinition of the conflict graph. We outline the derivation at the hand of the backward-break conflict constraints, represented by constraint (1.8). For all drivers $d \in D$, we construct a graph $\mathcal{G}^d_{\text{back\_break}} = (\mathcal{V}^d_{\text{back\_break}}, \mathcal{E}^d_{\text{back\_break}})$. The vertex set is defined as

$$\mathcal{V}^d_{\text{back\_break}} \coloneqq \mathcal{V}^d_X \cup \mathcal{V}^d_Y, \quad \text{with} \quad \mathcal{V}^d_X \coloneqq T^d \times \{1\}, \quad \mathcal{V}^d_Y \coloneqq T^d \times \{2\},$$

i.e. $\mathcal{V}^d_{\text{back\_break}}$ contains two copies of each node in T^d. For ease of notation, we will write $t_X \in \mathcal{V}_X$ and $t_Y \in \mathcal{V}_Y$ instead of $(t,1) \in \mathcal{V}^d_X$ and $(t,2) \in \mathcal{V}^d_Y$ respectively, with the understanding that $t \in T^d$ holds in each case. The vertices in $\mathcal{V}_X$ correspond to the respective x-variables, while those in $\mathcal{V}_Y$ correspond to the respective y-variables. The edge set is then defined as

$$\mathcal{E}^d_{\text{back\_break}} \coloneqq \{\{t_X, t_Y\} : t_X \in \mathcal{V}^d_X \wedge t_Y \in \mathcal{V}^d_Y \wedge t_X \in T^{\text{B-}}_{t_Y,d}\} \cup \{\{t^1_Y, t^2_Y\} : t^1_Y, t^2_Y \in \mathcal{V}^d_Y \wedge t^2_Y \in T^{\text{B-}}_{t^1_Y,d}\}.$$

There are two kinds of edges in $\mathcal{E}^d_{\text{back\_break}}$. The first of them corresponds to conflicts between x-variables and y-variables, and the second represents conflicts only between y-variables, which are implied by other constraints, as explained in the following.

Since for a given driver $d \in D$ no nodes in $\mathcal{V}^d_X$ are connected to each other, each clique in $\mathcal{G}^d_{\text{back\_break}}$ contains at most one node from the vertex set $\mathcal{V}^d_X$. For the construction of constraints, we are interested in the cliques which contain at least one node from $\mathcal{V}^d_X$, i.e. one x-variable. Such cliques will be referred to as *constraint-generating* cliques. Note that we can safely omit cliques containing only nodes from $\mathcal{V}^d_Y$, since – thanks to constraint (1.19) – the resulting conflict constraints will be dominated by the constraints generated from cliques including nodes from both $\mathcal{V}^d_X$ and $\mathcal{V}^d_Y$. One obvious extension of the approach would be to include all the edges induced by time conflicts (from $\mathcal{G}^d_{\text{time}}$) also between the nodes in $\mathcal{V}^d_X$. To save computation time, we decided to include only those conflicts which are not already represented in $\mathcal{G}^d_{\text{time}}$ – this reduces the number of edges, which led to a smaller number of cliques to be considered in our experiments in Section 5.

We now define $\mathscr{C}^d_{\text{back\_break}}$ to be the set of all maximal cliques in $\mathcal{G}^d_{\text{back\_break}}$ for all drivers $d \in D$. Based on this set, we define a set of constraint-generating cliques for each train $t \in T^d$ as the subset of maximal cliques containing the node $t_X \in \mathcal{V}^d_X$:

$$\mathcal{C}^{t,d}_{\text{back\_break}} \coloneqq \{C \in \mathscr{C}^d_{\text{back\_break}} : (t_X \in C \vee t_Y \in C) \wedge t_X \in \mathcal{V}^d_X\}.$$

For the illustrative case of a driver $d \in D$ with $T^d \coloneqq \{t_1, t_2, t_3, t_4\}$, $T^{\text{B-}}_{t_4,d} \coloneqq \{t_2, t_3\}$, $T^{\text{B-}}_{t_3,d} \coloneqq \{t_1, t_2\}$, an exemplary graph $\mathcal{G}^d_{\text{back\_break}}$ is presented in Figure 2. In particular, if train t_4 is selected as the first one in a shift of driver d, this driver may not serve trains t_2 and t_3 as well. Similarly, trains t_1 and t_2 may not be assigned to driver d if train t_3 was chosen as the first one in the shift.

A very similar derivation for constraints (1.9) and (1.10) leads to corresponding graphs $\mathcal{G}^d_{\text{forward\_break}}$ and

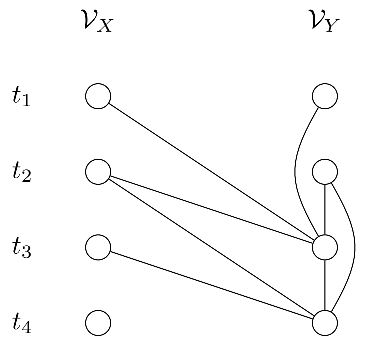


Figure 2: Example of a break conflict graph $\mathcal{G}^d_{\text{back\_break}}$

$\mathcal{G}^d_{\text{long\_break}}$ for all drivers $d \in D$ and respective sets of constraint-generating cliques $\mathcal{C}^{t,d}_{\text{forward\_break}}$ and $\mathcal{C}^{t,d}_{\text{long\_break}}$. With a slight abuse of notation, we also introduce the respective node sets $\mathcal{V}_V$ and $\mathcal{V}_Z$ as well as the respective node types t_V and t_Z for the trains $t \in T^d$. They allow us to simplify model (1) further.

Proposition 3.2. *The constraints* (1.8)–(1.10) *are equivalent to the following set of constraints:*

$$\sum_{t_Y \in C} y^{t_Y}_d + \sum_{t_X \in C} x^{t_X}_d \leq 1 \qquad (\forall d \in D)\ (\forall t \in T^d)\ (\forall C \in \mathcal{C}^{t,d}_{\text{back\_break}}) \tag{1.8b}$$

$$\sum_{t_V \in C} v^{t_V}_d + \sum_{t_X \in C} x^{t_X}_d \leq 1 \qquad (\forall d \in D)\ (\forall t \in T^d)\ (\forall C \in \mathcal{C}^{t,d}_{\text{forward\_break}}) \tag{1.9b}$$

$$\sum_{t_Z \in C} z^{t_Z}_d + \sum_{t_X \in C} x^{t_X}_d \leq 1 \qquad (\forall d \in D)\ (\forall t \in T^d)\ (\forall C \in \mathcal{C}^{t,d}_{\text{long\_break}}). \tag{1.10b}$$

To save computation time, we will determine the required clique edge covers in a heuristic fashion; see Section 5.1.

4 Solution methods

We will now derive a solution algorithm based on decomposition and cutting planes for the integrated locomotive scheduling and driver assignment problem introduced in the previous section. Figure 3 gives an outline of the approach as a flowchart; its ingredients are detailed in the following. First, in Subsection 4.1 the general decomposition scheme will be presented. A number of preliminary computational experiments had shown that the locomotive assignment part of the problem is far easier to solve than the driver part. This led to the idea to design the solution method in such a way that it first computes a best possible feasible locomotive assignment, relaxing all driver-related constraints. In Subsection 4.2, we will derive cutting planes in the locomotive-flow variables which are valid for the integrated problem (1). These cutting planes encode common reasons for the infeasibility of the driver part (as encountered for the real-world instances presented in Section 5), expressed in terms of the locomotive assignment variables. In our solution algorithm, we first solve the locomotive part as a master problem to obtain a candidate locomotive assignment. Then we search for violated cutting planes to cut the current locomotive assignment off from the locomotive master problem and solve it again. This procedure is iterated until no further cutting planes are found. The next step is to fix the resulting locomotive assignment in the integrated problem. This gives rise to the driver subproblem, whose task it is to find a feasible driver assignment which is compatible to the fixed locomotive assignment. If this is possible, we have solved the problem to global optimality. Sometimes, the driver part is infeasible but we cannot generate one of our problem-specific cutting planes to ensure feasibility of the driver subproblem. We then generate a combinatorial Benders cut instead which precisely cuts off the current locomotive assignment from the locomotive master problem and reiterate. This way, the algorithm will eventually converge to a global optimal solution to the joint locomotive and driver problem. Finally, we will present some algorithmic enhancements to reduce computation times. These are a preprocessing scheme (Subsection 4.3) as well as a heuristic (Subsection 4.4) whose aim it is to facilitate the solution of the driver subproblem, which is still hard to solve, even though the locomotive assignment is already fixed there.

4.1 Decomposition into locomotive and driver subproblem

As already discussed in Subsection 3.2, we can subdivide the constraints of the joint model (1) into three groups: (i) compatibility constraints (1.2) and (1.3) between drivers and locomotives, (ii) the driver-related constraints (1.4)–(1.22) and (1.27)–(1.33), and (iii) the locomotive-related constraints (1.23)–(1.26) and (1.34). When relaxing the compatibility constraints, model (1) decomposes into two independent subproblems. We call the model induced by the original objective function (1.1) and constraints (1.4)–(1.22) as well as (1.27)–(1.33) the *driver subproblem*.

In order to define the corresponding *locomotive master problem*, we first note that although the locomotive-related constraints are exactly those of a pure binary multi-commodity-flow problem, we need a slightly different kind of objective function. Rather than maximizing the flow through the network (which would amount to maximizing the number of used locomotives), we have to make sure that as many nodes/trains in the network are "visited" by some commodity/locomotive. To this end, we introduce a new binary decision variable which checks whether a given train was served by a compatible locomotive or not:

$$\lambda_t = \begin{cases} 1, & \text{if train } t \in T \text{ is served by a locomotive of compatible class } L \in \mathcal{L}^t \\ 0, & \text{otherwise.} \end{cases}$$

Using the λ-variables, we can state the objective function of the locomotive master problem maximizing the number of served trains:

$$\max \sum_{t \in T} \lambda_t. \tag{2.1}$$

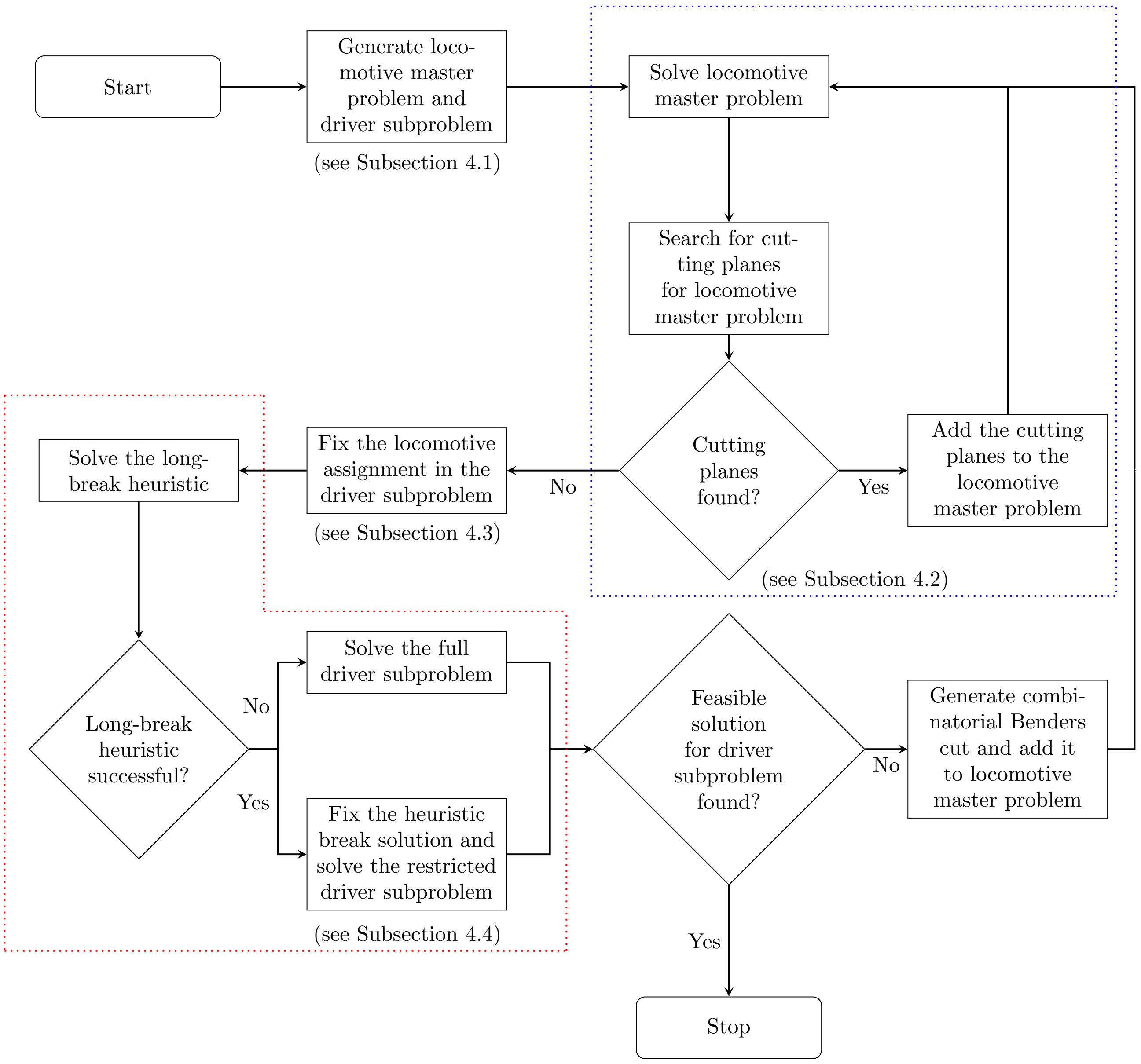

Start
Generate locomotive master problem and driver subproblem
(see Subsection 4.1)
Solve locomotive master problem
Search for cutting planes for locomotive master problem
Cutting planes found?
Yes
Add the cutting planes to the locomotive master problem
No
(see Subsection 4.2)
Fix the locomotive assignment in the driver subproblem
(see Subsection 4.3)
Solve the long-break heuristic
Long-break heuristic successful?
No
Solve the full driver subproblem
Yes
Fix the heuristic break solution and solve the restricted driver subproblem
(see Subsection 4.4)
Feasible solution for driver subproblem found?
No
Generate combinatorial Benders cut and add it to locomotive master problem
Yes
Stop

Figure 3: Flowchart of the exact version of the solution algorithm introduced in this work

We also need an additional constraint which couples the newly introduced λ-variables with the existing f-variables:

$$\lambda_t \quad \leq \quad \sum_{\substack{w \in \delta^+(t) \cap T^L, \\ L \in \mathcal{L}^t}} f_L^{t,w} \quad (\forall t \in T). \tag{2.2}$$

Finally, we need to make sure that the λ-variables are binary:

$$\lambda_t \quad \in \quad \{0,1\} \quad (\forall t \in T). \tag{2.3}$$

Altogether, the *locomotive master problem* has the objective function (2.1) as well as the constraints (1.23)–(1.26), (1.34), (2.2) and (2.3).

4.2 Valid inequalities for the locomotive subproblem

The driver subproblem contains ten constraint types which are potential sources of infeasibility in model (1) when fixing the f-variables corresponding to a given solution to the locomotive master problem. These are all the conflict constraints (1.5), (1.6), (1.7b), (1.8b), (1.9b) and (1.10b), the compatibility constraints (1.11), (1.12), long-break enforcement constraint (1.15) as well as constraint (1.18), which pertains to ensuring the Sunday breaks. Based on our computational experience, we determined that only a subset of these are violated for the real-world instances supplied by our industrial partner. For these, we show explicitly here how an infeasibility can arise. Further, we present corresponding valid inequalities classes for the locomotive master problem to ensure that its solution will not result in an infeasibility of the driver subproblem. For the remaining, potentially violated driver constraints, the valid locomotive inequalities can be derived in a similar fashion. ~~Although the valid inequalities introduced in this section are customized to model (1), a similar logic may be used to derive further valid inequalities in other models, which could benefit from a decomposition by constraint relaxing.~~

We start by noticing that all constraints of the driver subproblem not mentioned above cannot be a cause of infeasibility in model (1) when fixing a solution to the locomotive part of the problem

Observation 4.1. *Let* $(\bar{f}, \bar{\lambda})$ *be a feasible solution to the locomotive master problem given by constraints* (2.1), (1.23)–(1.26), (1.34), (2.2) *and* (2.3). *Then we can find values* $\bar{x}, \bar{y}, \bar{v}, \bar{\alpha}, \bar{\omega}, \bar{h}$ *such that* $(\bar{f}, \bar{x}, \bar{y}, \bar{v}, \bar{\alpha}, \bar{\omega}, \bar{h})$ *is a solution to the constraint system given by the driver-related constraints* (1.2), (1.3), (1.13), (1.14), (1.16), (1.17), (1.19), (1.20), (1.21) *and* (1.22).

Proof. Feasibility of the second driver-locomotive compatibility constraint (1.3) can be ensured for any feasible locomotive assignment, because it is sufficient to set at most one x-variable on its right-hand side to one in order to fulfil it. Recall that for each train-locomotive combination, at least one compatible driver can be found. By setting at most one x-variable on the right-hand side of constraint (1.3) to one, we choose any compatible driver for the locomotive-train assignment.

For constraints (1.13), (1.14), (1.16), (1.17) we can choose the y-, v-, α-, ω- and h-variable corresponding to the chosen x-variables and set them to one. By doing so, we do not have to set any other x-variables occuring on the right-hand sides of constraints (1.19), (1.20), (1.21) and (1.22) to one. Therefore, the first driver-locomotive compatibility constraint (1.2) cannot be violated either. $\square$

Let us now focus on the ten potentially problematic constraints mentioned in the beginning of this subsection. As they might well be violated by a given, fixed locomotive assignment, we need to consider them already at the stage of solving the locomotive master problem. We will now explain how such an infeasibility can arise and introduce additional, valid constraints for the locomotive master problem to ensure the feasibility of the locomotive assignment when computing a compatible driver assignment. As we will see, many of the infeasibilities targeted by these cuts are caused by an insufficient availability of drivers to certain train-locomotive combinations. Such a situation may occur if the total number of drivers is very small, or if some train-locomotive combinations can only be served by a small number of drivers. In particular, such cases could take place when the railway carrier chooses to serve a new customer, whose location can only be reached by a few of the drivers available, or when an unplanned closure of an important railway line occurs and a less-frequented detour route has to be taken.

In the case study in Section 5, we will see that these cutting planes are sufficient to ensure the feasibility of the driver assignment for the real-world instances provided by our industry partner, which means that we obtain globally optimal solutions via our method. As mentioned above, it might occur for general instances that a given locomotive solution cannot be cut off by the cutting planes we derive here. In this case, we can ensure the feasibility and global optimality of the decomposition approach by cutting off the current locomotive assignment from the locomotive master problem via a combinatorial Benders cut, as described in Codato and Fischetti (2006). Altogether, this allows us to cut off any locomotive assignment which cannot be completed to a full, feasible solution to model (1). Therefore, our algorithm will in the end converge to a feasible, globally optimal solution to the joint locomotive scheduling and driver assignment problem.

4.2.1 Valid cutting planes derived from the time conflict constraints

Constraint (1.7b) ensures that no driver is staffed to drive two trains which run simultaneously. We need to make sure that the computed optimum of the locomotive subproblem also respects these constraints. Recall that in Subsection 3.3 we introduced the graph $\mathcal{G}^d_{\text{time}}$ to represent the time conflicts between trains for a given driver $d \in D$. Let us now generalize this graph to cover time conflicts between *all* of the trains. For this purpose, we introduce the graph $G_{\text{time}} = (T, E_{\text{time}})$, where

$$E_{\text{time}} \coloneqq \{(t_1, t_2) : t_1 \in T \quad \wedge \quad t_2 \in T_{t_1}^{\text{time\_global}}\}$$

Now define $\mathscr{C}^{\text{time}}$ as the set of all maximal cliques in G_{time}. As the set of drivers available for a train depends both on the train itself and on the locomotive assigned to carry it, we need to consider all possible assignments of locomotives for all trains present in the cliques in $\mathscr{C}^{time}$. These assignments are represented by the set

$$\mathscr{C}^{\text{time}}_{\text{assignments}} := \{\{(t, L) : t \in C \wedge L \in \mathcal{L}^t\} : C \in \mathscr{C}^{time}\},$$

which allows us to derive valid cutting planes.

Theorem 4.2. *The following inequalities are valid for model* (1)*:*

$$\sum_{(t,L)\in C} \sum_{t_1 \in T^L \cap \delta^+(t)} f_L^{t,t_1} \quad \leq \quad \left| \bigcup_{(t,L)\in C} D^t \cap D^L \right| \quad (\forall C \in \mathscr{C}^{\text{time}}_{\text{assignments}}). \tag{2.4}$$

Proof. For the purpose of this proof, we define a set $C_{\text{trains}} := \{t_1 : (t_1, L) \in C\}$ for each $C \in \mathscr{C}^{\text{time}}_{\text{assignments}}$. By definition, for each of the sets $C \in \mathscr{C}^{\text{time}}_{\text{assignments}}$, all of the trains in the set C_{trains} are in time conflict. Hence, we need at least $|C_{\text{trains}}|$-many drivers compatible with the trains in C_{trains}. In other words,

$$|C_{\text{trains}}| \leq \left| \bigcup_{(t,L)\in C} D^t \cap D^L \right|$$

Moreover, from constraint (1.24), which stipulates that at most one locomotive can be assigned to a train, we know that

$$\sum_{(t,L)\in C} \sum_{t_1 \in T^L \cap \delta^+(t)} f_L^{t,t_1} \leq |C_{\text{trains}}|.$$

Hence, inequalities (2.4) are valid for the model (1).

□

To better illustrate the context in which the cut is being used, we introduce the following example.

Example 4.3. *Consider a subset $S := \{t_1, t_2, t_3\}$ of a larger set of trains. Assume that the trains in S are mutually in time conflict. Then the set S constitutes a clique in $\mathscr{C}^{\text{time}}_{\text{assignments}}$. Let us assume that this clique is maximal. Further, we assume the following locomotive compatibilities: $\mathcal{L}^{t_1} = \{L_1, L_2\}$, $\mathcal{L}^{t_2} := \{L_1, L_2, L_3\}$, $\mathcal{L}^{t_3} := \{L_2, L_3\}$. With regard to drivers, let $D^{t_1} = D^{t_2} = D^{t_3} := \{d_1, d_2, d_3, d_4\}$, and $D^{L_1} := \{d_1, d_2, d_3\}$, $D^{L_2} := \{d_1, d_2\}$, $D^{L_3} := \{d_2, d_3, d_4\}$.*

Now, consider the following assignment of locomotives to trains: $\{(t_1, L_2), (t_2, L_2), (t_3, L_2)\}$. Such a locomotive assignment will cause the need to assign two drivers d_1 and d_2 to three trains in time conflict, which is obviously infeasible (and violates constraint (1.7b)*).*

Such an infeasibility may be cut off by a cut introduced in this section. For the example considered, it would *look as follows:*

$$\sum_{t_a \in T^{L_2} \cap \delta^+(t_1)} f_{L_2}^{t_1,t_a} + \sum_{t_b \in T^{L_2} \cap \delta^+(t_2)} f_{L_2}^{t_2,t_b} + \sum_{t_c \in T^{L_2} \cap \delta^+(t_3)} f_{L_2}^{t_3,t_c} \leq |(D^{t_1} \cap D^{L_2}) \cup (D^{t_2} \cap D^{L_2}) \cup (D^{t_3} \cap D^{L_2})| = 2.$$

This way, we ensure that at most two of the trains in S are assigned locomotives of class L_2, thereby cutting off the infeasible solution.

In general, the count of maximal cliques in a graph could reach $3^{n/3}$, with n corresponding to the number of vertices in the graph (see Moon and Moser (1965)). Similarly, enumerating the set of all maximal cliques is rather difficult – it can be achieved in $\mathcal{O}(3^{n/3})$ (see Tomita et al. (2006)). However, for families of graphs in which the number of maximal cliques is bounded polynomially, we are able to list all of the maximal cliques in polynomial time (based on the results of Johnson et al. (1988)). Moreover, Fulkerson and Gross (1965) show that the number of maximal cliques for chordal graphs amounts to n.

Despite the seeming similarity, we cannot state with certainty that G_{time} is an interval graph. This is because the set E_{time} of edges of G_{time} is based on the set $T_t^{\text{time\_global}}$. By definition, $T_t^{\text{time\_global}}$ includes trains in time conflict with $t \in T$, as well as trains $t_1 \in T$ that cannot be served together with the train t due to an excessive transit time required to reach the departure station of t_1 after serving t. Hence, G_{time} is at most a supergraph of an interval graph. ~~This is due to the additional elements in the sets $T_t^{\text{time\_global}}$, which make G_{time} at most a supergraph of an interval graph.~~

Nonetheless, the results we obtained in our experiments (discussed in Part 5) suggest a linear relationship between the number of vertices in the graph G_{time} and both the number of maximal cliques ($|\mathscr{C}^{\text{time}}|$) and their enumeration time for our instances. Therefore, although our solution algorithm (discussed in more detail in Section 5.1) will require us to enumerate all the maximal cliques $\mathscr{C}^{\text{time}}$, the theoretical results suggest that neither will the count of the cliques be too large, nor will the enumeration time be too long.

4.2.2 Valid cutting planes derived from the break conflict constraints

Consider constraints (1.8b), (1.9b) and (1.10b). As they are relatively similar in structure, we will only treat constraint (1.9b) exemplarily. Here we deal with situations in which only one driver $d \in D$ is available to serve a certain set of train-locomotive combinations, whereas – due to working time regulations – the driver may be assigned to at most one of them. If the solution to the locomotive subproblem enforced an assignment of driver d to more than one of these trains, some of the working time constraints would be violated, and thus the locomotive assignment would be infeasible. Thanks to the cutting planes developed in this section, such locomotive assignments can be cut off.

Recall from Proposition 3.3 that for each driver $d \in D$ and for each train $t \in T^d$, all the constraint-defining cliques $\mathcal{C}_{\text{forward\_break}}^{t,d} \in \mathscr{C}_{\text{forward\_break}}^{d}$ contain exactly one node from the node set $\mathcal{V}_X^d$ and at least one node from node set $\mathcal{V}_V^d$. For each driver $d \in D$, for each train $t \in T^d$ and for each conflict set $\mathcal{C}_{\text{forward\_break}}^{t,d} \in \mathscr{C}_{\text{forward\_break}}^d$, we now define a set $S^{t,d}$ of train-locomotive assignments with $S^{t,d} := S_X^{t,d} \cup S_V^{t,d}$, where

$$S_X^{t,d} := \{(t, L) : L \in \mathcal{L}^t \quad \wedge \quad D^t \cap D^L = \{d\}\}$$

and

$$\begin{aligned} S_V^{t,d} := \{(t_v, L) : t_v \in \mathcal{C}_{\text{forward\_break}}^{t,d} \quad \wedge \quad t_v \in \mathcal{V}_V^d \quad \wedge \quad L \in \mathcal{L}^{t_v} \\ \wedge \quad D^{t_v} \cap D^L = \{d\} \quad \wedge \quad T_{t_v,d}^{\text{shift\_end}} = \{t_v\}\}. \end{aligned}$$

Again, we derive a set of valid cutting planes from this construction.

Theorem 4.4. *The following inequalities are valid for model* (1)*:*

$$\sum_{(t,L)\in S^{t,d}} \sum_{t_1 \in T^L \cap \delta^+(t)} f_L^{t,t_1} \quad \leq \quad 1 \quad (\forall d \in D) \ (\forall t \in T^d). \tag{2.6}$$

Proof. By definition of $S^{t,d}$, at most one of the f-variables pertaining to the pairs $(t, L) \in S^{t,d}$ may be selected, as otherwise constraint (1.9b) would be violated. Hence, inequalities (2.6) are valid for model (1).

□

To better illustrate the context in which the cut is being used, we introduce the following example.

Example 4.5. *Consider a subset $\{t_1, t_2\}$ of a larger set of trains. Moreover, let $t_2 \in \mathcal{C}_{\text{forward\_break}}^{t_1,d}$ for all $d \in D$. Additionally, assume $T_{t_2,d}^{\text{shift\_end}} := \{t_2\}$ for all $d \in D$. Further, we assume the following locomotive compatibilities: $\mathcal{L}^{t_1} := \{L_1, L_2\}$, $\mathcal{L}^{t_2} := \{L_1, L_2, L_3\}$. With regard to drivers, let $D^{t_1} = D^{t_2} := \{d_1, d_2, d_3, d_4\}$, and $D^{L_1} := \{d_1, d_2, d_3\}$, $D^{L_2} := \{d_1\}$, $D^{L_3} := \{d_2, d_3, d_4\}$.*

Now, consider the following assignment of locomotives to trains: $\{(t_1, L_2), (t_2, L_2)\}$. Such a locomotive assignment will cause the need to assign the same driver d_1 to the two trains t_1 and t_2. Since $t_2 \in \mathcal{C}_{\text{forward\_break}}^{t_1,d}$ for all $d \in D$ and $T_{t_2,d}^{\text{shift\_end}} = \{t_2\}$ for all $d \in D$, we will see an infeasibility with regard to constraint (1.9b) *– we may not assign the same driver to t_1 and t_2.*

To get rid of such an infeasibility, we use the cut introduced in this section. For the example considered, it would look as follows:

$$\sum_{t_a \in T^{L_2} \cap \delta^+(t_1)} f_{L_2}^{t_1,t_a} + \sum_{t_b \in T^{L_2} \cap \delta^+(t_2)} f_{L_2}^{t_2,t_b} \quad \leq \quad 1.$$

For constraints (1.8b) and (1.10b), a respective sets of valid constraints each can be constructed in a similar fashion.

Similar to the case of the graph graph G_{time} discussed in Subection 4.2.1, the results we obtained in our experiments (discussed in Part 5) suggest a linear relationship between the number of vertices in the graph

$\mathcal{G}^d_{\text{forward\_break}}$ and both the number of maximal cliques and their enumeration time for our instances. This could be due to the fact that while $\mathcal{G}^d_{\text{forward\_break}}$ is not an interval graph itself, one could intuitively expect that at least many of its subgraphs are, and hence the performance of clique enumeration will not approach the worst-case scenario. An exploration of the properties of graphs $\mathcal{G}^d_{\text{forward\_break}}$ would be an interesting future research avenue.

4.2.3 Valid cutting planes derived from the compatibility constraints

Let us now turn to constraints (1.11) and (1.12). These two are again similar in structure, so that we only treat constraint (1.11) explicitly here. As it was discussed in Section 3.1, the ω-variables only exist for trains which end in the home region of the driver.

This means that for a driver $d \in D$ and for a train $t \in T^d$ which does not end in the driver's home region, we need to ensure that at least one train $t_1 \in T^{\text{after\_break}}_{t,d}$ may be performed by driver d. This requires us to assign locomotives to trains $t_1 \in T^{\text{after\_break}}_{t,d}$ in such a way that the driver is compatible with at least one train-locomotive assignment (t_1, L) with $t_1 \in T^{\text{after\_break}}_{t,d}$ and $L \in \mathcal{L}^{t_1}$. More formally, for each driver $d \in D$, we define a subset $\mathscr{O}^d \subseteq T^d$ of trains which do not end in the driver's home region (and hence no corresponding ω-variable exists):

$$\mathscr{O}^d \coloneqq \{t \in T^d : a(t) \in S \setminus H(d)\}.$$

For the trains $t \in \mathscr{O}^d$, we consider all train-locomotive assignments which may be served only by driver d. These will be stored in a set $\mathscr{S}^d$, defined as

$$\mathscr{S}^d \coloneqq \{(t, L) : t \in \mathscr{O}^d \quad \wedge \quad L \in \mathcal{L}^t \cap \mathcal{L}^d \quad \wedge \quad D^t \cap D^L = \{d\}\}.$$

Now, for each $(t, L) \in \mathscr{S}^d$ we introduce the set $\mathscr{F}^d_{t,L}$, which contains all train-locomotive assignments for which driver $d \in D$ is able to drive a train after a break:

$$\mathscr{F}^d_{t,L} \coloneqq \{(t_1, L_1) : t_1 \in T^{\text{after\_break}}_{t,d} \quad \wedge \quad L_1 \in \mathcal{L}^{t_1} \cap \mathcal{L}^d\}.$$

We use it to construct valid inequalities cutting off solutions which meet two conditions jointly for a driver $d \in D$ and a train $t \in \mathscr{O}^d$:

1. for the train t they assign such a locomotive class $L \in \mathcal{L}^t$ that driver d is the only available one, i.e. $D^L \cap D^t = \{d\}$.

2. for each of the trains $t_1 \in T^{\text{after\_break}}_{t,d}$, they assign such a locomotive class $L_1 \in \mathcal{L}^{t_1}$ that driver d may not be assigned to them (i.e. $d \notin D^{t_1} \cap D^{L_1}$ for all $t_1 \in T^{\text{after\_break}}_{t,d}$ and $L_1 \in \mathcal{L}^{t_1}$).

Such solutions result in the violation of constraint (1.11) and hence are infeasible.

Theorem 4.6. *The following inequalities are valid for model* (1)*:*

$$\sum_{t_1 \in T^L \cap \delta^+(t)} f_L^{t,t_1} \quad \leq \quad \sum_{(t_2,L_2)\in \mathscr{F}^d_{t,L}} \sum_{t_3 \in T^{L_2}\cap\delta^+(t_2)} f_{L_2}^{t_2,t_3} \quad (\forall d \in D) \ \ (\forall (t,L) \in \mathscr{S}^d). \tag{2.7}$$

Proof. By definition, for all drivers $d \in D$ and for any solution $(t,L) \in \mathscr{S}^d$ we need to select at least one of the assignments $(t_1,L_1) \in \mathscr{F}^d_{t,L}$, as otherwise constraint (1.11) would be violated. Hence, (2.7) are valid inequalities with regard to the model (1)

□

The following example will now be used to illustrate an infeasibility cut off by the inequality we introduced in this section.

Example 4.7. *Let us consider a set of three trains* $\{t_1,t_2,t_3\}$*, being a subset of a larger set of trains* T*, as well as a set of three drivers* $\{d_1,d_2,d_3\}$ *which is a subset of a bigger set of drivers* D*. Moreover, assume that* $t_1 \in \mathscr{O}^d$ *for all* $d \in \{d_1,d_2,d_3\}$*. Additionally, let* $\{t_2,t_3\} \subseteq T^{\text{after\_break}}_{t_1,d}$ *for all* $d \in \{d_1,d_2,d_3\}$*. Further, we assume the following locomotive compatibilities:* $\mathcal{L}^{t_1} := \{L_1,L_2\}$*,* $\mathcal{L}^{t_2} := \{L_1,L_2,L_3\}$*,* $\mathcal{L}^{t_3} := \{L_2,L_3\}$*. With regard to drivers, let* $D^{t_1} = D^{t_2} = D^{t_3} := \{d_1,d_2,d_3,d_4\}$*, and* $D^{L_1} := \{d_2,d_3\}$*,* $D^{L_2} := \{d_1\}$*,* $D^{L_3} := \{d_2,d_3,d_4\}$*.*

For an assignment $\{(t_1,L_2),(t_2,L_1),(t_3,L_3)\}$*, we notice that the only driver who is able to drive* t_1 *is* d_1*. Since* $t_1 \in \mathscr{O}^{d_1}$*, we need to assign locomotives to trains* $t \in T^{\text{after\_break}}_{t_1,d_1}$ *in such a way that the driver* d_1 *is compatible with at least one such assignment.*

In the presented context, this is not the case, as for train t_2 *the set of feasible drivers is* $D^{t_2} \cap D^{L_1} = \{d_2,d_3\}$*, and for train* t_3 *it is* $D^{t_3} \cap D^{L_3} = \{d_2,d_3,d_4\}$*. Hence, we get a solution which is infeasible with regard to constraint* (1.11)*.*

To prevent this, we can use the cut introduced in this section. It will then make sure that the locomotives are assigned in such a way that for at least one of the trains $t \in T^{\text{after\_break}}_{t_1,d_1}$ *driver* d_1 *may be chosen as well.*

In the context of our example, the cut would look the following way:

$$\sum_{t_a \in T^{L_2}\cap\delta^+(t_1)} f_{L_2}^{t_1,t_a} \quad \leq \quad \sum_{t_b \in T^{L_2}\cap\delta^+(t_2)} f_{L_2}^{t_2,t_b} + \sum_{t_c \in T^{L_2}\cap\delta^+(t_3)} f_{L_2}^{t_3,t_c}.$$

In our solution algorithm, these valid inequalities are enumerated upfront. Their count is, in the worst case, $|T| \cdot |\mathcal{L}|$. Hence, since the count of locomotive classes among the railway carriers is usually limited (or $|\mathcal{L}|$ is usually constant among instances), one may hope for a relatively fast enumeration, dependent on the number of trains in the instance. This is confirmed by our numerical experiments, discussed in more detail in Section 5.

4.2.4 Valid cutting planes derived from the constraints related to Sunday breaks

The purpose of constraint (1.18) is to enforce that no driver works on more than c^{sunday}-many consecutive Sundays. We consider the set $T_{w,d}^{\text{sunday}}$ for each driver $d \in D$ and for each week $w \in W$. Now let us construct the graphs $G_d^{\text{sunday}} = (V_d^{\text{sunday}}, E_d^{\text{sunday}})$ for all $d \in D$, where

$$V_d^{\text{sunday}} := \{(t, L) : D^t \cap D^L = \{d\} \quad \wedge \quad \exists w \in W : t \in T_{w,d}^{\text{sunday}}\},$$

$$E_d^{\text{sunday}} := \{\{(t_1, L_1), (t_2, L_2)\}\} : t_1, t_2 \in V_d^{\text{sunday}} \quad \wedge \quad t_1 \in T_{w_1,d}^{\text{sunday}} \quad \wedge \quad t_2 \in T_{w_2,d}^{\text{sunday}}$$
$$\wedge \quad w_1, w_2 \in W \quad \wedge \quad 0 < |w_2 - w_1| \leq c^{\text{sunday}}.$$

Further, for each $d \in D$ let $\mathcal{C}_d^{\text{sunday}}$ be the set of all $(c^{\text{sunday}} + 1)-$cliques in G_d^{sunday}. Each such clique corresponds to a combination of train-locomotive assignments that would lead to a driver d working on $(c^{\text{sunday}} + 1)$ consecutive Sundays. This idea leads to a set of inequalities whose validity is easy to see.

Theorem 4.8. *The following inequalities are valid for model* (1)*:*

$$\sum_{(t,L)\in C} \sum_{t_1 \in \delta^+(t) \cap T^L} f_L^{t,t_1} \quad \leq \quad c^{\text{sunday}} \quad (\forall d \in D) \ (\forall C \in \mathcal{C}_d^{\text{sunday}}). \tag{2.9}$$

To better illustrate the situations in which the cut introduced above is applied, we present the following example.

Example 4.9. *Consider a subset $\{t_1, t_2, t_3, t_4\}$ of a larger set of trains T. We assume that the set of trains T spans over one month with four Sundays and that t_1 is scheduled to run on the first Sunday of the month, t_2 is planned to run on the second Sunday, t_3 will run on the third Sunday, and finally t_4 is expected on the fourth Sunday.*

Moreover, assume that the parameter c^{sunday} takes the value of 3. Now, assume that locomotives of four types L_1, L_2, L_3, L_4 are assigned to the trains $t_1, \ldots, t_4$, respectively, and that – as a result – there is only one driver d capable of serving the chosen assignments, i.e. $D^{L_1} \cap D^{t_1} = \{d\}$, $D^{L_2} \cap D^{t_2} = \{d\}$, $D^{L_3} \cap D^{t_3} = \{d\}$, $D^{L_4} \cap D^{t_4} = \{d\}$.

Such a locomotive assignment will result in an infeasibility in constraint (1.18) *– we set the maximal number of working Sundays (represented by parameter c^{sunday}) to three. To prevent such infeasibilities from occurring, we add the cut of the form* (2.9)*. In the presented context, the inequality would look as follows:*

$$\sum_{t_a \in \delta^+(t_1) \cap T^{L_1}} f_{L_1}^{t_1,t_a} + \sum_{t_b \in \delta^+(t_2) \cap T^{L_2}} f_{L_2}^{t_2,t_b} + \sum_{t_c \in \delta^+(t_3) \cap T^{L_3}} f_{L_3}^{t_3,t_c} + \sum_{t_d \in \delta^+(t_4) \cap T^{L_4}} f_{L_4}^{t_4,t_d} \quad \leq \quad c^{\text{sunday}} = 3.$$

As in all other cases, these valid inequalities are enumerated upfront as well. In the worst case, their number will be $|D| \cdot |\mathcal{C}_d^{\text{sunday}}|$. While the former depends directly on the instance size, the latter requires maximal clique enumeration, which could be expensive (see Section 4.2.1 for a more extensive discussion). However, in our computational experiments the enumeration of the valid inequalities turned out to take only little time.

In the presence of further working time constraints or objectives which are not considered in this article, (e.g. shift fairness, avoidance of night shifts etc.), the decomposition scheme will still work, since its core principle consists in excluding locomotive assignments which cause infeasibilities in the driver schedules. Our approach allows for a "translation" of driver-specific constraints into constraints which are valid for the locomotive master problem. Moreover, we would like to note that the new inequalities introduced in this section do not affect the validity of Observation 4.1, since they are valid with regard to the integrated model (1). Note that some more complex working time requirements may require an introduction of additional variables and constraints in the driver subproblem, for which one then has to find a way to transform them into new cutting planes for the locomotive master problem.

4.3 Preprocessing the driver subproblem

In this section, we will describe two preprocessing mechanisms we use to reduce the size of the driver subproblem after the locomotive master problem has been solved and to simplify the solution process of the driver subproblem.

Removing unnecessary driver-related variables As the assignment of locomotive-classes to the trains has already been performed by the locomotive master problem, part of the variables relevant to the train-driver assignment can be eliminated – in particular the variables which pertain to drivers who are unable to drive both a given train and its assigned locomotive.

Let $(\bar{f}, \bar{\lambda})$ be a solution to the locomotive master problem. Using that solution, for all trains $t \in T$ we can enumerate the subset of drivers who are compatible with both the train t and the locomotive class $L \in \mathcal{L}^t$ which was assigned to it. More formally, for all $t \in T$ let us denote the locomotive class selected to perform the train t by

$$\bar{L}^t := L, \text{ where } L \text{ is the unique } L \in \mathcal{L}^t \text{ with } \sum_{t_1 \in \delta^+(t)} \bar{f}_L^{t,t_1} = 1.$$

Now as we know the selected locomotive class for each train $t \in T$, we can introduce the corresponding set of feasible drivers as

$$\bar{D}^t := D^t \cap D^{\bar{L}^t}.$$

With these sets, we can now precisely generate those variables x, y, z, v, α and ω for all trains $t \in T$ and for all drivers $d \in \bar{D}^t$ compatible with both the train and the selected locomotive. This means we can use the solution of the locomotive master problem to reduce the number of variables generated. Furthermore, those constraints of the driver subproblem which only include variables for drivers $D^t \setminus \bar{D}^t$ on the left-hand side become trivial and can be eliminated as well.

Changing the sense of one of the constraints Recall the multiple-choice constraints (1.4) for the driver assignment, which are part of the driver subproblem. As the locomotive master problem already determines which trains shall be performed, we can change the optimization sense of the driver subproblem from maximization (of objective function (1.1)) into a mere feasibility problem if we change constraint (1.4) into an equation as follows:

$$\sum_{d \in \bar{D}^t} x_d^t \quad = \quad 1 \quad (\forall t \in T). \tag{1.4b}$$

This reformulation has proved to be computationally more efficient than the original, maximization version of the driver subproblem.

4.4 A long-break heuristic

Despite of the preprocessing, the driver subproblem can be still be difficult to solve. Therefore, we now describe a heuristic which is able to find feasible solutions more quickly in many cases. It is based on solving an auxiliary MIP including constraints (1.10b), (1.15), (1.17) and (1.18), which make sure that the drivers' requirements with regard to longer breaks (i.e. Sundays off and the long breaks) are respected. In our computational experiments, we saw that the driver subproblem was significantly easier to solve when these constraints had been relaxed. Accordingly, if we decide upfront which of the Sundays shall be a free day for each driver and after which train a long break shall begin, the solution to the remainder of the driver subproblem usually becomes much easier. To be precise, the mentioned auxiliary MIP maximizes objective function (1.1) over the above-mentioned constraints (1.10b), (1.15), (1.17) and (1.18), as well as the following two constraints:

$$\sum_{d \in \bar{D}^t} x_d^t \quad \geq \quad \left\lfloor \frac{|\bar{D}^t|}{2} \right\rfloor + 1 \qquad (\forall t \in T) \tag{3.1}$$

$$\sum_{t \in T_{w,d}^{\text{week}}} z_d^t \quad \geq \quad 1 \qquad (\forall w \in W)(\forall d \in D). \tag{3.2}$$

The purpose of constraint (3.1) is to ensure that there is sufficient choice of drivers for each train after fixing the breaks – we want to make sure that slightly more than half the drivers are available to the restricted driver subproblem solved afterwards, which empirically proved to be an adequate value. Constraint (3.2) is required to make sure that at least one long break for each week $w \in W$ is selected for each driver $d \in D$ (even if the driver was not pre-assigned to any train via constraint (3.1)). An optimal solution $(\bar{x}, \bar{z}, \bar{h})$ to this auxiliary model contains three important pieces of information for the driver subproblem. Firstly, it determines which

Sunday shall be off for each driver in the given month. Secondly, it fixes the points in time when the weekly 35-hour breaks of the drivers start. But most importantly, it indicates for all drivers $d \in D$ which trains $t \in T^d$ cannot be served when respecting the selected Sunday or 35-hour breaks. Based on the latter, we restrict the set of available drivers for a given train $t \in T$ to

$$D^t_{\text{available}} \coloneqq \{d \in \bar{D}^t : \bar{x}^t_d = 1\}.$$

If the above auxiliary MIP is feasible, we heuristically preprocess out further x-, y-, v-, α- and ω-variables accordingly and fix the decisions concerning long breaks to obtain a restricted driver subproblem. Should the heuristic preprocessing fail (i.e. either auxiliary MIP or restricted driver subproblem are infeasible), which it seldom did in our experiments, we instead solve the full, unrestricted driver subproblem directly.

We would like to point out that in more general settings, the efficiency of the heuristic introduced in this section – in its current shape – will probably decrease with more working time constraints. However, it is straightforward to extend it to accommodate other constraint types and this way to adjust its performance.

5 A real-world case study for Polish rail freight traffic

In this section, we demonstrate the efficiency of our methods for solving the joint locomotive scheduling and driver assignment problem in a real-world case study. We begin by describing how we will assess the enhancements to the solution algorithm presented in Section 4 in terms of reducing the computation times and mention some of the details which lead to a well-performing implementation. Then we describe the case study itself, performed at the hand of a country-wide problem instance stemming from Polish rail freight traffic. We also mention the values assigned to parameters critical to the generation of the working time constraints. After a description of the input data provided by our industry partner DB Cargo Polska, we analyse both the solution times of our algorithmic approaches and the quality of the computed solutions in terms of covering the order-book as far as possible.

To further show the efficiency of our approach, we also present how the number of trains running in the optimal solution changes under less favorable resource availability conditions.

5.1 Implementation details

We have run all the experiments presented in Section 5 on a compute server with two Intel Xeon E5-2643 v4 processors using all 12 cores and 256 GB of memory. Further, we have used Gurobi 9.5 (Gurobi Optimization, LLC (2022)) to solve the arising binary optimization problems. The models are built and solved via its Python interface. Finally, we have used NetworkX (Hagberg et al. (2008)) to represent the underlying graph structures.

Recall that for generating constraints (1.7b)–(1.10b) in the improved version of model (1), we need to find minimal clique edge covers in certain graphs. As this problem is NP-hard in general (see Kou et al. (1978)), we use the maximal-clique enumeration algorithm developed by Bron and Kerbosch (1973) as adapted by Tomita et al. (2006) and discussed in Cazals and Karande (2008) to solve it heuristically. This algorithm is implemented in the Python NetworkX package (see Hagberg et al. (2008)). The choice of this method is justified by its good running time behaviour (Tomita et al. (2006) showed that its worst-case time complexity amounts to $\mathcal{O}(3^{n/3})$ for a graph with n nodes), its ease of use as part of our implementation as well as the good results we obtained with it.

To further decrease model generation times, we also use a slightly simplified approach to the clique tightening (as described in Subsection 3.3). Namely, each of the graphs $\mathcal{G}^d_{\text{time}}$, $\mathcal{G}^d_{\text{back\_break}}$, $\mathcal{G}^d_{\text{forward\_break}}$ and $\mathcal{G}^d_{\text{long\_break}}$, generated for each driver $d \in D$, contains only trains relevant to the driver d as its nodes. In our implementation, we generate the graphs G_{time}, $G_{\text{back\_break}}$, $G_{\text{forward\_break}}$ and $G_{\text{long\_break}}$ introduced in Subsection 4.2 instead, which contain all trains $t \in T$. Then we use the maximum-clique algorithm mentioned in the previous paragraph to enumerate all the maximal cliques of these larger graphs. For each of the four sets of maximal cliques, we generate an adjusted version of the cliques for each driver $d \in D$ by removing the trains which are not compatible with the driver d. This simplification allows us to only perform the clique enumeration four times instead of $4 \cdot |D|$ times. As a result, we obtain a significant decrease in model generation time, at the expense of a slightly higher number of constraints generated.

In order to assess the impact of each of the improvements we developed on top of our basic solution approach, we will perform numerical experiments subdivided into eight different computational scenarios~~groups~~. A concise summary of each of these computational scenarios is presented in Table 4. Its first column states the names of the four main algorithmic improvements introduced in this work (with a reference to Subsection, in which the improvement is introduced). Columns 2-9 represent the computational scenarios introduced. An "X" denotes that an algorithmic improvement is included in a computational scenario, whereas a dash denotes the opposite. For a detailed description of the scenarios, we refer the reader to Appendix B.~~In the following, we describe each of these computational scenarios in detail.~~

Improvement	NoClq	Clq	D	D-NoClq	D-NoCut	D+H	D+H-NoClq	D+H-NoCut
Clique Tightening (3.3)	-	X	X	-	X	X	-	X
Decomposition (4.1)	-	-	X	X	X	X	X	X
Cutting Planes (4.2)	-	-	X	X	-	X	X	-
Presolve Heuristic (4.4)	-	-	-	-	-	X	X	X

Table 4: Summary of the algorithmic improvements included in the computational scenarios

5.2 A case study for February 2020 at DB Cargo Polska

In this section, we will present a case study based on the data supplied by our industrial partner. We will begin by describing the instance we generated from the data. Then we will present the results of the numerical experiments performed on these instances.

Our industry partner provided us with a high-quality real-world data set for the problem, covering a full month of planning (for February 2020). Firstly, it comprises the complete order-book for this month, i.e. a list of all the trains that need to be run, including their origin and destination stations, departure and arrival times and the respective calculation weeks they are counted to. The data set covers four calculation weeks, which always start on a Saturday and last until the next Friday. Further, we have obtained the full list of drivers, together with their respective licenses to locomotives, knowledge of routes and home regions. We were also provided with the list of available locomotives, stating their respective tractive power and energy source, as well as a mapping of stations to regions in Poland. To estimate the travel times for the assumed "car rides" of drivers between stations (as discussed in Subsection 3.1.1), we used the data available on 14 February 2020 from the Google Maps API.

To reflect the requirements of the Polish Labour Code and Polish Railway Transport Act, our industrial partner gave us the following values for parameters used in the generation of sets required for constraints construction. They are presented in Table 5.

Name	Value
c^{shift}	12
c^{short}	11
c^{long}	35
c^{sunday}	3

Table 5: Values for parameters of the working time constraints used in the case study

Since the data about the route knowledge of the drivers was limited to only inner-Polish routes, we restrict ourselves to trains departing and arriving in the territory of Poland. In cases where a given train was to terminate at the first station past the Polish border, we artificially shortened the route to the passed border station on the Polish side. The trains which only "commute" between two neighbouring border terminals of Poland and a neighbouring country were not taken into account, since they are rather to be considered as shunting connections, which are not in the focus of this work. In total, we needed to sort out 390 out of 2941 trains, which leaves us with a total of 2551 trains to be served.

Based on the data received from the industrial partner, we derived ten problem instances corresponding to different planning horizons, ranging in length from one week to the full month. In each instance, we have assumed all 217 drivers and 112 locomotives of DB Cargo Polska to be available. Table 6 below presents a summary of the instances. Their names correspond to the time period they entail (e.g. 1M – the whole month, 1W_1 – the first week of the month, 2W_3 – the third two-week period of the month, i.e. weeks 3 and 4, and so on). Note that for the instances spanning less than one month, we excluded the h-variables and the constraints pertaining to Sunday breaks.

Instance	#Days	#Trains	Avg. model generation time (in s)
1W_1	7	629	298
1W_2	7	610	278
1W_3	7	615	299
1W_4	7	613	288
2W_1	14	1239	737
2W_2	14	1242	744
2W_3	14	1228	744
3W_1	21	1854	1325
3W_2	21	1838	1313
1M	29	2551	2154

Table 6: Overview of instance characteristics and average model generation times

With a growing instance size, the computational complexity of model generation increases exponentially due to the effort required to perform the clique tightening. Especially for the larger instances, the model generation is a challenging task on its own. Here we report arithmetic averages of the model generation times over three runs, since they proved to be similar regardless of the chosen version of the algorithm.

5.2.1 Analysis of solution times

For the computational experiments, we have used the instances from Table 6 and compare their solutions times for the three different implementation scenarios described in Subsection 5.1. We have performed three runs of each implementation on each instance and report the average solution times in Table 7.

Instance	NoClq	Clq	D	D-NoCut	D-NoClq	D+H	D+H-NoCut	D+H-NoClq
1W_1	1610	3042‡	117	121	276	**74**	249	-
1W_2	878	6312†	199	106	262	**60**	188	-
1W_3	1950	3474‡	316	237	254	**73**	144	-
1W_4	1086	2989†	178	213	230	**77**	196	-
2W_1	-	-	1464	1651	654	**522**	1314	-
2W_2	-	-	1253	1608	602	**309**	668	-
2W_3	-	-	921	1133	638	**166**	604	-
3W_1	-	-	3739	3771	1269	**650**	1693	-
3W_2	-	-	804	1790	1263	**668**	1951	-
1M	-	-	2674†	-	2680	**2397**	4085	-

Table 7: Solution times of the different implementations for each instance (in seconds).

The first column presents the name of the instance. Columns 2 to 9 display the performance of the respective implementation against the listed instances. A dash denotes that no feasible solution has been found within the time limit of 7200 seconds for all three runs of each scenario-instance combination. A dagger (†) means that the presented result pertains to only two of the computations and that the remaining one timed out. Similarly, a double dagger (‡) indicates that the presented result pertains to only one of the computations and that the remaining two timed out. An integer number indicates that all three experiments for a given scenario-instance pair have returned an optimal solution within the allotted time, and also represents the average time in seconds required to find an optimal solution A number in **bold** denotes the best solution time for each instance.

Overall, it is obvious from the table that the use of the decomposition approach is necessary to obtain a solution for the largest instances. The joint model is only able to solve the instances spanning one week to optimality, but never in all three runs. For the longer planning horizons, it returned no integer-feasible solution within the allotted time. We can also observe that employing the long-break heuristic has produced significant speed-ups in the solution times.

Furthermore, it can be deduced from Table 6 that the exclusion of valid inequalities introduced in Section 4.2 leads to a deterioration of the solution times. When comparing D+H with D+H-NoCut, this phenomenon can be observed for all instances. We can also observe a similar pattern for D and D-NoCut – the latter is unable to solve the largest instance, it also performs worse for instances spanning two and three weeks. Looking at the weekly instances, D is better than D-NoCut for two instances, and the opposite is true for the other two instances.

Moreover, the results presented in Table 6 underline the usefulness of clique tightening for our solution algorithm. Our algorithm performs the best when both the custom cuts and our clique tightening are activated. Although scenario D-NoClq performs worse than D for four out of ten scenarios only, it is vital to ensure that the presolve heuristic introduced in Subsection 4.4 can succeed. This is visible in the results for the scenario D+H-NoClq, for which none of the instances could be solved – it is due to the excessively long solution times of the presolve heuristic, caused by the lack of the tightened clique constraints.

When the model is solved directly (scenarios NoClq and Clq), Gurobi's built-in clique tightening performs better for the instances spanning one week. For longertime horizons, none of these two methods is able to come up with a solution within the allotted time.

The results presented in Table 6 clearly show that the complete version of the algorithm introduced in this work solves the instances provided by DB Cargo Polska in the shortest time.

These solution times even exceeded the expectations of our industrial partner, for whom it would already have been sufficient to obtain a monthly schedule in less than 12 hours. For shorter planning horizons, they are even short enough to allow for an interactive use, e.g. for performing what-if analyses.

Finally, it should be mentioned that for all the experiments presented in this subsection, already one iteration of the algorithm was sufficient. Moreover, since the instances span over realistic resource availabilities, there was no need to generate the valid inequalities discussed in Section 4.2.

5.2.2 Assessment of solution quality

In Table 8, we compare the number of trains in each instance to the number of trains served in the optimal solution. The first column states the name of the instance, with the second column repeating the corresponding

Instance	#Trains	Optimum	Coverage
1W_1	629	629	100.0%
1W_2	610	610	100.0%
1W_3	615	615	100.0%
1W_4	613	613	100.0%
2W_1	1239	1238	99.9%
2W_2	1242	1224	98.6%
2W_3	1228	1228	100.0%
3W_1	1854	1845	99.5%
3W_2	1838	1829	99.5%
1M	2551	2529	99.1%

Table 8: Comparison of the number of trains in each instance and the coverage in its optimal solution

number of trains in that instance. The third column presents the globally optimal solution obtained via the decomposition approach developed in this work. Finally, the fourth column gives the coverage obtained with this solution, computed as the number of the trains performed in the optimal solution, divided by the total number of trains in the instance. Overall, we can conclude that the solutions our approach produces enable the railway company to perform all or nearly all trains in the order-book, regardless of the time horizon of the instance.

5.3 Performance of the method under less favourable conditions

To better evaluate the method introduced here, we will now test its performance against a set of instances derived from the 1M instance with limited availability of drivers and locomotives for the trains. We will consider two scenarios. In the first one, we will explore how the optima returned by our algorithm change when the least used resources are being removed. The second scenario will consider an upfront removal of random locomotives and drivers. For the numerical experiments discussed in this section, we have used the scenario D+H. This choice is justified by the best performance of this scenario, as shown in Table 6.

5.3.1 Exploring the optimal values with a decreasing number of locomotives and drivers

In the first scenario, we will explore how the optima change when only the assets which have performed the most work are kept in the asset base.

Description of the scenario In this scenario, we run our algorithm against the 1M instance in an iterative fashion. We begin with all the resources available. For each consecutive run, we remove approx. 10% of the least used resources (i.e. 11 locomotives and 21 drivers), resolving draws randomly. For locomotives, we ensure that at least one locomotive of each type remains present throughout the experiment. Since the setting involved an iterative approach, we let the solution process run for 24 hours, a maximum allowed by the compute cluster we used for our experiments.

Discussion of the results Table 9 presents the impact of iteratively removing the least used assets on the value of the optimal solution in the 1M instance. The first column denotes the sequence number of the iteration of the experiment. In this context, an iteration corresponds to one run of our algorithm on the (modified) 1M instance. Iteration 0 means that the algorithm runs with all the resources available. The second and third columns represent the number of locomotives resp. drivers in the consecutive iterations. Similarly, the fourth and fifth column represent the share of locomotives and drivers available in each instance to their respective total numbers. The sixth column represents the value of the objective function after each iteration, while the seventh presents it as a percentage of all trains in the order-book. In the seventh column, the solution times for each iteration are shown. Since the model generation times are similar to the ones reported in Subsection 5.2, we do not display them here.

Iteration	# Locos	# Drivers	% Locos	% Drivers	Value	% Served trains	Time (s)
0	112	217	100	100	2529	99.1	395
1	101	196	90	90	2527	99.1	1176
2	90	175	80	81	2519	98.7	6437
3	79	154	71	71	2504	98.2	936
4	68	133	61	61	2488	97.5	1296
5	57	112	51	52	2460	96.4	48214
6	46	91	42	42	2431	95.3	7222

Table 9: Changes in optimal value with decreasing availability of drivers and locomotives

Table 9 clearly shows that, despite decreasing resource availability by up to approx. 60%, solutions which cover some 95% of the order-book may still be achieved. Such an assignment is much more efficient due to the much higher utilization of the employed locomotives and drivers as well as the more economic choice of trains to serve. Therefore, the presented procedure may well serve as a generator of a core plan which may be used as a starting point for the creation of a full plan in the company. The planner could use the solution from Iteration 6 (as presented in Table 9) and use the remaining assets to power as many of the unserved trains as desirable. Further, the freed-up assets could be assigned to tasks which are not included in the instance. This may help determine the maintenance plan – unused locos may now be scheduled for (long-term) maintenance. With regard to the algorithmic performance, we also remark that for the presented results no combinatorial Benders cuts had to be added for all iterations. Moreover, since the considered resource availabilities are not extremely tight, there was no need to add the valid inequalities discussed in Section 4.2. When reducing the availability of locomotives and drivers below 40%, the instances could not be solved within the limit.

5.3.2 Limiting the availability of drivers and locomotives upfront

In the second scenario, we will consider the impact of decreasing the asset base by removing a certain percentage of randomly selected assets before proceeding with the optimization.

Description of the scenario In this scenario, we again run our algorithm against the order-book of the 1M instance. We test the performance of the method when the availability of the resources is limited. To this end, we limit the availability of assets in steps of 10%. The reduction of the locomotive availability is performed by randomly removing locomotives from the original data. With regard to drivers, we simulate holiday patterns which could be requested by the drivers affected. These include requesting one or two periods off in a month. If one period is selected, it may last 1, 2, 3, 7, 14, 21 or 28 days. If two periods off are requested, each one may last 1, 2, 3, 7 or 14 days. We randomly select the subset of drivers whose availability becomes limited. For these, we randomly select the number of periods of time off and then their respective duration and starting date(s).

Discussion of the solution Table 10 presents the optimal solutions to the 1M order-book with limited resource availability. Rows correspond to different availabilities of locomotives, while columns represent various degrees to which drivers are available. The numbers reported correspond to an average of the discussed result from three runs. If a dagger (†) is shown next to a number, it means that the number is an average of two results, since one experiment failed. If there is no number in a cell, then no experiment for such a configuration was conducted. Such a table structure follows throughout this paragraph.

Table 10 itself proves the versatility of the method we introduce in this article – even the instances with the smallest resource availabilities were solved to optimality. As the number of locomotives available decreases, so does the optimum. Even with only 20% of the locomotives and all drivers, the algorithm can still cover the order-book to a level of 77% (1962 out of 2551 trains). Conversely, with all the locomotives present and a decreasing availability of the drivers, the optimum does not change. Instances with smaller percentages of drivers did not solve to optimality within the allotted three hours time. For the much more realistic case of "proportional" decreases (equal shares of the original locomotive and driver sets), the optima returned for successful experiments do not differ from the scenario in which only locomotive set was reduced for all but one case.

Table 11 presents the average solution times across the instances. We may conclude that removing drivers complicates the model much more than removing locomotives and generally contributes to longer solution times. We may also conclude that solution times vary greatly for similar settings – for example, while the instance comprising 60% of drivers and 60% of locomotives solves to optimality in 1113 seconds on average, a more "relaxed" instance consisting of 60% of drivers and 100% of locomotives requires 1377 seconds on average.

With regard to cuts of type (2.7), as well as to the combinatorial Benders cuts, none of the instances considered required their addition. Additionally, for all of the successful experiments, the long-break heuristic turned out to be successful as well. Hence, we do not present these results in a tabular form.

Overall, our computational results clearly show that our method is capable of solving joint locomotive scheduling and driver roster problems on a real-world scale.

% of locomotives remaining	% of drivers remaining: **10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	1359									1359
20		1962								1962
30			2272							2272
40				2417						2438
50					2459					2459
60						2484				2484
70							250			2505
80								2519		2519
90									2522†	2522
100	2529	2529	2529	2529	2529	2529	2529	2529	2529	2529

Table 10: Upfront resource availability reduction – optimal values

% of locomotives remaining	% of drivers remaining: **10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
10	322									449
20		841								958
30			932							916
40				1428						1288
50					989					1088
60						1113				976
70							1314			1314
80								983		847
90									1306†	2430
100	1791	3195	1542	2017	2624	1377	1470	1348	1494	1377

Table 11: Upfront resource availability reduction – average model solution times (in seconds)

To conclude, we would like to add that the results presented in this section were satisfying to the industrial partner. This is due to the fact that our algorithm is capable of finding both a locomotive and a driver for all (or almost all) trains in any given orderbook. In case not all trains are running, the few remaining trains can likely be served by repositioning locomotives in a similar fashion as we did with the drivers. Together with the relatively short solution times we achieve, our approach can prove very favourable if integrated in a decision support tool, where planners can manually add such repositionings ("deadheading") or modify the input data by e.g. slightly modifying the departure times of trains to make sure that all of them can be run.

An example illustration of a schedule assigned to a locomotive and a shift assigned to a driver can be found in Appendix C.

6 Conclusions

We presented mathematical methods for the integrated optimization of locomotive scheduling and driver rostering in rail freight transport. Our aim was to serve as many of the scheduled trains as possible without making use of additional empty runs. To the best of our knowledge, we introduce the first comprehensive model for this integrated planning task, representing in particular the working time requirements of the drivers in a very complete fashion. We are able to solve the problem for large, countrywide instances supplied by a major player in the Polish market, DB Cargo Polska. This was possible by strengthening the initial formulation of the model, together with developing an exact decomposition approach which allows for a sequential solution of the problem. Problem-specific as well as general purpose valid inequality classes ensure the feasibility of solutions to the decomposed subproblems. To further decrease computation times, we devised an additional presolve heuristic. Our results show that the optimal solutions we obtained in many cases allow to run all of the scheduled trains. Further, these solutions can be obtained for planning horizons spanning up to one month in comparably short time (overnight). Altogether, the obtained results point to a very beneficial use of our methods in practice, in particular to significantly simplify the current planning process at rail freight carriers.

Acknowledgements

We would like to express our gratitude to Michał Batko, Jacek Babiuch, Piotr Cieśliński, Mariusz Jankiewicz, Tomasz Kleszcz, Grzegorz Studnik, Miłosz Warda and their teams at DB Cargo Polska for all their support. We also thank our Master's student Linda Schneider for her support with the literature overview. This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 765374 (ROMSOC). This research was further supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics – Data – Applications (ADA-Center) within the framework of "BAYERN DIGITAL II" (20-3410-2-9-8).

A Set definitions

In order to define all the index sets required for the construction of the constraints of model (1) in Section 3, we first need several real-valued parameters which are train-, week- or driver-specific. They are summarized in Table 12. In the definitions below, we use the constant $h \in \mathbb{R}$ to denote a real number equivalent to one hour.

Parameter	Definition
$s(t)$	Starting time of a train $t \in T$
$e(t)$	Ending time of a train $t \in T$
$s^{\text{week}}(w)$	Starting time of a calculation week $w \in W$
$e^{\text{week}}(w)$	Ending time of a calculation week $w \in W$
$s^{\text{sunday}}(w)$	Starting time of a Sunday falling in week $w \in W$
$e^{\text{sunday}}(w)$	Ending time of a Sunday falling in week $w \in W$
$o(t)$	Origin station of a train $t \in T$
$a(t)$	Destination station of a train $t \in T$
τ^{s_1,s_2}	Transit time between stations $s_1, s_2 \in S$
c^{shift}	Maximal duration of shift in hours
c^{short}	Minimal duration of short break in hours
c^{long}	Minimal duration of long break in hours
c^{sunday}	Maximal count of working Sundays in a month

Table 12: Summary of parameters required for set construction

Many of the parameters included in Table 12 were introduced earlier. We will now briefly comment on the five parameters, which were not introduced earlier:

- Parameters $s^{\text{week}}(w)$ and $e^{\text{week}}(w)$ denote the starting time and the ending time of a calculation week $w \in W$ respectively. They are necessary to build sets required for the construction of constraints (1.15).
- Parameters $s^{\text{sunday}}(w)$ and $e^{\text{sunday}}(w)$ denote the starting time and the ending time of a Sunday falling in the calculation week $w \in W$ respectively. They are necessary to build sets required for the construction of constraints (1.17).
- Parameter τ^{s_1,s_2} denotes the ransit time between stations $s_1, s_2 \in S$. The role of transit times is to reflect the possibility to transport a train driver to the station, from which the first train in their shift departs, by car. This is described in greater detail in Subsection 3.1.1.

In the paragraphs below, a definition of each index set used in model (1) is given.

Forward-looking daily break set $T^{\text{B+}}_{t,d}$ The set $T^{\text{B+}}_{t,d}$ is used to represent, for each driver $d \in D$ and for all $t \in T^d$, the trains $t_1 \in T^d$ which cannot be assigned to driver d if t is the last train in the shift before a short break. We can formally state the set $d \in D$ and $t \in T^d$ as

$$
\begin{aligned}
T^{\text{B+}}_{t,d} &:= \{t_1 \in T^d : s(t_1) \geq s(t) \ \wedge \ s(t_1) \leq e(t) + c^{\text{short}}\} \\
&\quad \cup \{t_1 \in T^d : s(t_1) > s(t) \wedge e(t) + c^{\text{short}} + \tau^{a(t),o(t_1)} > s(t_1)\}.
\end{aligned}
$$

The capital letter B used in the name of this set (as well as in the name of the set $T^{\text{B}-}_{t,d}$ introduced below) point towards the fact that its contains the trains which would violate a break constraint if they were assigned to driver d along with train t. This entails both trains departing before or after train t.

Weekly break set $T^{\textbf{LB+}}_{t,d}$ For each driver $d \in D$ and train $t \in T^d$ we require the set $T^{\text{LB+}}_{t,d}$ to denote those trains $t_1 \in T^d$ which cannot be assigned to driver d if train t is the last job before a long break. Formally, we define this set as:

$$\begin{aligned} T^{\text{LB+}}_{t,d} &:= \{t_1 \in T^d : s(t_1) \geq s(t) \ \wedge \ s(t_1) \leq e(t) + c^{\text{long}}\} \\ &\quad \cup \{t_1 \in T^d : s(t_1) > s(t) \ \wedge \ e(t) + c^{\text{long}} + \tau^{a(t),o(t_1)} > s(t_1)\} \end{aligned}$$

for all $d \in D$ and $t \in T^d$. For our computational experiments, we used a heuristic version of this set as it allowed for a simpler implementation. It is different from the above definition by the fact that instead of using exact transportation times between stations, we use maximal transportation time between stations which are the origin and destination stations of trains $t \in T$. Since the heuristic version of the set includes more trains than necessary, it is actually to the benefit of the drivers – their breaks could potentially be longer than the assumed c^{long} hours plus the transportation time. This restriction still allowed us to obtain optimal solutions for all the instances in our real-world case study, as we could deduce from a comparison of the number of trains covered by the locomotive master solution and the driver subproblem solution respectively.

Backward-looking trains blocked set $T^{\text{B}-}_{t,d}$ In the set $T^{\text{B}-}_{t,d}$, $d \in D$ and $t \in T^d$, we group together all trains $t_1 \in T^d$ which cannot be served by the driver d if train t is selected to be the first in one of the shifts of driver d. More formally, we have

$$\begin{aligned} T^{\text{B}-}_{t,d} &:= \{t_1 \in T^d : \ e(t_1) \geq s(t) - c^{\text{short}} \ \wedge \ e(t_1) \leq s(t)\} \\ &\quad \cup \{t_1 \in T^d : s(t_1) < s(t) \ \wedge \ s(t) - c^{\text{short}} - \tau^{a(t_1),o(t)} < e(t_1)\} \end{aligned}$$

for all $d \in D$ and $t \in T^d$.

Long break beginning set $T^{\text{week}}_{w,d}$ For every driver $d \in D$ and each week $w \in W$, we construct a set $T^{\text{week}}_{w,d}$ to collect those trains $t \in T^d$ which belong to calculation week w and which could serve as the last one before a long break of driver d. We need to make sure that at least one long break is scheduled in each calculation week. Hence, $T^{\text{week}}_{w,d}$ shall contain all the jobs $t \in T^d$ which end more than c^{long} hours before the end of the calculation week.

$$T^{\text{week}}_{w,d} := \{t \in T^d : e(t) \geq s^{\text{week}}(w) \ \wedge \ e(t) + c^{\text{long}} \ \leq \ e^{\text{week}}(w)\}$$

for all $w \in W$ and $d \in D$.

Calculation week set $T_{w,d}^{\text{week\_assignment}}$ For each calculation week $w \in W$ and for each driver $d \in D$, we need the set $T_{w,d}^{\text{week\_assignment}}$ to indicate the trains $t \in T^d$ which belong to calculation week w. It can be defined as follows:

$$T_{w,d}^{\text{week\_assignment}} := \{t \in T^d : s(t) \geq s^{week}(w) \ \wedge \ e(t) \ \leq \ e^{week}(w)\}$$

for all $w \in W$ and $d \in D$.

Feasible shift beginnings and ends $T_{t,d}^{\text{shift\_end}}$ and $T_{t,d}^{\text{shift\_beginning}}$ The two sets $T_{t,d}^{\text{shift\_beginning}}$ and $T_{t,d}^{\text{shift\_end}}$ are required for all drivers $d \in D$ and all trains $t \in T^d$ in order to accumulate those trains $t_1 \in T^d$ which could have been assigned to the driver d along with train t as the first and last train in a shift respectively. For a formal definition, for each $d \in D$ and each train $t \in T^d$ let us first introduce the following set of potential next jobs:

$$\begin{aligned} T_{t,d}^{\text{next\_driver}} := & \{t_1 \in T^d : s(t_1) > e(t) \ \wedge \ e(t_1) < s(t) + c^{\text{shift}} \wedge o(t_1) = a(t))\} \\ & \cup \{t_1 \in T^d : s(t_1) > e(t) + \tau^{o_{t_1}, a(t)} \ \wedge \ e(t_1) < s(t) + c^{\text{shift}}\}. \end{aligned}$$

Based on this set, we define the directed graph $G_d^{\text{shift}} = (T^d, A_d^{\text{shift}})$ with

$$A_d^{\text{shift}} := \{(t_1, t_2) : t_1, t_2 \in T^d \wedge t_2 \in T_{t_1,d}^{\text{next\_driver}}\}.$$

Based on this graph, we formally define the two required index sets:

$$T_{t,d}^{\text{shift\_beginning}} := \{t_1 : t_1 \in \delta^-(t) \wedge s(t_1) \geq e(t) - c^{\text{shift}}\} \cup \{t\}$$

and

$$T_{t,d}^{\text{shift\_end}} := \{t_1 : t_1 \in \delta^+(t) \wedge e(t_1) \leq s(t) + c^{\text{shift}}\} \cup \{t\}$$

for all $d \in D$ and $t \in T^d$. Here, $\delta^+(t)$ and $\delta^-(t)$ denote the set of outgoing and incoming arcs of a node $t \in T^d$ respectively.

Trains in time conflict set $T_{t,d}^{\text{time}}$ Using the set $T_{t,d}^{\text{time}}$, we gather, for all drivers $d \in D$ and for each train $t \in T^d$, the trains $t_1 \in T^d$ which are in time conflict with the train $t \in T^d$. It also contains trains $t_1 \in T^d$ which may not be served by the driver d who served the train t because of an excessively long transit time required to arrive at the departure station of t_1. This will allows us to prohibit situations which would require a driver to serve two trains at the same time. Formally, we introduce this set as

$$\begin{aligned}T_{t,d}^{\text{time}} := (&\{t_1 \in T^d : s(t_1) \leq s(t) \quad \wedge \quad e(t_1) \geq e(t)\}\\ &\cup \{t_1 \in T^d : s(t_1) \geq s(t) \quad \wedge \quad s(t_1) \leq e(t) \quad \wedge \quad e(t_1) \geq e(t)\}\\ &\cup \{t_1 \in T^d : s(t_1) \geq s(t) \quad \wedge \quad e(t_1) \leq e(t)\}\\ &\cup \{t_1 \in T^d : s(t_1) \leq s(t) \quad \wedge \quad e(t_1) \leq e(t) \quad \wedge \quad e(t_1) \geq s(t)\}\\ &\cup \{t_1 \in T^d : e(t) + \tau^{a(t),o(t_1)} > s(t_1) \quad \wedge \quad e(t_1) \leq s(t) + c^{\text{shift}} \quad \wedge \quad s(t_1) > s(t)\}) \setminus \{t\}\end{aligned}$$

for all $d \in D$ and $t \in T^d$.

Trains in time conflict set $T_t^{\text{time\_global}}$ Using the set $T_t^{\text{time\_global}}$, we gather, for each train $t \in T$, the trains $t_1 \in T$ which are in time conflict with the train $t \in T$. This will allows us to enumerate all of the trains which run in parallel to the considered train t. This set also contains trains $t_1 \in T$ which may not be served by a driver $d \in D$ who served the train t because of an excessively long transit time required to arrive at the departure station of t_1. Formally, we introduce this set as

$$\begin{aligned}T_t^{\text{time\_global}} := (&\{t_1 \in T : s(t_1) \leq s(t) \quad \wedge \quad e(t_1) \geq e(t)\}\\ &\cup \{t_1 \in T : s(t_1) \geq s(t) \quad \wedge \quad s(t_1) \leq e(t) \quad \wedge \quad e(t_1) \geq e(t)\}\\ &\cup \{t_1 \in T : s(t_1) \geq s(t) \quad \wedge \quad e(t_1) \leq e(t)\}\\ &\cup \{t_1 \in T : s(t_1) \leq s(t) \quad \wedge \quad e(t_1) \leq e(t) \quad \wedge \quad e(t_1) \geq s(t)\}\\ &\cup \{t_1 \in T : e(t) + \tau^{a(t),o(t_1)} > s(t_1) \quad \wedge \quad e(t_1) \leq s(t) + c^{\text{shift}} \quad \wedge \quad s(t_1) > s(t)\}) \setminus \{t\}\end{aligned}$$

for all $t \in T$.

Feasible next shift beginnings and ends $T_{t,d}^{\text{after\_break}}$ and $T_{t,d}^{\text{before\_break}}$ For all drivers $d \in D$ and trains $t \in T^d$, these two sets $T_{t,d}^{\text{after\_break}}$ and $T_{t,d}^{\text{before\_break}}$ are used to group together the trains $t_1 \in T^d$ which can be the first jobs of the next shift after the short break following the train t and, respectively, the trains $t_1 \in T^d$ which can be the last job of the previous shift before the short break preceding train t. For a formal definition, for each $d \in D$ let us first introduce a graph $G_d^b = (T^d, A_d^b)$ with

$$\begin{aligned}A_d^b := &\{(t_1, t_2) : t_1, t_2 \in T^d \wedge o(t_2) = a(t_1) \wedge s(t_2) > e(t_1) + c^{\text{short}}\}\\ &\cup \{(t_1, t_2) : t_1, t_2 \in T^d \wedge s(t_2) > e(t_1) + \tau^{o(t_2),a(t_1)} + c^{\text{short}}\}.\end{aligned}$$

Based on this graph, we formally define

$$T_{t,d}^{\text{after\_break}} := \delta^+(t)$$

and

$$T_{t,d}^{\text{before\_break}} := \delta^-(t).$$

for all $d \in D$ and $t \in T^d$. The two sets $\delta^+(t)$ and $\delta^-(t)$ denote the outgoing and incoming arcs of a node $t \in T^d$ respectively.

Locomotive potential next trains $T_{t,L}^{\text{next}}$ For all locomotive classes $L \in \mathcal{L}$ and for all trains $t \in T^L$, the set $T_{t,L}^{\text{next}}$ is used to gather all trains $t_1 \in T^L$ which can be selected as the successors to locomotive of locomotive class L if it serves train t. It can be formally stated as

$$T_{t,L}^{\text{next}} := \{t_1 \in T^L : s(t_1) > e(t) \ \wedge \ o(t_1) = a(t)\}$$

for all $t \in T$ and $L \in \mathcal{L}^t$.

Sunday set $T_{w,d}^{\text{sunday}}$ The set $T_{w,d}^{\text{sunday}}$ is used to determine which trains $t_1 \in T$ are scheduled for the period falling between Sunday, 6:00 a.m., and Monday, 6:00 a.m. (referred to as 'Sunday') in a calculation week w for a given driver $d \in D$ in order to make sure that at least every fourth Sunday is off. We define this set as

$$\begin{aligned} T_{w,d}^{\text{sunday}} := &\{t_1 \in T^d : s(t_1) \geq s^{\text{sunday}}(w) \quad \wedge \quad e(t_1) \leq e^{\text{sunday}}(w)\} \\ &\cup \{t_1 \in T^d : s(t_1) \geq s^{\text{sunday}}(w) \quad \wedge \quad s(t_1) \leq e^{\text{sunday}}(w) \quad \wedge \quad e(t_1) \geq e^{\text{sunday}}(w)\} \\ &\cup \{t_1 \in T^d : s(t_1) \leq s^{\text{sunday}}(w) \quad \wedge \quad e(t_1) \geq s^{\text{sunday}}(w) \quad \wedge \quad e(t_1) \leq e^{\text{sunday}}(w)\} \end{aligned}$$

for all $d \in D$ and $w \in W$.

B Detailed description of the computational scenarios

B.1 Baseline scenarios

The following two scenarios attempt at solving the model (1) directly, without the use of the decomposition scheme described in Subsection 4.1 or presolve heuristic introduced in Subsection 4.4.

Original model As a first step, we will attempt at solving the initial formulation of model (1). We refer to this scenario as NoClq.

Clique tightening Here will attempt at solving the initial formulation of model (1), with the modifications described in Subsection 3.3. This means we replace constraints (1.7), (1.8), (1.9) and (1.10) with (1.7b), (1.8b), (1.9b) and (1.10b) respectively We refer to this scenario as Clq.

B.2 Scenarios based on decomposition only

The following three scenarios attempt at solving the model (1), using the decomposition scheme introduced in Subsection 4.1. They vary in the inclusion of clique tightening in Subsection 3.3 or valid inequalities introduced in Subsection 4.2. None of the scenarios introduced here uses the presolve heuristic introduced in Subsection 4.4.

Decomposition Here we consider the decomposition approach as described in Subsection 4.1. To recall, it comprises the following three steps between which the algorithm iterates until convergence:

1. Solve the locomotive master problem given by objective function (2.1) and constraints (1.23)–(1.26), (1.34), (2.2) and (2.3) extended by the additional valid inequalities developed in Subsection 4.2. To be precise, inequalities (2.7) are all added from the beginning, while inequalities (2.4), (2.6) and (2.9) are added as cutting planes until no further cutting planes are found.

2. Preprocess the driver subproblem, as discussed in Subsection 4.3.

3. Solve the driver subproblem, i.e. the feasibility problem given by constraints (1.2), (1.3), (1.4b), (1.5)–(1.6), (1.7b)–(1.10b), (1.11)–(1.22) and (1.27)–(1.33), with the f-variables fixed according to the solution of the locomotive master problem. If the problem is infeasible, we generate a combinatorial Benders cut for the locomotive master problem and iterate. Otherwise, the algorithm terminates with a globally optimal solution.

We refer to this implementation as D.

Decomposition without clique tightening Here we consider the decomposition approach as described in paragraph "Decomposition", but with the use of constraints (1.7), (1.8), (1.9) and (1.10) instead of (1.7b), (1.8b), (1.9b) and (1.10b) respectively.

We refer to this implementation as D-NoClq.

Decomposition without valid inequalities In this scenario, we consider the decomposition approach as described in paragraph "Decomposition", but excluding the valid inequalities introduced in Subsection 4.2.

We call this implementation D-NoCut.

B.3 Scenarios based on decomposition and presolve heuristic

The following three scenarios attempt at solving the model (1), using the decomposition scheme introduced in Subsection 4.1 and the presolve heuristic introduced in Subsection 4.4. They vary in the inclusion of clique tightening in Subsection 3.3 or valid inequalities introduced in Subsection 4.2.

Decomposition + Heuristic In this complete implementation of our algorithm we use the complete algorithmic scheme shown in Figure 3 on page 23. Mainly, this is implementation Decomp enhanced by the long-break heuristic described in Subsection 4.4. In particular, the procedure loops over the following four steps until convergence:

1. Solve the locomotive master problem as in implementation D.
2. Preprocess the driver subproblem, as discussed in Subsection 4.3.
3. Solve the long-break presolve heuristic for the driver subproblem described in Subsection 4.4.
4. Solve the driver subproblem as in implementation Decomp with the following modification: we do not only fix the f-variables according to the solution of the locomotive master problem, but also the $(x, y, v, z, \alpha, \omega, h)$-variables as per the solution of the long-break presolve heuristic (if it is successful, otherwise solve the full driver subproblem with only the f-variables fixed).

This implementation is called D+H.

Decomposition + Heuristic without clique tightening In this implementation, we use the complete algorithmic scheme as presented in paragraph "Decomposition + Heuristic", but with the use of constraints (1.7), (1.8), (1.9) and (1.10) instead of (1.7b), (1.8b), (1.9b) and (1.10b) respectively.

We call this implementation D+H-NoClq.

Decomposition + Heuristic without valid inequalities Here we use the complete algorithmic scheme shown in Figure 3 on page 23. It differs from implementation DECOMP+HEUR by the exclusion of the valid inequalities introduced in Subsection 4.2.

This implementation is called D+H-NOCUT.

C Example shift reports

In this section, we present two exemplary monthly train assignments – one for a locomotive and another one for a driver. For both, we present a tabular summary as well as a chart which visualize an individual monthly assignment of trains. Each of the tables includes the individual number of each train assigned, its origin and destination stations as well as the departure and arrival times. In the chart, one can see a number of bars. Each of them corresponds to one train listed in the corresponding table. The trains are plotted against a grid resembling a monthly calendar. On the top of the grid, the names of the weekdays are mentioned.

Locomotive monthly plan

Loco type and item: Heavy Diesel Locomotive

Number of shifts: 61

Train no.	From	To	Departure time	Arrival time
29	PL035	PL020	2020-02-01 06:54:59	2020-02-01 13:18:59
149	PL020	PL035	2020-02-02 19:12:00	2020-02-02 21:54:59
206	PL035	PL085	2020-02-03 13:45:00	2020-02-03 14:39:59
226	PL085	PL035	2020-02-03 19:59:59	2020-02-03 20:46:00
290	PL035	PL020	2020-02-04 11:55:00	2020-02-04 14:15:00
352	PL020	PL035	2020-02-05 03:40:59	2020-02-05 07:14:59
459	PL035	PL020	2020-02-06 06:54:59	2020-02-06 13:18:59
521	PL020	PL035	2020-02-06 19:59:59	2020-02-06 23:06:00
587	PL035	PL046	2020-02-07 10:59:59	2020-02-07 12:00:00
600	PL046	PL035	2020-02-07 15:00:00	2020-02-07 16:00:00
667	PL035	PL020	2020-02-08 09:22:00	2020-02-08 12:45:00
777	PL020	PL015	2020-02-09 19:12:00	2020-02-10 03:54:59
850	PL015	PL101	2020-02-10 18:45:59	2020-02-10 23:35:59
873	PL101	PL045	2020-02-11 00:00:59	2020-02-11 06:11:00
917	PL045	PL003	2020-02-11 12:00:00	2020-02-11 12:29:00
1002	PL003	PL022	2020-02-12 10:50:00	2020-02-12 14:17:00
1027	PL022	PL003	2020-02-12 17:30:00	2020-02-12 21:22:00
1098	PL003	PL022	2020-02-13 10:50:00	2020-02-13 14:17:00
1122	PL022	PL003	2020-02-13 17:30:00	2020-02-13 21:22:00
1144	PL003	PL033	2020-02-13 22:30:59	2020-02-13 23:10:00
1165	PL033	PL003	2020-02-14 05:10:00	2020-02-14 05:46:00
1178	PL003	PL066	2020-02-14 09:20:59	2020-02-14 10:20:00
1198	PL066	PL003	2020-02-14 13:20:00	2020-02-14 14:23:00
1206	PL003	PL070	2020-02-14 15:35:00	2020-02-14 15:51:59
1221	PL070	PL003	2020-02-14 19:36:59	2020-02-14 19:54:00
1235	PL003	PL033	2020-02-14 22:30:59	2020-02-14 23:10:00
1257	PL033	PL003	2020-02-15 05:10:00	2020-02-15 05:46:00

Continued on next page

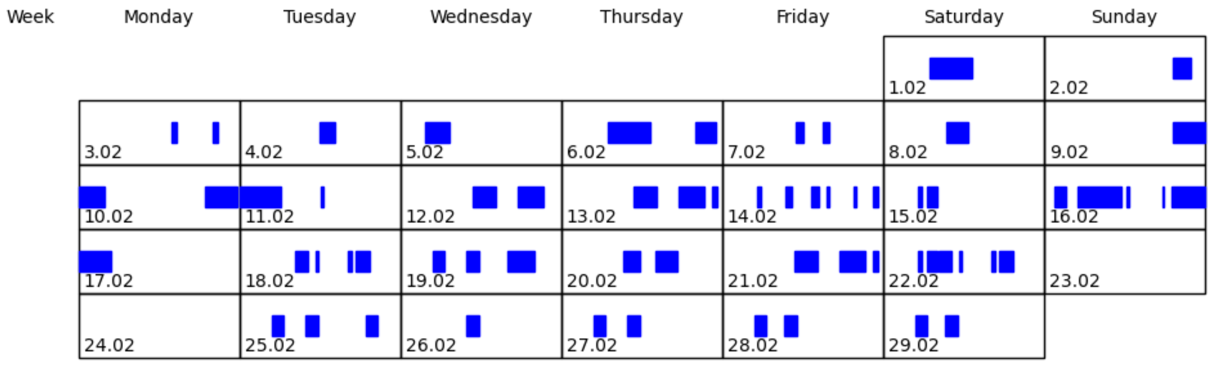
Table 13 – continued from previous page

Train no.	From	To	Departure time	Arrival time
1264	PL003	PL026	2020-02-15 06:27:59	2020-02-15 08:01:00
1327	PL026	PL003	2020-02-16 01:36:00	2020-02-16 03:13:59
1334	PL003	PL047	2020-02-16 04:54:00	2020-02-16 11:35:00
1361	PL047	PL012	2020-02-16 12:18:59	2020-02-16 12:35:00
1371	PL012	PL020	2020-02-16 17:35:00	2020-02-16 17:48:00
1375	PL020	PL026	2020-02-16 19:00:00	2020-02-17 04:50:00
1499	PL026	PL003	2020-02-18 08:12:00	2020-02-18 10:12:00
1517	PL003	PL033	2020-02-18 11:17:00	2020-02-18 11:48:00
1535	PL033	PL003	2020-02-18 16:11:00	2020-02-18 16:41:59
1539	PL003	PL026	2020-02-18 17:21:59	2020-02-18 19:18:59
1586	PL026	PL003	2020-02-19 04:43:59	2020-02-19 06:33:00
1605	PL003	PL026	2020-02-19 09:45:00	2020-02-19 11:44:59
1629	PL026	PL003	2020-02-19 16:00:00	2020-02-19 19:59:59
1696	PL003	PL080	2020-02-20 09:13:00	2020-02-20 11:41:00
1723	PL080	PL003	2020-02-20 13:59:59	2020-02-20 17:21:59
1806	PL003	PL022	2020-02-21 10:50:00	2020-02-21 14:17:00
1829	PL022	PL003	2020-02-21 17:30:00	2020-02-21 21:22:00
1850	PL003	PL033	2020-02-21 22:30:59	2020-02-21 23:10:00
1873	PL033	PL003	2020-02-22 05:10:00	2020-02-22 05:46:00
1880	PL003	PL026	2020-02-22 06:27:59	2020-02-22 08:01:00
1885	PL026	PL003	2020-02-22 08:12:00	2020-02-22 10:12:00
1901	PL003	PL033	2020-02-22 11:17:00	2020-02-22 11:48:00
1916	PL033	PL003	2020-02-22 16:11:00	2020-02-22 16:41:59
1921	PL003	PL026	2020-02-22 17:21:59	2020-02-22 19:18:59
2107	PL026	PL003	2020-02-25 04:43:59	2020-02-25 06:33:00
2123	PL003	PL026	2020-02-25 09:45:00	2020-02-25 11:44:59
2164	PL026	PL003	2020-02-25 18:53:00	2020-02-25 20:35:00
2221	PL003	PL026	2020-02-26 09:45:00	2020-02-26 11:44:59
2296	PL026	PL003	2020-02-27 04:43:59	2020-02-27 06:33:00
2316	PL003	PL026	2020-02-27 09:45:00	2020-02-27 11:44:59

Continued on next page

Table 13 – continued from previous page

Train no.	From	To	Departure time	Arrival time
2388	PL026	PL003	2020-02-28 04:43:59	2020-02-28 06:33:00
2405	PL003	PL026	2020-02-28 09:09:59	2020-02-28 11:10:59
2484	PL026	PL003	2020-02-29 04:43:59	2020-02-29 06:33:00
2501	PL003	PL026	2020-02-29 09:09:59	2020-02-29 11:10:59



Driver weekly plan

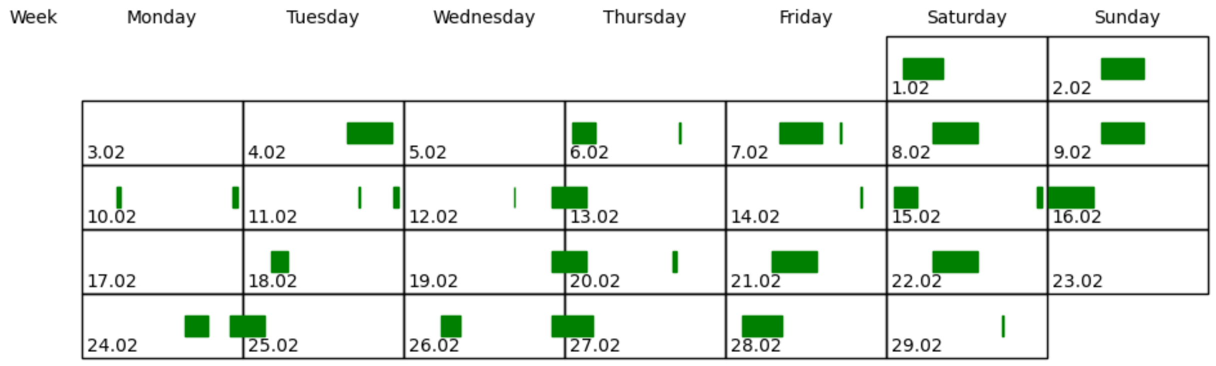
Name and surname: Driver045

Number of shifts: 24

Number of Sundays off: 1

Number of 35-hour breaks: 4

Train no.	From	To	Departure time	Arrival Time
11	PL032	PL050	2020-02-01 02:30:00	2020-02-01 08:23:00
113	PL032	PL050	2020-02-02 08:06:00	2020-02-02 14:19:00
305	PL050	PL032	2020-02-04 15:29:59	2020-02-04 22:16:00
437	PL032	PL094	2020-02-06 01:11:00	2020-02-06 04:41:59
507	PL011	PL094	2020-02-06 17:10:59	2020-02-06 17:19:00
568	PL032	PL050	2020-02-07 08:06:00	2020-02-07 14:19:00
606	PL011	PL094	2020-02-07 17:10:59	2020-02-07 17:19:00
659	PL050	PL032	2020-02-08 06:56:00	2020-02-08 13:34:00
743	PL032	PL050	2020-02-09 08:06:00	2020-02-09 14:19:00
803	PL033	PL003	2020-02-10 05:10:00	2020-02-10 05:46:00
868	PL003	PL033	2020-02-10 22:30:59	2020-02-10 23:10:00
932	PL011	PL094	2020-02-11 17:19:59	2020-02-11 17:28:00
958	PL003	PL033	2020-02-11 22:30:59	2020-02-11 23:10:00
1024	PL011	PL094	2020-02-12 16:25:00	2020-02-12 16:34:00
1052	PL032	PL050	2020-02-12 22:03:59	2020-02-13 03:11:00
1223	PL094	PL011	2020-02-14 20:10:00	2020-02-14 20:21:00
1243	PL032	PL094	2020-02-15 01:11:00	2020-02-15 04:41:59
1320	PL003	PL033	2020-02-15 22:30:59	2020-02-15 23:10:00
1324	PL050	PL032	2020-02-16 00:15:00	2020-02-16 06:51:00
1485	PL094	PL005	2020-02-18 04:14:00	2020-02-18 06:42:59
1659	PL032	PL050	2020-02-19 22:03:59	2020-02-20 03:11:00
1726	PL033	PL003	2020-02-20 16:11:00	2020-02-20 16:41:59
1786	PL050	PL032	2020-02-21 06:56:00	2020-02-21 13:34:00
1883	PL050	PL032	2020-02-22 06:56:00	2020-02-22 13:34:00
2058	PL094	PL032	2020-02-24 15:24:59	2020-02-24 18:51:59
2085	PL032	PL050	2020-02-24 22:03:59	2020-02-25 03:13:00
2206	PL094	PL032	2020-02-26 05:39:00	2020-02-26 08:26:59
2277	PL032	PL050	2020-02-26 22:03:59	2020-02-27 04:12:00
2380	PL032	PL050	2020-02-28 02:30:00	2020-02-28 08:23:00
2531	PL011	PL094	2020-02-29 17:19:59	2020-02-29 17:28:00


Week
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
1.02
2.02
3.02
4.02
5.02
6.02
7.02
8.02
9.02
10.02
11.02
12.02
13.02
14.02
15.02
16.02
17.02
18.02
19.02
20.02
21.02
22.02
23.02
24.02
25.02
26.02
27.02
28.02
29.02

References

Aksoy, A. and Altan, A. (2013). The Integrated Locomotive Assignment and Crew Scheduling Problem. *International Journal of Computational Engineering Research*, 3(8):18–24.

Amberg, B., Amberg, B., and Kliewer, N. (2011). Increasing Delay-Tolerance of Vehicle and Crew Schedules in Public Transport by Sequential, Partial-Integrated and Integrated Approaches. *Procedia - Social and Behavioral Sciences*, 20:292–301.

Amberg, B., Amberg, B., and Kliewer, N. (2019). Robust Efficiency in Urban Public Transportation: Minimizing Delay Propagation in Cost-Efficient Bus and Driver Schedules. *Transportation Science*, 53(1):89–112.

Bach, L., Dollevoet, T., and Huisman, D. (2016). Integrating Timetabling and Crew Scheduling at a Freight Railway Operator. *Transportation Science*, 50(3):878–891.

Borndörfer, R., Löbel, A., and Weider, S. (2008). A Bundle Method for Integrated Multi-Depot Vehicle and Duty Scheduling in Public Transit. In Hickman, M., Mirchandani, P., and Voß, S., editors, *Computer-aided Systems in Public Transport*, volume 600, pages 3–24. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title: Lecture Notes in Economics and Mathematical Systems.

Borndörfer, R., Sagnol, G., Schlechte, T., and Swarat, E. (2017a). Optimal duty rostering for toll enforcement inspectors. *Annals of Operations Research*, 252(2):383–406.

Borndörfer, R., Schulz, C., Seidl, S., and Weider, S. (2017b). Integration of duty scheduling and rostering to increase driver satisfaction. *Public Transport*, 9(1-2):177–191.

Boyer, V., Ibarra-Rojas, O. J., and Rios-Solis, Y. A. (2018). Vehicle and Crew Scheduling for Flexible Bus Transportation Systems. *Transportation Research Part B: Methodological*, 112:216–229.

Brito, S. S. and Santos, H. G. (2021). Preprocessing and cutting planes with conflict graphs. *Computers & Operations Research*, 128:105176.

Bron, C. and Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577. Publisher: Association for Computing Machinery (ACM).

Caprara, A., Monaci, M., and Toth, P. (2001). A Global Method for Crew Planning in Railway Applications. In Fandel, G., Trockel, W., Aliprantis, C. D., Kovenock, D., Voß, S., and Daduna, J. R., editors, *Computer-Aided Scheduling of Public Transport*, volume 505, pages 17–36. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title: Lecture Notes in Economics and Mathematical Systems.

Cazals, F. and Karande, C. (2008). A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1-3):564–568. Publisher: Elsevier BV.

Codato, G. and Fischetti, M. (2006). Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. *Operations Research*, 54(4):756–766.

Cordeau, J.-F., Desaulniers, G., Lingaya, N., Soumis, F., and Desrosiers, J. (2001a). Simultaneous locomotive and car assignment at VIA Rail Canada. *Transportation Research Part B: Methodological*, 35(8):767–787.

Cordeau, J.-F., Stojković, G., Soumis, F., and Desrosiers, J. (2001b). Benders Decomposition for Simultaneous Aircraft Routing and Crew Scheduling. *Transportation Science*, 35(4):375–388.

Cordeau, J.-F., Toth, P., and Vigo, D. (1998). A Survey of Optimization Models for Train Routing and Scheduling. *Transportation Science*, 32(4):380–404.

Dalal, M. and Jensen, L. (2001). Simulation modeling at Union Pacific Railroad. In *Proceeding of the 2001 Winter Simulation Conference (Cat. No.01CH37304)*, volume 2, pages 1048–1055, Arlington, VA, USA. IEEE.

Dauzère-Pérès, S., De Almeida, D., Guyon, O., and Benhizia, F. (2015). A Lagrangian heuristic framework for a real-life integrated planning problem of railway transportation resources. *Transportation Research Part B: Methodological*, 74:138–150.

De Leone, R., Festa, P., and Marchitto, E. (2011). A Bus Driver Scheduling Problem: a new mathematical model and a GRASP approximate solution. *Journal of Heuristics*, 17(4):441–466. Publisher: Springer.

Drexl, M., Rieck, J., Sigl, T., and Press, B. (2013). Simultaneous Vehicle and Crew Routing and Scheduling for Partial- and Full-Load Long-Distance Road Transport. *Business Research*, 6(2):242–264.

Dunbar, M., Froyland, G., and Wu, C.-L. (2014). An integrated scenario-based approach for robust aircraft routing, crew pairing and re-timing. *Computers & Operations Research*, 45:68–86.

Díaz-Ramírez, J., Huertas, J. I., and Trigos, F. (2014). Aircraft maintenance, routing, and crew scheduling planning for airlines with a single fleet and a single maintenance and crew base. *Computers & Industrial Engineering*, 75:68–78.

Dück, V., Ionescu, L., Kliewer, N., and Suhl, L. (2012). Increasing stability of crew and aircraft schedules. *Transportation Research Part C: Emerging Technologies*, 20(1):47–61.

Ernst, A., Jiang, H., Krishnamoorthy, M., Nott, H., and Sier, D. (2001). An Integrated Optimization Model for Train Crew Management. *Annals of Operations Research*, 108:211–224.

Freling, R., Huisman, D., and Wagelmans, A. P. M. (2003). Models and Algorithms for Integration of Vehicle and Crew Scheduling. *Journal of Scheduling*, 6(1):63–85.

Fulkerson, D. and Gross, O. (1965). Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855.

Gaffi, A. and Nonato, M. (1999). An Integrated Approach to Ex-Urban Crew and Vehicle Scheduling. In Fandel, G., Trockel, W., and Wilson, N. H. M., editors, *Computer-Aided Transit Scheduling*, volume 471, pages 103–128. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title: Lecture Notes in Economics and Mathematical Systems.

Goumopoulos, C. and Housos, E. (2004). Efficient trip generation with a rule modeling system for crew scheduling problems. *Journal of Systems and Software*, 69(1):43–56.

Gurobi Optimization, LLC (2022). Gurobi Optimizer Reference Manual.

Guttkuhn, R., Dawson, T., and Trutschel, U. (2003). A discrete event simulation for the crew assignment process in North American freight railroads. In *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, pages 1686–1692, New Orleans, LA, USA. IEEE.

Haase, K., Desaulniers, G., and Desrosiers, J. (2001). Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems. *Transportation Science*, 35(3):286–303.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring Network Structure, Dynamics, and Function using NetworkX. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11–15, Pasadena, CA USA.

Heil, J., Hoffmann, K., and Buscher, U. (2020). Railway crew scheduling: Models, methods and applications. *European Journal of Operational Research*, 283(2):405–425.

Hollis, B., Forbes, M., and Douglas, B. (2006). Vehicle routing and crew scheduling for metropolitan mail distribution at Australia Post. *European Journal of Operational Research*, 173(1):133–150.

Huisman, D. (2004). *Integrated and dynamic vehicle and crew scheduling.* Number 325 in Tinbergen Institute research series. Thela Thesis, Amsterdam. OCLC: 249616330.

Huisman, D., Freling, R., and Wagelmans, A. P. M. (2005). Multiple-Depot Integrated Vehicle and Crew Scheduling. *Transportation Science*, 39(4):491–502.

Huisman, D. and Wagelmans, A. P. (2006). A solution approach for dynamic vehicle and crew scheduling. *European Journal of Operational Research*, 172(2):453–471.

Ibarra-Rojas, O., Delgado, F., Giesen, R., and Muñoz, J. (2015). Planning, operation, and control of bus transport systems: A literature review. *Transportation Research Part B: Methodological*, 77:38–75.

Johnson, D. S., Yannakakis, M., and Papadimitriou, C. H. (1988). On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123.

Jütte, S. and Thonemann, U. W. (2012). Divide-and-price: A decomposition algorithm for solving large railway crew scheduling problems. *European Journal of Operational Research*, 219(2):214–223.

Khmeleva, E., Hopgood, A. A., Tipi, L., and Shahidan, M. (2014). Rail-Freight Crew Scheduling with a Genetic Algorithm. In Bramer, M. and Petridis, M., editors, *Research and Development in Intelligent Systems XXXI*, pages 211–223. Springer International Publishing, Cham.

Khmeleva, E., Hopgood, A. A., Tipi, L., and Shahidan, M. (2018). Fuzzy-Logic Controlled Genetic Algorithm for the Rail-Freight Crew-Scheduling Problem. *KI - Künstliche Intelligenz*, 32(1):61–75.

Koniorczyk, M., Talas, B., and Gedeon, F. (2015). Preconditioning in the Backtracking Duty Generation of Passenger Rail Crew Scheduling: A Case Study. *Communications - Scientific letters of the University of Zilina*, 17(2):23–29.

Kou, L. T., Stockmeyer, L. J., and Wong, C. K. (1978). Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Communications of the ACM*, 21(2):135–139.

Kumar, A., Vaidyanathan, B., and Ahuja, R. K. (2009). Railroad Locomotive Scheduling. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization*, pages 3236–3245. Springer US, Boston, MA.

Lam, E., Van Hentenryck, P., and Kilby, P. (2020). Joint vehicle and crew routing and scheduling. *Transportation Science*, 54(2):488–511. Publisher: INFORMS.

Laurent, B. and Hao, J.-K. (2008). Simultaneous Vehicle and Crew Scheduling for Extra Urban Transports. In Nguyen, N. T., Borzemski, L., Grzech, A., and Ali, M., editors, *New Frontiers in Applied Artificial Intelligence*, volume 5027, pages 466–475. Springer Berlin Heidelberg, Berlin, Heidelberg. Series Title: Lecture Notes in Computer Science.

Mercier, A., Cordeau, J.-F., and Soumis, F. (2005). A computational study of Benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32(6):1451–1476.

Mercier, A. and Soumis, F. (2007). An integrated aircraft routing, crew scheduling and flight retiming model. *Computers & Operations Research*, 34(8):2251–2265.

Mesquita, M., Moz, M., Paias, A., Paixão, J., Pato, M., and Respício, A. (2011). A new model for the integrated vehicle-crew-rostering problem and a computational study on rosters. *Journal of Scheduling*, 14(4):319–334.

Mesquita, M., Moz, M., Paias, A., and Pato, M. (2013). A decomposition approach for the integrated vehicle-crew-roster problem with days-off pattern. *European Journal of Operational Research*, 229(2):318–331.

Mesquita, M. and Paias, A. (2008). Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem. *Computers & Operations Research*, 35(5):1562–1575.

Moon, J. W. and Moser, L. (1965). On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23–28.

Perumal, S. S., Dollevoet, T., Huisman, D., Lusby, R. M., Larsen, J., and Riis, M. (2021). Solution approaches for integrated vehicle and crew scheduling with electric buses. *Computers & Operations Research*, 132:105268.

Petersen, J. D., Sölveling, G., Clarke, J.-P., Johnson, E. L., and Shebalov, S. (2012). An Optimization Approach to Airline Integrated Recovery. *Transportation Science*, 46(4):482–500.

Piu, F. and Speranza, M. G. (2014). The locomotive assignment problem: a survey on optimization models: The locomotive assignment problem: a survey on optimization models. *International Transactions in Operational Research*, 21(3):327–352.

Raff, S. (1983). Routing and scheduling of vehicles and crews. The state of the art. *Computers & Operations Research*, 10(2):63–211.

Shen, Y. and Xia, J. (2009). Integrated bus transit scheduling for the Beijing bus group based on a unified mode of operation. *International Transactions in Operational Research*, 16(2):227–242.

Steinzen, I., Becker, M., and Suhl, L. (2007). A hybrid evolutionary algorithm for the vehicle and crew scheduling problem in public transit. In *2007 IEEE Congress on Evolutionary Computation*, pages 3784–3789, Singapore. IEEE.

Tomita, E., Tanaka, A., and Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42. Publisher: Elsevier BV.

Vaidyanathan, B. and Ahuja, R. K. (2015). Crew Scheduling Problem. In Patty, B. W., editor, *Handbook of Operations Research Applications at Railroads*, volume 222, pages 163–175. Springer US, Boston, MA. Series Title: International Series in Operations Research & Management Science.

Vaidyanathan, B., Jha, K. C., and Ahuja, R. K. (2007). Multicommodity network flow approach to the railroad crew-scheduling problem. *IBM Journal of Research and Development*, 51(3.4):325–344.

Valouxis, C. and Housos, E. (2002). Combined bus and driver scheduling. *Computers & Operations Research*, 29(3):243–259.

Weide, O., Ryan, D., and Ehrgott, M. (2010). An iterative approach to robust and integrated aircraft routing and crew scheduling. *Computers & Operations Research*, 37(5):833–844.