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

Andreas Bärmann, Alexander Martin, and Jonasz Staszek

Department of Data Science
Friedrich-Alexander-Universität Erlangen-Nürnberg
Cauerstr. 11, 91058 Erlangen, Germany
{andreas.baermann,alexander.martin,jonasz.staszek}@fau.de

August 30, 2021

Abstract

In this work, we consider the integrated problem of locomotive scheduling and driver rostering 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 as well as a commodity aggregation for the multi-commodity-flow part. 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 rostering. 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 our method by solving a large-scale real-world problem instance provided by our industry partner DB Cargo Polska S.A.

Keywords: Integrated Locomotive and Driver Rostering, 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 rostering problem. We then derive two improvements of this formulation by performing a clique-based tightening of the original set-packing inequalities and a commodity aggregation for the multi-commodity-flow part. To deal with the computational complexity of the problem, we decompose the problem into a master locomotive assignment problem and a subproblem for driver rostering. 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.

Our study is – to the best of our knowledge – the first one to consider locomotive scheduling and driver rostering in an integrated fashion. Concerning staff planning, we directly focus on driver rostering – we skip the intermediate step of driver scheduling considered by the other authors, i.e. instead of coming up with a set of feasible driver shifts, our solution delivers an individual plan for each driver. We evaluate our method in a detailed real-world case study. As we will see, the efficiency of our 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. Moreover, our 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 practical use of our methods.

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 our mathematical model (formulated as a combination of multi-commodity flow and set covering with compatibility, conflict and multiple-choice constraints) as well as the two formulation-strengthening approaches. Further, in Section 4 we present our solution approach described above. In Section 5, we show the effectiveness of our method on problem instances provided by our industrial partner, DB Cargo Polska. Finally, in Section 6, we draw our conclusions and give 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 outline the literature on integrated vehicle and crew scheduling (which sometimes also includes crew rostering).

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.

Integrated approaches 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)).

Integrated vehicle and crew scheduling in railway planning and our contribution During our study of the 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 all the operational requirements of our industrial partner, DB Cargo Polska, 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 our model and the performance of our algorithm are demonstrated at the hand of real-world, country-scale problem instances by our industry partner.

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 our approach, 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.

Most notably, our paper introduces the *integrated vehicle scheduling and driver rostering 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). Further, our approach, enhanced by cutting planes, can solve instances which are about five times larger than those presented in Dauzère-Pérès et al. (2015).

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 two techniques for the improvement of the model formulation – clique tightening and commodity aggregation – will be discussed.

3.1 Overview of modelling requirements and assumptions

Our 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 that will allow as many of them as possible to 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. Additionally, our industrial partner supplied information about the *calculation weeks* – which are subsequent, non-overlapping planning periods, spanning seven days each, defined by DB Cargo Polska on a quarterly basis to comply with the stipulations of the Polish Labour Code. They are needed to ensure that at least one long break, explained in more detail later, is fully included in each of the calculation weeks. Given all the information at hand, in our approach we can use an indirect modelling of time and space. 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 our model. They pertain to all the three constraint groups mentioned above. 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 regulations of the Polish Labour Code and the Polish Railway Transport Code as well as the planning practice of the industrial partner.

Requirements The Polish Labour Code requires that each driver work for at most 12 hours in a shift and then rest for at least 11 hours. The industrial partner has to ensure that each driver has at least one long break, defined as a period of 35 hours of uninterrupted rest, per calculation week. The entire 35-hour long break must be incorporated in full within one calculation week. Additionally, at least one in four Sundays must be off. Following the Polish Railway Transport Code, 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.

Restrictions Our industrial partner assigns drivers to one of three planning regions, roughly corresponding to a geographic subdivision of Poland into three regions – roughly south, west and east. Based on the Polish Labour Code, 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 their 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. Although by labour code this transportation time can be counted as break time for the driver, we again incorporate company directive, which grants the drivers a break of 11 hours plus transportation time if a transport is required.

We 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 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 our model focusses on the operational activities.

We assume that at most one driver can be staffed to a train. This is standard operational practice at our industrial partner. Our model could be extended to allow for multiple drivers in one train by changing one constraint group from set packing to set cover, although 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. They respect the Polish Railway Transport Code as well as the 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 are free to select the starting location of each locomotive in our model. It can easily be extended to consider the actual starting locations of locomotives. 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, trips of locomotives without a train in order to place them where they are needed next. We instead assume that empty runs are already part of the order book. This sometimes results in a slight less-than-100% coverage of the trains scheduled therein.

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 exist, 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. 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

Last, we summarize the requirements stemming from the Polish Railway Transport Code and the restrictions imposed by our industry partner which are related to the mutual compatibilities between locomotives, drivers and trains.

Requirements In our model, 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.

3.2 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 an 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 $t \in T^d$, 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. Let T^l and T^d be the subsets of trains compatible with a locomotive $l \in \mathscr{L}$ or a driver $d \in D$ respectively. Finally, we use $W \subset \mathbb{N}$ to denote the set of calculation weeks.

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

As a consequence of the indirect modelling of time indicated in the previous subsection, we assume that each shift of a driver has a unique first and a unique last train (job). Moreover, 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 last ones before the long break. For these we allow that the arrival time of one job determines the beginning time of the long break for an arbitrary number of drivers.

Sets 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. Table 1 presents a summary of these sets. Their exact definitions can be found in Appendix A.

For ease of notation, we write $t_2 \leq t_1$ for two trains $t_1, t_2 \in T$ if t_2 departs at the same point in time or later than t_1.

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

Name	Description
$T_{t,d}^{B+}$	Trains unassignable to driver d if train t is the last job in their working day
$T_{t,d}^{\text{35h}}$	Trains unassignable to driver d if t is their last job before a 35-hour break
$T_{t,d}^{B-}$	Trains unassignable to driver d if t is their first job in a working day
$T_{w,d}^{\text{week\_assignment}}$	Trains which belong to calculation week w
$T_{w,d}^{\text{week}}$	Trains which belong to calculation week w and are suitable as the last ones before a 35-hour break
$T_{w,d}^{\text{sunday}}$	Trains which belong to the Sunday in calculation week w
$T_{t,d}^{\text{shift\_beginning}}$	Preceding trains assignable to driver d if he is assigned to train t
$T_{t,d}^{\text{shift\_end}}$	Future trains assignable to driver d if he is assigned to train t
$T_{t,d}^{\text{time}}$	Trains which are feasible for driver d and in time conflict with train t
$T_{t,d}^{\text{after\_break}}$	Trains which can be the first job of the next shift of driver t after his 11-hour break if t is the last train in his previous shift
$T_{t,d}^{\text{before\_break}}$	Trains which can be the last job of the previous shift of driver d before the 11-hour break if t is the first train in his next shift
$T_{t,l}^{\text{next}}$	Future trains assignable to locomotive l if it is assigned to train t

Table 1: Descriptions of the sets required for constraint construction

via $\mathscr{V} := T$ and

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

Let us also define $\Sigma, \Theta \in \mathscr{V}$ as the source and sink nodes of $\mathscr{G}$ respectively. In our basic model, we will represent each individual locomotive as a separate commodity. The choice of an arc $a = (t_1, t_2) \in \mathscr{A}$ by a locomotive means that this locomotive first carries 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

$$\delta^+(t) := \{(t, t_1) \in \mathscr{A} : t_1 \in \mathscr{V}\},$$
$$\delta^-(t) := \{(t_1, t) \in \mathscr{A} : t_1 \in \mathscr{V}\},$$

respectively. Per definition of the set $\mathscr{A}$, a locomotive can only choose 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.

Decision variables In our model, we need to make sure that each train $t \in T$ is staffed by exactly one suitable driver $d \in D^t$ and one suitable locomotive $l \in \mathscr{L}^t$ 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 $l \in \mathscr{L}^{t_1} \cap \mathscr{L}^{t_1}$ 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 shift in the workday 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 workday. Similarly, the binary variable v_d^t models the choice of the last job in a shift before a short (11-hour) break while the binary variable z_d^t does the same for the last job in a shift before a long (35-hour) break. 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. All the variables are summarized in Table 2.

Name	Description	Type
$f_l^{u,v}$	Trains u, v are served by locomotive 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 an 11-hour break	binary
z_d^t	Train t is the last job of driver d before a 35-hour break	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 2: Summary of decision variables used in the model

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

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

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

$$\sum_{v:(t,v) \in \mathscr{A}} f_l^{t,v} \leq \sum_{d \in D^l \cap D^t} x_d^t \qquad (\forall t_1 \in T \setminus \{\Sigma\}) \ (\forall l \in \mathscr{L}^t) \tag{1.3}$$

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

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

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

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

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

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

$$z_d^t + x_d^{t_1} \leq 1 \qquad (\forall d \in D) \ (\forall t \in T^d) \ (\forall t_1 \in T_{t,d}^{35\text{h}}) \tag{1.10}$$

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

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

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

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

$$v_d^t \leq \sum_{t_1 \in T_{t,d}^{\text{shift\_beginning}}} y_d^{t_1} \qquad (\forall d \in D) \ (\forall t \in T^d) \tag{1.15}$$

$$x_d^t \leq \sum_{t_1 \in T_{w,d}^{\text{week}}} z_d^{t_1} \qquad (\forall w \in W) \ (\forall d \in D) \ (\forall t \in T_{w,d}^{\text{week\_assignment}}) \tag{1.16}$$

$$\alpha_d^t \leq \sum_{\substack{t_1 \in T^d: \\ t_1 \geq t}} \omega_d^{t_1} \qquad (\forall d \in D) \ (\forall t \in T^d) \tag{1.17}$$

$$x_d^t \leq h_d^w \qquad (\forall d \in D) \ (\forall w \in W) \ (\forall t \in T_{w,d}^{\text{sunday}}) \tag{1.18}$$

$$\sum_{w \in W} h_d^w \leq 3 \qquad (\forall d \in D) \tag{1.19}$$

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

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

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

$$\omega_d^t \leq x_d^t \qquad (\forall d \in D) \ (\forall t \in T^d) \tag{1.23}$$

$$\sum_{t_0 \in \delta^-(t_1) \cap T^l} f_l^{t_0,t_1} = \sum_{t_2 \in \delta^+(t_1) \cap T^l} f_l^{t_1,t_2} \qquad (\forall t_1 \in T) \ (\forall l \in \mathscr{L}^{t_1}) \tag{1.24}$$

$$\sum_{l \in \mathscr{L}^{t_1} \cap \mathscr{L}^{t_2}} f_l^{t_1,t_2} \leq 1 \qquad (\forall (t_1,t_2) \in \mathscr{A}) \tag{1.25}$$

$$\sum_{l \in \mathscr{L}^{t_2}} \sum_{t_1 \in \delta^-(t_2) \cap T^l} f_l^{t_1,t_2} \leq 1 \qquad (\forall t_2 \in T) \tag{1.26}$$

$$\sum_{t:(\Sigma,t) \in \mathscr{A} \wedge t \in T^l} f_l^{\Sigma,t} \leq 1 \qquad (\forall l \in \mathscr{L}) \tag{1.27}$$

$$\sum_{t_1 \in T^l} f_l^{\Sigma,t_1} = \sum_{t_2 \in T^l} f_l^{t_2,\Theta} \qquad (\forall l \in \mathscr{L}) \tag{1.28}$$

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

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

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

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

$$\alpha_d^t \in \{0,1\} \qquad (\forall t \in T) \ (\forall d \in D^t) \tag{1.33}$$

$$\omega_d^t \in \{0,1\} \qquad (\forall t \in T) \ (\forall d \in D^t) \tag{1.34}$$

$$f_l^{t_1,t_2} \in \{0,1\} \qquad (\forall (t_1,t_2) \in \mathscr{A}) \ (\forall l \in \mathscr{L}^{t_1} \cap \mathscr{L}^{t_2}). \tag{1.35}$$

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 amounting to 11 hours shall not be violated. Additionally, with (1.10) we require the integrity of the long, 35-hour 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 thar 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), (1.14) and (1.15) incorporate the integrity of drivers' shifts and model the maximal length of a shift amounting to 12 hours. Via (1.16), 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 (1.17), 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 (1.18) and (1.19), we guarantee that a driver works on at most three Sundays in a given planning period. Constraints (1.20), (1.21), (1.22) and (1.23) tie each shift variable to the corresponding staffing variable.

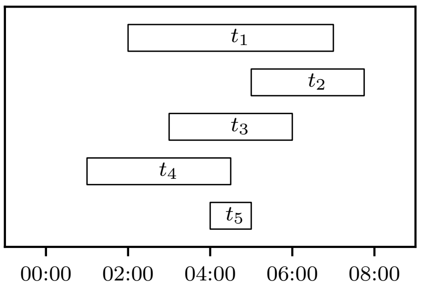
For the locomotive part of the model, (1.24) ensures that a locomotive that serves a train t arrives at its origin station and in the due time, and that similarly it later departs from the arrival station of train t. Using (1.25), we model that each train shall be served by at most one locomotive. With constraint (1.26), we make sure that an appropriate successor and predecessor are chosen for a locomotive $l \in \mathscr{L}$, given it serves a train $t \in T^l$. Constraint (1.27) ensures that each locomotive has a unique first and a unique last train in the plan. Via (1.28), we assure the integrity of the locomotive schedule. Finally, constraints (1.29)–(1.35) require all decision variables to be binary.

3.3 Model preprocessing

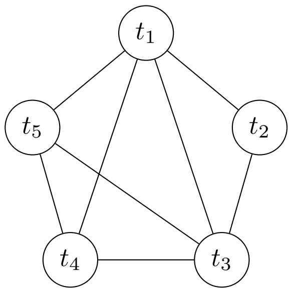
Model (1) is very complex, mostly due to its size. We will now consider ways to reduce the number of constraints and variables needed, which usually greatly benefits the computations times. To be more precise, we will describe two preprocessing techniques we use to simplify our model. The first of them – called *clique tightening* – will be used to improve the formulation of the conflict constraints. The other one will help us reformulate the multi-commodity-flow part in a more compact fashion.

3.3.1 Clique tightening for 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 constrains will grow exponentially. 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 $\mathcal{C}^d_{time}$ be a clique edge cover of the graph $\mathcal{G}^d_{time}$. Then constraint* (1.7) *is equivalent to the following constraint:*

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

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

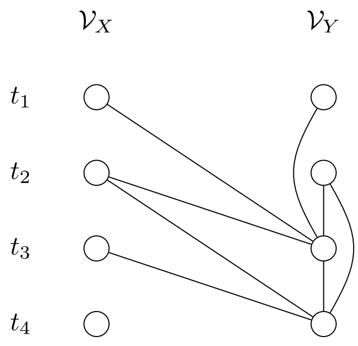


Figure 2: Example of a break conflict graph $\mathcal{G}^d_{\text{back\_break}}$

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

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

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.

For the illustrative case of a driver $d \in D$ with $T^d := \{t_1, t_2, t_3, t_4\}$, $T^{B-}_{t_4,d} := \{t_2, t_3\}$, $T^{B-}_{t_3,d} := \{t_1, t_2\}$, 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.

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 $\mathcal{V}^d_X$. Similarly, each node in $\mathcal{V}^d_X$ will appear in at most one maximal clique. 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 (1.20) – 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 and thus the number of cliques to be considered.

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 $t_X \in \mathcal{V}^d_X$:

$$\mathcal{C}^{t,d}_{\text{back\_break}} := \{C \in \mathscr{C}^d_{\text{back\_break}} : t \in C \wedge t_X \in \mathcal{V}^d_X\}.$$

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

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

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

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

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

3.3.2 Commodity aggregation for the locomotive part

Let us revisit the locomotive part of model (1). We notice that some of the locomotives $l \in \mathscr{L}$ have exactly the same set of compatible trains T^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 $\mathcal{L}^t \subseteq \mathscr{L}$ the set of locomotive classes compatible with a given train $t \in T$. Additionally, $\mathcal{L}^d \subseteq \mathscr{L}$ 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. Similarly we change the definition of the set $T^{\text{next}}_{t,l}$ to convey information about the locomotive class L. To reformulate model (1), we also need to slightly change the definition of the f-variables:

$$f_L^{t_1,t_2} = \begin{cases} 1, & \text{if nodes } (t_1,t_2) \in \mathscr{A} \text{ are served by a locomotive of class } L \in \mathcal{L}^{t_1} \cap \mathcal{L}^{t_2} \\ 0, & \text{otherwise.} \end{cases}$$

The reformulated constraints (1.2), (1.3), (1.24), (1.25), (1.26), (1.27), (1.28) and (1.35) look as follows:

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

$$\sum_{t_2:(t_1,t_2)\in\mathscr{A}} f_L^{t_1,t_2} \le \sum_{d\in D^L\cap D^t} x_d^t \qquad (\forall t_1 \in T \setminus \{\Sigma\})(\forall L \in \mathcal{L}^t) \tag{1.3b}$$

$$\sum_{t_0\in\delta^-(t_1)\cap T^L} f_L^{t_0,t_1} - \sum_{t_2\in\delta^+(t)\cap T^L} f_L^{t_1,t_2} = 0 \qquad (\forall L \in \mathcal{L})(\forall t_1 \in T^L) \tag{1.24b}$$

$$\sum_{L\in\mathcal{L}^{t_1}\cap\mathcal{L}^{t_2}} f_L^{t_1,t_2} \le 1 \qquad (\forall (t_1,t_2) \in \mathscr{A}) \tag{1.25b}$$

$$\sum_{L\in\mathcal{L}^{t_2}} \sum_{t_1\in\delta^-(t)\cap T^L} f_L^{t_1,t_2} \le 1 \qquad (\forall t_2 \in T) \tag{1.26b}$$

$$\sum_{t\in T^L} f_L^{\Sigma,t} \le |L| \qquad (\forall L \in \mathcal{L}) \tag{1.27b}$$

$$\sum_{t\in T^L} f_L^{\Sigma,t} - \sum_{t\in T^L} f_L^{t,\Theta} = 0 \qquad (\forall L \in \mathcal{L}) \tag{1.28b}$$

$$f_L^{t_1,t_2} \in \{0,1\} \qquad \begin{matrix}(\forall (t_1,t_2) \in \mathscr{A})\\ (\forall L \in \mathcal{L}^{t_1} \cap \mathcal{L}^{t_2}).\end{matrix} \tag{1.35b}$$

Note that this way of formulating the locomotive flow balance constraints entails a major reduction in the number of necessary constraints and variables as we do not distinguish individual locomotives and only consider locomotive classes. A similar modelling has been used in Lohatepanont and Barnhart (2004) for the optimization of aircraft rotation plans.

After solving the reformulated version of model (1), its trivial to concretize the assignment of a locomotive class to arcs and nodes in the flow network $\mathscr{G}$ to actual, individual locomotives. We obtain a solution in the form of directed paths through the network, connecting the source node Σ to the sink node Θ (as defined in Subsection 3.2) 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 $L \in \mathcal{L}$.

4 Solution methods

We will now derive a solution algorithm based on decomposition and cutting planes for the integrated locomotive scheduling and driver rostering 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, or, more precisely, the commodity-aggregated version of model (1). These cutting planes encode common reasons for the infeasibility of the driver part (as encountered for our real-world instances), 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 rises to the driver subproblem, whose task it is to find produces a feasible driver assignment which is compatible to the fixed locomotive assignment. If this is possible, we have solved the problem to global optimality. In case the driver part is infeasible but we cannot generate our problem-specific cutting planes were not sufficient 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)–(1.23) and (1.29)–(1.34), and (iii) the locomotive-related constraints (1.24)–(1.28) and (1.35). 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)–(1.23) as well as (1.29)–(1.34) 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:

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

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

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

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 (1.24b)–(1.28b), (2.2) and (2.3).

4.2 Valid inequalities for the locomotive subproblem

The driver subproblem contains ten constraints 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) and (1.17) as well as constraint (1.19), which pertains to ensuring the Sunday breaks.

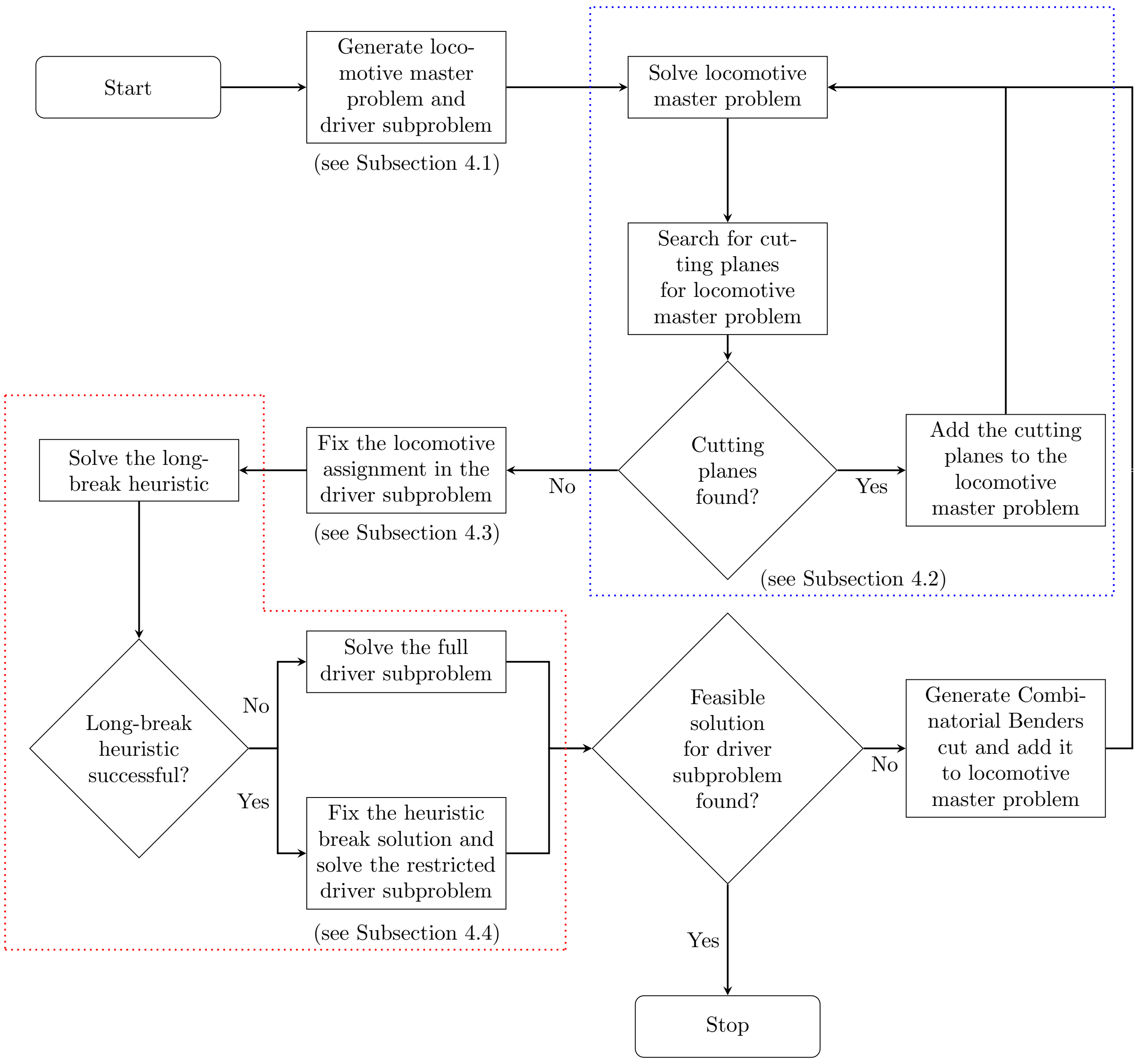

Start
Generate loco-motive master problem and driver subproblem
(see Subsection 4.1)
Solve locomotive master problem
Search for cut-ting planes for locomotive master problem
Cutting planes found?
Yes
Add the cutting planes to the locomotive master problem
No
Fix the locomotive assignment in the driver subproblem
(see Subsection 4.3)
(see Subsection 4.2)
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 Combi-natorial 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

Based on our computational experience, we determined that only a subset of these are violated for our real-world instances supplied by the 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.

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)*,* (1.24b)*–*(1.28b)*,* (2.2) *and* (2.3)*. Then we can find values $\bar{x}, \bar{y}, \bar{v}, \bar{\alpha}, \bar{\omega}$ such that $(\bar{f}, \bar{x}, \bar{y}, \bar{v}, \bar{\alpha}, \bar{\omega})$ is a solution to the constraint system given by the driver-related constraints* (1.2)*,* (1.3)*,* (1.13)*,* (1.14)*,* (1.15)*,* (1.17)*,* (1.18)*,* (1.20)*,* (1.21)*,* (1.22) *and* (1.23)*.*

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. This means, we choose any compatible driver for the locomotive-train assignment. For constraints (1.13), (1.14), (1.15), (1.17), (1.18) we can choose the y-, v-, α- and ω-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 (1.20), (1.21), (1.22) and (1.23), 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 subproblem. 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.

In our 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 rostering 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 3.3.1, we introduced the graph G^d_{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

$$E_{\text{time}} \coloneqq \{(t_1, t_2) : t_1 \in T \quad \wedge \quad \exists d \in D : t_2 \in T^{\text{time}}_{t_1,d}\}.$$

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

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

which allows us to derive valid cutting planes.

Theorem 4.2. *The following inequalities are valid for the commodity-aggregated version of model* (1)*:*

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

Proof. Let $(\bar{f}, \bar{\lambda}, \bar{x}, \bar{y}, \bar{v}, \bar{z}, \bar{\alpha}, \bar{\omega})$ be a feasible solution to model (1). Suppose this solution violates constraint (2.4). Then there exists a clique $C \in \mathscr{C}^{\text{time}}_{\text{assignments}}$ such that

$$\sum_{(t,L)\in C} \sum_{t_1 \in T^L \cap \delta^+(t)} f_L^{t,t_1} \quad > \quad \left| \bigcup_{(t,L)\in C} D^t \cap D^L \right| \tag{2.5}$$

holds. By definition of $\mathscr{C}^{\text{time}}_{\text{assignments}}$, all of the trains in the set $C_{\text{trains}} \coloneqq \{t_1 : (t_1, L) \in C\}$ are in time conflict. Hence, we at least need $|C_{\text{trains}}|$-many drivers compatible with the trains in C_{trains} and with the potential locomotive classes they are assigned in order to ensure that all these trains can be served. But in equation (2.5), we supposed that there are fewer compatible drivers. Therefore, the considered solution is infeasible, which is a contradiction. □

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

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

and

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

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

Theorem 4.3. *The following inequalities are valid for the commodity-aggregated version of 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. Let $(\bar{f}, \bar{\lambda}, \bar{x}, \bar{y}, \bar{v}, \bar{z}, \bar{\alpha}, \bar{\omega})$ be a feasible solution to model (1) and suppose this solution violates constraint (2.6). Then there exists a set $S^{t,d}$ for some $d \in D$ and $t \in T^d$ with

$$\sum_{(t,L)\in S^{t,d}} \sum_{t_1 \in T^L \cap \delta^+(t)} f_L^{t,t_1} \quad > \quad 1.$$

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. But we assumed that the feasible solution sets more than one of these variables to one. Therefore, the solution is infeasible Hence, we have a contradiction. □

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

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 be 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):

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

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

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

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:

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

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 $L_1 \in \mathcal{L}^{t_1}$ 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 4.4. *The following inequalities are valid for the commodity-aggregated version of model* (1)*:*

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

Proof. Let $(\bar{f}, \bar{\lambda}, \bar{x}, \bar{y}, \bar{v}, \bar{z}, \bar{\alpha}, \bar{\omega})$ be a feasible solution to model (1). Suppose this solution violates constraint (2.7). Then there exists a set $\mathscr{S}^d$ for some $d \in D$ and a train-locomotive assignment $(t, L) \in \mathscr{S}^d$ such that

$$\sum_{t_1 \in T^L \cap \delta^+(t)} f^{t,t_1}_L \quad > \quad \sum_{(t_2,L_2) \in \mathscr{F}^d_{t,L}} \; \sum_{t_3 \in T^{L_2} \cap \delta^+(t_2)} f^{t_2,t_3}_{L_2} \tag{2.8}$$

holds. 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. Since in inequality (2.8) we saw that the opposite case follows from our assumption, the solution is infeasible. Hence, we have a contradiction. □

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

The purpose of constraint (1.19) is to enforce that no driver works on more than three consecutive Sundays. We consider the set $T^{\text{sunday}}_{w,d}$ for each driver $d \in D$ and for each week $w \in W$. Now let us construct the graphs $G^{\text{sunday}}_d = (V^{\text{sunday}}_d, E^{\text{sunday}}_d)$ for all $d \in D$, where

$$\begin{aligned} V^{\text{sunday}}_d &:= \{(t, l) : D^t \cap D^l = \{d\} \quad \wedge \quad \exists w \in W : t \in T^{\text{sunday}}_{w,d}\}, \\ E^{\text{sunday}}_d &:= \{\{(t_1, l_1), (t_2, l_2)\} : t_1, t_2 \in V^{\text{sunday}}_d \quad \wedge \quad t_1 \in T^{\text{sunday}}_{w_1,d} \quad \wedge \quad t_2 \in T^{\text{sunday}}_{w_2,d} \\ &\qquad \wedge \quad w_1, w_2 \in W \quad \wedge \quad 0 < |w_2 - w_1| \leq 3\}. \end{aligned}$$

Further, for each $d \in D$ let $\mathcal{C}^{\text{sunday}}_d$ be the set of all 4−cliques in G^{sunday}_d. Each such clique corresponds to a combination of train-locomotive assignments that would lead to a driver d working on four consecutive Sundays. This idea leads to a set of inequalities whose validity is easy to see.

Theorem 4.5. *The following inequalities are valid for the commodity-aggregated version of model* (1)*:*

$$\sum_{(t,L) \in C} \; \sum_{t_1 \in \delta^+(t) \cap T^L} f^{t,t_1}_L \quad \leq \quad 3 \quad (\forall d \in D) \;\; (\forall C \in \mathcal{C}^{sunday}_d). \tag{2.9}$$

Note that for all four cutting planes (2.5), (2.6), (2.8), (2.9), which are valid for the commodity-aggregated version of model (1), it is straightforward to deduce equivalent cutting planes which are valid for the original version of model (1).

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), (1.16), (1.18) and (1.19), which make sure that the drivers' requirements with regard to longer breaks (i.e. Sundays off and the 35-hour 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 weekly 35-hour 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), (1.16), (1.18) and (1.19), 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 $d \in \bar{D}$ (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), what it seldom did in our experiments, we instead solve the full, unrestricted driver subproblem directly.

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 rostering problem in a real-world case study. We begin by describing how we will assess which of the enhancements to both the optimization model and the solution algorithm contribute how much to reducing the computation times and mention some of the details which lead to a well-performing implementation. Then we describe our case study at the hand of a country-wide problem instance stemming from Polish rail freight traffic. 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.

5.1 Implementation details

We have run all our experiments on a compute server with a 2x Intel Xeon E5-2643 v4 processor using all 12 cores and 256 GB of memory. Further, we have used Gurobi 9.1 (Gurobi (2021)) 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 3.3.1). 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 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 on 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 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 four different groups. In the following, we describe each of these computational scenarios in detail. We did not consider the joint model in its basic version to its complexity, in particular the large number of constraints and variables – in our computational experiments it failed to solve even the smallest of the instances.

Clique tightening In the second implementation, we will use the improved version of model (1) where constraints (1.7), (1.8), (1.9) and (1.10) are replaced by constraints (1.7b), (1.8b), (1.9b) and (1.10b) respectively (see Section 3.3.1). This implementation will be abbreviated by Clq.

Clique tightening + commodity aggregation In the next step, we solve the most efficient modification of model (1). This modification uses both clique tightening as in the previous paragraph and commodity aggregation as described in Subsection 3.3.2. This means we additionally replace constraints (1.2), (1.3), (1.24), (1.25), (1.26), (1.27), (1.28) and (1.35) with constraints (1.2b), (1.3b), (1.24b), (1.25b), (1.26b), (1.27b), (1.28b) and (1.35b) respectively. From now on, we will refer to this scenario as Clq+Agg.

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 (1.24b)–(1.28b), (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 (1.2b), (1.3b), (1.4b), (1.5)–(1.6), (1.7b)–(1.10b), (1.11)–(1.23) and (1.29)–(1.34), 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 Decomp.

Decomposition + Heuristic In the final, most advanced implementation, we use the complete algorithmic scheme shown in Figure 3 on page 14. 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 Decomp.

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 $(x, y, v, \alpha, \omega)$-variables as per the solution of the long-break presolve heuristic.

This implementation will be called Decomp+Heur in the following.

5.2 The case study

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 fuelling, 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.

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 3 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. With a growing instance size,

Instance	#Days	#Trains	Avg. model generation time (in s)
1W_1	7	629	328
1W_2	7	610	302
1W_3	7	615	325
1W_4	7	613	317
2W_1	14	1239	834
2W_2	14	1242	828
2W_3	14	1228	836
3W_1	21	1854	1558
3W_2	21	1838	1529
1M	29	2551	2618

Table 3: Overview of instance characteristics and average model generation times

the computational complexity of model generation increases exponentially due to the effort required to perform the clique tightening. Especially for the larger instances, 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 solution scenario.

5.2.1 Analysis of solution times

For the computational experiments, we have used the instances from Table 3 and compare their solutions times for the five different implementation scenarios described in Subsection 5.1. We have performed three runs of implementation on each instance, and report the average solution times in Table 4. The

Instance	Clq	Clq+Agg	Decomp	Decomp+Heur
1W_1	238%	0.64%	453	107
1W_2	243%	0.16%	104	73
1W_3	242%	79.80%	362	66
1W_4	241%	3135 ‡	383	77
2W_1	-	-	1250	293
2W_2	-	-	2261	263
2W_3	-	-	439	278
3W_1	-	-	7117	773
3W_2	-	-	3684 †	644
1M	-	-	-	2370

Table 4: Solution times of the different implementations for each instance (in seconds).

first column presents the name of the instance. Columns 2 to 6 display the performance of the respective solution scenario against the listed instances. A dash denotes that no 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 the remaining two timed out. An integer number denotes the average time in seconds required to find an optimal solution. If a percentage is shown instead, we report the lowest of the MIP gaps returned within the time limit over all three runs of the considered instance-algorithm combination.

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, even if improved, only seldom returns integer-feasible solutions, and an optimum could only be obtained for two instances in one of three attempts, with the remaining two attempts not finishing within within the time limit. We can also observe that employing the long-break heuristic has produced significant speed-ups in the solution times. Finally, it should be mentioned that the solution times 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.

5.2.2 Assessment of solution quality

In Table 5, 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

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	2528	99.1%

Table 5: Comparison of the number of trains in each instance and the coverage in its optimal solution

the corresponding number of trains in that instance. The third column presents the globally optimal solution obtained via the decomposition method approaches developed in this paper. 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. These results were satisfying to the industrial partner, because 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 our method obtains, our model can prove very favourable if integrated in a decision support tool, where planners can manually add such repositionings ("empty runs") to make sure that all the trains 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 B.

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 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 – using the example of the Polish labour code. 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 developing an exact decomposition approach which allows for a sequential solution of the problem. To further decrease computation times, we derived a number of problem-specific valid inequality classes (which suffice to ensure the feasibility of solutions to the decomposed subproblems) and 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 this 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 Jacek Babiuch, Michał Batko, 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. Our 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 6. In the definitions below, we use the constant $h \in \mathbb{R}$ to denote a real number equivalent to one hour. In the paragraphs below, a definition of each index set used in model (1) is given.

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 $t \in T$
$d(t)$	Destination station of a train $t \in T$
τ^{s_1,s_2}	Transit time between stations $s_1, s_2 \in S$

Table 6: Summary of parameters required for set construction

Forward-looking daily break set $T^{\mathbf{B+}}_{t,d}$ 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 daily, 11-hour break. We can formally state the set $d \in D$ and $t \in T^d$ as

$$\begin{aligned} T^{\text{B+}}_{t,d} \coloneqq & \{t_1 \in T^d : s(t_1) \geq e(t) \ \wedge \ e(t_1) \leq e(t) + 11h\} \\ & \cup \{t_1 \in T^d : s(t_1) < e(t) + 11h \ \wedge \ e(t_1) \geq e(t) + 11h\} \\ & \cup \{t_1 \in T^d : e(t) + 11h + \tau^{a(t),o(t_1)} > s(t_1)\}. \end{aligned}$$

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 $T^{35h}_{t,d}$ For each driver $d \in D$ and train $t \in T^d$ we require the set $T^{\text{35h}}_{t,d}$ 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 35-hour break. Formally, we define this set as:

$$\begin{aligned} T^{\text{35h}}_{t,d} \coloneqq & \{t_1 \in T^d : s(t_1) \geq e(t) \ \wedge \ e(t_1) \leq e(t) + 35h\} \\ & \cup \{t_1 \in T^d : s(t_1) < e(t) + 35h \ \wedge \ e(t_1) \geq e(t) + 35h\} \\ & \cup \{t_1 \in T^d : e(t) + 35h + \tau^{a(t),o(t_1)} > s(t_1)\} \end{aligned}$$

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 35 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 $T^{\mathbf{B-}}_{t,d}$ 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

$$\begin{aligned} T^{\text{B}-}_{t,d} \coloneqq & \{t_1 \in T^d : s(t_1) \leq s(t) - 11h \ \wedge \ e(t_1) \geq s(t) - 11h \ \wedge \ e(t_1) \leq s(t)\} \\ & \cup \{s(t_1) \geq s(t) - 11h \ \wedge \ e(t_1) \leq s(t)\} \\ & \cup \{s(t) - 11h - \tau^{a(t_1),o(t)} < e(t_1)\} \end{aligned}$$

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

Long break beginning set $T_{w,d}^{\textbf{week}}$ For every driver $d \in D$ and each week $w \in W$, we construct a set $T_{w,d}^{\text{week}}$ 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, 35-hour break of driver d. We need to make sure that at least one 35-hour break is scheduled in each calculation week. Hence, $T_{w,d}^{\text{week}}$ shall contain all the jobs $t \in T^d$ which end more than 35 hours before the end of the calculation week. This set can be formally defined as

$$T_{w,d}^{\text{week}} := \{t \in T^d : e(t) \geq s^{week}(w) \ \wedge \ e(t) + 35h \ \leq \ e^{week}(w)\}$$

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

Calculation week set $T_{w,d}^{\textbf{week\_assignment}}$ 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 $T_{t,d}^{\textbf{shift\_end}}$ and $T_{t,d}^{\textbf{shift\_beginning}}$ 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:

$$\begin{aligned} T_{t,d}^{\text{next\_driver}} := \{t_1 \in T^d : (s(t_1) > e(t) \ \wedge \ e(t_1) < s(t) + 12h \wedge o(t_1) = a(t))\} \\ \cup \{t_1 \in T^d : s(t_1) > e(t) + \tau^{o_{t_1}, a(t)} \ \wedge \ e(t_1) < s(t) + 12h\}. \end{aligned}$$

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:

$$T_{t,d}^{\text{shift\_beginning}} := \{t_1 : t_1 \in \delta^-(t) \wedge s(t_1) \geq e(t) - 12h\} \cup \{t\}$$

and

$$T_{t,d}^{\text{shift\_end}} := \{t_1 : t_1 \in \delta^+(t) \wedge e(t_1) \leq s(t) + 12h\} \cup \{t\}$$

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 $T_{t,d}^{\textbf{time}}$ 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 $t \in T^d$. 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

$$\begin{aligned} T_{t,d}^{\text{time}} := \{t_1 \in T^d : s(t_1) \leq s(t) \quad \wedge \quad e(t_1) \geq e(t)\} \\ \cup \{t_1 \in T^d : s(t_1) \geq s(t) \quad \wedge \quad s(t_1) \leq e(t) \quad \wedge \quad e(t_1) \geq e(t)\} \\ \cup \{t_1 \in T^d : s(t_1) \geq s(t) \quad \wedge \quad e(t_1) \leq e(t)\} \\ \cup \{t_1 \in T^d : s(t_1) \leq s(t) \quad \wedge \quad e(t_1) \leq e(t) \quad \wedge \quad e(t_1) \geq s(t)\} \\ \cup \{t_1 \in T^d : e(t) + \tau^{a(t), o(t_1)} \leq s(t_1) \quad \wedge \quad e(t_1) \leq s(t) + 12h \quad \wedge \quad s(t_1) > s(t)\} \end{aligned}$$

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

Feasible next shift beginnings and ends $T_{t,d}^{\textbf{after\_break}}$ and $T_{t,d}^{\textbf{before\_break}}$ 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 11-hour 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 11-hour 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

$$\begin{aligned} A_d^b &:= \{(t_1, t_2) : t_1, t_2 \in T^d \wedge o(t_1) = a(t) \wedge s(t_1) > e(t) + 11h\} \\ &\cup \{(t_1, t_2) : t_1, t_2 \in T^d \wedge s(t_1) > e(t)\tau^{o(t_1),a(t)} + 11h\}. \end{aligned}$$

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 $T_{t,l}^{\textbf{next}}$ For all locomotives $l \in \mathscr{L}$ 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 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 $l \in \mathscr{L}^t$.

Sunday set $T_{w,d}^{\textbf{sunday}}$ 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

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

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

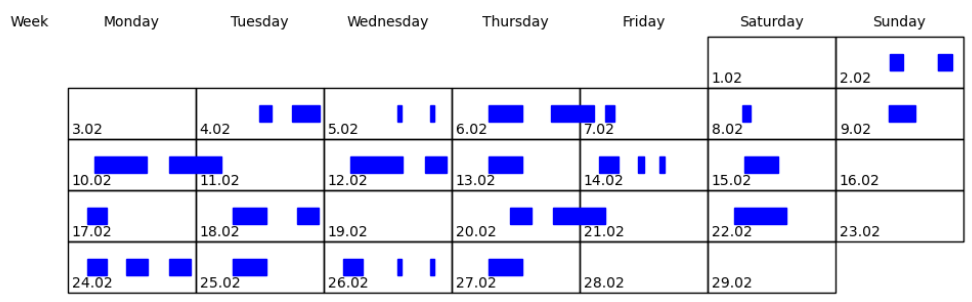
B Example shift reports

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 weekly plan: (Heavy Diesel Locomotive)

Loco type and item: (Heavy Diesel Locomotive)
Number of shifts: 34

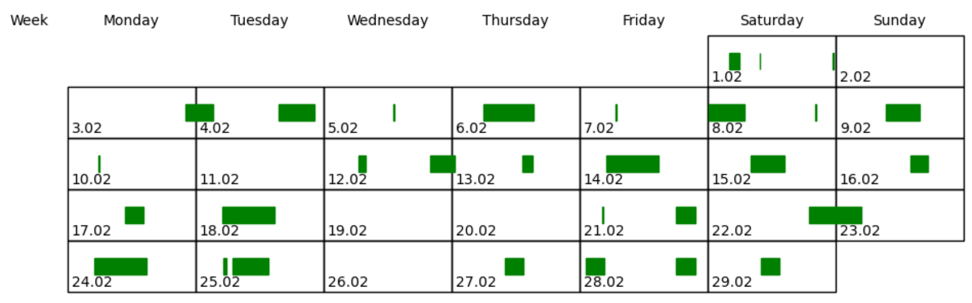
Train no.	From	To	Departure time	Arrival Time
124	PL035	PL020	2020-02-02 10:10	2020-02-02 12:46
149	PL020	PL035	2020-02-02 19:12	2020-02-02 21:55
290	PL035	PL020	2020-02-04 11:55	2020-02-04 14:15
315	PL020	PL035	2020-02-04 18:08	2020-02-04 23:15
395	PL035	PL085	2020-02-05 13:45	2020-02-05 14:40
418	PL085	PL035	2020-02-05 20:00	2020-02-05 20:46
459	PL035	PL020	2020-02-06 06:55	2020-02-06 13:19
515	PL020	PL026	2020-02-06 18:43	2020-02-07 02:46
555	PL026	PL003	2020-02-07 04:44	2020-02-07 06:33
656	PL003	PL026	2020-02-08 06:28	2020-02-08 08:01
751	PL026	PL026	2020-02-09 10:00	2020-02-09 15:00
802	PL026	PL020	2020-02-10 05:00	2020-02-10 14:50
853	PL020	PL026	2020-02-10 19:00	2020-02-11 04:50
977	PL026	PL020	2020-02-12 05:00	2020-02-12 14:50
1033	PL020	PL035	2020-02-12 19:00	2020-02-12 23:00
1080	PL035	PL020	2020-02-13 06:55	2020-02-13 13:19
1157	PL020	PL035	2020-02-14 03:41	2020-02-14 07:15
1191	PL035	PL046	2020-02-14 11:00	2020-02-14 12:00
1205	PL046	PL035	2020-02-14 15:00	2020-02-14 16:00
1266	PL035	PL020	2020-02-15 06:55	2020-02-15 13:19
1398	PL020	PL035	2020-02-17 03:41	2020-02-17 07:15
1496	PL035	PL020	2020-02-18 06:55	2020-02-18 13:19
1548	PL020	PL035	2020-02-18 19:00	2020-02-18 23:00
1709	PL035	PL020	2020-02-20 11:00	2020-02-20 15:00
1739	PL020	PL026	2020-02-20 19:00	2020-02-21 04:50
1872	PL026	PL020	2020-02-22 05:00	2020-02-22 14:50
2013	PL020	PL035	2020-02-24 03:41	2020-02-24 07:15
2042	PL035	PL020	2020-02-24 11:00	2020-02-24 15:00
2069	PL020	PL035	2020-02-24 19:00	2020-02-25 00:00
2114	PL035	PL020	2020-02-25 06:55	2020-02-25 13:19
2196	PL020	PL035	2020-02-26 03:41	2020-02-26 07:15
2240	PL035	PL085	2020-02-26 13:45	2020-02-26 14:40
2265	PL085	PL035	2020-02-26 20:00	2020-02-26 20:46
2304	PL035	PL020	2020-02-27 06:55	2020-02-27 13:19



Driver weekly plan: Driver069

Name and surname: Driver069
Number of shifts: 28
Number of Sundays off: 1
Number of 35-hour breaks: 4

Train no.	From	To	Departure time	Arrival Time
17	PL008	PL022	2020-02-01 04:00	2020-02-01 05:59
39	PL011	PL094	2020-02-01 09:45	2020-02-01 09:53
89	PL011	PL094	2020-02-01 23:30	2020-02-01 23:38
234	PL032	PL050	2020-02-03 22:04	2020-02-04 03:13
305	PL050	PL032	2020-02-04 15:30	2020-02-04 22:16
390	PL094	PL011	2020-02-05 13:00	2020-02-05 13:12
457	PL026	PL020	2020-02-06 06:00	2020-02-06 15:27
562	PL094	PL011	2020-02-07 06:42	2020-02-07 06:50
630	PL050	PL032	2020-02-08 00:12	2020-02-08 06:51
704	PL094	PL011	2020-02-08 20:10	2020-02-08 20:22
748	PL094	PL030	2020-02-09 09:19	2020-02-09 15:43
807	PL094	PL011	2020-02-10 05:50	2020-02-10 05:59
984	PL020	PL017	2020-02-12 06:28	2020-02-12 07:47
1038	PL016	PL008	2020-02-12 20:00	2020-02-13 00:33
1109	PL034	PL050	2020-02-13 13:10	2020-02-13 15:10
1163	PL026	PL020	2020-02-14 05:00	2020-02-14 14:50
1268	PL032	PL050	2020-02-15 08:06	2020-02-15 14:19
1364	PL011	PL032	2020-02-16 14:02	2020-02-16 17:20
1425	PL003	PL022	2020-02-17 10:50	2020-02-17 14:17
1488	PL026	PL020	2020-02-18 05:00	2020-02-18 14:50
1775	PL094	PL011	2020-02-21 04:15	2020-02-21 04:27
1832	PL032	PL094	2020-02-21 18:05	2020-02-21 21:44
1923	PL020	PL026	2020-02-22 19:00	2020-02-23 04:50
2018	PL026	PL020	2020-02-24 05:00	2020-02-24 14:50
2108	PL033	PL003	2020-02-25 05:10	2020-02-25 05:46
2115	PL050	PL032	2020-02-25 06:56	2020-02-25 13:34
2319	PL011	PL032	2020-02-27 10:01	2020-02-27 13:21
2375	PL032	PL094	2020-02-28 01:11	2020-02-28 04:42
2443	PL032	PL094	2020-02-28 18:05	2020-02-28 21:44
2507	PL011	PL032	2020-02-29 10:01	2020-02-29 13:21



References

Aksoy, A. and Altan, A. (2013). The Integrated Locomotive Assignment and Crew Scheduling Problem. *International Journal of Computational Engineering Research*, 3(8):7.

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.

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.

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).

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.

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.

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.

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.

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.

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.

Gurobi (2021). Gurobi optimizer reference manual.

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.

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.

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.

Lohatepanont, M. and Barnhart, C. (2004). Airline schedule planning: Integrated models and algorithms for schedule design and fleet assignment. *Transportation Science*, 38(1):19–32.

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.

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.

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.