OR Spectrum (2023) 45:1113–1151
https://doi.org/10.1007/s00291-023-00729-9

ORIGINAL ARTICLE

A matheuristic for tactical locomotive and driver scheduling for the Swiss national railway company *SBB Cargo AG*

Marie-Sklaerder Vié[1,2] · Nicolas Zufferey[1,2] · Stefan Minner[3,4]

Received: 14 June 2022 / Accepted: 8 May 2023 / Published online: 14 July 2023

Abstract
At the scale of Switzerland, the national railway company *SBB Cargo AG* has to schedule its locomotives and drivers in order to be able to pull all trains. Two objective functions are considered in a two-stage lexicographic fashion: (1) the locomotive and driver costs and (2) the driver time that is spent without driving. As the problem instances tend to reach really big sizes (up to 1900 trains), we propose to schedule locomotives and drivers in a sequential way, thus having a sequence of smaller problems to solve. Moreover, for smaller instances, we also propose to schedule jointly locomotives and drivers in an integrated way, therefore increasing the search space but possibly leading to better solutions. In this paper, we present a mathematical formulation and model for the problem. We also consider the contract-related constraints of the drivers, and we propose a way to integrate some time flexibility in the schedules. Next, we propose an innovative matheuristic to solve the problem, relying on a descent local search and a rolling horizon decomposition. An important goal of this method is to explore thoroughly at which extent a general-purpose solver can be used on this problem. Finally, the benefits of each aspect of the model and of the method are analyzed in detail on the results obtained for 20 real *SBB Cargo AG* instances.

Keywords Locomotive and driver scheduling · Problem decomposition for sequential optimization · Matheuristics · Lexicographic objectives

1 Introduction

SBB Cargo AG (www.sbbcargo.com) has to provide locomotives and drivers for about 500 long-distance (national) trains per day in Switzerland. The objective is to assign the locomotives and the drivers to the trains such that the implied costs (locomotive and driver costs) are minimized. For the primary objective of this study, we consider two types of costs: the fixed cost per locomotive or per

Extended author information available on the last page of the article

driver duty and the additional cost for light-traveling connection (i.e., when a locomotive and a driver travel without pulling a train). Moreover, a secondary objective is considered in order to make the driver schedules as smooth as possible (i.e., reduce their waiting times).

The family of solution methods relying on both metaheuristics and exact methods is called matheuristics. An extensive review of the matheuristics that have been developed for routing problems can be found in Archetti and Speranza (2014). In this review, the matheuristics are classified in three categories: *decomposition approaches*, *improvement heuristics*, and *column generation*. Another review (Ball 2011) considers an additional category, namely the *relaxation-based approaches*. Various papers have shown that matheuristics have the potential of delivering high-quality solutions for vehicle routing problems (e.g., (Leggieri and Haouari 2018)). This motivates us to design a dedicated matheuristic based on a rolling horizon (i.e., a decomposition approach) coupled with a descent search (i.e., an improvement heuristic) to solve the integrated version of the considered problem (i.e., locomotive and driver scheduling).

The initial research question raised by *SBB Cargo AG* is the following. Can we solve the entire problem (i.e., locomotive and driver scheduling) with a direct approach relying on an efficient, state-of-the-art general-purpose solver? If this is not possible, can we decompose the problem into a sequence of subproblems that are suitable for the solver? The integrated approach is proposed to tackle the initial research question. Next, as the experiments will show that the direct approach is not possible for most of the instances (except the smaller ones), we propose a sequential approach. Given the increasing efficiency of solvers, such types of research questions have been more often tackled in the last decade for various industrial applications, such as distribution network design problem in the automotive industry (Kchaou Boujelben et al. 2014), routing and scheduling problem in roll-on roll-off shipping (Hansen et al. 2022), lot-sizing and scheduling in the production of fruit-based beverages (Toscano et al. 2020), scheduling problems of the pharmaceutical industry in multiproduct multistage batch plants (Kopanos et al. 2010), periodic inventory routing problem in reverse logistics (Cardenas-Barron and Melo 2021), aircraft scheduling (Sama et al. 2019), and operational management of intermodal logistics platforms in the automotive industry (Coindreau et al. 2019). Unsurprisingly, the railway industry is also concerned by this research stream (e.g., (Bouzaiene-Ayari et al. 2014; Frisch et al. 2019; Haahr et al. 2016; Scheffler et al. 2020)). In such a context, and in line with our above research questions, a recent study has pointed out the relevance and importance of integrating vehicle and crew scheduling in transportation (Ge et al. 2022). The authors have raised a key research question that can be summarized as follows. *"With the continuous improvement of information and communication technology, as well as general solvers, is it possible to solve integrated problems, that had to be tackled by means of specialized heuristics years ago due to their inherent problem complexity, by means of currently available standard solvers and, if so, which instance sizes are to be solved in time limits deemed practical?"*.

The contributions of this paper rely on the following combination of features.

- We formulate the locomotive and driver scheduling problem as an integrated model that also satisfies important contractual aspects of the drivers and allows time flexibility on the light travels (i.e., when a locomotive and a driver travel without pulling a train, in order to relocate the locomotive at the next planned location). As pointed out by (Rählmann and Thonemann 2020), adding time flexibility is an important feature in a context where the drivers frequently wait at stations before driving the next train.
- Two objective functions are considered in a lexicographic fashion: the locomotive and driver costs, and the driver inactive times (i.e., the time between the end of a duty and the beginning of the next one).
- We propose a matheuristic for solving the model in an integrated way (i.e., the entire problem is tackled) and in a sequential way. The matheuristic employs a rolling horizon decomposition and a descent local search for improving solutions.
- Several studies have shown that the way of decomposing the problem significantly affects the solution quality (Jütte and Thonemann 2015). The proposed sequential approach is based on an efficient decomposition of the problem into a sequence of smaller subproblems that are suitable for a general-purpose solver. Moreover, the proposed sequence of subproblems is natural in the sense it is understandable and intuitive for decision makers. Indeed, as mentioned in (Silver 2004), decision makers are likely to better accept decision rules if they have an intuitive understanding of how the rules operate.
- As highlighted in (Burdett and Kozan 2010), an efficient graph representation helps in designing better algorithms. A pre-processing procedure is proposed for significantly reducing the size of the locomotive flow graph and the driver routing graph.
- The experiments are conducted on 20 real instances with up to 1900 trains. We thoroughly analyze which level of integration brings the best solutions and how each part of our method improves the obtained results

The paper is organized as follows. Section 2 introduces the problem without any formalism, as well as the assumptions of our study. Section 3 presents the related literature. Section 4 presents the mathematical formulation and model, where an innovative proposition to allow some time flexibility is also given. Section 5 describes the characteristics of the real instances and a pre-processing procedure for reducing their sizes. Section 6 proposes solution methods to solve the problem, while considering both the sequential and the integrated frameworks. Experiments are conducted in Sect. 7, followed by conclusions and extensions in Sect. 8.

2 Informal presentation of the problem and assumptions of this study

With respect to the railway industry, the reader is referred to (Rählmann and Thonemann 2020) for more information on various optimization problems that can occur from the strategic level to the real-time level. The overall railway planning process

is a complex task. As mentioned by Zhang et al. (2022) and shown in Table 1, to reduce its computational complexity, the process is usually hierarchically divided into several stages.

The locomotive scheduling problem consists in assigning a locomotive to each train to be pulled, whereas the driver scheduling problem consists in assigning a driver to each locomotive. At *SBB Cargo AG*, this task is currently accomplished sequentially by a set of planners (with some iterations), using separate optimization algorithms which require manual guidance. Indeed, the locomotive scheduling problem occurs at a tactical level, whereas the driver scheduling problem occurs at an operational level. Note that the optimization approach employed by *SBB Cargo AG* strongly differs from our sequential approach, as the latter relies on a single decision maker who has an overview on the entire problem (without using manual guidance and iterations). In that sense, the approach employed by the company cannot be formally labeled as a sequential approach.

The goal is to minimize two types of objective functions, namely the locomotive and driver costs (denoted as f^{L} and f^{D}, respectively), as well as the driver inactive times (denoted as f^{C}). The following costs are considered.

- The fix activation costs of the locomotives (depending for instance on the purchase and maintenance costs).
- The light traveling costs of the locomotives (i.e., when a locomotive and a driver travel without pulling a train, in order to relocate the locomotive at the next planned location).
- The driver costs (depending on the annual salaries of the drivers).

On the other hand, the driver inactive times are the relocation and waiting times of the drivers before their next duties. More precisely, when a driver has finished to drive a train, two types of features have to be taken into account before s/he drives the next train: the traveling relocation time (during which the driver is a passenger in a regular train) and the waiting time (i.e., the driver waits at the planned location before pulling the next train).

Consequently, the total cost $f = f^{\mathrm{L}} + f^{\mathrm{D}}$ and the total inactive time f^{C} have to be minimized. Based on the priorities of the company, these two objective functions are optimized in a lexicographic fashion: the cost (f) has the priority over the inactive time (f^{C}). In other words, no reduction of f^{C} can be performed if it augments f. This priority is very natural and in line with the literature (Portugal and co HRL, Paixão

Table 1 Railway planning process: from strategic to operational planning

Strategic level	Origin–destination demand matrix
	Network design
	Line planning
Tactical level	Train timetabling
	Rolling stock scheduling
	Crew scheduling
Operational level	Real-time train dispatching

JP, 2009): the indirect costs (which somewhat correspond to idle times in the job scheduling literature) are less important than the real, direct costs (which are easy to measure from an accounting perspective). This is also in line with other studies, in which no trade-off among objectives is possible. Among them, one can cite applications in the automotive industry (Solnon et al. 2008), in vehicle routing (Lehuédé et al. 2020), and in aviation (Prats et al. 2010).

The following constraints are considered: (1) the flow conservation constraints for the locomotives (with additional features to allow more flexibility); (2) the routing constraints for the drivers; (3) the break constraints for the drivers (in order to respect the driver contracts).

The following assumptions characterize this study.

- The considered problem is at a tactical level, i.e., for an annual planning horizon including the most important constraints. Some examples of the constraints that are relaxed at the tactical level are driver qualifications or locomotive-specific times for maneuvers.
- A simplified model with only one type of locomotive is used. This is however a conservative assumption as it results in a much larger solution space when compared to the case involving several locomotive types. This simplification has the advantage to allow better identifying the weaknesses and strengths of the proposed methods. The consideration of various locomotive types (i.e., adding the associated constraints and thus reducing the solution space) is left as a possible future work.

Consequently, the proposed tactical solutions cannot be used immediately by *SBB Cargo AG*, but they can be employed as input for the operational planning (i.e., short-term, and including all constraints and exceptions), for which the company uses the software IVU.rail (www.lbw-optimization.com/en/optimizer). However, the generated tactical solutions provide obvious insights on the structures of operational solutions and on their associated values. Moreover, the proposed method (with its different variations) is a proof of concept on which the company can rely to build their next generation of algorithms. Accurately exploring the practical conditions to ensure that the results can be used in practice is left as a future research step that should cover, as highlighted by Zhong et al. (2019), the type and different compositions of rolling stock, the limitations of rolling stock and crew, the maintenance requirements of a rolling stock, etc.

The 20 instances provided by *SBB Cargo AG* will be accurately presented in Sect. 7, and they all have 10 depots. For each instance, the initially planned number of locomotives will be given (see Table 3). Next, in Sect. 7.4, a sensitivity analysis will be performed if different numbers of locomotives are employed. It is important to note here that for each instance, the involved decision maker can—theoretically—pickup its preferred solution among the 160 provided solutions (we can thus reasonably assume that the picked solution will be efficient for the company). Indeed, we have two solution approaches (i.e., integrated, and sequential). Next, for each approach, we have 8 algorithms (depending on the procedure combination to use, as presented in Sects. 7.2 and 7.3). Finally, for each instance, we can run all the

16 available algorithms for 10 different but relevant numbers of locomotives (i.e., we will not only use the initially planned number of locomotives). At the end, the decision maker can decide to either employ the initially planned number of locomotives, or a different number of locomotives, based on locomotive availability and total costs.

Considering large NP-hard problems, it is well known in the optimization community that to be efficient, a solution method should have the ability to intensify and diversify its search in the solution space. Intensification refers to the ability to deeply investigate and exploit a promising zone of the solution space, whereas diversification refers to the ability to explore various zones of the solution space (which is particularly important for very large instances). When optimal solutions and tight bounds are not known for the considered problem (which regularly occurs for industrial problems), solution quality is often assessed by comparing the results returned by various algorithms. Such a comparative approach is in line with many papers in various industrial fields (e.g., production design (Vié et al. 2019), inventory deployment (Respen et al. 2017), network design (Amrani et al. 2011), production scheduling (Thevenin et al. 2017), and aircraft landing planning (Vié et al. 2022)), but also for academic problems (e.g., graph coloring (Malaguti and Toth 2010)). To be reliable, a comparative approach should involve methods that are made of different features (e.g., intensification and diversification procedures), and compare them with a common time limit according to the obtained solution values. This is exactly what we have done in our numerical comparisons (e.g., the below DLS feature will play an intensification role, whereas the below TFA feature will play a diversification role).

3 Literature review

The considered problem involves the driver scheduling problem and the locomotive scheduling problem. References for the former problem are given in the next paragraph, whereas papers for the latter are outlined in the third paragraph. The integration of both problems is discussed in the fourth paragraph. Given that hundreds of articles have been published in the field, we only give pointers in this section. The reader interested in more information is referred to most recent below-mentioned literature reviews.

The vehicle scheduling problem with a single depot can be formulated as a minimum-cost flow problem, and it is therefore solvable in polynomial time (Lenstra and Kan 1981). In contrast, the consideration of multiple depots makes the problem NP-hard (Bertossi et al. 1987). The crew scheduling problem can be formulated as a set covering problem and it is therefore NP-hard too (Jütte and Thonemann 2015). The railway crew scheduling problem consists of finding the best duty combination for railway crews in order to cover all trains over the planning horizon. The reader is referred to Heil et al. (2020) for an extensive literature review on crew scheduling, covering 123 publications on railway crew scheduling (with a focus on publications from 2000). The driver scheduling problem consists in selecting a set of duties for vehicle drivers. It is a well-known problem, and various models, possible objective

functions and constraints can be found in (Portugal and co HRL, Paixão JP, 2009; Wren et al. 2003), whereas some case studies are presented in Kwan (2011). The most common contract-related constraints for drivers, which are also present in our problem, are the following: maximum duty length, minimum break duration, maximum time without break, and multiple driver depots (Abbink et al. 2005; Boschetti et al. 2004; Fores et al. 2002; Yunes et al. 2005).

The locomotive scheduling problem consists of assigning a set of locomotives to a preplanned train schedule in order to be able to pull all the trains from the origins to the destinations. Various models and pointers to recent references can be found in Ahuja et al. (2005); Frisch et al. (2021); Piu and Speranza (2014); Vaidyanathan et al. (2008a, b). The literature provides various linear programming formulations and highlights the importance to minimize the real costs as a first objective. Moreover, the combination of MIP and heuristic features is also successfully employed (Scheffler et al. 2020).

The integrated vehicle and crew scheduling problem has been widely studied in the literature, and the reader is referred to (Caprara et al. 2006; Ge et al. 2022; Perumal et al. 2020) for reviews of the problem, solved either sequentially or jointly, and for pointers to the main papers and methods tackling this type of integration. The most common and popular method is column generation, coupled either with Lagrangian relaxations (Borndörfer et al. 2008; Freling et al. 2003; Huisman et al. 2005) or with Branch-and-Price (Friberg and Haase 1999; Haase et al. 2001; Horváth and Kis 2019; Mesquita et al. 2009). However, some papers propose metaheuristics such as Greedy Randomized Adaptive Search procedures (De Leone et al. 2011) or Large Neighborhood Search (Lam et al. 2020; Perumal et al. 2020). Considering these studies, the biggest solved instances have around 1500 timetabled trips (Borndörfer et al. 2008), which roughly corresponds to the instances faced by *SBB Cargo AG* (the biggest instance presented in this paper has 1899 trips).

With respect to the above literature review, and in line with the considered research stream, an important goal of our work is to explore thoroughly at which extent a general-purpose solver can be used on a challenging real-life problem that is faced by railway operators (while considering some specific features faced by *SBB Cargo AG*). The main motivations of using a general-purpose solver are explained in Sect. 1 when presenting the research question. Indeed, given the increasing efficiency of solvers, a growing research stream consists in determining if it is possible to solve industrial problems by means of currently available solvers and, if so, which instance sizes can be solved in acceptable time limits (with respect to practitioners). In such a context, state-of-the-art approaches such as column generation methods, fix-and-optimize techniques, and metaheuristics (e.g., large neighborhood search or variable neighborhood search algorithms) are out of the scope of the considered research stream. Such methods, as well as defining more refined MIP models (e.g., with strengthening cuts), are left as avenues of research. In our context, the proposed matheuristic contains original and unique algorithmic design choices, it is extensively tested, and it performs well on real-life instances. The proposed sequential approach is based on an efficient decomposition of the entire problem into a sequence of smaller subproblems that are suitable for a general-purpose solver. Moreover, the proposed sequence of subproblems is natural in the sense it is

understandable and intuitive for decision makers. This is particularly interesting and appealing from an industrial perspective.

4 Mathematical model

The mathematical model for the integrated locomotive and driver scheduling problem is presented as follows.

- Flow variables and constraints for the locomotives (Subsect. 4.1), with additional features to allow some time flexibility where possible (Subsect. 4.2).
- Routing variables and constraints for the drivers (Subsect. 4.3).
- Break-related additional variables and constraints that ensure feasible driver duties according to their contract (Subsect. 4.4).

The employed main notation is summarized in Table 2. Other models for the locomotive flow and driver routing already exist in the literature, and our contribution here relies mainly on the time flexibility feature. Moreover, the contract-related constraints presented here are the most important ones from the *SBB Cargo AG*'s perspective. They come from their own modeling and were only adjusted to fit this feature addition. Note that because of a non-disclosure agreement, all the costs presented here are based on estimations and approximate formulas. For instance, driver costs could also include training, insurance and some equipment that are not considered here.

As already highlighted above, other modeling decisions and formulations are of course possible (e.g., (Zhu et al. 2014)), including the determination of strengthening cuts. On the one hand, the reader interested in investigating the importance and possible impacts of the formulation can refer to (Bertsimas and Weismantel 2005; Pataki 2003), where strong formulations are discussed for example for the well-known facility location problem and the traveling salesman problem. Such additional modeling efforts are out of the research scope defined above for our paper, and thus left as possible future works. On the other hand, the reader interested in having more details on the rationale and justification of various modeling choices can refer to Ahuja et al. (2005); Frisch et al. (2021); Piu and Speranza (2014); Vaidyanathan et al. (2008a, b) for the locomotive scheduling problem, and to Heil et al. (2020); Portugal and co HRL, Paixão JP, (2009); Wren et al. (2003) for the driver scheduling problem.

4.1 Locomotives flow

For the planning of locomotives on a tactical level, *SBB Cargo AG* currently uses a multi-commodity flow-based optimization approach that computes (cyclic) locomotive tours/circulations. In this approach, locomotive types are scheduled collectively, exploiting possible type substitutions and respecting coupling constraints for the

Table 2 Mathematical notation

Sets	$\mathcal{A}^{\mathrm{D}}$	Arcs of $\mathcal{G}^{\mathrm{D}}$
	$\mathcal{A}^{\mathrm{L}}$	Arcs of $\mathcal{G}^{\mathrm{L}}$
	$\mathcal{B}$	Set of back-arcs in $\mathcal{G}^{\mathrm{L}}$
	$\mathcal{D}$	Driving nodes in $\mathcal{G}^{\mathrm{D}}$
	$\mathcal{G}^{\mathrm{D}}$	Graph for drivers
	$\mathcal{G}^{\mathrm{L}}$	Graph for locomotives
	$\mathcal{H}$	Home nodes in $\mathcal{G}^{\mathrm{D}}$
	$\mathcal{I}(u)$	Set of incoming arcs of u
	$\mathcal{L}$	Set of light-traveling arcs in $\mathcal{G}^{\mathrm{L}}$
	$\mathcal{N}^{\mathrm{D}}$	Nodes of $\mathcal{G}^{\mathrm{D}}$
	$\mathcal{N}^{\mathrm{L}}$	Nodes of $\mathcal{G}^{\mathrm{L}}$
	$\mathcal{O}(u)$	Set of outgoing arcs of u
	$\mathcal{S}$	Set of station locations
	$\mathcal{T}$	Set of train arcs in $\mathcal{G}^{\mathrm{L}}$
	$\mathcal{W}$	Set of waiting arcs in $\mathcal{G}^{\mathrm{L}}$
Data	A_i	Arrival time of train i
	C^{L}	Unit cost for locomotive type k
	C_i	Additional cost for light traveling on arc i
	C^{D}	Unit cost of a duty
	$D_i^{\max}$	Maximum delay before departure of a light traveling train i
	L	Length of the cyclic period
	M	Sufficiently big integer
	T_i	Driving time of node i
	T_{ij}	Traveling time between nodes i and j
	$\mathrm{duty}_{\min}$	Minimum duration of a duty
	$\mathrm{duty}_{\max}$	Maximum duration of a duty
	$\mathrm{break}_{\min}$	Minimum time at which a break can start
	$\mathrm{break}_{\max}$	Maximum time at which a break can start
Variables	b_i	Indicates if breaks have occurred in the duty before train i
	d_i	Actual delay before departure of a light traveling train i
	f	Joint costs
	f^{L}	Locomotive costs
	f^{D}	Driver costs
	f^{C}	Secondary contract objective
	t_i	Relative end time of node i within the duty of its corresponding driver
	w_i	Time spent waiting after driving on node i
	x_i	Number of locomotives that serves the arc i
	y_{ij}	Indicates if node i is served after node j by a driver
	z_i^h	Indicates if node i is served by a driver from home location h

locomotives. Further, the model respects specific traction requirements of the trains (e.g., double traction through long tunnels or along steep sections).

The flow variables and constraints for the locomotives are based on a space-time-expanded graph $\mathcal{G}^{\mathrm{L}} = (\mathcal{N}^{\mathrm{L}}, \mathcal{A}^{\mathrm{L}})$, and an example is represented in Fig. 1. Each node in $\mathcal{N}^{\mathrm{L}}$ is associated with a location and a timestamp, and the arcs in $\mathcal{A}^{\mathrm{L}}$ are of three types (we have here $\mathcal{A}^{\mathrm{L}} = \mathcal{T} \cup \mathcal{W} \cup \mathcal{L}$, and these sets are defined next). First, each train that must be pulled is represented by an arc between its departure and arrival nodes (e.g., arc $1 \rightarrow 2$ represents a train that connects location B to location A). We denote $\mathcal{T}$ as the set of such train arcs, and $\mathcal{S}$ as the set of station locations (in Fig. 1, $\mathcal{S} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$). Second, we have the waiting arc set $\mathcal{W}$, where each arc connects each arrival to the next departure at the same location (e.g., arc $2 \rightarrow 8$ represents waiting at location A), considering cyclic times over the week. Considering cyclic times here means that, at the end of the week, we come back to the beginning of the week (i.e., next Sunday = previous Sunday), which is equivalent to considering the days modulo 7. This makes perfect sense for *SBB Cargo AG*, as their clients demands are weekly periodic. Hence, there is always a next departure (as we are considering a cycle, there is no point where no train comes next), and $\mathcal{B} \subset \mathcal{A}^{\mathrm{L}}$ denotes the set of arcs going back in time, called the back-arcs (e.g., arc $14 \rightarrow 3$). We use back-arcs for cycling as back-arcs go back in the modulo (e.g., from Friday to Monday). Therefore, cutting them will tell us how many locomotive "cycles" (and thus how many locomotives) we use in total. Third, to allow a locomotive to travel without pulling a train, the light-traveling arcs in $\mathcal{L}$ connect each arrival node to each other location, at the earliest possible time (i.e., the timestamp of the arrival node plus the traveling time needed between the two locations). For instance, the arcs $2 \rightarrow 3$ and $2 \rightarrow 4$ connect the arrival of Train 1 at location A to each other location (i.e., B and C). The variables and constraints are represented as a min-cost flow problem under constraints, and therefore the locomotive scheduling problem alone is solvable in polynomial time (as long as we consider only one locomotive type, else it becomes a multi-commodity flow problem, which is in general NP-complete). Even though this information is not useful when solving the entire integrated problem, it is relevant for solution methods using decomposition approaches such as separating

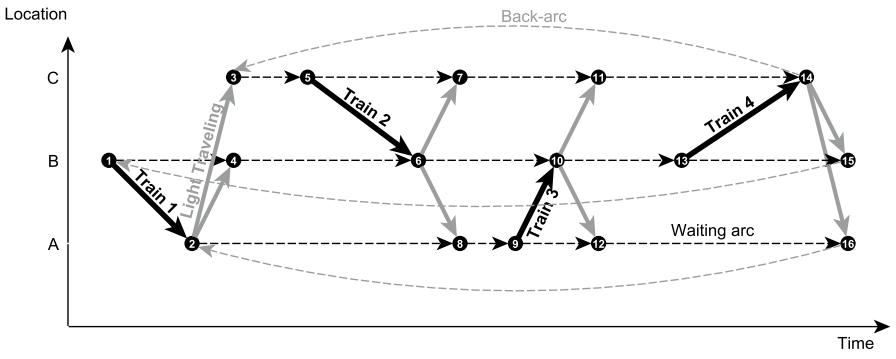

Fig. 1 Flow graph for the locomotives

locomotives and drivers. Regarding Fig. 1, nodes and arcs are inputs. They represent where we need locomotives (trains), or what locomotives can do (wait or light-travel). The variables are the number of locomotives on each arc, and if there are two locomotives, it does not matter which one is pulling which train as we consider one locomotive type. In other words, this graph consists in a small flow decomposition to perform to get the routes per locomotive. Note here that each train (i.e., each arc) can have more than one locomotive. Among the involved locomotives, one has to pull the train, and the other ones belongs to the train because they simply need to be relocated (i.e., such inactive locomotives can be considered as wagons, and they do not need any driver).

Let $x_i \in \mathbb{N}$ be the decision variable indicating the number of locomotives that serve the arc $i \in \mathcal{A}^{\mathrm{L}}$. Note that the set of back-arcs $\mathcal{B}$ is a cut of the flow graph, hence the sum of the flow variables on this set indicates the total flow value of the graph. Therefore, the fix activation cost for the locomotives is $\sum_{i\in\mathcal{B}} x_i \cdot C^{\mathrm{L}}$, where C^{L} is the unit cost for activating one locomotive. More precisely, the value of C^{L} is based on the following calculation: $\frac{\text{purchase + average maintenance costs}}{\text{average locomotive lifetime}} \times$ (one week). Let C_i be the additional cost for light traveling on arc $i \in \mathcal{L}$ (which is evaluated by *SBB Cargo AG*, and mainly relies on network access costs and additional driver costs). The additional pulling cost due to light traveling is thus $\sum_{i\in\mathcal{L}} x_i \cdot C_i$. To give an order of magnitude, we have $C^{\mathrm{L}} \approx 5000$ CHF/week, whereas $C_i \approx$ (length of i in km) $\times\, 3$ CHF/week. The total locomotive costs are therefore summarized in Eq. (1).

$$f^{\mathrm{L}} = \sum_{i\in\mathcal{B}} x_i \cdot C^{\mathrm{L}} + \sum_{i\in\mathcal{L}} x_i \cdot C_i \tag{1}$$

Moreover, the locomotives must satisfy three sets of constraints. First, constraints (2) impose that each scheduled trip i from $\mathcal{T}$ has at least one locomotive. Second, constraints (3) are the flow conservation constraints on each node $u \in \mathcal{N}^{\mathrm{L}}$, where $\mathcal{I}(u)$ (resp. $\mathcal{O}(u)$) is the set of incoming (resp. outgoing) arcs of u. Constraints (4) are the domain constraints.

$$x_i \geq 1 \quad \forall i \in \mathcal{T} \tag{2}$$

$$\sum_{i\in\mathcal{I}(u)} x_i = \sum_{j\in\mathcal{O}(u)} x_j \quad \forall u \in \mathcal{N}^{\mathrm{L}} \tag{3}$$

$$x_i \in \mathbb{N} \quad \forall i \in \mathcal{A}^{\mathrm{L}} \tag{4}$$

4.2 Light traveling time flexibility addition

In the previous graph (Fig. 1), the locomotives do the light traveling as soon as possible (as it does not impact the locomotive flow solution). However, for the drivers, as they have many duty constraints, we want to allow flexible departures for these trains. In order to do so, for each light-traveling arc $i \in \mathcal{L}$, we define

the maximum delay $D_i^{\max}$ as the maximum time a locomotive can wait before its departure, but still reaches the next train arc departure. For instance, the light-traveling arc $2 \rightarrow 3$ can be delayed up to the departure time corresponding to node 5 (i.e., the locomotive performing $2 \rightarrow 3$ must only be on time for starting its mission associated with Train 2, which means that arc $2 \rightarrow 3$ can be slightly shifted to the right, as long as node 3 is not on the right of node 5). For this purpose, we introduce a decision variable d_i that chooses the actual delay waited by the locomotive before departure, which must therefore be smaller than $D_i^{\max}$. To be consistent with the locomotive flow model, the d_i variables are also computed in cyclic time (i.e., modulo one week, to allow the repetition of the schedule every week).

Now, considering that we might want the possibility to delay even more these light-traveling arcs (e.g., we might want to put the arc $2 \rightarrow 3$ even after the departure time of node 5 in order to allow assigning Train 2 to another locomotive), we add new light-traveling arcs. More precisely, we would have to add light-traveling arcs from each location to arrive just after each train departure of another location. As in our model, we have discretized the time with steps of one minute (i.e., the time bucket is one minute), arriving just after means one minute after (arriving two minutes after would result in overconstraining this arc). The resulting graph after such operation on the graph from Fig. 1 is presented in Fig. 2.

The operation Time Flexibility Addition (TFA) of adding these delay variables and new light-traveling arcs exactly doubles the number of light-traveling arcs, and adds one extra decision variable for each traveling arc (new or not). To simplify the time consistency constraints in the following subsection, we define variables with null values ($d_i = 0$ for each train arc $i \in \mathcal{T}$). In other words, we have time flexibility only for light traveling (so the d_i variables only exist for those arcs), but to simplify the writing of some equations, we add time flexibility also on regular trains and set it to 0 (which is actually equivalent to not adding it). Note that adding this new set of light-traveling arcs is only useful when solving the integrated problem. If solving the locomotive and driver problems sequentially, adding time flexibility to the light travels only means adding the variables d_i to the activated light-traveling arcs.

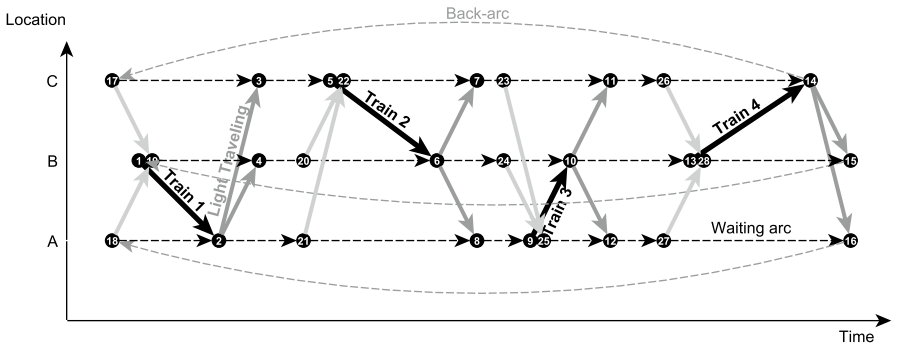

Fig. 2 Flow graph with light travel time flexibility for the locomotives

4.3 Drivers routing

At this stage, the model has been set for the locomotives. Now, we integrate the drivers dimension. We need a driver for each train or light travel, knowing that a driver comes from a home location and needs to get back to it at the end of the shift. As a driver can take a passenger train between two trains that s/he drives, representing these constraints is not straightforward when relying on the above locomotive graphs. Therefore, we introduce a routing graph for the drivers, where each node represents a locomotive (i.e., a train or a light travel) that requires a driver. This type of representation is in line with well-known vehicle routing problem representations.

Let $\mathcal{G}^{\mathrm{D}} = (\mathcal{N}^{\mathrm{D}}, \mathcal{A}^{\mathrm{D}})$ be the routing graph for the drivers. The drivers must go from one locomotive to another; therefore, the set $\mathcal{A}^{\mathrm{D}}$ of nodes actually corresponds to train arcs of the locomotive graph. More precisely, the drivers must serve all nodes of $\mathcal{D} = \mathcal{T} \cup \mathcal{L}$, composed of the trains $\mathcal{T}$ that need a driver, including the light traveling trains $\mathcal{L}$ (only the activated ones in the case of sequential solving, but all of them in the case of integrated solving). Also, we are given a set of home nodes $\mathcal{H}$, from which the drivers start their duty, and to which they must return at the end of the duty. Therefore, we have $\mathcal{N}^{\mathrm{D}} = \mathcal{D} \cup \mathcal{H}$. An example with the train set of Fig. 1 is given in Fig. 3 which is an aggregated graph for all drivers (i.e., there is a small decomposition to perform to know which driver is assigned to which train, which is quite straightforward as their respective duty loops do not really overlap). Each arc of $\mathcal{N}^{\mathrm{D}}$ links either a pair of driving nodes of $\mathcal{D}$ (such arcs are represented by plain black arrows), or a driving node of $\mathcal{D}$ with a home node of $\mathcal{H}$ (such arcs are represented by dashed gray arrows). Note for

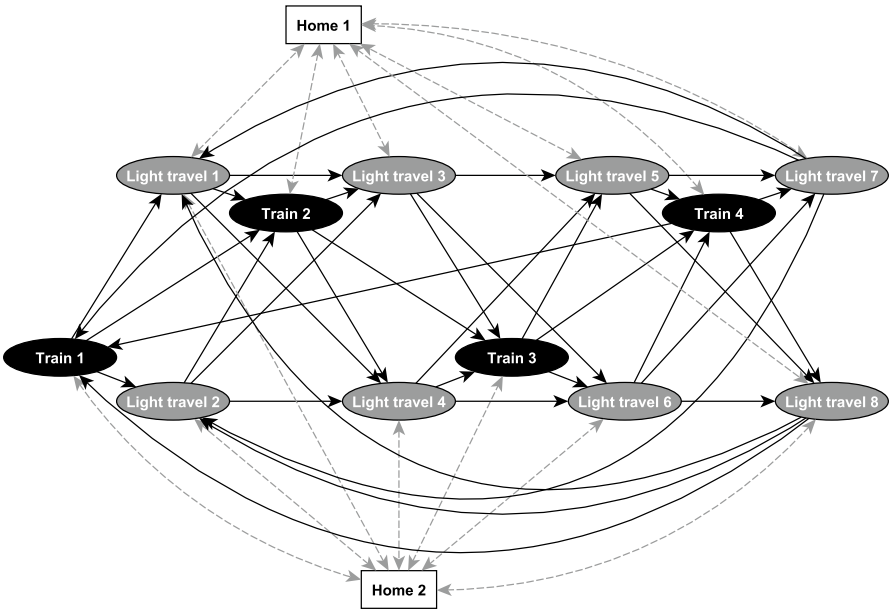

Fig. 3 Routing graph for the drivers

example that there is no arrow from home depot to Train 1. Therefore, the driver coming from depot home 1 cannot serve the mission of Train 1, meaning that s/he has to do some light traveling first (the departure from Train 1 is too far away from her/his home). The size of the graph $\mathcal{G}^{\mathrm{D}}$ will be significantly reduced by the pre-processing procedure presented in Sect. 5.

For this model, we define four additional sets of decision variables. For each arc $(i,j) \in \mathcal{A}^{\mathrm{D}}$, $y_{ij} \in \{0,1\}$ indicates if node i is served after node j by a driver. For each driving node $i \in \mathcal{D}$ and home node $h \in \mathcal{H}$, $z_i^h \in \{0,1\}$ indicates if node i is served by a driver from home location h. This additional set of variables helps to keep track of where a driver comes from, to ensure that s/he goes back home and not to another depot. For each driving node $i \in \mathcal{D}$, $t_i \in \mathbb{N}$ indicates the relative end time of node i within the duty of its corresponding driver, and $w_i \in \mathbb{N}$ is the time spent waiting after driving on node i and traveling to the next location. The composition of a driver's duty is illustrated in Fig. 4: for each driving node, the driver first drives the train, then travels to the departure location of the next train (or to home if there is no next train), and finally waits until the departure. Defining the waiting time w at the end of the nodes allows the driver to wait before going back home, which guarantees the feasibility of the contract-duty-length minimum constraint, even if her/his duty is really short (as we have a minimal length constraint). We denote T_{ij} as the traveling time between two nodes i and j, and as T_i the driving time of node i.

With these variables, the total number of duties is the sum of the drivers that leave a home location $h \in \mathcal{H}$, and the total driver costs are presented in Eq. (5), with C^{D} being the unit cost of a duty. More precisely, the value of C^{D} is based on the following ratio: $\frac{\text{annual salary of a driver}}{\text{number of duties per driver per year}}$. The order of magnitude of this cost is $C^{\mathrm{D}} \approx 800$ CHF/week.

$$f^{\mathrm{D}} = C^{\mathrm{D}} \cdot \sum_{h \in \mathcal{H}} \sum_{i \in \mathcal{D}} y_{hi} \tag{5}$$

The global objective of our integrated problem is to minimize the total costs $f = f^{\mathrm{L}} + f^{\mathrm{D}}$ (i.e., locomotive costs + driver costs).

In addition to minimizing f^{D}, the driver routing problem must satisfy various structural constraints.

- *Driver assignment constraints.* Constraints (6–7) ensure that the y and z variables indicate the same number of drivers per driving node. Constraints (8) state that each driving node i must have exactly one driver if it is a regular train. With the help of a sufficiently big integer M (each M value was chosen as small as pos-

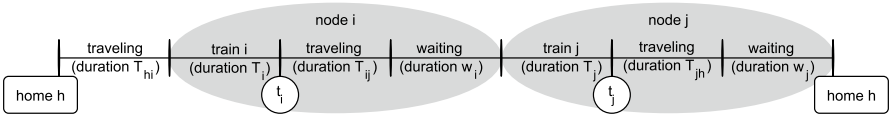

Fig. 4 Example of a duty with two train nodes i and j

sible, see Sect. 6), constraints (9–11) verify that each light traveling train has one driver if it is activated (i.e., if $x_i > 0$ – remember that it can be bigger than 1) and zero if not (i.e., if $x_i = 0$).

- *Home constraints.* Constraints (12–15) ensure the link between the y and z variables, by translating that ($y_{ih} = 1$ or $y_{hi} = 1$) implies $z_i^h = 1$, and $y_{ij} = 1$ implies $z_i^h = z_j^h$ (we need both the last two constraints to ensure that there is equality). Thanks to the z variables, these constraints ensure that each duty starts and ends at the same home location, by transitivity.
- *Sub-tour elimination constraints.* Constraints (16–19) ensure the time consistency of the duties, as shown in Fig. 4. More precisely, if i is the first node of a duty that starts at home location h (i.e., if $y_{hi} = 1$), then it sets $t_i = T_{hi} + T_i$; and if j follows i in a duty (i.e., if $y_{ij} = 1$), then it sets $t_j = t_i + w_i + T_{ij} + T_j$. These constraints prevent sub-tours without a home location at the start and at the end, as the t variables can only increase when going through a driving node $i \in \mathcal{D}$, whereas they are not set in the home location nodes $h \in \mathcal{H}$.
- *Train arrival constraints.* Constraints (20–23) ensure the consistency between the train arrival times and the t_i variables. It ensures that the difference between the actual arrival times $A_i + d_i$ and $A_j + d_j$ of the trains of nodes i and j (A_i and A_j being the arrival times if the associated locomotive does not wait before departure) is the same as the difference between the relative duty times t_i and t_j. The case where $A_j < A_i$ (that can happen as we consider cyclic times) is handled by adding the length of the cyclic period L. Note that if $y_{ij} = 1$, then $t_j > t_i$, and therefore this ensures that if $A_j \geq A_i$, then $A_j + d_j \geq A_i + d_i$, which is why the different cases between constraints (20–23) can be differentiated by comparing only the A_is without the d_is.
- *Domain constraints*. Constraints (24–28) specify the allowed values for the variables.

$$\sum_{j \in \mathcal{N}^{\mathrm{D}}} y_{ij} = \sum_{h \in \mathcal{H}} z_i^h \quad \forall i \in \mathcal{D} \tag{6}$$

$$\sum_{j \in \mathcal{N}^{\mathrm{D}}} y_{ji} = \sum_{h \in \mathcal{H}} z_i^h \quad \forall i \in \mathcal{D} \tag{7}$$

$$\sum_{h \in \mathcal{H}} z_i^h = 1 \quad \forall i \in \mathcal{T} \tag{8}$$

$$\sum_{h \in \mathcal{H}} z_i^h \leq 1 \quad \forall i \in \mathcal{L} \tag{9}$$

$$\sum_{h \in \mathcal{H}} z_i^h \leq x_i \quad \forall i \in \mathcal{L} \tag{10}$$

$$M \cdot \sum_{h \in \mathcal{H}} z_i^h \geq x_i \quad \forall i \in \mathcal{L} \tag{11}$$

$$z_i^h \geq y_{hi} \quad \forall i \in \mathcal{D}, \forall h \in \mathcal{H} \tag{12}$$

$$z_i^h \geq y_{ih} \quad \forall i \in \mathcal{D}, \forall h \in \mathcal{H} \tag{13}$$

$$z_i^h - z_j^h \geq y_{ij} - 1 \quad \forall (i,j) \in \mathcal{D}^2, \forall h \in \mathcal{H} \tag{14}$$

$$z_j^h - z_i^h \geq y_{ij} - 1 \quad \forall (i,j) \in \mathcal{D}^2, \forall h \in \mathcal{H} \tag{15}$$

$$t_i \geq T_{hi} + T_i - M \cdot (1 - y_{hi}) \quad \forall i \in \mathcal{D}, \forall h \in \mathcal{H} \tag{16}$$

$$t_i \leq T_{hi} + T_i + M \cdot (1 - y_{hi}) \quad \forall i \in \mathcal{D}, \forall h \in \mathcal{H} \tag{17}$$

$$t_j \geq t_i + w_i + T_{ij} + T_j - M \cdot (1 - y_{ij}) \quad \forall (i,j) \in \mathcal{D}^2 \tag{18}$$

$$t_j \leq t_i + w_i + T_{ij} + T_j + M \cdot (1 - y_{ij}) \quad \forall (i,j) \in \mathcal{D}^2 \tag{19}$$

$$(A_j + d_j) - (A_i + d_i) \leq t_j - t_i + M \cdot (1 - y_{ij}) \quad \forall i \in \mathcal{D}, \forall j \in \mathcal{D}, \text{ s.t. } A_j \geq A_i \tag{20}$$

$$(A_j + d_j) - (A_i + d_i) \geq t_j - t_i - M \cdot (1 - y_{ij}) \quad \forall i \in \mathcal{D}, \forall j \in \mathcal{D}, \text{ s.t. } A_j \geq A_i \tag{21}$$

$$(A_j + d_j) - (A_i + d_i) + L \leq t_j - t_i + M \cdot (1 - y_{ij}) \quad \forall i \in \mathcal{D}, \forall j \in \mathcal{D}, \text{ s.t. } A_j < A_i \tag{22}$$

$$(A_j + d_j) - (A_i + d_i) + L \geq t_j - t_i - M \cdot (1 - y_{ij}) \quad \forall i \in \mathcal{D}, \forall j \in \mathcal{D}, \text{ s.t. } A_j < A_i \tag{23}$$

$$y_{ij} \in \{0, 1\} \quad \forall (i,j) \in \mathcal{A}^{\mathrm{D}} \tag{24}$$

$$z_i^h \in \{0, 1\} \quad \forall i \in \mathcal{D}, \forall h \in \mathcal{H} \tag{25}$$

$$t_i \in \mathbb{N} \quad \forall i \in \mathcal{D} \tag{26}$$

$$w_i \in \mathbb{N} \quad \forall i \in \mathcal{D} \tag{27}$$

$$d_i \leq D_i^{\max} \quad \forall i \in \mathcal{L} \tag{28}$$

4.4 Contract-related constraints

SBB Cargo AG must respect their driver contracts in order to obey work-time regulation. Therefore, an additional set of decision variables is required: $b_i \in \{0, 1\}$ indicates if breaks have occurred in the duty before train $i \in \mathcal{D}$. With these variables, and to fulfill the contract specifications, the constraint set formally described below must be added.

- *Allowed interval for a duty.* Constraints (29) state that a duty (in minutes) must happen between $\text{duty}_{\min}$ and $\text{duty}_{\max}$, respectively, fixed to 360 and 660 min in this study (i.e., 6 and 11 h). Note that constraints (29) consider the last train before going back home (else if $y_{ih} = 0$, the equation is obviously satisfied as only $\text{duty}_{\max}$ remains), as we only look at the total duty length and the relative time of the trains from the start of the duty.
- *Break occurrence.* Constraints (30) ensure that there is one paid break in each duty. Note that a paid break is necessary a waiting arc (see Sect. 4.1), but a waiting arc is not necessary a paid break (indeed, a driver can wait multiple times).
- *Forbidden intervals for break scheduling.* Constraints (31–32) impose that the start of this break does neither occur in the first $\text{break}_{\min}$ minutes of the duty nor after $\text{break}_{\max}$ minutes (as we consider the start of the break for reference, these constraints are tied just before the waiting, which is why w_i does not appear). In this study, $\text{break}_{\min}$ and $\text{break}_{\max}$ are fixed to 90 and 300 min (i.e., 5 h), respectively. Note that the understanding of constraints (31–32) should start with what happens in the parenthesis: $y_{ij} = 0$ means that the involved arc was not chosen and thus no constraint applies; $b_i = 1$ means that the break has occurred and thus the second constraint must apply (i.e., we must be at least 90 min away from the duty start); $b_i = 0$ means that the break has not occurred yet, and thus the first constraint must apply (i.e., we must not be after 5 h within the duty).
- *Break duration.* Constraints (33–34) specify that the break lasts at least one hour.
- *Variable consistency.* Constraints (35) ensure the consistency between the b and y variables (i.e., it ensures that $b_j \geq b_i$ if $y_{ij} = 1$). Note that constraints (34–35) are just propagation constraints: if a break has happened before train i and next in the duty, train j comes just afterward, it means that a break has occurred before train j in the duty (remember that the b variables are an indicator of whether a break has happened somewhere before or not, not an indicator about if the break has happened exactly before the train).
- *Domain constraints.* Constraints (36) specify the allowed values for the variables.

$$\text{duty}_{\min} \cdot y_{ih} \leq t_i + w_i + T_{ih} \cdot y_{ih} \leq \text{duty}_{\max} \quad \forall i \in \mathcal{D}, \forall h \in \mathcal{H} \tag{29}$$

$$y_{ih} \leq b_i \leq 1 \quad \forall i \in \mathcal{D}, \forall h \in \mathcal{H} \tag{30}$$

$$t_i + T_{ij} \geq \text{break}_{\min} \cdot (b_i + y_{ij} - 1) \quad \forall i \in \mathcal{D}, \forall j \in \mathcal{D} \cup \mathcal{H} \tag{31}$$

$$t_i + T_{ij} \leq \text{break}_{\max} + M \cdot (b_i + 1 - y_{ij}) \quad \forall i \in \mathcal{D}, \forall j \in \mathcal{D} \cup \mathcal{H} \tag{32}$$

$$b_i \leq \frac{w_i}{60} + (1 - y_{hi}) \quad \forall i \in \mathcal{D}, \forall h \in \mathcal{H} \tag{33}$$

$$b_j - b_i \leq \frac{w_j}{60} + (1 - y_{ij}) \quad \forall (i,j) \in \mathcal{D}^2 \tag{34}$$

$$b_j - b_i \geq y_{ij} - 1 \quad \forall (i,j) \in \mathcal{D}^2 \tag{35}$$

$$b_i \in \{0,1\} \quad \forall i \in \mathcal{D} \tag{36}$$

In addition to these constraints, minimizing the driver time that is paid without driving is relevant as well. Therefore, the function presented in Eq. (37) is considered as a secondary objective (i.e., time spent traveling $\sum_{(i,j)\in\mathcal{A}^{\text{D}}} y_{ij} \cdot T_{ij}$ plus time spent waiting $\sum_{i\in\mathcal{D}} w_i$).

$$f^C = \sum_{(i,j)\in\mathcal{A}^{\text{D}}} y_{ij} \cdot T_{ij} + \sum_{i\in\mathcal{D}} w_i \tag{37}$$

This objective function is in line with some studies that have considered the integration of the timetable and the crew schedule on an operational level (Rählmann and Thonemann 2020). Indeed, those problems are usually solved sequentially, which results in suboptimal schedules for the drivers due to large idle times between two train rides.

5 Presentation of the instances and the pre-processing procedure

In this study, we consider 20 instances extracted from the *SBB Cargo AG* data of 2018–2020. Each instance corresponds to a weekly demand extracted from different scenarios (e.g., a type of locomotive, or a specific sub-network). The 20 instances and their characteristics are presented in Table 3.

For each instance, we first indicate the size $|\mathcal{T}|$ of the set of scheduled trains and the size $|\mathcal{S}|$ of the set of the corresponding station locations. Column "Max-parallel" presents the maximum number of trains that are running at the same time; this information gives a lower bound on the numbers of locomotives and drivers that are required. The last two columns give information on the solution of the locomotive flow problem alone (and were obtained by solving this problem). More precisely, column "Locomotives" indicates the minimum number of locomotives required, and column "Light travel" gives the number of light-traveling arcs used in the optimal solution found for the locomotive problem alone.

The instances are divided into four groups, where each group is built based on the instance sizes and minimum numbers of locomotives needed. Indeed, the instances of the first group have up to 112 trains and need at most 10 locomotives, whereas

Table 3 Characteristics of the considered instances

Instance	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{S}\rvert$	Max-parallel	Locomotives	Light travel
I1	46	30	2	3	30
I2	39	32	4	4	24
I3	72	16	4	4	21
I4	101	28	6	6	60
I5	112	31	10	10	8
I6	190	50	9	9	155
I7	315	34	13	19	159
I8	288	41	11	12	191
I9	262	78	9	10	87
I10	403	63	14	17	288
I11	301	97	11	12	109
I12	348	97	15	17	174
I13	340	74	16	19	211
I14	486	102	16	24	189
I15	759	96	17	25	239
I16	1245	148	29	36	449
I17	1122	113	34	37	749
I18	1551	139	36	43	517
I19	1168	77	47	49	576
I20	1899	179	43	52	661
Average	552	76	17	20	245

the instances of the last group have at least 1122 trains and 36 locomotives. This instance separation will allow us to study how our solution methods behave with the increase of the difficulty level (i.e., when moving from the first to the last group).

In line with other studies (e.g., Babayev and Mardanov (1994); Coindreau et al. (2019); Hertz et al. (2005); Weintraub et al. (2008)), we want to reduce the problem size as much as possible before using the proposed algorithms. When using a general-purpose solver, this task can be important, for instance, when memory problems can occur (which is the case for the integrated approach, see Subsect. 7.1). To accomplish this, we propose a pre-processing procedure that reduces first the size of the locomotive flow graph (Subsect. 5.1) and second the size of the driver routing graph (Subsect. 5.2).

5.1 Reducing the locomotive flow graph

In practice, *SBB Cargo AG* barely chooses light traveling trains lasting more than 2 h. Also, in all the considered instances, removing them from the graph does not impact the number of locomotives used in the optimal solution of the locomotive-only flow problem. Therefore, we remove all light travel arcs that travel more than 2 h from our locomotive graph. This observation is only empirical, and the resulting reduced graph could prevent us from finding better solutions. We have however

decided to perform this reduction because it leads to many arc removals, and we conjecture that the risk of missing a better solution is extremely poor (this holds true for most of our graph reductions). Indeed, such an operation removes on average 35% of the traveling arcs from the locomotive flow graph, and therefore, it removes 35% of the traveling nodes from the driver routing graph, which consists of $\mathcal{T} \cup \mathcal{L}$. This is a significant reduction, as the number of arcs of the driver routing graph (before the pre-processing presented in Subsect. 5.2) scales quadratically with the number of nodes in the driver routing graph (as we have an arc between each pair of nodes).

Table 4 shows the effect of the locomotive graph reduction over the instances, both on the graph with and without TFA. The last column shows the average reduction percentage of $|\mathcal{D}|$, computed over the two cases (i.e., with or without TFA). Note that the average reduction percentage of $|\mathcal{D}|$ without TFA and the average reduction percentage of $|\mathcal{D}|$ with TFA are always very close (they differ by at most 1%). We observe that TFA roughly doubles the number of arcs in the locomotive flow graph, as the procedure doubles the number of light-traveling arcs, and as the

Table 4 Locomotive flow graph reduction

Instance	Without TFA		With TFA		
	$\|\mathcal{D}\|$ before pre-processing	$\|\mathcal{D}\|$ after pre-processing	$\|\mathcal{D}\|$ before pre-processing	$\|\mathcal{D}\|$ after pre-processing	Reduction percentage (%)
I1	887	704	1731	1380	21
I2	946	715	1851	1386	24
I3	951	781	1903	1553	18
I4	1948	1570	3998	3261	19
I5	3361	2821	6610	5530	16
I6	5842	4536	12,406	9829	22
I7	9260	5017	18,142	9954	46
I8	8951	6219	17,125	11,628	31
I9	17,050	13,013	34,359	26,083	24
I10	19,441	12,174	38,944	24,238	37
I11	24,104	18,386	48,206	36,659	24
I12	30,639	20,656	55,038	36,321	33
I13	16,491	10,279	33,297	20,281	38
I14	39,894	25,649	83,475	54,387	36
I15	57,734	35,108	120,683	73,009	39
I16	159,400	101,626	324,484	207,438	36
I17	93,227	69,730	198,736	146,843	25
I18	184,396	116,066	382,191	238,853	37
I19	74,885	50,843	153,204	102,234	32
I20	309,261	195,026	604,616	380,490	37
Average	52,933	34,546	107,050	69,568	35

number of train arcs is negligible in comparison. Indeed, as we construct light-traveling arcs between each arrival at a station to the next departure of each other station, $|\mathcal{L}|$ is of the same magnitude as $|\mathcal{T}| \times |\mathcal{S}|$) (i.e., on average 76 times bigger than $|\mathcal{T}|$). However, discarding long light travels reduces the size of $\mathcal{D} = \mathcal{T} \cup \mathcal{L}$ by a percentage ranging from 16 to 46%. Also, this reduction percentage tends to increase with the number of trains (it is between 16 and 24% for the instances with less than 200 trains, and over 24% for the other instances).

5.2 Reducing the driver routing graph

Note that in this section, for sake of clarity, we present the graph reduction that is induced by the rules of the pre-processing procedure. But in the implemented methods, we directly build the graph with the "interesting" arcs, instead of initially considering all the arcs and then removing the "uninteresting" ones.

Without considering the contract-related constraints (29–36) in the driver routing graph, we would need all possible arcs between each pair of nodes of $\mathcal{D}$, and in both ways between each node of $\mathcal{D}$ and each node of $\mathcal{H}$, as explained in Subsect. 4.3. Therefore, we would have a total of $|\mathcal{D}| \cdot (|\mathcal{D}| - 1 + 2 \cdot |\mathcal{H}|)$ arcs in the graph. But as we can see in Table 4, even after the light-traveling arcs reduction, $|\mathcal{D}|$ ranges from 704 to 195,026. Therefore, we can have up to around $4 \cdot 10^{10}$ arcs in the driver graph, which is obviously too much to be handled by a general-purpose solver. However, we can significantly reduce this number by taking into consideration the contract constraints stated in Subsect. 4.4 and by using the *SBB Cargo AG* insights on the arcs that are barely used in practice.

Indeed, we can first remove each obviously inconsistent arc, that is, each arc between two train nodes i and j (from $\mathcal{D} \cup \mathcal{H}$) if a driver cannot physically visit both within one duty (knowing that a duty must not last more than 11 h). Consider for example the trains presented in Fig. 5. In this case, a driver could drive Train 3 after Train 1, or Train 4 after Train 2. However, s/he could not drive Train 2 after Train 1 (as the two trains overlap), Train 4 after Train 1 (as the corresponding duty would last more than 11 h), or Train 4 after Train 3 (as s/he would not have time to travel from B to C between the two trains).

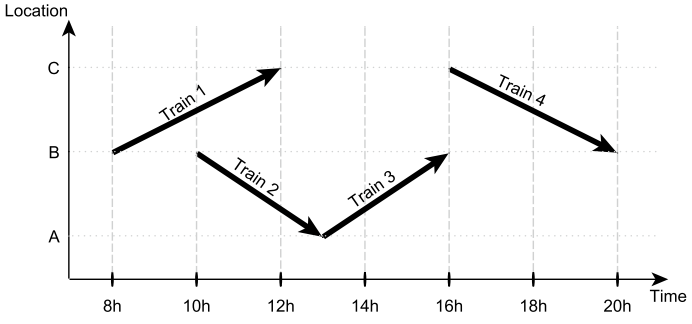

Fig. 5 Example for train-driver duty consistency

In addition, due to practice observation (these arcs were barely used in a duty), we remove each arc between two driving nodes $(i, j) \in \mathcal{D} \times \mathcal{D}$ if traveling between i and j is longer than 5 h, or if waiting between i and j (without considering d_i or d_j) is longer than 4 h, and we remove each arc between a driving node $i \in \mathcal{D}$ and a home node $h \in \mathcal{H}$ (in both directions) if traveling between h and i takes more than 3 h.

Table 5 has the same structure as Table 4. It presents the effect of the driver graph reduction over the different instances, when solving the integrated problem, both with and without TFA. Each column with the label "possible" gives the number of all possible arcs, whereas a column with the label "feasible" only indicates the arcs that are actually feasible. The number of arcs obtained after the pre-processing steps are given in the columns with the label "processed". As the number $|\mathcal{L}|$ roughly doubles when adding TFA, we observe that this option multiplies $|\mathcal{A}^{\mathrm{D}}|$ by a factor of around 4. On average, the number of arcs is reduced by 97% if we consider all possible arcs, and by 46% if we consider the feasible arcs only. However, and in particular for the largest instances, the size of the reduced graphs remains significant: for the biggest instance, we still have a billion arcs in the graph after reduction. Here again, a general-purpose solver is clearly not appropriate to tackle the problem, and therefore justifies the design of matheuristics. Indeed, as it will be shown in Sect. 7, the solver is only able to solve the integrated problem up to around half a million arcs (i.e., only the instances of the first group).

The very large size of the driver routing graph, when solving the locomotive and driver problems together, also fully justifies the fact that these two problems are often solved sequentially. Indeed, if solving the locomotive flow problem first and the driver routing problem second, instead of considering all light travel possibilities of $\mathcal{L}$ in the driver routing, we would only consider the ones that have been activated when solving the locomotive flow problem (i.e., the ones that have a strictly positive flow). More precisely, we would remove all light travel arcs that are not used in the locomotive flow optimal solution (presented in Table 3) from the set $\mathcal{D}$ (presented in Table 4), thus resulting in a much smaller set, as shown in the second column of Table 6. Note that in this case, this set is the same with or without TFA, as when solving the problems sequentially, adding this option does not change the number of light-traveling arcs considered. In the next columns, we indicate the number of arcs that would compose $\mathcal{A}^{\mathrm{D}}$ with such a set, before (column 3 if looking at all possible arcs, column 4 if looking only at feasible ones) and after (column 5) the pre-processing, and the reduction percentage (column 6) brought by the pre-processing (which is of the same order of magnitude than for the integrated problem, except for the smallest instances where it goes to 75%). Finally, the last column shows the ratio between the size of the graph for the integrated problem (without TFA) when compared to the size of the graph for the driver problem only. Looking at the problems sequentially reduces the size of the graph significantly for the smallest instances (the graph size is at least divided by 16) and drastically for the largest instances (the graph size is divided by around 5000 for I20, as we go from around $1.2 \cdot 10^9$ to $2.4 \cdot 10^5$ arcs).

Table 5 Driver routing graph for the integrated approach

Instance	Without TFA			With TFA			Reduction	
	$\lvert\mathcal{A}^D\rvert$ possible	$\lvert\mathcal{A}^D\rvert$ feasible	$\lvert\mathcal{A}^D\rvert$ processed	$\lvert\mathcal{A}^D\rvert$ possible	$\lvert\mathcal{A}^D\rvert$ feasible	$\lvert\mathcal{A}^D\rvert$ processed	From possible (%)	From feasible (%)
I1	$5.1{\cdot}10^5$	$4.9{\cdot}10^4$	$2.9{\cdot}10^4$	$1.9{\cdot}10^6$	$1.7{\cdot}10^5$	$9.3{\cdot}10^4$	94	41
I2	$5.2{\cdot}10^5$	$4.5{\cdot}10^4$	$3.1{\cdot}10^4$	$1.9{\cdot}10^6$	$1.5{\cdot}10^5$	$1.0{\cdot}10^5$	94	31
I3	$6.2{\cdot}10^5$	$4.8{\cdot}10^4$	$3.4{\cdot}10^4$	$2.4{\cdot}10^6$	$1.7{\cdot}10^5$	$1.0{\cdot}10^5$	95	30
I4	$2.5{\cdot}10^6$	$1.9{\cdot}10^5$	$1.1{\cdot}10^5$	$1.1{\cdot}10^7$	$7.5{\cdot}10^5$	$4.3{\cdot}10^5$	95	39
I5	$8.0{\cdot}10^6$	$8.2{\cdot}10^5$	$4.6{\cdot}10^5$	$3.1{\cdot}10^7$	$3.2{\cdot}10^6$	$1.8{\cdot}10^6$	96	43
I6	$2.1{\cdot}10^7$	$1.5{\cdot}10^6$	$7.7{\cdot}10^5$	$9.7{\cdot}10^7$	$6.9{\cdot}10^6$	$3.6{\cdot}10^6$	96	49
I7	$2.5{\cdot}10^7$	$1.1{\cdot}10^7$	$8.2{\cdot}10^5$	$9.9{\cdot}10^7$	$4.2{\cdot}10^7$	$3.2{\cdot}10^6$	97	46
I8	$3.9{\cdot}10^7$	$2.4{\cdot}10^6$	$1.3{\cdot}10^6$	$1.4{\cdot}10^8$	$8.3{\cdot}10^6$	$4.4{\cdot}10^6$	97	44
I9	$1.7{\cdot}10^8$	$2.1{\cdot}10^7$	$5.6{\cdot}10^6$	$6.8{\cdot}10^8$	$8.3{\cdot}10^7$	$2.3{\cdot}10^7$	97	47
I10	$1.5{\cdot}10^8$	$8.3{\cdot}10^6$	$4.5{\cdot}10^6$	$5.9{\cdot}10^8$	$3.3{\cdot}10^7$	$1.8{\cdot}10^7$	97	46
I11	$3.4{\cdot}10^8$	$1.5{\cdot}10^6$	$1.1{\cdot}10^7$	$1.3{\cdot}10^9$	$6.0{\cdot}10^6$	$4.4{\cdot}10^7$	97	46
I12	$4.3{\cdot}10^8$	$2.5{\cdot}10^7$	$4.1{\cdot}10^7$	$1.3{\cdot}10^9$	$7.9{\cdot}10^7$	$4.1{\cdot}10^7$	97	47
I13	$1.1{\cdot}10^8$	$6.2{\cdot}10^6$	$3.5{\cdot}10^6$	$4.1{\cdot}10^8$	$2.4{\cdot}10^7$	$1.4{\cdot}10^7$	97	44
I14	$6.6{\cdot}10^8$	$4.0{\cdot}10^7$	$2.3{\cdot}10^7$	$3.0{\cdot}10^9$	$1.8{\cdot}10^8$	$1.0\cdot10^8$	97	44
I15	$1.2{\cdot}10^9$	$7.1{\cdot}10^7$	$3.8{\cdot}10^7$	$5.3{\cdot}10^9$	$3.1{\cdot}10^8$	$1.7{\cdot}10^8$	97	46
I16	$1.0{\cdot}10^{10}$	$6.1{\cdot}10^8$	$3.2{\cdot}10^8$	$4.3{\cdot}10^{10}$	$2.5{\cdot}10^9$	$1.4{\cdot}10^9$	97	46
I17	$4.9{\cdot}10^9$	$2.9{\cdot}10^8$	$1.5{\cdot}10^8$	$2.2{\cdot}10^{10}$	$1.3{\cdot}10^9$	$6.7{\cdot}10^8$	97	49
I18	$1.3{\cdot}10^{10}$	$7.7{\cdot}10^8$	$4.2{\cdot}10^8$	$5.7{\cdot}10^{10}$	$3.3{\cdot}10^9$	$1.8{\cdot}10^9$	97	46
I19	$2.6{\cdot}10^9$	$1.5{\cdot}10^8$	$8.0{\cdot}10^7$	$1.0{\cdot}10^{10}$	$6.1{\cdot}10^8$	$3.2{\cdot}10^8$	97	46
I20	$3.8{\cdot}10^{10}$	$2.2{\cdot}10^9$	$1.2{\cdot}10^9$	$1.4{\cdot}10^{11}$	$8.3{\cdot}10^9$	$4.5{\cdot}10^9$	97	46
Average	$3.6{\cdot}10^9$	$2.1{\cdot}10^8$	$1.1{\cdot}10^8$	$1.4{\cdot}10^{10}$	$8.4{\cdot}10^8$	$4.5{\cdot}10^8$	97	46

6 Loco-driver matheuristic

The loco-driver matheuristic, presented in Algorithm 1, relies on two main ideas: (1) solve the locomotive flow problem with a fixed number of locomotives (with Algorithm 2); (2) improve the driver duties of the so-obtained solution with the help of a general-purpose solver. Two approaches are possible for (2): (a) use a rolling horizon decomposition (RHD) over the cyclic week (with Algorithm 4); (b) use a direct approach that launches the general-purpose solver on the entire problem, with the total available computing time. Note that this direct approach also provides the general-purpose solver with the initial solution obtained in Step (1). These two approaches (a) and (b) are numerically compared in Sect. 7.

Steps 1 and 2 of Algorithm 1 are performed for different numbers n^{L} of locomotives, ranging from $n^{\mathrm{L}}_{\min}$ to $n^{\mathrm{L}}_{\max}$. On the one hand, the value of $n^{\mathrm{L}}_{\min}$ is presented in Table 3. It is computed by solving (to optimality) the locomotive problem only (i.e., the driver constraints are ignored). On the other hand, $n^{\mathrm{L}}_{\max}$ was fixed to $n^{\mathrm{L}}_{\min} + 9$ (after preliminary experiments). Indeed, larger values for $n^{\mathrm{L}}_{\max}$ never led to better solutions (with respect to objective function values). The allowed computing time of 140 min is explained in Sect. 7. Consequently, Algorithm 1 can be performed within one day of computing time, as 10 values for n^{L} are tested. Note that the number of locomotives being fixed, the remaining main objective function to optimize corresponds to the light traveling distance plus the duty costs (the second objective function f^{L} remains unchanged, as this cost is inherent to number of locomotives).

Algorithm 1 Loco-Driver Matheuristic

Input: $n^L_{\min}$ (smallest possible number of locomotives, as presented in Table 3)
Initialization: set $n^L = n^L_{\min}$, $f^\star = \infty$, $\mathcal{S}^\star = \emptyset$, Continue = True
For ($n^L = n^L_{\min}$ **to** $n^L_{\max}$), **do**:

1. Generate an initial solution $\mathcal{S}$ (using exactly n^L locomotives) with Algorithm 2
2. Improve the driver duties in $\mathcal{S}$ with either (a) or (b) (depending on the chosen method), with a time limit of 140 minutes
   (a) Rolling-Horizon Decomposition (see Algorithm 4)
   (b) Direct approach: launch a general-purpose solver on the entire problem
3. Update the record: **if** $f(\mathcal{S}) < f^\star$, **then** set $f^\star = f(\mathcal{S})$ and $\mathcal{S}^\star = \mathcal{S}$

Output: best-encountered solution $\mathcal{S}^\star$

Algorithm 2 generates an initial solution to the entire problem. It is based on the following steps: solve the locomotive flow problem only (with a general-purpose solver); greedily construct a driver routing solution by assigning a duty per regular train and per activated light-travel train; try to construct longer duties by performing a descent local search (this step is optional and numerical comparisons will be performed with and without it).

Table 6 Driver routing graph when fixing locomotives (i.e., for the sequential approach)

Instance	$\lvert\mathcal{D}\rvert$ without unactivated light arcs	$\lvert\mathcal{A}^{\mathrm{D}}\rvert$ possible	$\lvert\mathcal{A}^{\mathrm{D}}\rvert$ feasible	$\lvert\mathcal{A}^{\mathrm{D}}\rvert$ processed	Reduction percentage (%)	Reduction ratio (vs. integrated)
I1	76	7220	2014	1679	77	17
I2	63	5166	1514	1284	75	24
I3	93	10,416	2231	2083	80	16
I4	162	29,322	4835	4064	86	28
I5	120	16,680	3245	2750	84	169
I6	345	125,580	14,813	10,780	91	74
I7	474	233,682	14,324	15,034	94	55
I8	479	238,542	21,528	16,246	93	89
I9	349	128,432	18,181	10,110	92	557
I10	691	490,610	36,227	26,465	95	171
I11	410	175,890	19,193	12,663	93	873
I12	522	282,402	25,067	17,445	94	762
I13	550	312,950	25,767	19,110	94	182
I14	675	468,450	38,378	27,067	94	834
I15	998	1,012,966	72,605	47,131	95	814
I16	1694	2,901,822	188,849	118,344	96	2743
I17	1840	3,420,560	221,427	133,082	96	1121
I18	2073	4,336,716	277,109	163,894	96	2541
I19	1738	3,053,666	174,671	120,789	96	666
I20	2569	6,648,572	411,414	241,924	96	4847
Average	796	1,195,082	78,670	49,597	91	829

Algorithm 2 Initial Solution Generation

Input: number n^L of locomotives that must be used in the solution

1. Solve the locomotive problem alone with a general-purpose solver, constrained to use n^L locomotives (this gives initial values for all x-variables, and it runs in just a few minutes)
2. Build the driver routing graph according to the chosen approach (sequential/integrated, with/out **TFA** – these two options impact which light traveling are considered in the graph)
3. Construct a driver solution in a greedy fashion, with one driver duty per regular train and activated light-travel train (a driver from the closest home location drives the train and then goes home)
4. (Optional) Perform the Descent Local Search (**DLS**) to try to construct longer duties

Output: solution $\mathcal{S}$ resulting from the above steps

The descent local search (DLS) is presented in Algorithm 3. Its purpose is to reduce the number of duties of a given solution *S*, without changing the parts of the solution regarding the locomotives (i.e., the number of locomotives used and the

light-traveling arcs activated). In order to accomplish that, it performs one type of move: group two duties together, with the possibility of changing the drivers home location (in order to allow grouping two duties with drivers coming from two different home locations). Note that if two duties are combined, then only one driver is needed (i.e., the second driver is not used anymore, as we have no constraint on the number of drivers). At each step, we merge the two duties having the best impact on f^C, and we stop when we cannot merge duties anymore without violating the maximum duty length constraint. We look only at the second objective function f^C when grouping two duties, as it has the same impact on the first objective function whatever are the two duties chosen: it does not change the locomotive costs f^{L} (because we do not change the locomotive solution), and it reduces the drivers cost by $1 \cdot C^{\mathrm{D}}$ (as we decrease the number of duties by 1).

Algorithm 3 Descent Local Search (DLS)

Input: initial solution $\mathcal{S}_0$
Initialization: set $\mathcal{S} = \mathcal{S}_0$, Continue = True
While (Continue = True), **do**:

1. Set $f^\star = \infty$ and $\mathcal{S}^\star = \mathcal{S}$ (where $f^\star$ is the value of the best-encountered solution $\mathcal{S}^\star$)
2. **For each** pair of duties (Δ_1, Δ_2) in $\mathcal{S}$:
   if there is an arc in $\mathcal{A}^D$ from the last train of Δ_1 to the first train of Δ_2:
   - **group** Δ_1 and Δ_2: create one duty $\Delta_{1\cup 2}$ with all the trains that compose the two duties Δ_1 and Δ_2, and with the driver's home location that minimizes the duty's length
   - **if** this new duty $\Delta_{1\cup 2}$ is feasible (i.e., with a length smaller than 11 hours), **then** let $\mathcal{S}'$ denote the solution resulting from $\mathcal{S}$ where Δ_1 and Δ_2 have been replaced by $\Delta_{1\cup 2}$
   - **if** $f^C(\mathcal{S}') < f^\star$, **then** set $\mathcal{S}^\star = \mathcal{S}'$ and $f^\star = f^C(S')$ (i.e., we update the record)
3. **if** $f^\star \neq \infty$, **then** set $\mathcal{S} = \mathcal{S}^\star$ (i.e., the solution $\mathcal{S}$ has been successfully improved), **else** Continue = False.

Output: solution $\mathcal{S}$ (which is a local minimum)

In the rolling horizon decomposition (RHD), presented in Algorithm 4, we try to improve the solution by re-optimizing, with a general-purpose solver (but with a time limit of 5 min), all the duties that start in interval $[t, t + 24$ hours$]$, and we roll this window (i.e., we increase t by a time step of 12 h) over the planning horizon (i.e., the cyclic week) twice. We also relax the light traveling edges that are used in this interval, so that we can choose another one if it helps improving the solution. It is relevant to do it twice as the duties will be longer in the second run, and hence more trains will be considered when including all duties starting in this window. Note that, as we roll the horizon by time steps of 12 h, and as we run it twice over the total horizon of 1 week, we solve the problem 28 times. Consequently, with a time limit of 5 min for the general-purpose solver, this step can take up to $28 \times 5 = 140$ minutes.

Algorithm 4 Rolling-Horizon Decomposition (**RHD**)

Input: initial solution $\mathcal{S}$

1. Set t as the earliest train departure time of the instance
2. Fix the variables of all driver duties that do not start within $[t, t + 24$ hours$]$ and of their involved trains
3. Solve the remaining problem with a general-purpose solver, with a computation time limit of 5 minutes (remember that this cannot change the number of locomotives as they are cyclic in one week, but it can reduce both the light traveling distance activated and the number of duties)
4. Re-do the three above operations while rolling t over the horizon (by 12-hour steps)
5. Stop when t has run through the cyclic week twice

Output: solution $\mathcal{S}$ (which is likely to be improved)

Note that, as described below, this algorithm has a run time in O(number of duties)^3, but this can be easily reduced to O(number of duties)^2 by storing the grouping scores between duties, sorting it, and only computing the grouping scores with the new duty at each iteration (and it is impossible to go below this order of magnitude, because we need at least the scores between each pair for our algorithm, which requires squared computing time). However, we do not describe such improvements here, as they are well-known, and such considerations would unnecessarily complicate the presentation of the method (note that in our case, the computing time of DLS remains reasonable anyway).

For each occurrence of M in the model, it has been implemented in the solver by replacing it by the smallest possible upper bound. More precisely, we have chosen the following values;

- $M = n^{\mathrm{L}}_{\min} + 10$ for constraint (11), as it is an upper bound on the number of locomotives x_i to use;
- $M = 660$ for constraint (16), as if $T_{hi} + T_i > 660$, then the arc $h \to i$ would have been removed from the graph during the pre-processing phase;
- $M = 660$ for constraints (17) and (19), as 660 is an upper bound for t_i due to constraint (29);
- $M = 2 \cdot 660$ for constraint (18) as $t_i + w_i \leq 660$ due to constraint (29), and $T_{ij} + T_i \leq 660$ due to the pre-processing phase;
- $M = D^{\max} + 2 \cdot 660$ for constraints (20–23), with $D^{\max} = \max_{i \in \mathcal{L}}(D^{\max}_i)$ being an upper bound of the d_i possible values, as $t_i \leq 660$ due to constraint (29) and $A_j - A_i \leq 660$ (or $A_j - A_i + L \leq 660$, if $A_j < A_i$) due to the pre-processing phase;
- $M = 2 \cdot 660 - 300$ for constraint (32), as t_i and T_{ij} or both are upper bounded by 660.

7 Results

The proposed approaches were coded in Python 3.6 and run on the Baobab server of the University of Geneva (with Intel 8 cores under a maximum memory constraint of 60GB). As a general-purpose solver, we use Gurobi V9.1. The allowed time limit to solve the entire problem is one day of computation. Indeed, as the problem is at a tactical level, there is no need to provide a solution within a couple of minutes. As 10 values of n^{L} (the number of locomotives) are considered for each instance (see Sect. 6), it means that we have a computing time of 140 min for each considered number of locomotives. We observed that the solver never stops before its allowed time limit (which is not surprising given the complexity of the problems/instances). For these reasons (i.e., tactical nature of the problem, full employment of the available computing time), we will not compare the methods according to speed, but only according to quality (i.e., solution values).

The two considered objective functions f (i.e., locomotive costs + driver costs) and f^C (i.e., the driver time that is spent without driving) are related. Indeed, if there are less light travel in the solution, the duties total length will very probably be shorter. As explained in Sect. 2, f and f^C are optimized in a lexicographic fashion: the cost f has the priority over the inactive time f^C. In other words, no reduction of f^C can be performed if it augments f. In this regard, for each n^{L} value, our optimization approach contains two phases.

- Phase 1. We first try to minimize f for 80% of the time limit (i.e., 112 min).
- Phase 2. Considering the solution generated in Phase 1 as input, we try to reduce f^C for the remaining 20% of the allowed computing time (i.e., 28 min), but without augmenting the value of f.

Doing so (i.e., allocating 80% of the computing time for optimizing f and 20% of the computing time for optimizing f^C) uses a tool proposed directly by the Gurobi solver, and we have observed in preliminary experiments that this time repartition gives the best results according to the lexicography.

For all the presented tables, the average cost f per train connection (i.e., we divide the cost by the number of trains $|\mathcal{T}|$, for normalization over all instances) is presented in Swiss Francs (CHF), the average values of f^C in minutes, and the light distance traveled in kilometers. We decided to show average values in order to better compare the results for different instance sizes. Indeed, the biggest instance has more than 40 times the number of trains contained in the smallest one. For each table, the best values are highlighted in bold font.

In the following subsections, we first briefly compare the results of the integrated and the sequential approaches (Subsect. 7.1). Next, for each of these two approaches, we thoroughly analyze the impact of each component of the method, namely, DLS, RHD and TFA (Subsects. 7.2 and 7.3). It is important to note that DLS and RHD are two algorithms of the Loco-Driver Matheuristic, whereas TFA is related to the modeling part. Finally, we provide managerial insights on the number of locomotives to use and on instance splitting (Subsect. 7.4).

7.1 Sequential versus integrated approach

As a first observation, solving the integrated problem consumes too much memory for our method. Indeed, even without TFA, we are only able to solve the first 8 instances (the ones where $|\mathcal{D}| < 7000$ after pre-processing) if we use RHD, and only 5 instances (the ones where $|\mathcal{D}| < 3000$ after pre-processing) if we solve the problem over the whole week directly. When a memory problem occurs, it happens very early for the five biggest instances (i.e., on the initial linear program), whereas it occurs on the branching phase for the other instances. In contrast, the sequential approach is able to solve every instance with each configuration.

For the 8 instances that could be solved by both approaches, and for each approach (i.e., sequential, and integrated), Table 7 presents the smallest obtained value (over all possible configurations) of the cost per train connection (the details will be discussed in the next subsections). The integrated approach finds a slightly better solution for the instance I1, finds the same solutions as the sequential approach for instances I2 and I3, but it is outperformed by the sequential approach for the other instances. The explanation for this is certainly that the integrated problem becomes too big for the computing time limit given for those instances. For such instances, considering the above common time limit restrictions (i.e., the solver can only provide its best encountered solution during the available time limit), solving a sequence of smaller problems is thus more efficient than solving the entire problem. In other words, these results for the small instances I1–I8 validate the quality of our sequential approach.

7.2 Results for the sequential approach

Table 8 presents the average value of the cost f per train connection obtained for the sequential approach, with all possible configurations of our proposed method: with or without RHD; with or without DLS; with or without TFA. Remember that all these methods use the general-purpose solver, and they differ on the problem to solve and on the employed initial solution. The first three lines present the method configuration, where a check mark indicates the activation of the considered option. For example, the first column presents the result of the method where DLS, RHD and TFA are employed, whereas the fourth column only activates DLS. The last column corresponds to running the solver on the whole problem, given the simplest initial solution (i.e., one duty per train).

The following observations can be made.

- DLS reduces the costs drastically: the average cost is 671.5 with DLS (over the four algorithms), versus 798.3 without it. This improvement is likely to be due to the fact that this operation saves a lot of computational effort for the solver.
- RHD also improves the solutions significantly, as we obtain an average cost of 683.8 with it, versus 786 without it. However, this improvement mostly occurs for large instances. This is likely to mean that: the solution provided by DLS is

already very good for solving the driver routing problem alone; relaxing the light traveling edges only becomes useful if the instance is big enough.
- TFA does not bring a significant advantage: it slightly reduces the average cost when combined with RHD, but it worsens the solutions without it. This implies that TFA leads to a problem that is too big for the solver, unless we use RHD.
- The improvements brought by the components under study are more important for the larger instances.
- The simplest method (i.e., without using any component, see the last column) still generates good solutions for small and medium instances. Indeed, we obtain an average cost per train connection of 787 (resp. 711, 739) for instances I1–I5 (resp. I6–I10, I11–I15). Unfortunately, the method fails for the large instances, as the average cost per train connection is 1201 for instances I16–I20.
- In contrast, the most refined method (i.e., using all three options) does not suffer at all with the augmentation of the instance complexity. Indeed, if we look at the average cost per train connection by group of five instances: we have an average cost of 779 (resp. 639, 621, and 596) for instances I1–I5 (resp. I6–I10, I11–I15, and I16–I20).

For each method configuration, the last three lines of Table 8 show how the total cost (averaged over all the instances) is distributed over its three components: (A) cost due to the number of locomotives (first component of f^{L} in Eq. (1)); (B) cost due to the number of duties (f^{D} in Eq. (5)); (C) cost due to the light distance traveled (second component of f^{L} in Eq. (1)). Without the use of DLS nor RHD, the best solutions found have larger (A) and (B) values, but smaller (C) values, leading, however, to an overall larger total cost. Except for this case, most methods provide almost the same costs with respect to (A) and (C), and the cost differences are explained by (B).

Table 9 has the same structure as Table 8, but it presents the average values of the cost f^C per train connection. Overall, we observe similar behaviors as for the first objective (i.e., $f = f^{\mathrm{L}} + f^{\mathrm{D}}$): RHD and DLS are clearly beneficial, whereas TFA only improves the results slightly if combined with RHD. Moreover, f^C does not really increase with the instance size when using DLS, RHD and TFA: it ranges from 111 to 169, and the results do not appear to be linked with the instance size (the minimum is reached for I9, whereas the maximum is obtained for I10). On the contrary, and as observed for f, the f^C values are very high for the five largest

Table 7 Average cost per train connection when solving the integrated or the sequential problem, over the 8 instances that could be solved by both approaches

Instance	I1	I2	I3	I4	I5	I6	I7	I8
Sequential	738	983	690	**675**	**786**	**596**	**733**	**617**
Integrated	**735**	983	690	710	815	769	876	789

The best results are indicated in bold faces

Table 8 For each method configuration of the sequential approach: average cost per train connection for each instance; distribution of the average cost per train connection over the three cost components

Component	DLS	✓	✓	✓	✓				
	RHD	✓	✓			✓	✓		
	TFA	✓		✓		✓		✓	
Instance	I1	**738**	790	772	790	**738**	790	772	790
	I2	**983**	1004	**983**	**983**	**983**	**983**	**983**	**983**
	I3	701	712	**690**	701	**690**	701	**690**	701
	I4	**675**	691	**675**	683	715	707	711	**675**
	I5	800	793	**786**	**786**	**786**	**786**	**786**	**786**
	I6	630	627	613	**596**	699	697	711	634
	I7	752	749	749	**733**	871	848	1011	871
	I8	634	652	645	**617**	710	744	820	713
	I9	477	480	486	**459**	532	535	599	483
	I10	702	**680**	702	**680**	833	837	1024	856
	I11	494	505	497	**475**	563	536	634	543
	I12	643	646	650	**611**	742	751	940	808
	I13	745	733	742	**705**	860	871	1098	898
	I14	650	649	625	**616**	725	725	862	740
	I15	574	565	567	**525**	607	600	872	708
	I16	566	**565**	622	574	579	579	867	1044
	I17	**641**	642	718	720	666	664	1227	1225
	I18	525	**521**	597	563	546	539	1193	1185
	I19	716	**697**	799	748	732	722	1373	1349
	I20	532	**531**	853	959	537	535	1207	1204
Average cost		**659**	662	689	676	706	708	919	860
Cost distribution	nb. locos	**251**	254	254	255	254	255	269	261
	nb. duties	**376**	377	403	390	421	421	623	569
	light dist.	33	32	32	31	32	31	**27**	29

The best results are indicated in bold faces

instances when none of the method components is active (it reaches an average of 368 for the simplest method).

7.3 Results for the integrated approach

Tables 10 and 11 have an already presented structure. Table 10 (resp. Table 11) presents the average value of f (resp. f^C) per train connection obtained for the integrated approach. Only instances I1 to I4 are considered, because at least one method configuration ran out of memory for the other instances. Overall, the different methods behave as for the sequential approach. Again, DLS improves the results significantly. However, the benefit of RHD is less clear: the average costs are slightly reduced

(resp. augmented) with (resp. without) DLS. Finally, the benefit of TFA seems to be more promising for the integrated approach, even though it leads to worse average results over the four instances. Indeed, looking at them separately, TFA always improves the solution for I1, almost always for I3, does not change it for I2, and worsens it for I4. In other words, TFA has a good potential to improve the solutions, but unfortunately, it increases too much the size of the problem with respect to the solver ability (except for the small instances with less than 100 trains).

7.4 Managerial insights on the number of locomotives and on instance splitting

An interesting managerial insight consists in measuring how the increase of the number of locomotives impacts the results. For each instance, Table 12 compares the solution found with the smallest possible number $n^{\mathrm{L}}_{\min}$ of locomotives with the best solution returned by our methods. In this table, the results in bold highlight the instances for which these two solutions are different. For all the 14 first instances but I7, the same solutions are encountered. In contrast, for the larger instances (I15–I20) and I7, the number of locomotives increases (on average by 10%) in order to decrease the light distance traveled (on average by 33%) and the number of duties (on average by 3%), leading to an overall cost reduction of around 1%. In other words, using more than the minimum possible number of locomotives (to satisfy the demand) is only interesting if the number of trains to be pulled is really significant (more than 500).

This naturally leads to the following question: would it be better to decompose the big instances into smaller ones, and therefore having smaller problems to solve and hence a larger chance of finding the best solution in the allowed computing time? To investigate this aspect, we consider the three instances I11, I16 and I20, which were actually built by combining the train connections of two other instances (involving two compatible locomotive types). More precisely, we have $\mathcal{T}(\mathrm{I11}) = \mathcal{T}(\mathrm{I2}) + \mathcal{T}(\mathrm{I9})$, $\mathcal{T}(\mathrm{I16}) = \mathcal{T}(\mathrm{I14}) + \mathcal{T}(\mathrm{I15})$, and $\mathcal{T}(\mathrm{I20}) = \mathcal{T}(\mathrm{I12}) + \mathcal{T}(\mathrm{I18})$. In this context, Table 13 compares the results obtained when solving the two instances separately (e.g., I1 and I9), versus when solving the instances with their combined trains (e.g., I11). Two methods are compared: the simplest (i.e., without any feature among DLS, RHD and TFA) and the most refined (i.e., with all the features activated in the method). The following two main observations can be made. First, the results of the refined method highlight again the need for DLS, RHD and TFA. Second, for the refined method, the results are clearly in favor of looking at the entire instance (versus decomposing it), even if more duties are created. However, for the simplest method, the whole instance is too big for two of the three cases. This observation is particularly interesting. It indicates that a sequential approach cannot simply be reduced to instance splitting, highlighting again our contribution with respect to the proposed problem decomposition of the sequential approach.

Table 9 Average solution values with respect to f^C, for each method configuration of the sequential approach, and for each instance

Component	DLS	✓	✓	✓	✓				
	RHD	✓	✓			✓	✓		
	TFA	✓		✓		✓		✓	
Instance	I1	143	146	158	**132**	147	176	157	133
	I2	164	184	160	**128**	162	158	176	129
	I3	127	141	138	**98**	117	125	132	**98**
	I4	149	149	141	**103**	183	178	137	105
	I5	123	124	126	**84**	116	123	125	85
	I6	136	137	136	**92**	147	170	165	99
	I7	135	147	146	**96**	164	231	230	148
	I8	128	131	132	**83**	152	181	197	124
	I9	111	113	115	**72**	125	134	179	85
	I10	169	**117**	169	**117**	192	207	314	178
	I11	111	119	116	**76**	130	133	174	103
	I12	155	151	160	**103**	186	123	263	179
	I13	166	163	167	**109**	207	192	297	183
	I14	143	142	139	**101**	166	165	274	147
	I15	157	158	153	**101**	150	140	269	181
	I16	144	144	165	**129**	138	139	283	354
	I17	157	157	188	165	150	**147**	427	373
	I18	135	133	168	136	127	**126**	377	373
	I19	150	144	192	150	**135**	141	368	358
	I20	**126**	127	265	294	127	127	384	383
Average cost		141	141	157	**118**	151	156	246	191

The best results are indicated in bold faces

Table 10 Average cost per train connection for each method configuration of the integrated approach

Component	DLS	✓	✓	✓	✓				
	RHD	✓	✓			✓	✓		
	TFA	✓		✓		✓		✓	
Instance	I1	750	782	**735**	782	765	765	753	765
	I2	**983**	**983**	**983**	**983**	1045	1045	1004	1004
	I3	**690**	701	712	712	701	724	741	701
	I4	956	**710**	1460	716	810	739	810	739
Average cost		845	**794**	973	798	830	818	827	802

The best results are indicated in bold faces

8 Conclusion

Scheduling locomotives and drivers is one of the main tasks of each railway company. In this study, we first propose a mathematical lexicographic model (including

Table 11 Average solution values with respect to f^C for each method configuration of the integrated approach

Component	DLS	✓	✓	✓	✓				
	RHD	✓	✓			✓	✓		
	TFA	✓		✓		✓		✓	
Instance	I1	**128**	153	**128**	138	140	157	**128**	141
	I2	129	**128**	130	132	153	199	162	179
	I3	116	108	109	113	**101**	118	114	104
	I4	136	**132**	432	148	225	144	225	144
Average cost		143	**126**	200	133	155	155	157	142

The best results are indicated in bold faces

all driver-contract-related constraints as well) to solve this problem, relying on a general-purpose solver. We also propose an extension of this model that allows time flexibility on the light travels. Next, we develop a general matheuristic that uses the solver to find solutions to this problem, either in an integrated or a sequential way, with the help of additional optimization features (e.g., a rolling horizon decomposition, a descent local search for improving solutions).

Table 12 Comparison of the best solution found over all algorithms according to the chosen number of locomotives

		Solution obtained with $n^{\mathrm{L}}_{\min}$				Best solution obtained over all n^{L} values			
		Number of locos	Number of duties	Light distance	Average cost	Number of locos	Number of duties	Light distance	Average cost
Instance	I1	3	16	2002	735	3	16	2002	735
	I2	4	19	1047	983	4	19	1047	983
	I3	4	37	31	690	4	37	31	690
	I4	6	44	995	675	6	44	995	675
	I5	10	47	146	786	10	47	146	786
	I6	9	76	2506	596	9	76	2506	596
	I7	**19**	**163**	**1948**	**734**	**20**	**158**	**1488**	**733**
	I8	12	137	2701	617	12	137	2701	617
	I9	10	80	2071	459	10	80	2071	459
	I10	17	211	6756	680	17	211	6756	680
	I11	12	95	2357	475	12	95	2357	475
	I12	17	138	5764	611	17	138	5764	611
	I13	19	169	3122	705	19	169	3122	705
	I14	24	201	6243	616	24	201	6243	616
	I15	**25**	**312**	**8186**	**526**	**30**	**293**	**4594**	**525**
	I16	**36**	**610**	**14,880**	**572**	**39**	**594**	**11,042**	**565**
	I17	**37**	**642**	**11,078**	**652**	**41**	**613**	**7895**	**641**
	I18	**43**	**707**	**13,347**	**529**	**48**	**677**	**8776**	**521**
	I19	**49**	**697**	**6035**	**703**	**52**	**679**	**3441**	**697**
	I20	**52**	**884**	**16,134**	**535**	**57**	**863**	**11,055**	**531**

The best results are indicated in bold faces

Table 13 Instance combination comparison: for three instances, result variations if we consider them split in two or as a whole

		Without: DLS, RHD, TFA				With: DLS, RHD, TFA			
		Number of locos	Number of duties	Light distance	Average cost	Number of locos	Number of duties	Light distance	Average cost
Instance	I2+I9	14	107	3118	548	14	**105**	3118	**543**
	I11	13	115	**2131**	543	**12**	**102**	2357	**494**
	I14+I15	51	755	**12,535**	720	**49**	**580**	14,211	**604**
	I16	45	1317	**7063**	1044	**39**	**595**	11,042	**566**
	I12+I18	71	2161	**11,715**	1116	**62**	**848**	16,780	**547**
	I20	61	2442	**8958**	1204	**57**	**866**	11,055	**532**

The best results are indicated in bold faces

Considering 20 real instances, the first result of this study is that the general-purpose solver is overwhelmed by the integrated problem, as it runs out of memory for 80% of the instances. In contrast, our matheuristic is able to solve the sequential problem efficiently, but it requires the use of specific features (i.e., rolling horizon decomposition and descent local search). Two managerial insights can be deduced from our work. First, if a company would like to solve an optimization problem with a commercial solver but the problem is too big for it, it is recommended to decompose the problem into an efficient (with respect to solution quality) and natural (with respect to the involved decision maker) sequence of smaller subproblems that are suitable for the commercial solver at hand. As highlighted by the experiments conducted in Sect. 7.4, we cannot simply split an instance into smaller parts to be efficient (considering the entire instance was indeed more efficient, even if more duties are generated). Second, we have shown that increasing the number of locomotives (to reduce the light travel distance and the number of duties) has the potential of reducing the overall total cost for the biggest instances.

Among the straightforward future works, we suggest the development of solution methods to tackle the integrated locomotive and driver scheduling problem, or to add driver considerations when solving the first locomotive flow. Promising candidates that have been successfully applied for other problems with common features (e.g., transportation scheduling, consideration of various objectives, and even crew scheduling) are column generation (Gaur and Singh 2017; Crainic and Rousseau 1987), variable neighborhood search (Thevenin and Zufferey 2019), and fix-and-optimize techniques (Coindreau et al. 2021). Moreover, this would allow to investigate further the potential of our time flexibility mechanism for the light travels, as we have shown that it could improve the solutions of the integrated problem, but unfortunately it increases too much the size of the model with respect to the solver ability. Finally, one could also explore how these algorithms could be parallelized to make a better use of the allocated computing time.

Acknowledgements This study was funded in part by *SBB Cargo AG*. The authors would like to thank Philipp Germann and Carsten Moldenhauer from *SBB Cargo AG* for their availability and advices. We thank the three anonymous reviewers for their valuable comments.

Funding Open access funding provided by University of Geneva.

Data availability No data are publicly available.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

Abbink E, Fischetti M, Kroon L, Timmer G, Vromans M (2005) Reinventing crew scheduling at Netherlands Railways. Interfaces 35(5):393–401

Ahuja R, Liu J, Orlin J, Sharma D, Shughart L (2005) Solving real-life locomotive-scheduling problems. Transp Sci 39(4):503–517

Amrani H, Martel A, Zufferey N, Makeeva P (2011) A variable neighborhood search heuristic for the design of multicommodity production-distribution networks with alternative facility configurations. Oper Res Spectr 33(4):989–1007

Archetti C, Speranza M (2014) A survey on matheuristics for routing problems. EURO J Comput Optim 2(4):223–246

Babayev DA, Mardanov SS (1994) Reducing the number of variables in integer and linear programming problems. Comput Optim Appl 3:99–109

Ball M (2011) Heuristics based on mathematical programming. Surv Oper Res Manag Sci 16(1):21–38

Bertossi A, Carraresi P, Gallo G (1987) On some matching problems arising in vehicle scheduling models. Networks 17(3):271–281

Bertsimas D, Weismantel R (2005) Optimization over integers. Dynamic Ideas, Belmont, Massachusetts

Borndörfer R, Löbel A, Weider S (2008) A bundle method for integrated multi-depot vehicle and duty scheduling in public transit. In: Hickman M, Mirchandani P, Voss S (eds) Computer-aided systems in public transport, vol 600. Springer, Berlin, Heidelberg, pp 3–24

Boschetti M, Mingozzi A, Ricciardelli S (2004) An exact algorithm for the simplified multiple depot crew scheduling problem. Ann Oper Res 127(1–4):177–201

Bouzaiene-Ayari B, Cheng C, Das S, Fiorillo R, Powell WB (2014) From single commodity to multiattribute models for locomotive optimization: a comparison of optimal integer programming and approximate dynamic programming. Transp Sci 50(2):366–389

Burdett RL, Kozan E (2010) A sequencing approach for creating new train timetables. OR Spectr 32:163–193

Caprara A, Kroon L, Monaci M, Peeters M, Toth P (2006) Passenger railway optimization. Handbooks in operations research and management science: transportation. Elsevier, Amsterdam, pp 129–188

Cardenas-Barron LE, Melo RA (2021) A fast and effective MIP-based heuristic for a selective and periodic inventory routing problem in reverse logistics. Omega 103:102394

Coindreau MA, Gallay O, Zufferey N, Laporte G (2019) Integrating workload smoothing and inventory reduction in three intermodal logistics platforms of a European car manufacturer. Comput Oper Res 112:104762

Coindreau MA, Gallay O, Zufferey N, Laporte G (2021) Inbound and outbound flow integration for cross-docking operations. Euro J Oper Res 294(3):1153–1163

Crainic T, Rousseau JM (1987) The column generation principle and the airline crew scheduling problem. INFOR Inf Syst Oper Res 25(2):136–151

De Leone R, Festa P, Marchitto E (2011) A bus driver scheduling problem: a new mathematical model and a GRASP approximate solution. J Heuristics 17(4):441–466

Fores S, Proll L, Wren A (2002) TRACS II: a hybrid IP/heuristic driver scheduling system for public transport. J Oper Res Soc 53(10):1093–1100

Freling R, Huisman D, Wagelmans A (2003) Models and algorithms for integration of vehicle and crew scheduling. J Sched 6(1):63–85

Friberg C, Haase K (1999) An exact branch and cut algorithm for the vehicle and crew scheduling problem. Comput Aided Transit Schedul. Springer, Berlin, Heidelberg, pp 63–80

Frisch S, Hungerländer P, Jellen A, Weinberger D (2019) a mixed integer linear program for optimizing the utilization of locomotives with maintenance constraints. In: Fortz B, Labbé M (eds) Operations research proceedings 2018. Springer International Publishing, Cham, pp 103–109

Frisch S, Hungerländer P, Jellen A, Primas B, Steininger S, Weinberger D (2021) Solving a real-world locomotive scheduling problem with maintenance constraints. Transp Res Part B Methodol 150:386–409

Gaur D, Singh R (2017) A heuristic for cumulative vehicle routing using column generation. Discret Appl Math 228:140–157

Ge L, Kliewer N, Nourmohammadzadeh A, Voss S, Xie L (2022) Revisiting the richness of integrated vehicle and crew scheduling. Public Transp. https://doi.org/10.1007/s12469-022-00292-6

Haahr JT, Wagenaar JC, Veelenturf LP, Kroon LG (2016) A comparison of two exact methods for passenger railway rolling stock (re)scheduling. Transp Res Part E Logist Transp Rev 91:15–32
Haase K, Desaulniers G, Desrosiers J (2001) Simultaneous vehicle and crew scheduling in urban mass transit systems. Transp Sci 35(3):286–303
Hansen JR, Fagerholt K, Meisel F (2022) A MIP-based heuristic for a single trade routing and scheduling problem in roll-on roll-off shipping. Comput Oper Res 146:105904
Heil J, Hoffmann K, Buscher U (2020) Railway crew scheduling: models, methods and applications. Eur J Oper Res 283(2):405–425
Hertz A, Schindl D, Zufferey N (2005) Lower bounding and tabu search procedures for the frequency assignment problem with polarization constraints. 4OR 3(2):139–161
Horváth M, Kis T (2019) Computing strong lower and upper bounds for the integrated multiple-depot vehicle and crew scheduling problem with branch-and-price. CEJOR 27(1):39–67
Huisman D, Freling R, Wagelmans A (2005) Multiple-depot integrated vehicle and crew scheduling. Transp Sci 39(4):491–502
Jütte S, Thonemann U (2015) A graph partitioning strategy for solving large-scale crew scheduling problems. OR Spectr 37:137–170
Kchaou Boujelben M, Gicquel C, Minoux M (2014) A distribution network design problem in the automotive industry: MIP formulation and heuristics. Comput Oper Res 52:16–28
Kopanos GM, Mendez CA, Puigjaner L (2010) MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. Eur J Oper Res 207(2):644–655
Kwan R (2011) Case studies of successful train crew scheduling optimisation. J Sched 14:423–434
Lam E, Van Hentenryck P, Kilby P (2020) Joint vehicle and crew routing and scheduling. Transp Sci 54(2):488–511
Leggieri V, Haouari M (2018) A matheuristic for the asymmetric capacitated vehicle routing problem. Discret Appl Math 234:139–150
Lehuédé F, Péton O, Tricoire F (2020) A lexicographic minimax approach to the vehicle routing problem with route balancing. Eur J Oper Res 282(1):129–147
Lenstra J, Kan A (1981) Complexity of vehicle routing and scheduling problems. Networks 11(2):221–227
Malaguti E, Toth P (2010) A survey on vertex coloring problems. Int Trans Oper Res 17(1):1–34
Mesquita M, Paias A, Respício A (2009) Branching approaches for integrated vehicle and crew scheduling. Public Transp 1(1):21–37
Pataki G (2003) Teaching integer programming formulations using the traveling salesman problem. SIAM Rev 45(1):116–123
Perumal S, Dollevoet T, Huisman D, Lusby R, Larsen J, Riis M (2020) Solution approaches for vehicle and crew scheduling with electric buses. Department of Technology, Management and Economics, Technical University of Denmark, Tech. rep
Perumal S, Lusby R, Larsen J (2020) A review of integrated approaches for optimizing electric vehicle and crew schedules. Department of Technology, Management and Economics, Technical University of Denmark, Tech. rep
Piu F, Speranza M (2014) The locomotive assignment problem: a survey on optimization models. Int Trans Oper Res 21:327–352
Portugal R, Co HRL, Paixão JP (2009) Driver scheduling problem modelling. Public Transp 1:103–120
Prats X, Puig V, Quevedo J, Nejjari F (2010) Lexicographic optimisation for optimal departure aircraft trajectories. Aerosp Sci Technol 14(1):26–37
Rählmann C, Thonemann U (2020) Railway crew scheduling with semi-flexible timetables. OR Spectr 42:835–862
Respen J, Zufferey N, Wieser P (2017) Three-level inventory deployment for a luxury watch company facing various perturbations. J Oper Res Soc 68(10):1195–1210
Sama M, D'Ariano A, Palagachev K, Gerdts M (2019) Integration methods for aircraft scheduling and trajectory optimization at a busy terminal manoeuvring area. OR Spectr 41:641–681
Scheffler M, Neufeld J, Hölscher M (2020) An MIP-based heuristic solution approach for the locomotive assignment problem focussing on (dis-)connecting processes. Transp Res Part B Methodol 139:64–80
Silver EA (2004) An overview of heuristic solution methods. J Oper Res Soc 55:936–956

Solnon C, Cung VD, Nguyen A, Artigues C (2008) The car sequencing problem: overview of state-of-the-art methods and industrial case-study of the ROADEF 2005 challenge problem. Eur J Oper Res 191(3):912–927
Thevenin S, Zufferey N (2019) Learning variable neighborhood search for a scheduling problem with time windows and rejections. Discret Appl Math 261:344–353
Thevenin S, Zufferey N, Potvin JY (2017) Makespan minimization for a parallel machine scheduling problem with preemption and job incompatibility. Int J Prod Res 55(6):1588–1606
Toscano A, Ferreira D, Morabito R (2020) Formulation and MIP-heuristics for the lot sizing and scheduling problem with temporal cleanings. Comput Chem Eng 142:107038
Vaidyanathan B, Ahuja R, Liu J, Shughart L (2008a) Real-life locomotive planning: new formulations and computational results. Transp Res Part B Methodol 42:147–168
Vaidyanathan B, Ahuja R, Orlin J (2008b) The locomotive routing problem. Sci Transp 42(4):492–507
Vié MS, Zufferey N, Cordeau JF (2019) Solving the wire-harness design problem at a European car manufacturer. Eur J Oper Res 272(2):712–724
Vié MS, Zufferey N, Leus R (2022) Aircraft landing planning under uncertainties. J Sched 25:203–208
Weintraub A, Pereira M, Schultz X (2008) A priori and a posteriori aggregation procedures to reduce model size in MIP mine planning models. Electr Notes Discret Math 30:297–302
Wren A, Fores S, Kwan AKR, Parker M, Proll L (2003) A flexible system for scheduling drivers. J Sched 6:437–455
Yunes T, Moura A, De Souza C (2005) Hybrid column generation approaches for urban transit crew management problems. Transp Sci 39(2):273–288
Zhang Y, Peng Q, Lu G, Zhong Q, Yan X, Zhou X (2022) Integrated line planning and train timetabling through price-basedcross-resolution feedback mechanism. Transp Res Part B Methodol 155:240–277
Zhong Q, Lusby RM, Larsen J, Zhang Y, Peng Q (2019) Rolling stock scheduling with maintenance requirements at the Chinese high-speed railway. Transp Res Part B Methodol 126:24–44
Zhu E, Crainic TG, Gendreau M (2014) Scheduled service network design for freight rail transportation. Oper Res 62(2):383–400

Authors and Affiliations

Marie-Sklaerder Vié[1,2] · Nicolas Zufferey[1,2] · Stefan Minner[3,4]

✉ Nicolas Zufferey
n.zufferey@unige.ch

Marie-Sklaerder Vié
marie-sklaerder.vie@unige.ch

Stefan Minner
stefan.minner@tum.de

[1] Geneva School of Economics and Management, GSEM, University of Geneva, Geneva, Switzerland

[2] UniMail, Bd du Pont-d'Arve 40, 1211 Geneva 4, Switzerland

[3] TUM School of Management, Technical University of Munich, Munich, Germany

[4] Munich Data Science Institute (MDSI), Garching, Germany