1 Introduction

DB Cargo Polska works with the Chair of Analytics and Mixed Integer Optimization at Friedrich-Alexander-Universität Erlangen-Nürnberg to build a prototype of a new planning tool, which would come up with locomotive schedules and driver rosters simultaneously. As of today, the industry standard is to plan locomotives and drivers separately. The reason for this is the fact that both the planning areas are challenging on their own, and – without computational support – it would be impossible to consider both the areas at the same time. Such fragmented approaches were criticised in the literature already in 1980s' – see for example Raff (1983).

The purpose of the presented benchmarks is to present a set of instances, which we derived in collaboration with the industrial partner, as well as the procedures required to come up with complete Binary Programming models for these instances. In this work, we will only introduce the naive formulation of the mentioned binary models. Formulation improvements, as well as the algorithms to efficiently solve these models will be introduced in Deliverables D4.2 and D4.3.

1.1 A joint model for locomotive scheduling and driver rostering in rail freight transport

We model the joint locomotive scheduling and driver rostering problem as a combination of a set packing problem with compatibility, conflict and multiple-choice constraints and a multicommodity flow problem. Our objective is to maximize the number of trains performed, i.e. to maximize the number of trains for which both a locomotive and a driver were found. The inputs to the model are: a set T of trains to be performed, a set of locomotives $\mathscr{L}$ and a set of drivers D. To denote the sets of locomotives compatible with a driver $d \in D$ or a train $t \in T^d$, we use $\mathscr{L}^d$ and $\mathscr{L}^t$ respectively. Let D^l and D^t represent a set of drivers compatible with a locomotive $l \in \mathscr{L}$ or a train $t \in T$ respectively. Finally, let T^l and T^d be a set of trains compatible with a locomotive $l \in \mathscr{L}$ or a driver $d \in D$ respectively.

Sets required for constraints construction We also introduce a number of sets required to build the constraints of the model. Table 1 presents a summary of the sets required for constraint construction. Their exact definitions can be found in the Appendix A.

Name	Description
$T^{B+}_{t,d}$	Trains which cannot be assigned to a driver d if t is his last job in a working day
$T^{35h}_{t,d}$	Trains which cannot be assigned to a driver d if t is his last job before a weekly 35h break
$T^{B-}_{t,d}$	Trains which cannot be assigned to a driver d if t is his first job in a working day
$T^{week}_{w,d}$	Trains which belong to a calculation week w
$T^{sunday}_{w,d}$	Trains which belong to a Sunday falling on a week w
$T^{shift\_beginning}_{t,d}$	Past trains which could have been assigned to driver d if he is assigned to a job t.
$T^{shift\_end}_{t,d}$	Future trains which can be assigned to driver d if he is assigned to a job t.
$T^{time}_{t,d}$	Trains which are feasible for driver d and in time conflict with the train t
$T^{after\_break}_{t,d}$	Trains which could be the first jobs of the next shift after the 11h break following train t
$T^{before\_break}_{t,d}$	Trains which could have been the last job of the previous shift before the 11h break preceding train t
$T^{next\_l}_{t,l}$	Future trains which could be assigned to locomotive l if is assigned to a job t

Table 1: Descriptions of the sets required for constraint construction

1.1.1 Multi-commodity flow part of the model

We consider the set $\mathscr{L} = \{l_1, l_2, l_3, \ldots\}$ of locomotives to be modeled as commodities which need to be "pushed" through a directed graph $G = (V, A)$, which could be defined as $V \coloneqq T$ and

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

Let us also define Σ and $\Theta \in V$ as the source and sink nodes of the graph G, respectively. Additionally let us assume that we have one item of each commodity. Further, let each arc $a \in A$ have unit capacity i.e. it can host at most one commodity. We also assume the limited compatibility of each arc with regard to commodities – that means we may not be allowed to push every commodity through every arc.

1.1.2 Decision variables

In the model, we need to make sure that each train is staffed by exactly one suitable driver and one suitable locomotive. These decisions are modelled with binary variables x_d^t (for drivers) and $f_l^{t_1,t_2}$ (for locomotives).

To comply with the working time requirements, we need to distinguish between the first job in a shift, denoted by a binary variable y_d^t, the last job in a shift before a short (11 hour) break, denoted by a binary variable v_d^t and the last job in a shift before a long (35 hour) break, denoted by a binary variable z_d^t. We also need to know whether a driver has worked on a given Sunday. This is denoted by a binary variable h_d^w.

Finally, for modelling purposes we also need to know which trains t are the first and the last job for drivers in the planning period. We do that with the help of binary variables α_d^t and ω_d^t, which denote that the train t is respectively the first or the last one in the planning period for a driver $d \in D$. All the variables are summarized in Table 2.

Name	Description	Type
$f_l^{u,v}$	Trains u, v are served by a locomotive l	binary
x_d^t	Train t is served by a driver d	binary
y_d^t	Train t is the first job of a driver d in their shift	binary
v_d^t	Train t is the last job of a driver d before a 12h break	binary
z_d^t	Train t is the last job of a driver d before a 35h break	binary
α_d^t	Train t is the first train of driver d in the planning period	binary
ω_d^t	Train t is the last train of driver d in the planning period	binary
h_d^w	Driver d has worked on the Sunday of the week w	binary

Table 2: Summary of decision variables used in the model

1.1.3 Model formulation

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

s.t.

$$x_d^t \leq \sum_{\substack{l \in \mathscr{L}^d \cap \mathscr{L}^t \\ u:(t,u) \in E}} f_l^{t,u} \qquad (\forall t \in T)(\forall d \in D^t) \tag{1.2}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\sum_{v\in\delta^{in}(t)\cap\, T^l} f_l^{v,t} - \sum_{w\in\delta^{out}(t)\cap\, T^l} f_l^{t,w} = 0 \qquad (\forall t\in T)\ (\forall l\in\mathscr{L}^t) \tag{1.24}$$

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

$$\sum_{l\in\mathscr{L}^t}\sum_{t_0\in\delta^{in}(t)\cap T^l} f_l^{t_0,t} \leq 1 \qquad (\forall t\in T) \tag{1.26}$$

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

$$\sum_{t_1\in T^l} f_l^{\Sigma,t_1} - \sum_{t_2\in T^l} f_l^{t_2,\Theta} = 0 \qquad (\forall l\in\mathscr{L}) \tag{1.28}$$

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

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

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

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

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

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

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

With objective function (1.1), we maximize the number of trains running. Constraints (1.2) and (1.3) make sure that either both a locomotive and a driver or none of them are assigned to the train; they also take care that driver and locomotive are mutually compatible. With (1.4), we ensure that at most one driver is assigned to a train. Constraints (1.5) and (1.6) ensure that there is no more than one schedule per driver in the plan. Using constraint (1.7), we ensure that no two trains which run simultaneously are assigned to the same driver. With (1.8) and (1.9) we model that the minimal length of a short break amounting to 11 hours is not violated. Additionally, with (1.10) we ensure the integrity of the long, 35-hour break. Using (1.11), we make sure that each last job t in a shift is succeeded by a first job of the next shift, or that the job t is the last one assigned to driver d in the plan. Similarly, with (1.12) we model that each first job t in a shift was predecessed by a last job of the previous shift, or that the job t is the first one assigned to driver d in the plan. Constraints (1.13), (1.14) and (1.15) ensure the integrity of drivers' shifts and model the maximal length of a shift amounting to 12 hours. Using (1.16) we make sure that at least one long break per week is assigned to each driver in every week falling in the planning period. With the help of (1.17) we make sure that the last job of a driver in the plan is the same or a later one as the first job in the plan. Using constraints (1.18) and (1.19) we make sure that a driver works on at most three Sundays in a given planning period. Constraints (1.20), (1.21), (1.22) and (1.23) tie each "indicator" variable to the actual decision variable.

For the locomotive part of the model, (1.24) ensures that a locomotive that serves a train t arrives at its origin station and in the due time, and similarly it later departs from train t's arrival station. Using (1.25) we ensure that at most one locomotive serves each train. With the use of (1.26), we make sure that an appropriate successor and predecessor are chosen for a locomotive $l\in\mathscr{L}$ given it serves a train $t\in T^l$. We use (1.27) to warrant that at most one first train is chosen for each locomotive. Constraint (1.27) ensures that each locomotive has a unique first and a unique last train in a plan. With the help of (1.28) we ensure the integrity of the locomotive schedule. Finally, constraints (1.29) to (1.35) ensure that the decision variables are binary.

2 Input data description

Our industry partner provided us with a high-quality real-world data set for the problem. They represent the trains the industrial partner planned to serve in February 2020, as well as the information about drivers

and locomotives which was up to date on February 14, 2020. It comprises six files – we will now describe each in detail.

Order book It is a list of all the trains that need to be performed, including their origin and destination stations, as well as departure and arrival times and assignment to calculation weeks. In the supplied instance, there are four calculation weeks, starting on Saturday and lasting till next Friday. It is contained in file `trains.csv`.

List of drivers This file comprises the list of all the drivers, including their licenses to locomotive types, knowledge of routes and assignments to regions. It is included in file `drivers.csv`.

List of locomotives In this file, information about all the available locomotives is included. In particular, it comprises their class, source of energy (electric / diesel) and power. For each train powered by a locomotive which is not the property of the industrial partner, an artificial entry is made, stipulating only the required locomotive class. All that information can be found in the file `unique_locos.csv`.

Distances between stations This file includes estimated distances and travel times between all the stations present in the order book. This information is required to be able to allow drivers to move between various stations while not driving a train during their shift. These times were estimated using the API of Google Maps. They were up to date as of February 14, 2020. These distances and travel times are included in the file `distance_matrix.csv`.

Assignment of stations to regions Here, each station present in the order book is assigned to one of the driver regions. It is included in file `station_region_mapping.csv`.

Assignment of drivers to regions This is an auxiliary source of information about the assignment of drivers to planning regions. It It is included in file `driver_region_mapping.csv`.

Based on the data we received, we have developed ten instances. Table 3 below presents a summary of parameters of each instance we develop.

instance	# days	# trains	# drivers	# locomotives
1M	29	2551	217	112
3W_1	21	1854	217	112
3W_2	21	1838	217	112
2W_1	14	1239	217	112
2W_2	14	1242	217	112
2W_3	14	1228	217	112
1W_1	7	629	217	112
1W_2	7	610	217	112
1W_3	7	615	217	112
1W_4	7	613	217	112

Table 3: Overview of instance parameters and model generation times

3 Step-by-step procedure

In this section, the requirements for the scripts to run will be given. We will also provide a step-by-step manual for the usage of the benchmarks introduced.

3.1 Requirements

Our code is written in Python 3.7. Hence, the host machine needs to have a distribution of Python 3.7 or newer installed. Apart from packages available in the Python Standard Library, we also used `numpy` (for some numeric manipulations) and `networkx` (for the representation of graph objects and graph-related algorithms).

Our model was built with `gurobipy`, which is Python's API to the routines of the Gurobi solver. We have used Gurobi 9.1 in our work. Although Gurobi is a proprietary software, a free non-commercial license is available to all the members of academic community.

3.2 Usage of the benchmarks

Step 0: Installation Provided a Python interpreter and the required packages are available on your machine, you can simply clone from the ROMSOC Github repository:
`git clone https://github.com/ROMSOC/benchmarks-mip-rail-scheduling`

Step 1: Choice of instance Navigate to `instances` directory. From there, navigate to a directory whose name matches the instance you would like to consider.

Step 2: Run model construction There are two keywords which need to be supplied when running the scripts:

- {`weekly`,`monthly`} – determines the scope of the model. If you chose `1M` as your instance, use `monthly`, otherwise use `weekly`.
- {`write`,`nowrite`} – determines whether or not the resulting model will be written to an output file `model.lp`.

For example, if you wish to consider the instance `1W_1` and to generate an output file, you need to type:
`python main.py weekly write`
If you wish to consider the instance `1M` and **not** to generate an output file, you need to type:
`python main.py monthly nowrite`

4 Description of output data

If the user wishes so, the benchmarks introduced may generate an `model.lp` file which contain a text description of the integer model. It presents the objective function, variables and constraints of each model. The `.lp` files can be opened with any text editor (such as Notepad). We generally advise against storing models for larger instances, since their size may easily grow to gigabytes, which will render them useless for manual browsing.

References

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