A set packing approach for scheduling passenger train drivers: the French experience

Aurélien Froger, Olivier Guyon, Eric Pinson

A set packing approach for scheduling passenger train drivers: the French experience

A. Froger [a], O. Guyon [b,1], E. Pinson [a]
[a] LUNAM Université, Université Catholique de l'Ouest, LARIS EA7315, Angers, France
3 Place André Leroy F-49008 Angers, France
[b] SNCF, Innovation and Research Direction
40, Avenue des Terroirs de France, F-75611 Paris CEDEX 12, France
[1] E-mail: olivier.guyon@sncf.fr, Phone: +33 (0)1 57 23 62 74

Abstract
In this paper, we describe a method to solve the passenger crew scheduling problem for SNCF (the French national railway company). From rolling-stock rosters, the primary objective of the problem we address is to build shifts to maximize the number of trains that are assigned to drivers. Other objectives are mainly concerned with limiting the number of times where drivers have to rest away from their home, and with minimizing taxi trips. The problem is solved with a day-by-day approach, while guaranteeing a consistent chaining on consecutive days for shifts which include an external rest for drivers. Each day, a set of shifts is first generated according to regulation and business rules using a depth-first search algorithm. Then an iterative procedure based on a Lagrangian heuristic is used to solve the resulting set packing model. This procedure relies on a three-step algorithm: a subgradient method, a constructive heuristic and a fixation technique for selecting efficient shifts. The algorithm has been implemented in a proprietary software module: PLAISANCE. Numerical experiments have been performed on several real-life instances with up to 2,300 passenger trains to schedule. The results correspond to the business requirements and prove the effectiveness of the described method.

Keywords
Crew Scheduling, French, Railways, Set packing

Introduction

SNCF (the French national railway company) is faced regularly with building shifts for its drivers. Scheduling problems dealing with human resources are mostly strongly combinatorial and require the use of optimization techniques. Through years, many models, resolution techniques and software tools applied to varied domains have been developed to solve this kind of problems (Ernst et al., 2004). Different criteria such as the quality of service, the cost, or the robustness to accommodate delays have been used to determine the quality of a planning. Among resource planning problems of railway companies, scheduling the shifts of drivers is an important issue. Taking advantage of OR techniques, we report about a new planning tool (PLAISANCE) for passenger train drivers at SNCF. The passenger train driver planning problem is complex and therefore classically divided into a crew scheduling problem and a crew rostering problem, while respecting all working constraints. In this

paper, we restrict our attention to the crew scheduling problem which consists in finding an optimized set of shifts.

The paper is organized as follows. Section 1 describes the driver planning problem of the french national railway operator. Section 2 reviews the literature. Section 3 presents the solution of this problem. Computational results are given in Section 4. Finally, Section 5 draws some conclusions and describes future extensions.

1 Problem description

The planning process at SNCF (see Figure 1) typically starts from planning train paths and goes further on to planning rolling stock and train drivers. First, on demand estimation basis, public services and marketing department design lines with an associated frequency. Lines consist of a route with potentially stops between the departure and the arrival stations. Then, train paths are defined under economic considerations. In the next phase of the planning process, a rolling stock unit is assigned to each train. Driver scheduling comes last: legal work are built in such a way that each train has an assigned driver. Constraints control the construction of these shifts as well as their chaining on consecutive days.

This paper deals only with the last part of this process. The train driver scheduling problem at SNCF is defined as follows.

Each train is characterized by a departure time, a departure station, an arrival time and an arrival station. The rolling stock unit used to operate the train is known as well as the traveled line. Trains are passenger trains and also empty trains that are useful to reposition rolling stock units between locations. Each train has to be assigned to one unique driver from its departure station to its arrival station. Multiple units trains are considered as one single train that has to be assigned to one driver.

In some cases, drivers can take trains as passengers to move to another location so that they can drive their next train. We refer to this kind of trip as a *travel as passenger*. Travels as passenger are especially useful for bringing drivers to their depot before sign off. For some French *regions*, 30% of the current shifts involve at least one travel as passenger. In worst case, taxi trips can also be used to reach some locations, especially when train service is infrequent or to travel between two stations not easily reachable by train. We use a list of all possible trips that a driver is allowed to take to travel from one station to another one.

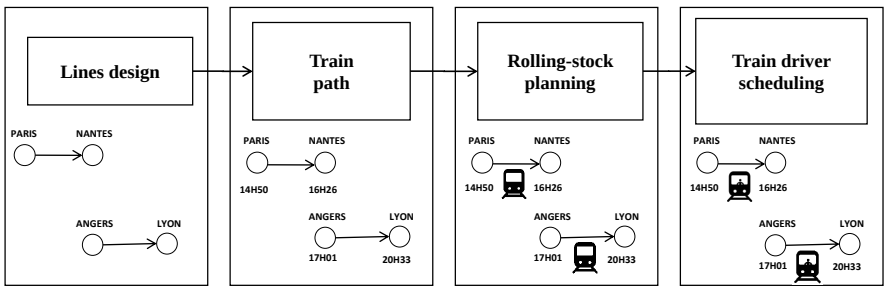

Figure 1: The planning process at SNCF

A shift is the daily work to be performed by a driver. It is composed of driving work and auxiliary work (sign-in and sign-off activities, meal and compulsory breaks, travels as passenger, taxi trips). Driving work are composed of sequence of trains. Driver home bases are called depots. A driver roster is defined as a group of drivers with the same skills in terms of rolling stock units and lines they can operate. These skills are referred as *traction knowledge* and *line knowledge*. Multiple driver rosters can be associated to one depot. Individual drivers are not explicitly considered in the planning; we focus only on driver rosters. The capacity of each driver roster is known and has to be respected.

Building daily shifts that bring back driver to their home depots is not always possible. In this case, drivers rest next to the final station of their shift. The term *external rest* is used to refer to this kind of situation. Drivers cannot rest away from their home twice in a row. The next day, they have to be assigned to shifts which bring them back to their depot. Therefore, shifts can be split into three categories:

- **Round shift** : a shift which starts and finishes at the same depot.
- **Outward shift** : a shift which starts at a depot and finishes at either a depot different from the previous one or at a station.
- **Return shift** : a shift which starts either at a depot or at a station and finishes at a depot different from the previous one.

A shift can be at the same time an outward shift and a return shift according to the driver roster which is taken into account. Outward and return shifts involve an external rest for drivers. We define a pairing as the combination of an outward shift A and a return shift B if A and B are related to consecutive days, the departure depot of A is the same as the arrival depot of B, and the arrival station of A is the same as the departure station of B. In the remainder of the paper, a *driver schedule* relates to either a round shift or a pairing. Saying that a driver schedule cover a train means that the train is part of the driving work of this schedule. Similarly, a train is said *covered* if it is part of a built driver schedule.

Shifts and pairings have to respect union agreements. The total working time of a shift is computed according to the type of work to operate. For example, the duration of a travel as passenger is not taken fully into account in the working hours as well as for some types of breaks. Therefore, the total working time may not be equal to the length of a shift. Some regulatory constraints are summarized below:

- The maximum length for a shift is 11 hours or 8 hours if a work occurs during the night period [22h,6h].
- The total working time cannot be less than 5 hours. It cannot exceed 9 hours or 8 hours if it includes more than 1h30 of work in the interval [23h,6h]
- In a shift, at most one rest break lasting more than one hour is authorized.
- A meal break is required if the total driving time exceed 8 hours. It can only be planned between 11h30 and 13h30 or between 18h30 and 20h30. Drivers must have meal in the stations equipped with a micro wave.
- In a pairing, the difference between the end of the outward shift and the beginning of the return shift cannot be less than 9 hours.

- The number of pairings assigned to a driver roster is bounded for each day.

Facing a multi-objectives optimization, a lexicographical order is used for ranking these objectives. Given rolling-stock planning, the problem consists in building regulatory shifts such as to maximize, as a priority, the number of trains covered by a driver schedule (a round shift or a pairing). The secondary objectives deal with the quality of the schedules. Having too many pairings is extremely expensive (accommodation costs, extra-work costs) and is therefore penalized, although their number are bounded. Coupling between drivers and rolling stock units are targeted to obtain more robust solutions. Taxi trips are also penalized.

2 Literature review

The train driver scheduling problem is a classic problem of the railway industry. More generally, it is known as a *Crew Scheduling Problem (CSP)*. This type of problems consists in building daily legal works for crews in such a way that each train, flight or bus trip included in the model has an assigned crew. Objectives deal mainly with cost or/and crew number minimization. The *Crew Pairing Problem (CPP)* may be seen as a special case of the first problem: shifts or sequence of shifts have to bring crew back to their home depot. Train driver scheduling at SNCF is a Crew Pairing Problem. The *Crew Rostering problem (CRP)* consist in planning the weekly or/and monthly work of each individual driver. The number of drivers required to operate on the shifts is sometimes minimized. Costly-based objectives may be still used. Crew planning management consists in both a CSP and a CRP. Our study focuses only on the CSP part.

CSP has been widely studied. The *Bus Driver Scheduling Problem* has been the first one on which researches have focused since the 1960s. The assignment of crew to planes is also quite similar to the one of railways. However, Crew Scheduling Problems in the railway sector cannot generally use the techniques developed for these transportation systems or cannot be adapted to other means of transport: network and the union agreements constraints are often too specific. CSP in the railway industry are known to be complex and NP-hard. In spite of this complexity and the size of the problems occurring in the railroad world, solution strategies have been widely proposed for improving the related solution quality.Finding optimal solutions is not prioritized, resolution times being often prohibitively high.

Generate-And-Select approach is a classical method used to solve crew scheduling problems. A part or all of the regulatory shifts are first generated. Additional rules may be incorporated to reduce the pool of generated shifts. A set covering model is next solved: shifts are selected to meet the problem requirements. Various techniques for solving SCP were proposed in the literature (Caprara et al., 2000). It may be impossible to generate thoroughly the entire pool of shifts for real-life instances. Column generation is then used to solve the problem by generating shifts on the fly. Successive iterations between generation and selection stages allow to limit the combinatorial explosion. The shift generation stage associated with the pricing problem matches with a shortest path problem with resource constraints, which can be solved using a pseudo-polynomial labeling algorithm. The following researches are mainly based on this technique.

The Dutch company NS Reizigers addresses Crew Scheduling in railways since 1990s. A software called CREW has been used initially. Planning generation was based on a search tree using A*-algorithm. An optimization software called TURNI has been developed in the early 2000s. The problem is modeled as a set covering problem (Kroon and Fischetti, 2000).

TURNI system is based on a technique introduced by Caprara et al. (1999b). It combines column generation, Lagrangian relaxation and heuristics. A strong-coupling is desired between drivers and train trips, so unnecessary train changes are penalized or bounded. A system called LUCIA software is currently used by NS Reizigers. Based on the same principles than TURNI, pricing sub-problems and parts of the master problem may take advantage of parallelization (Abbink et al., 2011). Instances with up to 15000 trips can be solved.

Caprara et al. (1999a) study the problem for Ferrovie dello Stato Italiene SpA, the public italian rail system company. A CPP is described and modeled as a SCP. The resolution technique is based on a previous work (Caprara et al., 1999b). A depth-first search builds all regulatory shifts. Heuristic choices are introduced to speed up the process. The algorithm iterates between three steps. A subgradient step is first executed to find near optimal Lagrangian multipliers. Then, an heuristic produces feasible solutions. Pairings are scored according to their cost and the Lagrangian multipliers associated with the trains operated in the pairing. A list algorithm based on these scores produces solutions. Finally, some efficient shifts with high probability to be part of the optimal solutions are fixed as elements of the solution. A new iteration begins afterwards on the resulting problem. Instances with up to 5000 trips are reported to be solved in a reasonable amount of time. Caprara et al. (2001) improve the pairing generator and develop a method to optimize jointly CSP and CRP in a single stage.

Japanese company Carmen, a subsidiary firm of Boeing, has designed a complete system for airline CRP. This system has been tested for some railway applications, in particular the German state railways Deutsche Bahn (Kohl and Karisch, 2004; Bengtsson et al., 2007). It exploits a rule modeling language. The SCP is solved with a Dantzig-Wolfe decomposition. Pairings are generated dynamically when solving the pricing sub-problem. However, this sub-problem may be time-consuming. An improved k-shortest path algorithm is therefore used along with a resource merging technique. Lower and upper bounds are respectively computed using a subgradient method and a dual-ascent heuristic coupled with a connection fixing strategy.

A crew scheduling system called TrainTRACS has been developed by the University of Leeds and has been proven to be efficient for the UK rail industry (Kwan et al., 2001). It consists in a *Generate-And-Select* approach. A large set of shifts are first generated. The set covering model is then solved with column generation associated with a branch-and-bound. Kwan and Kwan (2007) present an hybrid technique to explore efficiently the search space when finding integer solutions.

Yaghini et al. (2013) propose a matheuristic technique to solve the SCP. It consists of a LP-based neighborhood structure embedded in a tabu search. This method is applied to face the train driver scheduling problem of the Iranian railways. The shift generation is first performed with a breadth-first search strategy. In one hour, instance with approximately 3.000 trips are solved very efficiently.

Ernst et al. (2001) define a specific model to manage crew scheduling for Australian railways. All feasible shifts are first generated and the integer model is directly solved taking advantage of the sparseness of the network.

Metaheuristics have also been used. Cavique et al. (1999) design a tabu search for Lisbon Underground using a subgraph ejection chain method. Li (2005) presents a genetic algorithm associated with a fuzzy evaluation which depends on some shifts features.

Most planning systems are based on a set covering model. However, schedules produced with such a model may contain overlapping shifts: some trains trips are covered by more than one selected shift. Travels as passenger are then used to get out of these situations. The conflicted train trip is maintained in only one shift and considered as a travel as passenger for the other shifts. However, shifts must be still valid with these transformations. This works only if removing the overcovered train trips from all of the shifts except one leads to still valid shifts. This is not verified for the french national railway company and that's why we do not consider overlapping train trips as passenger travel. Additionally, the problem may be restricted if traction and route knowledge are defined. Indeed, in this situation, a driver cannot be passenger of a train he cannot be assigned as a driver. All studies dealing with a set covering modelization are therefore not relevant for us. As a result, we propose a particular set packing model for the problem described in section 1.

Freight train drivers scheduling have already been studied at SNCF (Djellab and Bionnier, 2011). The CARACO prototype is based on a column generation technique. Crew Rostering problem is then solved with a method combining a constructive heuristic with a local search. This work does not apply very well for the passengers transport. First of all, travels as passenger are not explicitly considered in the model; only estimated travel times are used to facilitate some connections in the network. Moreover, when a shift cannot be fitted with other shifts to form a pairing, an artificial shift is created. To prevent these drawbacks, we address simultaneously the shift and pairing generation phases.

3 PLAISANCE driver scheduling system

In this section, we present the PLAISANCE driver scheduling tool developed for solving the problem described in section 1.

The time horizon is one week. Because of the size of the problem (see Table 1) and to prevent lack of memory with our approach, we first decompose the weekly problem into daily sub-problems. Figure 2 describes such a decomposition. The day associated with each sub-problem is represented by a red line. It means that the sub-problem focuses only on the trains circulating during this day in the objective function. Pairing used for a sub-problem can only involve trains circulating on the days associated with the black arrow. The process starts by optimizing sub-problem P1. Then we freeze the selected shifts (round shifts or shifts associated with pairings), and we optimize the problem P2 in which all the trains of day 2, that are not retained on the solution of P1, are included as well as those of day 3. We repeat the same scheme until all days have been optimized. Of course, splitting the problem in several daily sub-problems leads to suboptimal solutions.

In order to take drivers days-off, vacations, and training courses into account, an upper bound is settled for each driver roster to limit the number of shifts that can be daily assigned to it. This upper bound is a linear function of the capacity of the roster, and helps to fix the number of pairings allowed for each driver roster. This technique is currently used by SNCF experts and feasible solutions have always been guaranteed. This upper bound is very useful for our process.

In the following sections, a two-step approach is proposed to solve each daily sub-problem described above. Let j be the day associated with the sub-problem to be solved (represented by a red line in Figure 2). First, a pool of regulatory driver schedules (round shifts and pairings) is generated. Then a set packing problem is solved to select these schedules according to the primary objective which is to maximize the number of trains

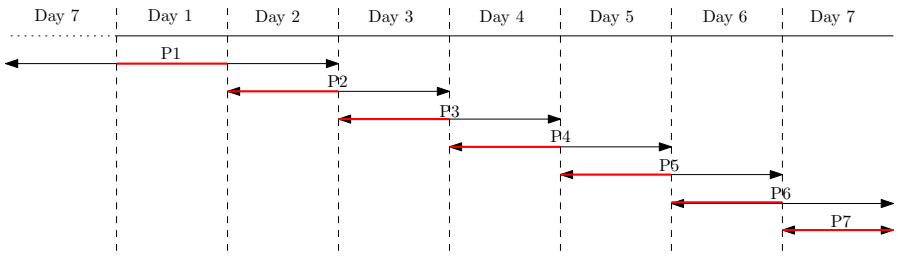

Figure 2: The separation in periods of the planning process

that are assigned to drivers. Secondary objectives are taken into account at the end of the second stage.

3.1 Generation stage : a depth first search customized with some heuristics

The generation stage is divided in two parts. At first, all regulatory shifts are generated using a depth first search. The generation is conceptually based on the time-indexed graph depicted in Figure 3. In this graph, vertices are associated with stations time-indexed on the periods related to the time horizon, while arcs correspond to driving or auxiliary works performed by drivers, and temporally consistent. These arcs are partitioned in three types : waiting, train and connection, relating respectively to waiting or rest periods at stations, to driving work or travels as passenger, and to the minimum time required between two trips. Secondly, legal pairings are built with outward and return shifts that can be paired. A pool of valid driver schedules is the result of this stage.

Reducing the pool of generated schedules
A statistical study has been performed to identify some additional operational rules allowing to limit the number of shifts and pairings generated in the first stage. For example, the number of travels as passenger and taxi trips authorized in a shift are bounded. An outward

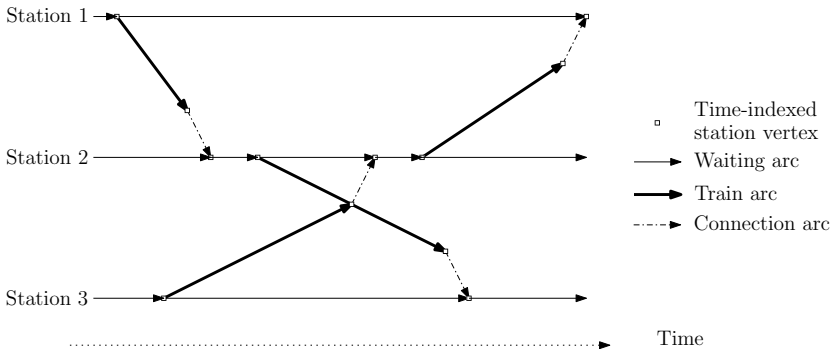

Figure 3: Activities network for shifts generation

shift (resp. a return shift) cannot finish too early (resp. too late). In a pairing, the difference between an outward shift and the beginning of a return shift cannot exceed X hours. Most of the generated shifts include an external rest. Because pairings generation is subject to a combinatorial explosion, having too much shifts including an external rest is undesirable. Additionally, a filter is implemented to limit the number of pairings. A pairing is generated only if it covers a train which is included in less than θ already generated schedules. The parameter θ has been fixed experimentally. Furthermore, taxi trips inventoried in a pre-specified list are not allowed.

Building additional schedules
Sometimes, some trains remain uncovered by the current pool of schedules (round shifts and pairings) or are to insufficiently covered according to the parameter θ (they are part of less than θ driver schedules). In this case, some heuristics are applied to generate additional relevant schedules. Operational rules are first removed. If some trains still remain uncovered, drivers are allowed to take all trains that are part of the problem as passenger and not only those in the pre-specified list. Finally, taxi trips are included in the generation.

3.2 Selection stage : an iterative Lagrangian relaxation heuristic

The selection stage of our approach can be formulate as an integer linear programming problem. Let V be the set of trains involved in the problem to solve (trains operating on the days associated with the black arrow in Figure 2) and R the set of driver rosters. The capacity of each driver roster $r \in R$ is η_r and the maximum number of pairings allowed is $\bar{\eta_r}$. Denote E the set of all driver schedules (round shifts and pairings) generated at the first stage. Let V_e be the pool of all trains covered by a driver schedule $e \in E$ and p_e be equal to 1 if schedule e is a pairing. We introduce similarly E_v the set of all schedules covering train $v \in V$, and E_r as the set of all schedules associated with the driver roster $r \in R$. By construction, E satisfies that for each $r_1, r_2 \in R$, $E_{r_1} \cap E_{r_2} = \emptyset$. Furthermore, the binary decision variables x_e indicate whether a driver schedule is selected or not. c_e is the number of trains operating during day j covered by schedule $e \in E$. Our problem is modelled as a set packing with some additional constraints :

$$[TPP] \max \sum_{e \in E} c_e x_e \tag{1}$$

$$\sum_{e \in E_v} x_e \leq 1 \ , \forall v \in V \tag{2}$$

$$\sum_{e \in E_r} x_e \leq \eta_r \ , \forall r \in R \tag{3}$$

$$\sum_{e \in E_r} p_e x_e \leq \bar{\eta_r} \ , \forall r \in R \tag{4}$$

$$x_e \in \{0, 1\} \ , \forall e \in E \tag{5}$$

The objective function (1) represents the number of trains operating during day j covered by the selected driver schedules. Set packing constraints (2) prevent trains to be part of more than one selected schedule. Constraints (3) and (4) limit respectively the number of schedules and the number of pairings that can be assigned to each driver roster. $[TPP]$ is solved using an iterative technique adapted from the heuristic method presented by Caprara

et al. (1999b) and based on Lagrangian relaxation. The main part of the algorithm, that we will note *ILH*, is summarized in procedure 3-PHASE. Notice that we use globally the same notations as in the original article.

Procedure 3-PHASE

Let Ω be the problem to solve
Let x^* be the best solution found by the algorithm
Denote ϵ the desired precision
$UB \leftarrow +\infty$, $LB \leftarrow -\infty$
repeat
  Apply subgradient phase to Ω
  Update UB if a better upper bound is found
  Compute near-optimal Lagrangian multiplier vector u^*
  Apply heuristic phase to Ω
  From u^*, build feasible solutions using procedure greedy(E,u^*)
  Update x^* and LB if a better solution is found
  Apply fixing phase to Ω
  Fix x_e variables to 1 for some relevant schedules
  $\Omega \leftarrow$ the resulting sub-problem.
until $\Omega = \emptyset$ *or* $UB - LB \leq \epsilon$;

Subgradient phase: getting upper bounds

The first phase of the algorithm is based on a Lagrangian relaxation of our model (1)-(5). Constraints (2) are dualized. For every vector $u \in \mathbb{R}_+^{|V|}$ of Lagrangian multipliers associated with the constraints (2), the Lagrangian sub-problem noted $[LP_u]$ reads :

$$L(u) = \max \sum_{e \in E} c_e(u) x_e + \sum_{v \in V} u_v \tag{6}$$

$$\sum_{e \in E_r} x_e \leq \eta_r \ \ , \forall r \in R \tag{7}$$

$$\sum_{e \in E_r} p_e x_e \leq \bar{\eta}_r \ \ , \forall r \in R \tag{8}$$

$$x_e \in \{0,1\} \ \ , \forall e \in E \tag{9}$$

where $c_e(u) = c_e - \sum_{v \in V_e} u_v$ denotes the Lagrangian cost of driver schedule $e \in E$. Clearly, $[LP_u]$ can be split into $|R|$ independent sub-problems denoted $[LP_u]_r$. We parallelize therefore the resolution of the Lagrangian problem in our experiments. $[LP_u]_r$ is a particular 0-1 knapsack problem with two constraints. Indeed, constraint (7) limits only the number of schedules that can be part of the solution and constraint (8) bounds the number of pairings allowed. Finding an optimal solution to $[LP_u]_r$ is therefore quite easy. Let E_r^+ be the set of all schedules with a strictly positive Lagrangian cost. We set $x_e = 1$ for the driver schedules in E_r^+ with the highest values $c_e(u)$ as long as the constraints (7) and (8) are satisfied. For all other driver schedules $e \in E_r$, we fixed $x_e = 0$.

This phase gives a valid upper bound for the problem. It produces also a near-optimal vector of Lagrangian multipliers whose information is useful to discriminate relevant driver

schedules. We solve the Lagrangian relaxation of our model by a standard subgradient method. We introduce the subgradient vector $s(u) = 1 - \sum_{v \in V} x_e(u)$ where $x_e(u)$ denote the optimal solution of $[LP_u]$. The subgradient method produces a sequence $u^0, u^1, ..$ of vectors of Lagrangian multipliers computed by the following formula :

$$u_v^{k+1} = max \left\{ u_v^k - \lambda^k \frac{L(u^k) - LB}{\|s(u^k)\|^2} s(u^k), 0 \right\} \tag{10}$$

where $\lambda \in]0; 1[$ is a given parameter and LB is a lower bound of the optimal value of $[TPP]$. At the first iteration, LB is computed with a basic listing algorithm. LB is then equal to the best solution founded so far.

Notice that if the generation stage produces too much schedules, schedules with the lowest Lagrangian costs $c_e(u)$ at the end of the subgradient algorithm are removed. However, it does not happen in our experiments.

Heuristic phase : building feasible solutions

From the near-optimal vector u^* of Lagrangian multipliers found in previous phase, a pool of vectors U is generated applying subgradient method with a specific Lagrangian multipliers update. Afterwards, for every vector $u \in U$, a score $SCORE(e, u)$ associated with each driver schedule $e \in E$ is computed.

$$SCORE(e, u) = \begin{cases} \dfrac{c_e(u)}{|V_e|} & \text{si } c_e(u) > 0 \\ c_e(u) \cdot |V_e| & \text{si } c_e(u) < 0 \end{cases}$$

For each vector $u \in U$, an heuristic solution of $[TPP]$ is computed using procedure greedy(E,u).

Procedure greedy(E,u)

```
Let E* := E be the set of all generated driver schedules
S ← ∅ (S is the set of the selected driver schedules)
Compute SCORE(e, u) , ∀e ∈ E*
repeat
    Let e* ∈ E* be the schedule with maximum score
    S ← S ∪ {e*}
    foreach v ∈ V_e* do
        E* ← E* \ E*_v
    r* ← driver roster associated with e*
    Check constraint (3)
    if η_r* schedules have been selected in S then
        E* ← E* \ E*_r*
    else
        Check constraint (4)
        if η̄_r* pairings have been selected in S then
            Delete all the pairings of driver roster r* in E*
until E* = ∅;
```

Fixing phase : reducing problem size
Let u^* be the distribution of Lagrangian multipliers leading to the best solution at previous phase. We define Q as the set of driver schedules $e \in E$ such as $c_e(u^*) > 0.01$. We fixed all schedules $e \in Q$ if $|E_v \cap Q| = 1$ for at least one train in $v \in V_e$. The first driver schedule selected by procedure greedy(E,u^*) is also fixed. In the next iteration of the 3-PHASE algorithm, only the resulting problem is considered. The capacities η_r and $\bar{\eta}_r$ of each driver roster $r \in R$ are iteratively updated all along the process.

Optimizing the algorithm
As suggested in Caprara et al. (1999b), we use some tricks to optimize the algorithm. The 3-PHASE procedure is applied only on a subset $\hat{E}$ of the entire set E to speed up the process. $\hat{E}$ is computed according to the Lagrangian costs and the cover of each train. We update $\hat{E}$ during specific subgradient iterations. Moreover, at the end of the procedure, some "good" driver schedules (the first ones selected by the greedy heuristic) are fixed in a similar way than in the fixing phase. The 3-PHASE procedure is performed once again to the resulting sub-problem until the difference between the best upper bound and the best solution is less than the precision value ϵ.

Dealing with the secondary objectives
While the ILH method processes, we store a pool of solutions leading to the same value of the primary objective function (1). The number of trains covered in each of these solutions is therefore equal. At the end of the selection stage, the solutions included in the pool are ranked according to the secondary objectives (given in order of priority) : minimizing the number of external rests, taxi trips and the number of different rolling stock units operated. The best solution according to these objectives is retrieved.

4 Computational results

In this section, we present results based on the experiments that we carried out. All numerical experiments have been performed on a PC, equipped with an Intel Core i7-2670QM processor, running Windows 7 Professional and working with 4 Go of memory. The iterative algorithm have been implemented in Java 1.7. The performance of the proposed method was evaluated on two instances (cf. Table 1).

Table 1: Instances

Instance	Trains	Depots	Driving rosters
REG_1	1,872	5	7
REG_2	2,299	7	12

Results of the generation stage are summarized in Table 2. This stage requires respectively 568 and 3812 CPU seconds for REG_1 and REG_2. Approximately, 94% of the driver schedules are pairings, which is not surprising since pairing generation is extremely combinatorial. If they were not filtered during the generation process, we would have generated near 60% more pairings, which proves the relevance of this filter. It is quite complex to set up a technique allowing to reduce this number while working as expected for all instances. Building additional shifts adds respectively 15% and 3% more schedules to the pool used in the selection stage. This building block may be, however, time-consuming, representing on

Table 2: Global results of the generation stages

Instances	Round shifts			Pairings		
	Max/day	Avg/day	Week	Max/day	Avg/day	Week
REG_1	69,710	39,120	273,838	1,274,526	684,738	4,793,163
REG_2	141,500	92,344	646,410	2,610,505	1,823,700	12,765,905

average a quarter of the overall running time of the generation stage, reaching up to almost half of this time for some cases. Results prove to be worse if we skip this step.

To assess the quality of the *ILH* method described in section 3 for the selection stage, we solve directly the MIP model $[TPP]$ with CPLEX MIP Solver 12.6. We set the same time limit for both methods in order to do an effective comparison. For the solver, the model is always initialized with the driver schedules built during the generation stage and it takes into account the decisions of the previous days made by the ILH technique. We aim also to compute the gap to the optimal solution. Let $\underline{Opt}^j$ be the optimal solution found by CPLEX MIP solver (with no time limit) for the selection stage of day j. $\underline{Opt}^j$ is not necessarily the optimal solution of the problem associated with day j for two main reasons. First, not all valid driver schedules are generated. Secondly, the problem being split up into several sub-problems, decisions made for day j impact on the solutions found for the following days when pairings are involved.

Tables 3 and 4 show respectively the results of the selection stage for REG_1 and REG_2 according to the primary objective. Because of the day-by-day approach, resolutions times decrease when it comes to the end of the week. On average, the ILH method is respectively almost three and two times faster than CPLEX MIP Solver running to compute the optimal solution. For 4 out of 7 days, this technique proves to be as good as or better than the mathematical programming solver when it is set up with the same time limit.

Table 5 summarizes the previous results of the selection stage for the whole week. A negative gap indicates that the ILH technique find better results than the other technique. First of all, we are not able to cover all the trains for the two instances. 76,7% and 81% of the trains are respectively covered in the final solution for REG_1 and REG_2. The quality of the results and especially the percentage of uncovered trains are caused by mainly two factors. First, shifts cannot easily be fitted together without overlapping with the pre-specified lists of travels as passenger and taxi trips allowed. Especially, too few taxi trips are specified but we aim at minimizing these costly trips. Secondly, in the selection stage for each daily sub-problem, driver schedules are successively fixed during the iterative process of the ILH method. This fixing phase performs well for the set covering model but might lead sometimes to a significant decrease of the solution quality for our model. Moreover, solving the weekly problem with a day-by-day approach impacts strongly on the number of uncovered trains. Indeed, when solving the problem at day j, selecting a pairing made up of an outward shift on day j and a return shift the next day affects the decisions that can be made at day $j + 1$. This can only lead to more uncovered trains. To illustrate this point, we identify that respectively 30% and 53% of these uncovered trains do not share in any driver schedule after the generation stages (one for each sub-problem) for the two instances studied in this paper. However, the average percentage gap for each day between the best computed solutions and the optimal solution is respectively 2,9% and 4,4%. These results are very encouraging especially when compared with those found by CPLEX with the same resolution time limit. Notice that all these values must be taken with precaution because of

the day-by-day approach.

Table 3: Results of the selection stages for REG_1 according to the primary objective

Day j	**Number of trains**	**ILH**		**CPLEX Solver**		$\mathbf{Opt^j}$	
		C.T.[1]	Time[2]	C.T.[1]	Time[2]	C.T.[1]	Time[2]
1	319	247	228	0	228	258*	1,915
2	316	238	225	198	225	246*	785
3	319	239	209	185	209	244*	752
4	318	235	193	0	193	242*	557
5	325	236	107	238	107	247*	192
6	154	134	104	136*	3	136*	3
7	121	106	26	107*	1	107*	1
Week	1,872	1,435	1,092	864	1,092	1,480	4,205

[1] C.T. : number of trains covered / [2] : time in CPU seconds / * optimal solution

Table 4: Results of the selection stages for REG_2 according to the primary objective

Day j	**Number of trains**	**ILH**		**CPLEX Solver**		$\mathbf{Opt^j}$	
		C.T.[1]	Time[2]	C.T.[1]	Time[2]	C.T.[1]	Time[2]
1	182	163	701	125	701	173*	1,015
2	383	307	793	322	793	328*	2,777
3	380	299	639	270	639	315*	719
4	385	315	763	254	763	327*	995
5	379	308	639	317	639	321*	815
6	388	324	445	331	445	341*	812
7	192	146*	47	146*	1	146*	1
Week	2,299	1,862	4,027	1765	3,981	1,951	7,134

[1] C.T. : number of trains covered / [2] : time in CPU seconds / * optimal solution

Table 5: Global results of the selection stages according to the primary objective

Instances	**ILH**			
	C.T.	E.T.[1]	Gap[2] CPLEX	Gap[2] to optimal
REG_1	76,7%	6,8%	-41%	2,9%
REG_2	81%	10%	-5,2%	4,4%

[1] E.T. : number of trains non covered after the generation stage / [2] average gap for each day

At the end of the ILH method, the secondary objectives are taken into account to retrieve the best solution. Results concerning this stage are described in Table 6. Since the pool of solutions is not very large, there is not a lot of diversity between the solutions but they correspond to the business requirements. Lastly, operational rules defined in section 3 prove to be relevant; respectively 85% and 86% of the selected driver schedules for REG_1 and REG_2 respect these rules.

Table 6: Global results according to the secondary objectives

Instances	**Average number of solutions in the pool**	**Solution attributes**		
		External rest	Taxi	Rolling stock units
REG_1	57	30,7%	14,3%	2,2
REG_2	31	37,1%	8,7%	2

5 Conclusion

In this paper, we present a new driver scheduling tool based on a day-by-day approach. Each resulting sub-problem is solved with a two-step technique. The first stage of this technique builds efficiently valid shifts and pairings. Filters and operational rules are defined to limit the driver schedule pool size. In the second stage, a particular set packing problem is solved by an iterative Lagrangian heuristic in a reasonable amount of time. A comparison made with the direct use of a mathematical programming solver proves the effectiveness of the described method. However, the method can be still improved to have less uncovered trains. Indeed, the current splitting process affects the quality of the solutions. More expertise is also required to fix relevant travels as passenger and taxi trips in order to find useful connections. Futures works include additional experiments on real-life instances. We intend also to test a global or a less split approach to find better results.

References

Abbink, E., Albino, L., Dollevoet, T., Huisman, D., Roussado, J., and Saldanha, R. (2011). Solving large scale crew scheduling problems in practice. *Public Transport*, 3(2):149–164.

Bengtsson, L., Galia, R., Gustafsson, T., Hjorring, C., and Kohl, N. (2007). Railway crew pairing optimization. In Geraets, F., Kroon, L., Schoebel, A., Wagner, D., and Zaroliagis, C., editors, *Algorithmic Methods for Railway Optimization*, volume 4359 of *Lecture Notes in Computer Science*, pages 126–144. Springer Berlin Heidelberg.

Caprara, A., Fischetti, M., Guida, P., Toth, P., and Vigo, D. (1999a). Solution of large-scale railway crew planning problems: the italian experience. In Wilson, N., editor, *Computer-Aided Transit Scheduling*, volume 471 of *Lecture Notes in Economics and Mathematical Systems*, pages 1–18. Springer Berlin Heidelberg.

Caprara, A., Fischetti, M., and Toth, P. (1999b). A heuristic method for the set covering problem. *Operations research*, 47(5):730–743.

Caprara, A., Monaci, M., and Toth, P. (2001). A global method for crew planning in railway applications. In Vo, S. and Daduna, J., editors, *Computer-Aided Scheduling of Public Transport*, volume 505 of *Lecture Notes in Economics and Mathematical Systems*, pages 17–36. Springer Berlin Heidelberg.

Caprara, A., Toth, P., and Fischetti, M. (2000). Algorithms for the set covering problem. *Annals of Operations Research*, 98:353–371.

Cavique, L., Rego, C., and Themido, I. (1999). Subgraph Ejection Chains and Tabu Search for the Crew Scheduling Problem. *The Journal of the Operational Research Society*, 50(6):608.

Djellab, H. and Bionnier, J. (2011). CARACO: A solution for crew planning problems, the French freight experience. In *WCRR, 9th World Congress on Railway Research*.

Ernst, A., Jiang, H., and Krishnamoorthy, M. (2001). An integrated optimization model for train crew management. *Annals of Operations Research*, 108:211–224.

Ernst, A., Jiang, H., and Krishnamoorthy, M. (2004). An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127(1-4):21–144.

Kohl, N. and Karisch, S. (2004). Airline crew rostering: Problem types, modeling, and optimization. *Annals of Operations Research*, 127(1-4):223–257.

Kroon, L. and Fischetti, M. (2000). Scheduling train drivers and guards: The Dutch" Noord-Oost" case. In *33rd Annual Hawaii International Conference on System Sciences*, pages 1–10.

Kwan, R. and Kwan, A. (2007). Effective search space control for large and/or complex driver scheduling problems. *Annals of Operations Research*, 155(1):417–435.

Kwan, R., Kwan, A., and Wren, A. (2001). Evolutionary driver scheduling with relief chains. *Evolutionary Computation*, 9(4):445–460.

Li, J. (2005). A Self-Adjusting Algorithm for Driver Scheduling. *Journal of Heuristics*, 11(4):351–367.

Yaghini, M., Karimi, M., and Rahbar, M. (2013). A set covering approach for multi-depot train driver scheduling. *Journal of Combinatorial Optimization*, pages 1–19.