A railroad maintenance problem solved with a cut and column generation matheuristic

Sébastien Lannez, Christian Artigues, Jean Damay, Michel Gendreau

A railroad maintenance problem solved with a cut and column generation matheuristic

Sébastien Lannez[1,2,3], Christian Artigues[2,3],
Jean Damay[4], Michel Gendreau[4]

[1]SNCF I&R/SRO ; 45 rue de Londres, 75008 Paris, France,
[2]CNRS; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse, France
[3]Université de Toulouse ; UPS, INSA, INP, ISAE ; UT1, UTM, LAAS ;
F-31077 Toulouse, France
[4]CIRRELT, Université de Montréal, C.P. 6128, Montréal (Québec), H3C 3J7 Canada

Abstract

In this paper we address a real life optimization problem, the Rail Track Inspection Scheduling Problem (RTISP). This problem consists of scheduling railway network inspection tasks. The objective is to minimize the total deadhead distance while performing all inspection tasks. Different 0-1 integer formulations for the problem are presented. A heuristic based on both Benders and Dantzig-Wolfe decompositions is proposed to solve this rich arc routing problem. Its performance is analyzed on a real life dataset provided by the French national railway company (SNCF). The proposed algorithm is compared to a dynamic programming-based heuristic. Its ability to schedule the inspection tasks of one year on a sparse graph with thousand nodes and arcs is assessed.

Keywords: arc routing, column generation, cutting planes, matheuristic, railtrack maintenance

1 Introduction

One of the major problems that railway companies have faced since the very beginning are failures in tracks. Defects in rails, the basic parts of a track, may result in serious accidents. Réseau Ferré de France (RFF), the company that owns and manages the French railway infrastructure, has delegated a part of the railway maintenance responsibility to the Société Nationale des Chemins de Fers (SNCF), a French railway company. SNCF is committed to ensure the safety of the railway network. One of the maintenance activities is to prevent track failures. In order to quickly inspect the French network, SNCF is using ultrasonic defectoscopy to detect and survey imperfections in rails.

Inspection frequencies range from six months to twenty years depending on the train speed and the cumulated weight of the considered track type. Two third of the total inspections (35 000 km) are performed on tracks which should be visited once or twice a year. These tracks are called primary tracks. All the remaining inspections (secondary tracks) have much lower inspection frequencies. The subject of the present study is the primary track inspection scheduling

problem. This problem is currently solved in a centralized way by the SNCF national logistics department while inspection of secondary tracks, which is less critical, is decomposed geographically into several sub-problems, each being solved by a local logistics department.

Ultrasonic inspections are performed with specialized rolling stock units, which will be called vehicles in the remaining of the text. Vehicles differ from one another in maximum speed and working capacity. The detection of defects inside the track is performed by a reverberation analysis of the ultrasonic waves passing through the rails. The inspection of primary tracks is subject to several limitations due to vehicle usage. First, a vehicle's speed depends on whether it is inspecting or not. A deadhead trip is a trip during which the vehicle is moving without inspecting the track. During inspection, speed can be more than three times slower than during a deadhead trip. The second limitation is due to the team working inside the vehicle (driver and technicians). There is a maximum shift duration and a vehicle can move during at most six hours per day. As a third limitation, there is a maximum daily inspection distance. Water is needed to keep sensors and rails coupled during the measurements. The inspection distance is limited by the water tank capacity. These tanks can only be refilled at special stations. Over the 200 stations on the primary tracks, only 90 are equipped with water supply. Furthermore, water tank refill is time consuming and may need available operators at the station. Hence, it is not desired to do more than one refill per shift. That is why the problem is structured in such a way that each vehicle starts and ends each shift at a refill node. A fourth limitation is that, for organizational purposes, vehicle maintenances should be performed periodically at specific stations. Last, as the same vehicles are used for secondary track inspection, each vehicle must be made available to local logistics departments and, to that purpose, has to be sent to other specific station during predefined periods. The inspection activity is also limited by the existence of track outages, that can alter the vehicle speed or even prevent it from circulating during specific periods.

The problem SNCF is dealing with is to visit the set of primary tracks taking into account the above-presented operational constraints and satisfying inspection frequencies.

The main cost indicator is a common logistic performance ratio for each vehicle which is equal to the amount (in distance) of tracks that have been inspected, divided by the total traveled distance (on a yearly basis).

This problem can be modeled as an arc routing problem related to the ones describing road deicing, waste collection or network weeding as described in the surveys from [18–21]. It involves complicating constraints, among which limited shift duration, water supply, track outages and heterogeneous fleet, as described above. Another difficulty is the network size which makes it a real challenge to solve.

The remaining of this article is structured as follows. In Section 2, we briefly present the relevant literature about arc routing problems. In Section 3, we present the notation and describe mathematical formulations and decompositions of the problem. Section 4 is devoted to the presentation of a heuristic, which, being based on the considered mathematical decompositions, can be called a matheuristic [17]. In Section 5, an alternative dynamic programming-based heuristic is presented. In Section 6, we introduce the different considered real datasets and we provide computational results that demonstrate the ability of

the proposed matheuristic to schedule the inspection tasks of one year on a sparse graph with some thousand nodes and arcs. In terms of performance, the matheuristic compares favorably with the heuristic based on dynamic programming, at the expense of additional CPU time. To further analyze the heuristic's behavior, a comparison between the obtained performance ratio and a linear programming relaxation-based upper bound is carried out. The impact of the heuristic components on the performance is also evaluated. Section 7 concludes the paper with considerations on the practical implementation of the method and perspectives for further research.

2 Literature review

2.1 Industrial arc routing problems

In [12], the authors state that industrial vehicle routing problems can be rich in the sense that "*they include aspects of the VRP that are essential to the routing of vehicles in real-life*".

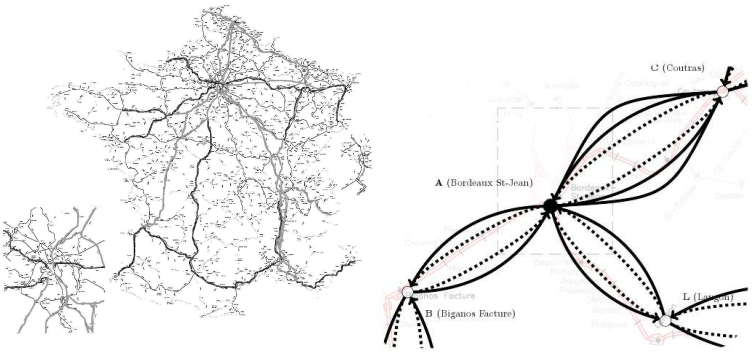
Road related problems have supplied researchers with a lot of (rich) arc routing problems. A review of problems arising during winter road maintenances has been published in the articles [18–22]. They also present industrial applications. Waste collection or postal deliveries are also an active field of arc routing problems. In [15], a description of a waste collection problem is presented. A nation wide postal delivery problem has been modeled as an industrial arc routing problem in [13].

2.2 Arc routing problems

In this section, some arc routing problems and their applications are listed. For a more detailed description of them, books [2] and [5] and survey articles [6, 7] are a good introduction.

The RTISP is related with the capacitated arc routing problem (CARP), described in [11]. It consists in visiting a set of arcs with a single vehicle. Each visited arc reduces the remaining working capacity of the vehicle by a given amount. In the RTISP, tasks and deadhead circulations can be modeled with arcs. The working capacity of vehicles is constrained by the vehicle's water tank capacity and the duration of a shift. The capacitated arc routing problem with time windows (CARP-TW) extends the CARP by constraining the possible visits of arcs to a set of periods. Paper [14] contains a description of a column generation procedure. In [23], this is solved problem with a greedy randomised adaptive search procedure (GRASP) associated with path relinking. Another extension is the capacitated arc routing problem with intermediate facilities (CARP-IF) presented in [10] also called the the capacitated arc routing problem with refill points (CARP-RP) presented in [1]. It extends the CARP by adding refill facilities to specific nodes.

We have not been aware of published work about methods for solving a problem having all these features. However, as the capacity of each vehicle is two-dimensional, the RTISP can be defined as a multi-capacitated arc routing problem with time windows, refill points and a heterogeneous fleet.



(a) Map of primary tracks

(b) Graph model of railtracks around Bordeaux

Figure 1: French railway network model

3 Problem definition, mathematical formulations and decompositions

In this section we start with providing preliminary definition and notations. Then, we propose three 0-1 formulations for the problem. The first one is an arc-flow model. The second one is obtained through a Dantzig-Wolfe decomposition of the first model and can be defined as a shift-flow model. The last one is obtained by applying a combinatorial Benders decomposition to the shift-flow model. The latter formulation yields an exact solving scheme consisting in solving iteratively the Benders master problem and the Benders sub-problems that also act as repairing methods for the master solution.

3.1 Definitions and notations

Modeling hypothesis Vehicle moves are modeled by arcs. They represent either inspection tasks, deadhead traversals of track portions, or complex moves like unit switch back or station traversal. Furthermore, arcs can represent deadhead trips from a node in the network to another node (for example to model a trip to a maintenance station which is not part of the inspected network). Arcs are suitable for the description of unidirectional railway tracks. As only primary tracks are directly modeled, bidirectional railway tracks are not considered in the study. Nodes describe stations, communication between railway tracks, or locations in the network where the vehicles can change their circulation mode. A map representing these tracks is presented in Figure 1a. A schematic zoom around Bordeaux is shown in Figure 1b. The presented graph represents the inspected railroad network. The arcs in dotted style represent inspection tasks to perform, and the other ones represent deadhead moves. The black node represents a refill station and the other nodes are normal stations without water facility.

In order to make the schedule being easily adapted during operations, multiple shifts per day are not allowed. As already mentioned, each shift consists of a trip between two refill stations with a total distance to inspect smaller than the maximal distance induced capacity of the water tank and a total trip duration

smaller than the duration of a work shift. Given all the feasible shift paths, the RTISP becomes the problem of selecting and scheduling them in order to satisfy all inspections at the lowest cost.

Graph and vehicle representation A multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ containing arcs ($\mathcal{A}$) and nodes ($\mathcal{V}$) models the railway network. Arcs can represent tasks ($\mathcal{A}^*$), deadhead traversals ($\mathcal{A}^d$) or waiting times ($\mathcal{A}^W$). Note that a task arc represent either a track inspection task or a vehicle maintenance task or a vehicle requirement for secondary track inspection. An arc from node i to node j is denoted $(i,j)^*$ if it represents an inspection task and $(i,j)^d$ if it represents a deadhead trip. A waiting arc is a loop (i,i). Nodes can represent rest and refill stations ($\mathcal{V}^r$), or communications between railroad tracks ($\mathcal{V}^c$). For each vehicle, we assume that there exists a depot refill node $D^k \in \mathcal{V}^r$ from which the vehicle must start its trip and to which it must return.

The parameter d_{ak} contains the time needed for vehicle k to traverse arc a. In practice, the average vehicle speed depends of multiple factors, such as the importance of the traffic in the considered track. In this study, the speed used for each vehicle is taken to be lower than the mean value computed during year 2009. This choice has been made in order to be able to use the schedule in practice. A waiting arc $a \in \mathcal{A}^W$ has a unit time duration $d_{ak} = 1$. The parameter l_{ak} is the distance involved by the traversal of arc a by vehicle k. This parameter is actually vehicle-dependent for inspection arcs as the "distance" represents the water consumption, which may vary from one vehicle to the other. However, for deadhead arcs, this parameter can be considered as vehicle-independent as it represents the actual traveled distance. For each vehicle k, L_k represents the maximum cumulated distance which can be inspected per shift and T_k denotes the maximum working time during a shift. This time is different from the shift duration as waiting times are authorized and not included in the working time.

Calendar The calendar $\mathcal{H}$ is assumed to contain only working days. It is composed of integer values representing the number of "shift seconds" elapsed since the first period of the planning horizon. The need for small time slots comes from the wide range of task durations and the relatively high speed of vehicles (ranging from 20 km/h to 120 km/h). t is a time slot in $\mathcal{H}$, s is the duration of a shift. We assume that $|\mathcal{H}|$ is a multiple of s and define $\mathcal{S} = \{0, \ldots, |\mathcal{H}|/s - 1\}$ as the shift index set.

For example, consider a shift duration s =4 and an horizon $\mathcal{H} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$. The shift index set is $\mathcal{S} = \{0, 1, 2\}$ corresponding to periods $\{1, 2, 3, 4\}$, $\{5, 6, 7, 8\}$ and $\{9, 10, 11, 12\}$, respectively.

Track inspection is also time-constrained. Due to the small frequencies mentioned in Section 1, the planning period (less than one year) contains at most two inspections of the same track. Hence, periodicity is handled by assigning time windows to inspection tasks which sets a minimum time lag between two consecutive inspections (and also allows to consider the current state of the network). As mentioned in Section 1, maintenance tasks and local vehicle requirement tasks have also time windows. The subset $\overline{\mathcal{H}}_{a,k} \subseteq \mathcal{H}$ contains the set of periods during which vehicle k can not traverse arc a. A period $t \in \overline{\mathcal{H}}_{a,k}$ reflects one of the following cases.

- The sub-network containing a is not accessible to vehicle k at period t due to a technical constraint or to a space-time vehicle preassignment.
- There is an outage on the track portion corresponding to arc a at period t.
- Arc a corresponds to a task and period t is outside its time window.

Costs The objective being to minimize the total deadhead, the cost of an arc is the length of the arc if this arc is a deadhead one while no cost is considered when traversing a task arc.

$$c_{ak} = \begin{cases} l_{ak}, \text{if } a \in \mathcal{A}^d, \\ 0, \text{else.} \end{cases} \tag{1}$$

3.2 Arc flow model

Using time discretization through calendar $\mathcal{H}$, the problem can be represented as a multi-commodity flow problem on a space-time network with additional constraints. As mentioned above, the path of a each vehicle can be decomposed in shifts. A shift starts at a refill node $v \in \mathcal{V}^r$ at the beginning of period 1 of the shift and ends at a refill node $v' \in \mathcal{V}^r$ at the end of period s of the shift (with possibly $v = v'$).

For each shift index σ and each vehicle $k \in \mathcal{K}$, we define a super-source o^k_σ with a set of additional arcs $\delta^+(o^k_\sigma) = \{(o^k_\sigma, v) | v \in \mathcal{V}^r\}$ and a super-sink d^k_σ with arcs $\delta^-(d^k_\sigma) = \{(v, d^k_\sigma) | v \in \mathcal{V}^r\}$. All these arcs have by convenience a duration equal to 1.

For each node $v \in \mathcal{V}$, let $\delta^{\sigma,k-}_\tau(v)$ denote the subset of arcs in $\mathcal{A} \cup \{(o^k_\sigma, v)\}$ ending at v that can be traversed by vehicle k during period $\sigma s+\tau$. Symmetrically, let $\delta^{\sigma,k+}_\tau(v)$ denote the subset of arcs in $\mathcal{A} \cup (v, d^k_\sigma)$ starting at v that can be traversed by vehicle k during period $\sigma s + \tau$. These subsets reflect the constraints that arcs cannot be traversed during an unavailability period. For example, if, for a period τ and a vehicle k, there is an arc a and an unavailability period $t \in \overline{\mathcal{H}}_{a,k}$ in the interval $[\sigma s + \tau - d_{ak} + 1, \sigma s + \tau]$, arc a is excluded from $\delta^{\sigma,k-}_\tau(v)$. The same principle is applied to model the fact that the super-source and the super-sink can only be connected to refill nodes at the beginning and at the end of each shift, respectively.

Let $x^{k\sigma}_{a\tau}$ denote the binary flow variable which equals 1 if vehicle $k \in \mathcal{K}$ starts traversing arc $a \in \mathcal{A}$ at the beginning of period $t = \sigma s + \tau$ and ends traversing it at the end of period $t + d_{ak} - 1$, where $\sigma \in \mathcal{S}$ is the shift index and $\tau \in \{1, \ldots, s\}$ is the shift period index. To represent the start and the end of each shift, we also introduce variables $x^{k\sigma}_{a0}$ for $a \in \delta^+(o^k_\sigma)$ and variables $x^{k\sigma}_{a,s+1}$ or $a \in \delta^-(d^k_\sigma)$.

A shift is defined as a feasible time-stamped $o^k_\sigma \to d^k_\sigma$ path for a given vehicle k and a given shift index σ, visiting a certain number of required arcs. The set of feasible shifts for vehicle k and shift index σ is defined by the following integer points.

$$Q^{k\sigma} = \left\{ x^{k\sigma} \in \{0,1\}^{|\mathcal{A}| \times s} \,\middle|\, x^{k\sigma} \text{satisfies (2)} - \text{(8)} \right\}, k \in \mathcal{K}, \sigma \in \mathcal{S}$$

where constraints (2)-(8) are detailed hereafter.

$$\sum_{a\in\delta^+(o^k_\sigma)} x^{k\sigma}_{a0} = 1 \tag{2}$$

$$\sum_{a\in\delta^-(d^k_\sigma)} x^{k\sigma}_{a,s+1} = 1 \tag{3}$$

$$\sum_{a\in\delta^{\sigma,k-}_{\tau-1}(v)} x^{k\sigma}_{a,\tau-d_{ak}} - \sum_{a\in\delta^{\sigma,k+}_{\tau}(v)} x^{k\sigma}_{a,\tau} = 0 \qquad v\in\mathcal{V}, \tau\in\{1,\dots,s+1\} \tag{4}$$

$$\sum_{\tau=1}^{s} x^{k\sigma}_{a\tau} \le 1 \qquad a\in\mathcal{A}^* \tag{5}$$

$$\sum_{a\in\mathcal{A}^*}\sum_{\tau=1}^{s} l_{ak} x^{k\sigma}_{a\tau} \le L_k \tag{6}$$

$$\sum_{a\in\mathcal{A}}\sum_{\tau=1}^{s} d_{ak} x^{k\sigma}_{a\tau} \le T_k \tag{7}$$

$$x^{k\sigma}_{a,\tau} \in \{0,1\} \qquad a\in\mathcal{A}, \tau\in\{1,\dots,s\} \tag{8}$$

Constraints (2) and (3) enforce each shift to start and end at a refill node (as only refill nodes are connected to the super-source and to the super-sink). Constraints (4) ensure flow conservation at each node. Constraints (5) ensure that each task is at most performed once during a shift. Constraints (6) state that the maximum cumulated distance during a shift due to water capacity in inspection mode is not exceeded. Constraints (7) limit the maximum working time during each shift. Note that to simplify the presentation, we assume that variables $x^{k\sigma}_{a,\tau}$ are equal to 0 for $\tau \le 0$. From this model, it follows that a shift can be defined as a resource-constrained path in a time-space network where constraints (5-7) are the resource-constraints.

With $Q^{k\sigma}$ as defined before, the arc flow model ($\mathcal{M}_{\text{arc}}$) is given by the following 0-1 formulation .

$$(\mathcal{M}_{\text{arc}}) \quad \min \sum_{a\in\mathcal{A}}\sum_{\sigma\in\mathcal{S}}\sum_{\tau=1}^{s}\sum_{k\in\mathcal{K}} c_{ak} x^{k,\sigma}_{a,\tau} \tag{9}$$

subject to

$$\sum_{k\in\mathcal{K}}\sum_{\sigma\in\mathcal{S}}\sum_{\tau=1}^{s} x^{k\sigma}_{a\tau} = 1 \qquad a\in\mathcal{A}^* \tag{10}$$

$$x^{k0}_{(o^k_0,D^k),0} = 1 \qquad k\in\mathcal{K} \tag{11}$$

$$x^{k,|\mathcal{S}|-1}_{(D^k,d^k_{|\mathcal{S}|-1}),s+1} = 1 \qquad k\in\mathcal{K} \tag{12}$$

$$x^{k,\sigma}_{(v,d^k_\sigma),s+1} - x^{k,\sigma+1}_{(o^k_{\sigma+1},v),0} = 0 \qquad \sigma,\sigma+1\in\mathcal{S}, v\in\mathcal{V}^r, k\in\mathcal{K} \tag{13}$$

$$x^{k\sigma} \in Q^{k\sigma} \qquad \sigma\in\mathcal{S}, k\in\mathcal{K} \tag{14}$$

Objective (9) minimizes the total cost. Constraints (10) state that each task must be performed exactly once. Constraints (11) ensure that each vehicle will leave its depot at the beginning of the first shift while constraints (12) make it

return to the depot at the end of the last shift. Constraints (13) ensure flow conservation for each vehicle at each refill node between two consecutive shifts by enforcing that the first visited refill node of shift $\sigma+1$ is the last visited refill node of shift σ. Constraints (14) enforce the feasibility of each shift for each vehicle.

Consider the example network of Figure 2. Bold arcs represent inspection tasks while thin arcs represent deadhead trips. Loops are omitted. Nodes 1, 2 and 3 are assumed to be refill stations. There are two shifts of 4 time periods each. We assume there is an outage at time period $t = 4$ on both task arc $(1, 2)^*$ and deadhead arc $(1, 2)^d$. Later, there is an outage at time periods $t \in \{7, 8\}$ on task arc $(3, 2)^*$ and deadhead arc $(3, 2)^d$. Arcs are labeled with the travel time, which is assumed to be equal to the cost for each deadhead arc. Suppose there is a single vehicle.

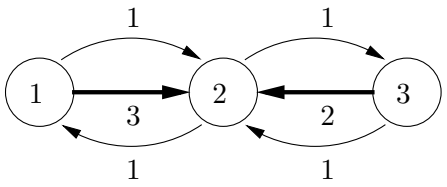


Figure 2: A simple network

The space-time networks corresponding to shifts $\sigma = 0$ and $\sigma = 1$ are represented in Figure 3. Each node is labeled with the node index v and the

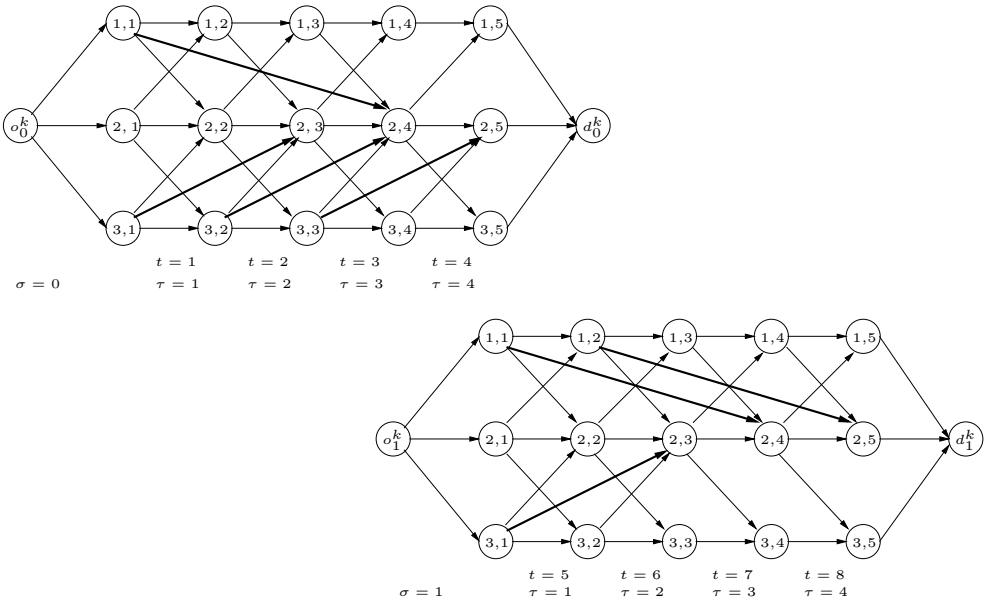


Figure 3: Space-time networks representing feasible shifts

shift time period index τ. Each arc from v, τ to v', τ' corresponds to a variable $x_{a\tau}^{k\sigma}$ with $a = (v, v')^*$ (bold arcs) or $a = (v, v')^d$ (thin arcs) with a travel time $d_{ak} = \tau' - \tau$. Each horizontal arc from v, τ to $v, \tau+1$ is a waiting arc. The set

of all $o^k_\sigma \to d^k_\sigma$ paths satisfying the resource constraints (5)-(7) in each space-time network maps the feasible shift set $Q^{k\sigma}$. The differences between the two networks are caused by the unavailability periods, as no vehicle can traverse arcs $(1,2)^d$ and $(1,2)^*$ at time period 4 (impacting shift $\sigma = 0$) nor arcs $(3,2)^d$ and $(3,2)^*$ at time periods 7 and 8 (impacting shift $\sigma = 1$). The concatenation of the shift paths is a feasible solution to ($\mathcal{M}_{\text{arc}}$) if all inspection tasks are covered (Constraints 10) and if the consecutive shifts end and start at the same node (Constraints 13). An example of feasible shifts (if the depot is refill node 3) is given by path $1,1 \to 2,4 \to 3,5$ for shift $\sigma = 0$ and $3,1 \to 2,3 \to 2,4 \to 1,5$ for shift $\sigma = 1$. Its total cost is equal to 2 (deadhead trips from 2 to 3 and from 2 to 1).

3.3 Dantzig-Wolfe decomposition and shift flow model

The arc flow model naturally decomposes into several problems (one per shift/ vehicle pair) which are linked through Constraints (10) and (13). Following the Dantzig-Wolfe decomposition principle, let us assume that the elements of $Q^{k\sigma}$ are indexed from 1 to $|Q^{k\sigma}|$ for each vehicle $k \in \mathcal{K}$ and for each shift index $\sigma \in \mathcal{S}$. We introduce binary variables $z^{k\sigma}_q$ equal to 1 if shift q is selected in $Q^{k\sigma}$, and to 0 otherwise.

Given all the sets of feasible trips between two refill stations (sets $Q^{k\sigma}$), vehicle circulation can still be modeled as a flow on a multicommodity network, in which each arc represents now a feasible shift. This yields a shift-flow model, with set partitioning constraints to model task covering.

Let $\mathcal{L}^{k\sigma}_q$ denote the sequence of visited arcs for shift $q \in Q^{k\sigma}$. The cost of a shift is defined as follows:

$$c^{k\sigma}_q = \sum_{a \in \mathcal{L}^{k\sigma}_q} c_{ak},\ k \in \mathcal{K}, \sigma \in \mathcal{S}, q \in Q^{k\sigma}. \tag{15}$$

Let $Q^{k\sigma}_a$ denote the subset of $Q^{k\sigma}$ containing shifts that visit arc a. Let $Q^{k\sigma}_{v+}$ ($Q^{k\sigma}_{v-}$) denote the subset of $Q^{k\sigma}$ containing shifts that start (end) at refill node $v \in \mathcal{V}^r$, respectively. The shift flow model ($\mathcal{M}_{\text{shift}}$) is as follows:

$$(\mathcal{M}_{\text{shift}}) \quad \min \sum_{k \in \mathcal{K}} \sum_{\sigma \in \mathcal{S}} \sum_{q \in Q^{k\sigma}} c^{k\sigma}_q z^{k\sigma}_q \tag{16}$$

subject to

$$\sum_{k\in\mathcal{K}}\sum_{\sigma\in\mathcal{S}}\sum_{q\in Q_a^{k\sigma}} z_q^{k\sigma} = 1 \qquad a \in \mathcal{A}^* \qquad (17)$$

$$\sum_{q\in Q_{D^k+}^{k0}} z_q^{k0} = 1 \qquad k \in \mathcal{K} \qquad (18)$$

$$\sum_{q\in Q_{D^k-}^{k,|S|-1}} z_q^{k,|S|-1} = 1 \qquad k \in \mathcal{K} \qquad (19)$$

$$\sum_{q\in Q_{v+}^{k,\sigma+1}} z_q^{k,\sigma+1} - \sum_{q\in Q_{v-}^{k\sigma}} z_q^{k\sigma} = 0 \qquad \sigma \in \mathcal{S}\setminus\{|\mathcal{S}|-1\}, v \in \mathcal{V}^r, k \in \mathcal{K} \qquad (20)$$

$$\sum_{q\in Q^{k\sigma}} z_q^{k\sigma} = 1 \qquad k \in \mathcal{K}, \sigma \in \mathcal{S} \qquad (21)$$

$$z_q^{k\sigma} \in \{0,1\} \qquad k \in \mathcal{K}, \sigma \in \mathcal{S}, q \in Q^{k\sigma} \qquad (22)$$

The objective function (16) ensures that from all feasible solutions the one with minimum total deadhead cost will be selected. Constraints (17) ensure that the set of selected shifts allows to perform all inspection tasks. Constraints (18) and (19) ensure that each vehicle starts and ends at its depot. Constraints (20) ensure for each vehicle that two consecutive shifts end and start at the same node. Constraints (21) enforce for each vehicle the assignment of exactly one shift per calendar day. Constraints (22) ensure that solutions are integer. Remark that the mathematical program presented above contains an exponential number of columns. Its relaxation could be solved by column generation and integer solutions could be searched by branch-and-price techniques. In this paper we take advantage of the structure of the problem to further decompose the problem, as explained in the next section.

3.4 Combinatorial Benders decomposition

Motivation and principle We explain the proposed decomposition scheme by describing how the planning is actually performed manually by operators. When they are scheduling the inspection tasks, the operators do not take into account the precise due date of each task. Instead, they prefer using the spatial information related to each task. Time windows are enforced at the end of the planning process. A simple graphical representation of where and when the tasks should be performed actually highlights the correlation between space and time. This information has been used to design a Benders decomposition scheme applied to the shift flow model. The master problem of this decomposition consists in finding a subset of shift patterns which covers every task and which can hopefully be transformed in a feasible inspection tour. The sub-problem checks if this subset violates constraints of the complete mathematical problem. If this is the case, a combinatorial cut is generated and added to the master problem and the process restarts. Otherwise the problem an optimal solution for the problem was found.

Benders master problem If each set of unavailability periods $\overline{\mathcal{H}}_{a,k}$ is empty, each time-stamped path $q \in Q^{k\sigma}$ is valid for any other shift index $\sigma' \in \mathcal{S}$ by a simple translation of the shift start time from $\sigma s+1$ to $\sigma s'+1$. In presence of unavailability periods, this statement is not true in general but we can associate to each element $q \in Q^{k\sigma}$ a set of shift indices $\mathcal{H}_q \subseteq \mathcal{S}$ for which q is valid. With such a flexibility, all shifts q that can be obtained from one another by a shift period translation defines a unique *shift pattern*. Consider the example of Figure 3. Path $1,1 \to 2,4 \to 1,5$ is a valid shift pattern for shift indices $\sigma = 0$ and $\sigma = 1$. However, path $1,1 \to 2,2 \to 3,3 \to 2,5$ is valid for shift index $\sigma = 0$ while, even if we relax the constraint to finish the trip at the depot, this path is not valid for shift index $\sigma = 1$ due to the unavailability of track $(3,2)$ at periods 7 and 8.

Dropping index σ, we define Q^k as the set of all shift patterns which are valid for vehicle k for at least one shift index $\sigma \in \mathcal{S}$. According to expression (15), the cost of a shift is independent of σ. Let c_q^k denote the cost of shift pattern $q \in Q^k$. We define Q_a^k as the subset of Q^k containing shift patterns that visit arc a and Q_{v+}^k (Q_{v-}^k) as the subset of Q^k containing shifts that start (end) at refill node $v \in \mathcal{V}^r$, respectively.

The principle of the Benders decomposition is to focus, in the master problem, on the selection of a set of shift patterns to cover all inspection tasks, without assigning the shift patterns to shift periods. The sub-problem aims, in a second phase, at finding a feasible schedule for the selected shift patterns. Let y_q^k denote a binary variable which takes value 1 if the shift pattern q is included in the path of vehicle k.

The Benders master problem ($\mathcal{M}_{\text{Benders}}$), which is a set partitioning problem with additional constraints, can be written as follows:

$$(\mathcal{M}_{\text{Benders}}) \quad \min \sum_{k \in \mathcal{K}} \sum_{q \in Q^k} c_q^k y_q^k \tag{23}$$

subject to

$$\sum_{k \in \mathcal{K}} \sum_{q \in Q_a^k} y_q^k = 1 \qquad \forall a \in \mathcal{A}^* \tag{24}$$

$$\sum_{q \in Q^k | \bar{y}_q^k = 1} (1 - y_q^k) + \sum_{q \in Q^k | \bar{y}_q^k = 0} y_q^k \geq 1 \qquad k \in \mathcal{K}, \bar{y}^k \in \mathcal{I}^k \tag{25}$$

$$y_q^k \in \{0,1\} \qquad k \in \mathcal{K}, q \in Q^k \tag{26}$$

Objective (23) and Constraints (24) are the time-independent expressions of objective (16) and Constraints (17), respectively. Constraints (24) state that all inspection tasks must be covered by the selected shift patterns. Constraints (25) are the combinatorial Benders cuts (or no-good cuts) as proposed in [4]. Set $\mathcal{I}^k$ contains the set of shift pattern selections for vehicle k violating the constraints of the original mathematical problem which are not represented in the master problem. These sets are not known a priori, they are build at each iteration by solving the subproblem presented below. Each constraint (25) removes one solution of $\mathcal{I}^k$ at a time.

Benders sub-problem Let $\bar{y}$ be the solution of the master problem at some iteration. The Benders sub-problems, whose solutions determine if the partial

solution from the master can be used to create a feasible inspection journey, can be split for each vehicle. A set of shift patterns $Q^k(\bar{y})$ covering the required tasks has been selected by the master problem. The role of the sub-problem is to assign each shift pattern selected in $\bar{y}$ to a precise shift, so that (i) track unavailability periods are respected and (ii) the shift sequence defines a feasible path for the vehicle. To increase the possibility of finding a feasible solution, more flexibility is added by relaxing the constraint that the last refill node of a shift q must be equal to the first refill node of the immediately following shift q'. Indeed, if these nodes differ, a deadhead trip from the last node of q to the first node of q' can be added at the end of shift q or at the beginning of shift q' to make the link. If the feasibility of the impacted shift is preserved by the modification, assigning q to σ and q' to $\sigma+1$ is feasible but incurs an additional deadhead cost, denoted $c^{k\sigma}_{qq'}$. The same principle can be applied to the shifts that start or end at the depot. We define $c^{k0}_{D^k q}$ ($c^{k,|\mathcal{S}|-1}_{qD^k}$) as the cost incurred if shift q starts (ends) the schedule, respectively. To model the shift/period assignment possibly incurring an additional cost, we reverse the shift-flow model presented in section 3.3 in which each shift was represented by an arc between refill stations in a time-space network. Each shift is now represented as a node while an arc connects two shifts that can be scheduled contiguously and each arc is weighted by the shift transition cost.

We introduce a dummy starting shift D^{k+} made of a single waiting arc at the depot D^k that must be assigned to a dummy shift period $\sigma = -1$ and a dummy ending shift D^{k+}, including the same arc for period $\sigma = |\mathcal{S}|$. We define $P^k(\bar{y})$ as the set of "shift arcs" (q, q') with $q \in Q^k(\bar{y}) \cup \{D^{k+}\}$ and $q' \in Q^k(\bar{y}) \cup \{D^{k-}\}$ such that q and q' can be scheduled contiguously w.r.t. the above-described adjustments. Note that, given the set of shifts selected by the master problem, the cost matrices and the shift arc sets are computed efficiently if the only allowed changes consist in adding dead-head trips at the start or the end of the two considered shifts and removing waiting times accordingly.

Consider again the example of Figure 3, ignoring depot constraints. Suppose the master problem has selected shift patterns q $(1,1 \rightarrow 2,4 \rightarrow 2,5)$ and q' $(3,1 \rightarrow 2,3 \rightarrow 2,4 \rightarrow 2,5)$ both with cost 0. Combining these shift patterns is not possible as such because their start and end nodes do not coincide. However, assigning q to shift 0 and q' to shift 1 becomes feasible by inserting deadhead trips (more precisely by replacing waiting period $2,4 \rightarrow 2,5$ by trip $2,4 \rightarrow 3,5$ in q and waiting period $2,4 \rightarrow 2,5$ by trip $2,4 \rightarrow 1,5$ in shift q'). Figure 4 displays the modification of q (left side) to make it compatible with q' (right side) incurring costs $c^{k0}_{qq'} = 1$ and $c^{k1}_{q'D^k} = 1$, i.e. a total cost of 2.

Let $z^{k\sigma}_{qq'}$ be a binary variable equal to 1 if shift pattern q' of vehicle k is assigned to shift period σ and preceded by shift pattern q. The Benders sub-problems can be written, for each vehicle k, as follows:

$$(\mathcal{SP}^k_{\text{Benders}}) \quad \min \sum_{(q,q') \in P^k(\bar{y})} \sum_{\sigma=0}^{|\mathcal{S}|} c^{k\sigma}_{qq'} z^{k\sigma}_{qq'} \tag{27}$$

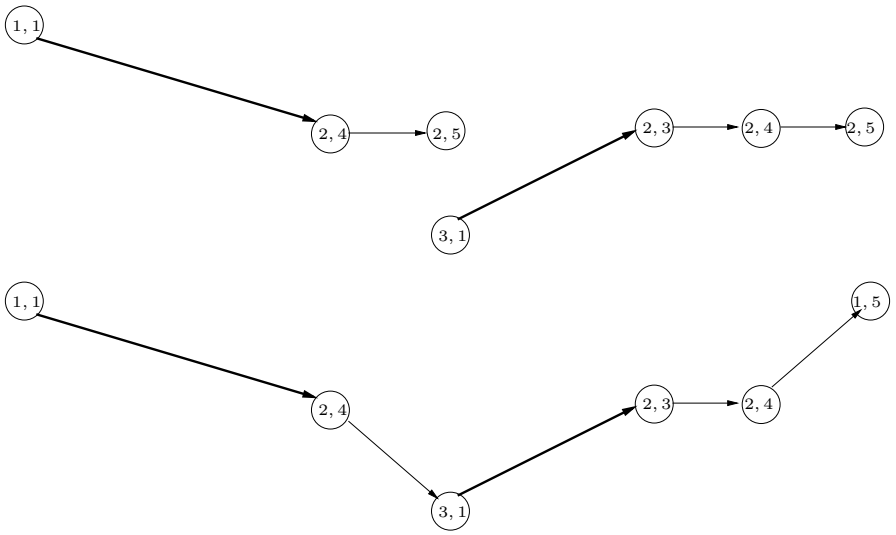


Figure 4: Benders subproblem: assigning shift patterns to time periods and repairing the master solution

$$\sum_{q|(D^k+q)\in P^k(\bar{y})} z^{k0}_{D^k+q} = 1 \tag{28}$$

$$\sum_{q|(qD^k-)\in P^k(\bar{y})} z^{k|\mathcal{S}|}_{qD^k-} = 1 \tag{29}$$

$$\sum_{q'|(q',q)\in P^k(\bar{y})} z^{k,\sigma-1}_{q'q} - \sum_{q'|(q,q')\in P^k(\bar{y})} z^{k\sigma}_{qq'} = 0 \quad \sigma\in\{1,\ldots,|\mathcal{S}|\}, q\in Q^k(\bar{y})\cup\{D^k-\} \tag{30}$$

$$\sum_{\sigma\in\mathcal{S}\setminus\{0\}} \sum_{q'|(q,q')\in P^k(\bar{y})} z^{k,\sigma}_{qq'} = 1 \qquad q \in Q^k(\bar{y}) \tag{31}$$

$$\sum_{\sigma\in\mathcal{S}} \sum_{q'|(q',q)\in P^k(\bar{y})} z^{k,\sigma}_{q'q} = 1 \qquad q \in Q^k(\bar{y}) \tag{32}$$

$$z^{k\sigma}_{qq'} = 0 \quad (q,q')\in P^k(\bar{y}),(\sigma-1,\sigma)\notin\mathcal{H}_q\times\mathcal{H}_{q'} \tag{33}$$

$$z^{k\sigma}_{qq'} \in \{0,1\} \qquad \sigma\in\mathcal{S}\cup\{|\mathcal{S}|\}, (q,q')\in P^k(\bar{y}) \tag{34}$$

Objective (27) is to find a solution that minimizes the total cost of the shift modifications. Constraints (28) indicate that a shift pattern starting at the depot must be scheduled at the first period while Constraints (29) express that the last shift must end at the depot. Constraints (30) express flow conservation constraints for each shift at each shift period. All together, Constraints (28)-(30) ensure that exactly one shift will be assigned to each time period and that consecutive shifts have to be compatible (a single path is selected in the shift time-space network). Constraints (31) and (32) state in addition that each shift must be assigned to exactly one shift period. Constraints (33) prevent a shift pattern from being assigned to an incompatible shift. To sum-up, the so-modeled shift scheduling problem is a special case of the traveling salesman problem with time windows.

A tightened Benders master problem To improve the Benders master problem, we consider adding in ($\mathcal{M}_{\text{Benders}}$) the following "equilibrium" constraints, yielding ($\mathcal{M}^*_{\text{Benders}}$).

$$\sum_{q \in Q^k_{v+}} y^k_q - \sum_{q \in Q^k_{v-}} y^k_q = 0 \quad v \in \mathcal{V}^r, k \in \mathcal{K} \tag{35}$$

Constraints (35) are the shift flow conservation constraints in space. They state that, for each vehicle, the number of shifts that end with a given refill node (including the depot) is equal to the number of shifts that start with the same refill node, independently of the time periods. They are not necessary for the proposed decomposition, as they are also tackled in the sub-problem. However, without these constraints, the master problem lacks information about how the tasks can be sequenced, either in space or time. This would lead to the generation of combinatorial Benders cuts (25) solely used to satisfy the flow constraints. This is illustrated by the two shifts selected by the master problem without Constraints (35) displayed on top of Figure 4. If we add Constraints (35) to the master problem, the selection of these two shifts becomes infeasible.

An exact Benders method From these decompositions, an exact Benders method that consist in iteratively solving the Benders master problem and the benders sub-problems can be derived.

Even if the set of shift patterns is smaller that the set of time-stamped shifts, the Benders master problem still has an exponential number of variables. Hence to solve ($\mathcal{M}^*_{\text{Benders}}$) exactly, a branch-and-price procedure has to be used, where at each node of a branch-and-bound tree, a column generation procedure solves the linear programming (LP) relaxation of ($\mathcal{M}^*_{\text{Benders}}$).

Once the optimal solution of ($\mathcal{M}^*_{\text{Benders}}$) is obtained, sub-problems ($\mathcal{SP}^k_{\text{Benders}}$) have to be solved, one for each vehicle $k \in \mathcal{K}$. The three following cases have to be considered after solving the sub-problems.

- If all sub-problems have an objective value equal to 0, the global optimal solution has been found.
- If one of the sub-problems is infeasible, $\bar{y}^k$ can be discarded by generating the corresponding no-good *feasibility* cut (25) and adding it to the master problem, which will be solved again during a new iteration.
- If the sub-problems are all feasible but at least one has a non-zero cost, a feasible solution for the RTISP is found and an upper bound on the total dead-head cost is obtained. Yet, solution $\bar{y}^k$ can be discarded from the search space of the master problem by generating the corresponding no-good *optimality* cut (25), which will be added to the master problem and the process is reiterated. Note also that, since the master is solved through column generation, the feasible shifts that are found can be added as possibly new columns.

4 Column and cut generation matheuristic

4.1 Motivation and general principles

The exact Benders method described in Section 3.4 cannot be applied to solve the RTISP instances provided by SNCF for at least two reasons. First, the instances are too large to be solved by an exact method. Second, as explained in Section 4.2, the combinatorial Benders cuts make the column generation sub-problem difficult to solve. To use the decomposition scheme in practice, we have chosen to develop alternatively a cut and column generation matheuristic, which is also able to yield lower bounds. The method is a four step heuristic, described as follows:

1. (ColGen) An LP relaxation of the Benders master problem ($\mathcal{M}^*_{\text{Benders}}$) is solved through column generation. At this step, a lower bound can be obtained. Compared to ($\mathcal{M}^*_{\text{Benders}}$), the considered model has two main differences. First, in a standard way, highly penalized slack variables are introduced in set partitioning constraints (24) to ensure feasibility. More importantly, the generated cuts may have a different form expression (25) , and/or the cut generation algorithm can be parametrized in a way that it become heuristic (in the sense that the generated cuts may exclude potentially optimal solutions). In the latter case, the obtained lower bound is not valid and the ColGen procedure is mainly used to obtain a diversified and relevant set of shift patterns. The procedure is described in mode details in Section 4.2 and the alternative Benders cut generation process is made at step 3 (RelCutGen).

2. (HeurCover) From the solution of this relaxation, a rounding heuristic for the covering problem proposed in [3] is adapted to find a good integer solution to the Benders master problem ($\mathcal{M}^*_{\text{Benders}}$). At this point, a set of shift patterns satisfying the equilibrium constraints (35) is obtained. The procedure is described in Section 4.3.

3. (RelCutGen) The selected shift patterns have to be scheduled in compatible shift periods. For each vehicle k, a relaxation of the Benders sub-problem ($\mathcal{SP}^k_{\text{Benders}}$) is solved. If one of the relaxations is infeasible, a Benders infeasibility cut is added to the master problem to exclude the solution found at step 2 and we return to step 1. As mentioned in step 1, the Benders infeasibility cuts produced at this step have a different form than combinatorial cuts (25). This is either due to the structure of the relaxation (aa assignment problem) or to a heuristic cut generation scheme. The procedure is described in Section 4.4.

4. (HeurSched) If the relaxations of the Benders sub-problem are all feasible, a heuristic is used to solve ($\mathcal{SP}^k_{\text{Benders}}$), i.e. to schedule the selected shift patterns, for each vehicle k. If one of the problems is infeasible, a combinatorial Benders cut (25) or a heuristic cut can be generated and we return to Step 1. Otherwise the solution is returned. The scheduling heuristic is described in Section 4.5.

4.2 ColGen: solving the relaxed Benders master problem ($\mathcal{M}^*_{\text{Benders}}$) with column generation

Initial column set The Benders master problem ($\mathcal{M}^*_{\text{Benders}}$) is considered with a restricted set of columns, corresponding to a restricted set of shift patterns $\tilde{Q}^k$.

The initial column set is initialized by considering for each required arc $a \in \mathcal{A}^*$ a shift pattern covering this arc. This allows to satisfy partitioning constraints (24). Furthermore, for each pair of refill nodes, a column corresponding to a dead-head trip joining these two nodes is added to $\tilde{Q}^k$ to satisfy the equilibrium constraints (35).

Restricted master problem solving The LP relaxation of the so-obtained restricted master problem is denoted as $(\tilde{\mathcal{M}}^*_{\text{Benders}})$. It is solved with the simplex algorithm. At the first iteration, there is no Benders cut (25). At the subsequent iterations, no-good cuts are incorporated to take the infeasibility of the corresponding Benders sub-problems $(\mathcal{SP}^k_{\text{Benders}})$ into account. As explained in Section 4.1, the cuts do not necessarily correspond to expression (25). This is due to several reasons that will be explained in Section 4.4.

Column generation subproblems As only a restricted number of columns is considered, the solution is not necessarily the optimal LP solution, as there may exist non considered columns with negative reduced cost. To explain how such columns are searched, let us first give the expression of a variable's reduced cost, by considering problem $(\mathcal{M}^*_{\text{Benders}})$ without any no-good cut. The reduced cost of a variable y^k_q of the master problem is a function of the dual solution vector. Let α^k_a, for all task arcs $a \in \mathcal{A}^*$, denote the dual variables related to partitioning constraints (24). Let β^k_v, for all refill nodes $v \in \mathcal{V}^r$ and vehicles $k \in \mathcal{K}$, denote the dual variables related to equilibrium constraints (35). If we ignore no-good cuts, the reduced cost of a variable y^k_q for a vehicle $k \in \mathcal{K}$ and a shift pattern $q \in Q^k$ starting at refill node v and ending at refill node v' is

$$\tilde{c}^k_q = c^k_q - \sum_{a \in \mathcal{A}^* | q \in Q^k_a} \alpha_a + \beta^k_v - \beta^k_{v'}$$

The column generation sub-problem aims at finding for each vehicle $k \in \mathcal{K}$ a feasible shift pattern $q \in Q^k$ of negative $\tilde{c}^k_q$, or at proving that no such shift pattern exists. To be feasible, a shift pattern has to correspond to a valid shift for at least one shift period σ, i.e. a shift in some $Q^{k\sigma}$. Hence, we obtain a column generation sub-problem $(\mathcal{SP}^{k\sigma}_{\text{ColGen}})$ per vehicle $k \in \mathcal{K}$ and shift index $\sigma \in \mathcal{S}$, which can be written as an arc flow model, as follows.

$$(\mathcal{SP}^{k\sigma}_{\text{ColGen}}) \quad \min \sum_{a \in \mathcal{A}} \sum_{\tau=1}^{s} \tilde{c}_{ak\tau} x^{k,\sigma}_{a,\tau} \tag{36}$$

subject to

$$x^{k\sigma} \in Q^{k\sigma}$$

where

$$\tilde{c}_{(i,j)k\tau} = c_{(i,j)k} - \tilde{\alpha}_{(i,j)} + \tilde{\beta}^k_{i\tau} - \tilde{\gamma}^k_{j,\tau+d_{(i,j)k}-1},$$

$$\tilde{\alpha}_a = \begin{cases} \alpha_a & \text{if } a \in \mathcal{A}^* \\ 0 & \text{otherwise,} \end{cases} \qquad \tilde{\beta}^k_{i\tau} = \begin{cases} \beta^k_i & \text{if } i \in \mathcal{V}^r \text{ and } \tau = 1 \\ 0 & \text{otherwise,} \end{cases}$$

$$\tilde{\gamma}^k_{i\tau} = \begin{cases} \beta^k_i & \text{if } i \in \mathcal{V}^r \text{ and } \tau = s \\ 0 & \text{otherwise.} \end{cases}$$

Given the expression of $Q^{k\sigma}$, each column generation sub-problem ($\mathcal{SP}^{k\sigma}_{\text{ColGen}}$) is a resource-constrained shortest path problem. The resulting path gives a new shift pattern corresponding to a variable y^k_q with a negative reduced cost (or it is proved that no such path exists).

If we want to integrate the no-good constraints in the column generation sub-problem along the method iterations, we have to take care of the following issues. While set partitioning constraints (24) and equilibrium constraints (35) are defined on arcs and nodes of the space-time network, the Benders infeasibility cuts (25) are defined on a shift pattern. This characteristics can slow down the dynamic programming algorithm used for solving the RCESP. This was for example recently experienced in [24] with clique inequalities defined on the set partitioning master problem of the vehicle routing problem with time windows, which required adding multiple resources in the RCESP. In this work, we ignore the dual information carried by no-good constraints when solving the column generation subproblems. As the reduced cost is simplified, the column generation algorithm could generate already existing columns. Hence, a duplicate detection mechanism is implemented and the algorithm stops in case of cycling.

It should be noticed that it could be time consuming to compute $|\mathcal{K}| \cdot |\mathcal{S}|$ constrained shortest paths at each column generation iteration. Our implementation enables finding valid shortest paths for several consecutive shifts. It can be parametrized to generate from $|\mathcal{K}|$ to $|\mathcal{K}| \cdot |\mathcal{S}|$ subproblems. At the first extreme, the feasible solution space of each of the $|\mathcal{K}|$ subproblems is large. Solving one of them may take time. In the other extreme, the feasible solution space of each of the $|\mathcal{K}| \cdot |\mathcal{S}|$ subproblems is narrow and solving one of them is fast.

Column generation algorithm For solving the sub-problems, we implemented a variant of the dynamic programming algorithm of [8] for the resource-constrained elementary shortest path problem. This algorithm has been selected because it is very flexible and adapting it to different types of vehicles cis easy. This is important in our case because the solving scheme should be general enough to allow its use for different problems.

The dynamic programming is a labelling algorithm. At each iteration, a node and an associated label are picked from a queue. The label is then extended through the traversal of the outgoing arcs, consuming resources according to the arc type. Dominance rules allow to prune labels which are known not to lead to an optimal solution as explained in [8]. The main difference is that the path does not need to be strictly elementary, unlike in paper [8]. In our setting, communication arcs and nodes can be traversed as needed. We just have to be sure that each task arc is visited at most once, as stated by Constraints 5.

4.3 HeurCover: greedy rounding heuristic for solving the Benders master problem ($\mathcal{M}^*_{\text{Benders}}$)

Starting with the continuous solution $\tilde{y}$ of the Benders master problem ($\mathcal{M}^*_{\text{Benders}}$) obtained by procedure ColGen (see Section 4.2), a greedy rounding heuristic inspired by the one proposed in [3] is used to get an integer solution (satisfying only the Benders master problem constraints). The principle of the heuristic is the following. For each vehicle $k \in \mathcal{K}$, let $\bar{Q}^k$ denote the set of shift patterns which can be selected, initially containing all shift patternS q such that $\tilde{y}^k_q \neq 0$.

Let $\mathcal{A}^*_{qk}$ be the set of tasks covered by a shift pattern $q \in \bar{Q}^k$. Let C denote the tasks still not covered, initially equal to $\mathcal{A}^*$. Let C^k_q denote the set of uncovered inspection tasks that can be covered by $q \in \bar{Q}^k$, initially set to $\mathcal{A}^*_{qk}$. Let $f(q,k)$ denote a priority function associated to shift pattern $q \in \bar{Q}^k$.

At each iteration, the greedy heuristic selects the shift pattern q^* for the vehicle k^* with the minimum priority function value. q^* is removed from $\bar{Q}^k$. Each inspection task $a \in \mathcal{A}^*_{q^*k}$ is removed from C and from all C^k_q with $k \neq k^*$ or $q \neq q^*$. Each shift pattern $q \in \bar{Q}^k$ such that C^k_q becomes empty is removed from $\bar{Q}^k$. The process is repeated until C is empty or no shift pattern can be selected anymore.

We propose several priority functions f. In [3], function $f_0(q,k) = c^k_q/|C^k_q|$ gives the ratio of the variable objective coefficient to the number of uncovered tasks it would cover if q was selected. We also propose alternative priority functions. $f_1(q,k) = c^k_q(1 - |C^k_q|/(|\mathcal{A}^* \setminus C| + 1))$ is a dynamic priority function that multiplies the variable cost by the ratio of the number of tasks that would remain uncovered if q was selected to the number of tasks already covered . Finally, $f_2(q,k) = c^k_q(1 - |C^k_q|/(|(\mathcal{A}^* \setminus C) \cap \mathcal{A}^*_{qk}| + 1))$ multiplies the variable cost by 1 minus the ratio of the number of uncovered tasks that q would cover if it was selected to the number of already covered tasks also covered by q. Both dynamic priority functions give priority to the shift patterns covering a lot of tasks at the first iterations and, when the number of covered tasks becomes large, to the shift patterns with a low cost.

In addition, weighted variants of these priority functions can be obtained by taking into account the value of the variable in the solution of the LP relaxation. Let $nint(i)$ denote the nearest integer to i. We define the following weighted functions:

$$f'_x(q,k) = f(q,k)(1 + \text{nint}(\tilde{y}^k_q) - \tilde{y}^k_q).$$

Last, the procedure is significantly improved by iteratively fixing the selected variables and resolving the linear relaxation of ($\mathcal{M}^*_{\text{Benders}}$). At each iteration of HeurCover, the linear relaxation of ($\mathcal{M}^*_{\text{Benders}}$) (with the last generated restricted set of columns) is solved with the additional constraint that $y^k_q = 1$. $\bar{y}$ is updated before selecting the next variable in the next HeurCover iteration, and the rpocess is iterated until all tasks are covered or no column are added. This incremental variant of HeurCover allows handling every constraint of the Master problem (not only the ones which ensure that every required arc belongs to a feasible trip).

4.4 RelCutGen: cut generation based on relaxations of the Benders sub-problems ($\mathcal{SP}^k_{\text{Benders}}$)

At this point, an integer solution $\bar{y}$ to the Benders master problem ($\mathcal{M}^*_{\text{Benders}}$) has been computed by the greedy procedure HeurCover (see Section 4.3). The Benders sub-problems ($\mathcal{SP}^k_{\text{Benders}}$), for $k \in \mathcal{K}$ can be solved to generate no-good cuts in case of infeasibility or sub-optimality. However, no-good cuts (25) invalidate a single solution $\bar{y}$. Their impact to the LP relaxation ($\tilde{\mathcal{M}}^*_{\text{Benders}}$) is weak and we do not use any branch-and-price procedure where they could have a pruning power. Furthermore, each Benders sub-problem is a special instance of the TSPTW, and, consequently, is hard to solve. We propose two ways to deal with this issue.

Solving relaxations of the Benders sub-problems Dropping flow conservation constraints (30) in each ($\mathcal{SP}^k_{\text{Benders}}$) for $k \in \mathcal{K}$ yields an easy-to-solve assignment problem.

To explain how cuts can be obtained by this relaxation, let us focus only on the feasibility of the Benders sub-problem for shift/period assignment. We add slack variable $h_{D^{k-}}$ to Constraint (29) and slack variables h_q, $\forall q \in Q^k(\bar{y})$ to constraints (32). Consider solving the assignment problem min $\sum_{q \in \{D^{k-}\} \cup Q^k(\bar{y})} h_q$ subject to (29), (32), (33), non negativity of variables h_q and relaxing integrity constraints for variables $z^{\sigma k}_{qq'}$. The objective function gives a lower bound on the number of shifts of $Q^k(\bar{y})$ selected by the master problem that cannot be assigned to a feasible period, given the availability periods in $\mathcal{H}_q$. This yields the following Benders cut

$$\sum_{q \in Q^k(\bar{y})} y^k_q \leq \sum_{q \in Q^k(\bar{y})} \sum_{\sigma \in \mathcal{S}} \sum_{q' | (q',q) \in P^k(\bar{y})} \tilde{z}^{k\sigma}_{q'q}$$

where $\tilde{z}^{k\sigma}_{q'q}$ is the (integer) solution of the relaxed Benders sub-problem. By introducing different assignment constraints in the linear program, different cuts can be obtained by applying the same principle.

Pseudo-cuts Pseudo-cuts, i.e. cuts that may exclude valid integer solutions of the master problem, can be defined to invalidate continuous solutions of the master problem, so that the integer solution has little chance to be generated by the HeurCover heuristic. More concretely, let $\tilde{y}$ be a continuous solution of the master problem, and let $\bar{y}$ denote its integer solution found by HeurCover.

Combinatorial or assignment-problem-based Benders cuts are generally of the following form.

$$\sum_{q \in Q^k(\bar{y})} y^k_q \leq n$$

where n is the number of variables that cannot be selected. As these cuts are violated by the integer solution we have

$$\sum_{q \in Q^k(\bar{y})} \bar{y}^k_q > n$$

Then, if we assume, in addition, that $\bar{y}^k_q \leq \lceil \tilde{y}^k_q \rceil$, the following constraint invalidate continuous solution $\tilde{y}$

$$\sum_{q \in Q^k(\bar{y})} \frac{y^k_q}{\tilde{y}^k_q} \leq n$$

Note that the closer the value $\tilde{y}^k_q$ is to an integer solution the more the cuts resemble to the original Benders cuts.

4.5 HeurSched: heuristic for the solving the Benders sub-problems ($\mathcal{SP}^k_{\text{Benders}}$)

If the relaxations presented in Section 4.4 are feasible for the current solution to the master problem, the original combinatorial Benders subproblems ($\mathcal{SP}^k_{\text{Benders}}$), for $k \in \mathcal{K}$, should be solved to determine the actual feasibility of this solution and possibly repairing it.

The corresponding traveling salesman problem with time windows is solved using a guided multi-start heuristic. Consider a chronological search tree with a maximum depth equal to $|\mathcal{S}|$ in which each node corresponds to a partial assignment of the selected shifts up to shift index σ. An iteration of the heuristic consists in a depth-first search on the search tree, yielding a complete shift/period assignment. At a given node of depth σ, let $q_{\sigma-1}$ be the previously selected shift, where $q_{-1} = D^{k+}$ for the root node $\sigma = 0$. The depth first search algorithm selects the decision (a shift q such that $\sigma \in \mathcal{H}_q$ and $(q_{\sigma-1}, q) \in P^k(\bar{y})$) that minimizes a dynamic transition function $t(q_{\sigma-1}, q)$. The transition function is initially set to the dead-head cost $t(q, q') = c^k_{qq'}$ for each pair $(q, q') \in P^k(\bar{y})$. However, for each possible decision at a node, two depth-first explorations are performed by using the current transition cost: one which selects the first shift according to the transition function, yielding a complete solution with a cost c^+_q and one which cannot select this shift yielding a complete solution with a cost c^-_q. A price π_q is associated with the decision to schedule shift q right after shift $q_{\sigma-1}$ where $\pi_q = c^+_q - c^-_q$. The transition cost matrix is updated with $t(q_{\sigma-1}, q) \leftarrow t(q_{\sigma-1}, q) + \pi_q$. The algorithm iterates until the matrix transition cost modification becomes negligible. For the considered dataset, this method proved to be the most efficient in terms of quality and solution time. This principle can be mixed with backtracking, when a decision does not yield a feasible solution.

5 A dynamic programming heuristic

In order to evaluate the proposed decomposition scheme with another heuristic which is closer to what would be performed by a human operator, we additionally designed a greedy algorithm to solve the complete RTISP based on the dynamic programming method used to generate the shift patterns for the column generation procedure. This algorithm uses the resource constrained shortest path solver to compute at each iteration the set of best shift patterns which can be performed starting from the last visited node. Hence, the algorithm is greedy in terms of shift scheduling, while the dynamic programming method used to generate the shift still performs an implicit enumeration. The shift of lowest cost is appended to the current partial solution. We select a vehicle and, starting from its depot, we apply this procedure until the end of the schedule horizon is reached. Then, we start with the next vehicle.

If no task is reachable from the current node, a deadhead move is performed to find the nearest node which enables performing a task. Deadhead speed is fast enough to allow every vehicle to travel from one node to every other node in less than a shift. This simple choice allows the potential generation of solutions which cover every task.

The cost function drives the algorithm towards a good solution and should be carefully chosen based on the characteristics of the dataset, and on whether the problem is more constrained in time or space. For task selection during the shortest path computation, different weight update rules have been tested. The best performing one uses information about task time windows and durations.

Let w^k_a denotes the cost of performing task a on vehicle k. This cost is defined as follows:

$$w_a^k = -M + c_a(2.0 - \frac{es_a}{ls_a}),$$

with es_a and ls_a, denoting respectively the earliest and latest start of task a. M should have a value such that the algorithm will always prefer performing a task over performing a deadhead trip.

6 Computational tests

6.1 Real data from 2009

The computational test was conducted on real data provided by SNCF. Full results are available in the PhD thesis [16]. These data has been used to calculate the time windows for each of the 2009's inspection tasks. The data model contains three vehicles and a realistic average working speed. Based on this data, multiple scenarios have been established by varying the track outages duration, the size of the considered network, the time horizon and other practical characteristics.

Complete network characteristics The complete graph has 2100 arcs and 760 nodes of which 90 are refill stations and about 700 tasks to perform. Task time windows have a fixed size of 28 days and their duration range from a few minutes to six hours. The duration of a shift is fixed to 7 hours.

Data sets The first two data sets (RTISP1.1 and RTISP1.2) involve a single vehicle with no track outage (the vehicle may circulate at any time in any part of the network). The first scenario captures data from the first half of 2009 and the second scenario extends it by adding the rest of the year. The second series of data sets (RTISP2.1 and RTISP2.1) still involves no track outage but consider special tasks limiting vehicle availability. Data set RTISP2.1 includes the special tasks corresponding to regional demands. Recall that, as only primary tracks inspection is precisely scheduled, regional departments can ask for one vehicle during a fixed period to inspect the secondary tracks . This is modeled by a fictitious inspection task located at the node where the vehicle must be provided and will be returned. As this period can be large, such scenarios are more constrained. Three vehicles are considered. Data set RTISP2.2 includes the regional demand tasks and maintenance tasks which are also special tasks representing a time period during which the vehicle must be sent to maintenance at a specific node. This set also includes three vehicles. The third data set series (RTISP3.1, RTISP3.2, RTISP3.3, RTISP3.4) contains 100 tasks and a one month time horizon but includes an increasing number of track outages. RTISP3.1 includes no outages, RTISP3.2, RTISP3.3 and RTISP3.4 include all tracks outages of more than 6 hours, 3 hours and 1 hour, respectively. The fifth series is made of the single data set RTISP4.1, which is identical to set RTISP3.1, except that one of the two vehicles is significantly slower than the other one. The last series of data sets (RTISP5.1, RTISP5.2, RTISP5.3, RTISP5.4) involves both a time restriction (may 2009) and a network restriction (Brittany region). It was generated mainly to have "small" non trivial instances (56 tasks). It involves 2 vehicles and possibly track outages. The data set characteristics are synthesized in Table 1.

Instance	#tasks	#vehicle	Comment
RTISP1.1	328	1	first semester 2009, no outages
RTISP1.2	694	1	2009, no outages
RTISP2.1	525	3	2009, regional demands
RTISP2.2	573	3	2009, regional demands & maintenance
RTISP3.1	100	2	February 2009, no outages
RTISP3.2	100	2	February 2009, outages of more than 6 hours
RTISP3.3	100	2	February 2009 ,outages of more than 3 hours
RTISP3.4	100	2	February 2009, outages of more than 1 hours
RTISP4.1	100	2	February 2009, no outages, 1 slow vehicle
RTISP5.1	56	2	May 2009, 12 outages
RTISP5.2	56	1	May 2009, 12 outages
RTISP5.3	56	1	May 2009, no outage
RTISP5.4	56	2	May 2009, no outage

Table 1: Instance characteristics

6.2 Key indicators

For each algorithm and each of these scenarios, the completion rate r defines the number of task which are covered by the solution. The performance ratio p gives information about the total distance traveled without inspecting (deadhead). The higher the ratio, the better the solution from the point of view of the end user. The performance ratio p is calculated to reflect the rate between the total inspected distance (d_i) and the cumulated deadhead length (d_d):

$$p = \frac{d_i}{d_i + d_d}$$

The task completion rate r is used to get information about the hardness of the instance. When comparing the decomposition heuristic with the dynamic programmming heuristic, this indicator gives valuable information about how hard it is to schedule the instance.

6.3 Global results

The experiments were carried out on an Intel Xeon (64bits) 3.0GHz, having 4 cores. We use IBM CPLEX 12.2 for linear programming and IBM CP 1.6 and SCHEDULER for the guided multi-start scheduling heuristic. All programs were written in C++. We parallelize the column generation procedure since the subproblems are independent.

In Table 2, the results of the proposed heuristic on the above presented data set is presented. The alloted CPU time has been adapted to the instance size with 4 hours for the largest instances, 30 min for the 100-task instances and 5 min for the small instances. In terms of completion rate, the global performance of the heuristic is good, except for instance set RTISP4.1 for which less than 30% of the tasks are performed. This set is identical to set RTISP3.1, except that a vehicle is significantly slower than the other, which makes the instance infeasible. There are also two sets (RTISP2.1 and RTISP2.2) for which less than 90% of

Instance	#tasks	Completion rate	Performance ratio	Solving time (s)
RTISP1.1	328	100%	88.4%	4 hours
RTISP1.2	694	97.4%	89.7%	4 hours
RTISP2.1	525	89.1%	49.3%	4 hours
RTISP2.2	573	86.9%	49.3%	4 hours
RTISP3.1	100	100%	41%	30 min
RTISP3.2	100	100%	40%	30 min
RTISP3.3	100	100%	38%	30 min
RTISP3.4	100	100%	43%	30 min
RTISP4.1	100	28%	13%	30 min
RTISP5.1	56	100%	30.5%	5 min
RTISP5.2	56	100%	35.5%	5 min
RTISP5.3	56	100%	34.4%	5 min
RTISP5.4	56	100%	29.4%	5 min

Table 2: Results of the proposed heuristic on the real data sets

the tasks are performed. These sets correspond to the cases where vehicles can be required for maintenances and/or regional demands. This shows that maintenance and regional demands should be carefully planned, given the limited number of inspection vehicles. In terms of performance ratio, the presence of maintenance and regional demand activities penalizes the results (as shown by the difference between series 1 and 2). It must be noticed, however, that the performance ratio naturally decreases as the number of inspection tasks decreases, which explains the difference between instances RTISP1.1/RTISP1.2 on the one hand and instance RTISP3.1 on the other hand (which all have no outage). Indeed, when a reduced number of inspection tasks is considered in the full network, the total dead-head distance increases comparing to the inspection distance.

6.4 Comparison with the dynamic programming heuristic

To assess the performance of the proposed heuristic we first compare it to the dynamic programming heuristic proposed in Section 5. We compare the two heuristics (*AlgoBenders* and *AlgoGreedy*) on the instance series 3 since the number of rail track outages appears to be discriminant. The results are displayed in Table 3. The dynamic-programming-based heuristic solves all datasets in less than 5 minutes. However both its completion rate and its performance ratio decrease as the number of outage increases. This seems to show that the heuristic takes local decisions which badly impact the global quality of the solution. In comparison, the decomposition algorithm performs well: every task is executed and the performance ratio is much better (more than twice better for the larger number of outages). Furthermore, the matheuristic is less sensitive to outage variations.

		RTISP3.1	RTISP3.2	RTISP3.3	RTISP3.4
Completion rate	(*AlgoGreedy*)	100%	88%	86%	86%
	(*AlgoBenders*)	100%	100%	100%	100%
Performance ratio	(*AlgoGreedy*)	32%	23%	23%	18%
	(*AlgoBenders*)	41%	40%	38%	43%
Solving time (s)	(*AlgoGreedy*)	246	283	254	178
	(*AlgoBenders*)	1500	1500	1500	1500

Table 3: Comparison between the decomposition algorithm and the dynamic programming-based heuristic

	RTISP5.1	RTISP5.2	RTISP5.3	RTISP5.4
Gap (IP-LP)/LP	0%	1%	8,57%	6,22%
Performance UB	43,96%	46,95%	45,74%	48,84%
Performance LB	30,53%	35,49%	34,40%	29,41%

Table 4: Comparison with the LP relaxation

6.5 Comparison with linear programming relaxation

To further evaluate the quality of the proposed heuristic, we compare it with the linear programing relaxation of the Benders master problem, using the column generation procedure ColGen (see Section 4.2). For large instances, the column generation procedure is stopped by the predefined time limit before a valid lower bound on the total dead-head distance is obtained. A valid lower bound can be obtained only on the fifth instance series, which contains 56 tasks. The results are displayed in Table 4. Row "Gap (IP-LP)/LP" gives the gap between the dead-head distance of the LP relaxation obtained by ColGen and the total dead-head distance obtained by the rounding heuristic HeurCover presented in Section 4.3. Row "Performance UB" gives the upper bound on the performance ratio given by the LP relaxation while row "Performance LB" recalls the performance ratio of the solutions computed by the matheuristic.

First, it has to be noticed that the rounding heuristic gives results very close to the LP relaxation. The performance gap between the proposed heuristic and the LP relaxation is much larger, although. This means that the dead-head distance is added by the repairing Benders sub-problem solving procedure when sequencing the selected shifts. However, the upper bound on the performance ratio provided by the LP relaxation is under 45% in average which tempers the apparently low performance of the the proposed heuristic (around 30% in average). Indeed, this provides the valuable information that reaching a performance of 50% is an unachievable goal in this subnetwork. However, to explain the 15% difference, we suspect that the upper bound quality is not so good. By analyzing the continuous solutions, we observe that they include a lot of subtours related to the visited refill nodes, and they do not take into account deadhead moves between these subtours. Furthermore, these data sets consider only the Brittany trgion and only a few weeks, whereas the initial algorithm was developed to solve a one year planning on the whole network. Hence, in these

	RTISP1.1			*RTISP1.2*		
	30	60	120	30	60	120
f_0	35	25	25		90	90
f_1	45	45	45	120	120	120
f_2	45	45	45	110	82	62

Table 5: Comparison of priority function for HeurCover

scenarios, tasks are neither contiguous in space nor in time. Consequently, the results do no necessarily generalize to the complete network. By contrast, for data sets RTISP1.1, RTISP1.2, RTISP1.3 and RTISP1.4 (extracted on a larger period and for the complete network), the shift sequencing heuristic did not add any additional dead-head trips (the solution provided by the Benders master problem was feasible).

6.6 Impact of algorithm components

The complexity of the algorithm and the necessity of industrial standard lead the design of the algorithm implementation towards a very modular solution. Indeed, the building blocks are inspired by the design patterns proposed in [9]. Each algorithm is a building block (set cover heuristic, column generation, assignment problem, ...) and can have multiple implementations.

In this section, we study the impact of variants of algorithm component on the global performance. The most important impact on the solving time was obtained by tuning the algorithm used to solve the Benders master problem ($\mathcal{M}^*_{\text{Benders}}$). Multiple priority functions in the HeurCover heuristic for finding integer solutions to ($\mathcal{M}^*_{\text{Benders}}$) have been tested. The conducted tests also demonstrate the great importance of adding the redundant equilibrium constraint in ($\mathcal{M}^*_{\text{Benders}}$). The interest of the proposed cuts is also assessed.

Multiple priority functions in HeurCover The different priority rules f_0, f_1 and f_2 are detailed in Sections 4.3 for the rounding heuristic HeurCover.

Table 5 presents the value of the best integer solution (total dead-head distance in TKm) for different computation times (in minutes) for data sets RTISP1.1 and RTISP1.2. It demonstrates the effectiveness of the rule f_0 proposed in [3] for solving the instance RTISP1.1. It also shows the new rule we propose f_3 is much more efficient on the instance RTISP1.2. The analysis of the 2009 data shows that vehicle workload is higher in the second part of the year than in the first. It should be noticed that rule f_0 only takes into account the ratio between the variable cost in the objective function and the number of tasks it would cover if it is selected. This rule may lead to the selection of columns which cover multiple times the same task, hence reducing the working capacity of the vehicle. The rule f_2 reduces this issue by taking the overlap of multiple columns into account in the priority weight. These remarks explain the differences between the two results.

Equilibrium cuts in the Benders master problem The redundant equilibrium cut (35) in the Benders master problem has proven to be very efficient to

	RTISP1.1			*RTISP1.2*		
	30	60	120	30	60	120
$flowCst_{-1}$	35	35	35	110	100	100
$flowCst_0$	35	35	35	120	110	100
$flowCst_{10e3}$	35	35	35	100	92	92
$flowCst_{10e4}$	35	35	35		120	120

Table 6: Impact of the flow conservation cut in the column generation procedure

accelerate the convergence of the algorithm. These constraints reduce the set of partial solutions which lead to infeasible subproblems hence limiting the number of generated cuts. Furthermore, they seem to improve the conditioning of the elementary shortest paths by reducing the number of non-dominated partial paths. Indeed, the dual information obtained from these equilibrium constraints can be interpreted as a deficit or surplus cost for the given node at the beginning of a shift. These weights highly impact the way the dynamic program selects the very first arcs of the solution by improving the efficiency of the dominance rule, hence speeding up the dynamic programming algorithm.

In order to highlight this behavior, a slack variable has been added to each flow conservation constraint. The penalty cost of each variable defines how much the constraint is enforced. Table 6 evaluates four variants, presented below:

1. $flowCst_{-1}$ No slack variable has been added.
2. $flowCst_0$ Penalty cost of slack variables is 0.
3. $flowCst_{10e3}$ Penalty cost of slack variables is 10000.
4. $flowCst_{10e4}$ Penalty cost of slack variables is 100000.

Table 6 provides the objective function values (in TKm) and the CPU times (in minutes) of the column generation procedure for the different variants on data sets RTISP1.1 and RTISP1.2.

The comparison of columns “30”, “60” and “120” for instance RTISP1.2 shows that adding the equilibrium constraint helps the column generation procedure a lot , leading to a solution with a smaller cost. It can also be noticed that the best continuous solution is found more quickly (in less than 30 minutes with the equilibrium constraint, in more than 100 minutes without the equilibrium constraint). Interestingly, a well chosen penalty price for the violation of the constraint (as the one used in the third case) can drastically improve the convergence and the quality of the solution.

Impact of Benders cuts In most of the above-presented results, the cuts proposed in Section 4.4 do not have a visible impact. More precisely, cuts are generated on columns with low quality that are unlikely to be selected. We have to recall that the cuts presented in Section 4.4 are feasibility cuts obtained by relaxations of the Benders sub-problem, as the combinatorial cuts could not easily be incorporated in the heuristic. Also, for data sets that are feasible or nearly feasible, this low impact could be explained. This is not true anymore for instance RTISP4.1 for which the heuristic performs less than 30% of the

	With cuts	Without cuts
Completion rate	28%	10%
Performance	13%	11%
CPU time (s)	1500	1500

Table 7: Impact of the Benders cuts of instance RTISP4.1

tasks. The comparison of the heuristic with and without the cuts proposed in Section 4.4 is displayed in Table 7. The results show the highly positive impact of the generated cuts which allow to increase the number of performed tasks by a factor three. After analyzing the results closely, we ntived that 15 shifts are assigned to vehicle 1 and 12 shifts are assigned to vehicle 2 when the cuts are applied while 20 shifts are assigned to vehicle 1 and only 6 shifts are assigned to vehicle 2 when no cuts are applied. Hence the Benders feasibility cuts avoid overloading the fastest vehicle.

7 Conclusion

An exact decomposition procedure for the RTISP has been proposed. This procedure is general enough to be used for others arc routing problemsand it has been used to design a matheuristic which is able to schedule real datasets.

The proposed solution is flexible enough to be adapted by the operator. Starting from the schedule given by the solver, the expert can adapt it to take into account the real speed of the vehicles, the precise date of the outages and most importantly the pathways. The industrialization of the proposed tool is currently in an evaluation phase at SNCF. In terms of lessons learned from a practical point of view, the method has shown, especially through the upper bound on the performance ratio, that maintenance and regional demands are limiting factors due to the limited availability of vehicles and should consequently be planned carefully. Indirectly, the generated data sets also show that performance ratio of the solutions proposed by the matheuristic is much higher for the larger problems considering the whole network and planning horizon, in comparison with restricted sets. This may seem counterintuitive at first sight, but planning on the whole network allows the algorithm to smooth the vehicle load over the entire horizon and significantly increase the proportion of inspections w.r.t. dead-heads. This fully justifies the need for an integrated planning of the primary track inspection that originated the project.

The use of the relaxed Benders subproblem may be of interest in other problems where the resolution of the Benders subproblem is the part which is consuming the most computation power. Furthermore, this approach can provide stronger cuts.

From the industrial point of view, the RTISP can be generalized to other railroad maintenance problems. The proposed approach is also usable to solve other type of maintenance problems.

However, the proposed method has limitations in the sense that the solutions proposed by the Benders master problem include subtours that needs introduction of extra deadhead costs in the repairing phase. Generating cuts to eliminate the subtours is a promising way of improving the performance gap.

References

[1] A. Amaya, A. Langevin, and M. Trépanier. The capacitated arc routing problem with refill points. *Operations Research Letters*, 35(1):45–53, 2007.

[2] S. Raghavan B.L. Golden and E.A. Wasil, editors. *The vehicle routing problem: latest advances and new challenges*. Springer US, 2008.

[3] V. Chvàtal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[4] G. Codato and M. Fischetti. Combinatorial Benders' cuts. In D. Bienstock and G. Nemhauser, editors, *Integer Programming and Combinatorial Optimization (IPCO X)*, volume 3064 of *Lecture Notes in Computer Science*, pages 178–195. Springer, 2004.

[5] M. Dror, editor. *Arc routing: theory, solutions and applications*. Springer, 2000.

[6] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part I: the chinese postman problem. *Operations Research*, 43(2):231–242, 1995.

[7] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part II: the rural postman problem. *Operations Research*, 43(3):399–414, 1995.

[8] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An exact algorithm for the elementary shortest path problem with resource constraints: application to some vehicle routing problems. *Networks*, 44(3):216–229, 2004.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addisson Wiley, 1994.

[10] G. Ghiani, G. Improta, and G. Laporte. The capacitated arc routing problem with intermediate facilities. *Networks*, 37(3):134–143, 2001.

[11] B.L. Golden and R.T. Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.

[12] G. Hasle and O. Kloster. Industrial vehicle routing. In G. Hasle, K.-A. Lie, and E. Quak, editors, *Geometric modelling, numerical simulation, and optimization*, pages 397–435. Springer, 2007.

[13] S. Irnich. Solution of real-world postman problems. *European Journal of Operational Research*, 190(1):52 – 67, 2008.

[14] E.L. Johnson and S. Wøhlk. Solving the capacitated arc routing problem with time windows using column generation. CORAL Working Papers L-2008-09, University of Aarhus, Aarhus School of Business, Department of Business Studies, January 2009.

[15] B. Kim, S. Kim, and S. Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33:3624–3642, 2006.

[16] S. Lannez. *Optimisation des tournées d'inspection des voies ferroviaires*. PhD thesis, INSA Toulouse, 2010. LAAS report 10893.

[17] V. Maniezzo, T. Stutzle, and S. Voss, editors. *Matheuristics: hybridizing metaheuristics and mathematical programming*. Springer-Verlag, 2009.

[18] N. Perrier, A. Langevin, and J.F. Campbell. A survey of models and algorithms for winter road maintenance. Part I: system design for spreading and plowing. *Computers & Operations Research*, 33:209–238, 2006.

[19] N. Perrier, A. Langevin, and J.F. Campbell. A survey of models and algorithms for winter road maintenance. Part II: system design for snow disposal. *Computers & Operations Research*, 33:239–262, 2006.

[20] N. Perrier, A. Langevin, and J.F. Campbell. A survey of models and algorithms for winter road maintenance. Part III: vehicle routing and depot location for spreading. *Computers & Operations Research*, 34:211–257, 2007.

[21] N. Perrier, A. Langevin, and J.F. Campbell. A survey of models and algorithms for winter road maintenance. Part IV: vehicle routing and fleet sizing for plowing and snow disposal. *Computers & Operations Research*, 33:239–262, 2007.

[22] N. Perrier, A. Langevin, and C.A. Amaya. Vehicle routing for urban snow plowing operations. *Transportation Science*, 42:44–56, 2008.

[23] M. Reghioui, C. Prins, and N. Labadi. GRASP with path relinking for the capacitated arc routing problem with time windows. In *Proceedings of the 2007 EvoWorkshops 2007 on EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog*, pages 722–731, Berlin, Heidelberg, 2007. Springer-Verlag.

[24] S. Spoorendonk and G. Desaulnier. Clique inequalities applied to the vehicle routing problem with time windows. *INFOR: Information Systems and Operational Research*, 48(1):53–67, 2010.