Column generation heuristic for a rich arc routing problem

Sébastien Lannez, Christian Artigues, Jean Damay, Michel Gendreau

To cite this version:

Sébastien Lannez, Christian Artigues, Jean Damay, Michel Gendreau. Column generation heuristic for a rich arc routing problem. 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10), Sep 2010, Liverpool, United Kingdom. pp.130–141, ⟨10.4230/OASIcs.ATMOS.2010.130⟩. ⟨hal-00560749⟩

HAL Id: hal-00560749

https://hal.science/hal-00560749v1

Submitted on 29 Jan 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

Column generation heuristic for a rich arc routing problem

Sébastien Lannez[1,2,3], Christian Artigues[2,3],
Jean Damay[1], 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
sebastien@lannez.fr, artigues@laas.fr
jean.damay@sncf.fr, michel.gendreau@cirrelt.ca

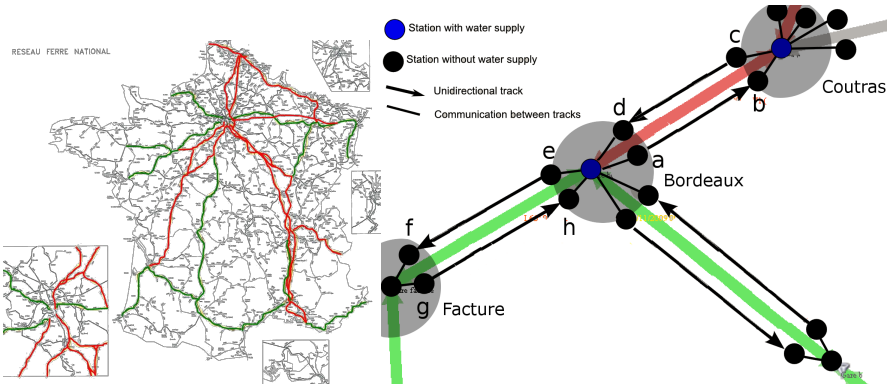
Abstract

In this paper we address a real world optimisation problem, the Rail Track Inspection Scheduling Problem (RTISP). This problem consists of scheduling network inspection tasks. The objective is to minimise total deadhead distance. A mixed integer formulation of the problem is presented. A column generation based algorithm is proposed to solve this rich arc routing problem. Its performance is analysed by benchmarking a real world dataset from the French national railway company (SNCF). The efficiency of the algorithm is compared to an enhanced greedy algorithm. Its ability to schedule one year of inspection tasks on a sparse graph with thousand nodes, arcs and edges is assessed.

arc routing, column generation, heuristic, 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, as the basic part of a track may result in serious accidents. Réseau Ferré Français (RFF), the French railway infrastructure manager, have delegated some railway maintenances 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 these maintenances is to prevent tracks failures. In order to quickly inspects the French network,



(a) Map of primary tracks (b) Graph model of railtracks around Bordeaux

Figure 1: French railway network model

SNCF is using ultrasonic defectoscopy to detect and survey imperfections in rails. Inspection frequencies increase with speed and cumulated train weight.

Inspection frequencies range from six months to twenty years. 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*) are performed by local logistic departments. A map representing these tracks is presented in figure 1a. A schematic zoom around Bordeaux is shown in figure 1b. Ultrasonic inspections are performed with three specialised rolling stock units, thereafter called vehicles. Their maximum speed and working capacity are different. The detection of defects inside the track is performed by reverberation analysis of the ultrasonic waves passing through the rails. These vehicles can move during at most six hours per day. This limitation is due to maximum shift duration and maximum daily inspection distance. This distance is limited by the water tank capacity needed to keep sensors and rails coupled during measure. These tanks can only be refilled at special stations. Over the 200 stations on the *primary tracks*, 90 are equipped with water supply. For organisational purposes, vehicle's moves are geographically constrained and their maintenances should be performed periodically.

The problem SNCF is dealing with is to visit a given set of tracks taking into account some operational constraints. Tracks outages can alter vehicle's speed or prevent them from circulating during certain days. Vehicle's speed depends whether it is inspecting or deadheading: during deadhead trips speed can be more than three times faster than when inspecting. Vehicle's daily inspection capacity is limited by the total amount of water which can be brought on board. Water tank refill is time consuming and needs rarely available operators at the station. Hence, it is not desired to do more than one refill per shift. The main cost indicator is a common logistic performance ratio based on the quantity of tracks inspected per year divided by the total traveled distance in a year.

The total quantity of *primary tracks* to be inspected every year is constant, so minimising this performance ratio reduces to the minimisation of the total deadhead distance.

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 survey from [17–20]. It involves complicating constraints, namely shift limited duration, water supply, track outages and heterogeneous fleets. Another difficulty is the network size which makes it a real challenge to solve.

2 Literature review

2.1 Industrial arc routing problems

In [13], the authors notified that industrial vehicle routing problems are rich: models are generalisations of lots of academic ones, and input data dimension can be huge.

Road related problems have supplied researchers with a lot of arc routing problems. A review of problems arising during winter road maintenances has been published in the articles [17–21]. They also present industrial applications. Waste collection or postal deliveries are also an active field from 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 [14].

2.2 Arc routing problems

In this section, some arc routing problems and their applications are presented. For a more complete catalog of them, a good introduction might be the books [6] and [4] and the survey articles [7, 8].

One problem the RTISP is related with is the capacitated arc routing problem (CARP), described in [12]. It consists in visiting a set of arcs with a single vehicle. Each visited arc reduces by a given amount the remaining working capacity of the vehicle. In the RTISP, tasks and deadheads circulation 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 belong to a set of periods. Paper [9] contains a description of a column generation procedure. In [22], a procedure to solve this problem with a greedy randomised adaptive search procedure (GRASP) associated with path relinking is described. Another extension is the capacitated arc routing problem with refill points (CARP-RP) presented in [2], also called the capacitated arc routing problem with intermediate facilities (CARP-IF) in [11]. It extends the CARP by adding refill facilities to certain nodes.

We have not been aware of published work about methods for solving a problem having all these features. However, this problem, which can be called

multi-capacitated arc routing problem with time windows, refill points and heterogeneous fleet (H-MCARP-RP-TW), is suitable for the description of the RTISP and also of use for others transportation problems.

3 Assumptions and models

3.1 Hypothesis

Vehicle moves are modeled with arcs and edges. They represent either inspection tasks and deadhead traversals of track portions or complex moves like unit switch back or station traversal. Arcs are suitable for the description of unidirectional railway tracks whereas edges are for bidirectional railway tracks. Nodes describe stations, communication between railway tracks, or locations in the network where the vehicles can change their circulation mode. Only *primary tracks* are directly modeled.

For the schedule to be easily adapted during operations, multiple shifts per day are not taken into account. Each shift consists of a trip between two refill stations with a total distance to inspect smaller than the capacity of the water tank and a total trip duration smaller than the duration of a work shift. Given all the feasible shift pattern paths, the RTISP becomes the problem of selecting and scheduling them in order to satisfy all inspections at the lowest cost.

3.2 Graph and vehicle representation

A multigraph $G = (V, A)$ containing arcs and edges (A) and nodes (V) models the railway network. Arcs and edges can represent tasks ($\bar{A}$), deadhead traversal ($\tilde{A}$) or wait ($\hat{A}$). Nodes can represent rest and refill stations ($\bar{V}$), or communications between railtracks and measure interruption possibility ($\tilde{V}$). The corresponding arcs describe the set ($\underline{A}$). All these sets are indexed by k when they are related to the subnetwork which can be inspected by vehicle $k \in K$. The parameter l_a is the length in kilometers of arc a. The parameter d_{ak} is the traversal duration of arc a for vehicle k. The parameter w_k is the working capacity, in kilometers, of the vehicle k. Loop arcs ($\hat{A}$) represent dead shifts and have a traversal duration of one shift.

3.3 Calendar

The calendar H is assumed to not contains any non working day. It is composed of integer values representing number of "shift seconds" since the first period of the planning horizon. The need for a small timeslot comes from the wide range of task duration and the relatively high speed of vehicles. t is a timeslot in H, s the duration of a shift and p the first period of the calendar. The subset $D \subseteq H$ contains the first "shift seconds" of each shift. The subset $\bar{H}_{a,k} \subseteq H$ contains the set of periods during which vehicle k can not traverse arc a.

3.4 Mathematical model

The binary integer program presented in this section is used to better clarify the mathematical representation of the RTISP. The model $\mathcal{M}$ contains an exponential number of variables, each one representing a feasible shift pattern.

Objective function Minimise total deadhead: the cost of an arc is the length of the arc if this arc is a deadhead one. No cost is imputed for other arcs.

$$c_a = \begin{cases} l_a, \text{if } a \in \tilde{A}, \\ 0, \text{else.} \end{cases} \tag{1}$$

Shift flow model - ($\mathcal{M}$) Given the complete set of feasible trips between two refill stations, vehicles circulation can be modeled as a flow on a multicommodity network, each arc representing feasible daily trips.

The set Q contains all shift patterns. The subset Q^k contains all shift patterns valid for vehicle k. Each shift pattern q is associated to a path between two nodes having refill facilities. Let $\mathcal{P}_q$ denote the sequence containing the visited arcs in their visiting order.

Let H_q denote the set of periods during which the shift q can start. Let s be the duration of a shift. Let z_q^t equal one if shift pattern q is performed during calendar day t. Let A_{aq} be a parameter which equal one if arc a is inspected during shift pattern q. Let S_{aq} and E_{aq} be parameters which equals one if arc a is respectively the first and the last of the shift pattern q.

Let $\delta^+(v)$ and $\delta^-(v)$ denote the set of outgoing arcs and the set of ingoing arcs of node v.

The cost of a shift pattern is defined as follows:

$$c_q = \sum_{a \in \mathcal{P}_q} c_a, \ \forall k \in K, q \in Q^k. \tag{2}$$

The mixed integer program is as follows:

$$\text{minimise} \sum_{q \in Q} \sum_{t \in D} c_q z_q^t \tag{3}$$

$$\text{subject to}$$

$$\sum_{t \in D} \sum_{q \in Q} A_{aq} z_q^t \geq 1, \qquad \forall a \in \bar{A} \tag{4}$$

$$\sum_{q \in Q^k} \sum_{a \in \delta^+(v)}^{a \in \underline{A}} S_{aq} z_q^{t+s} - \sum_{a \in \delta^-(v)}^{a \in \underline{A}} E_{aq} z_q^t = 0, \qquad \forall v \in \bar{V}, k \in K, t \in D \tag{5}$$

$$\sum_{q \in Q^k} z_q^t \leq 1, \qquad \forall k \in K, t \in D \tag{6}$$

$$z_q^t = 0, \qquad \forall t \notin H_q \tag{7}$$

$$z_q^t \in \{0, 1\}, \qquad \forall t \in D, q \in Q \tag{8}$$

The objective function (3) ensures that from all feasible solutions the one with minimum total deadhead will be selected. Constraints (4) ensure that the set of selected shift permits to perform all inspection tasks. Constraints (5) ensure for each vehicle that two consecutive shifts end and start at the same node. Constraints (6) enforce for each vehicle the assignment of at most one shift per calendar day. Constraints (7) ensure that shift are scheduled during valid periods. Constraints (8) ensure that solutions are integer.

4 Optimal algorithm

4.1 Reformulation

The mathematical program presented above contains an exponential number of columns. The equivalent mathematical program presented below is used to highlight the decision corresponding to the selection of the shift patterns to be scheduled in ordre to perform every inspection tasks in time. This reformulation uses the variable y_q which takes the value 1 if the shift pattern q is to be included in the schedule.

$$\text{minimise} \sum_{q \in Q} c_q y_q \tag{9}$$

$$\text{subject to}$$

$$\sum_{q \in Q} A_{aq} y_q \geq 1, \qquad \forall a \in \bar{A} \tag{10}$$

$$y_q - \sum_{t \in D} z_q^t = 0, \qquad \forall q \in Q \tag{11}$$

$$\sum_{q \in Q^k} \sum_{a \in \delta^+(v)}^{a \in \underline{A}} S_{aq} z_q^{t+s} - \sum_{a \in \delta^-(v)}^{a \in \underline{A}} E_{aq} z_q^t = 0, \qquad \forall v \in \bar{V}, k \in K, t \in D(5) \tag{12}$$

$$\sum_{q \in Q^k} z_q^t \leq 1, \qquad \forall k \in K, t \in D(6) \tag{13}$$

$$z_q^t = 0, \qquad \forall t \notin H_q(7) \tag{14}$$

$$y_q \in \{0,1\}, \qquad \forall q \in Q \tag{15}$$

$$y_q, z_q^t \in \{0,1\}, \qquad \forall t \in D, q \in Q(8) \tag{16}$$

4.2 Combinatorial Benders decomposition

The way operators are actually scheduling the inspection tasks seems to show that inspection tasks are correlated in time and space. Furthermore, the analysis of the input data shows that a great number of tasks which are near in space have overlapping time windows. This fact is used to efficiently solves the problem by

applying a Benders decomposition algorithm. This algorithm first consider the problem of selecting a subset of shit pattern which covers every tasks and which can be hopefully scheduled to create a feasible trip. At each iteration, a master problem is solved to determine this set of shift. Given this potential solution a set of subproblem are solved to determine if the candidate solution is feasiuble for the whole problem. If it is, this solution is optimalm, otherwise a combinatorial cut is generated and added to the master problem. The simplest implementation uses the following mathematical programs, respectively as master problem and subproblem.

The Benders master problem, which is a set covering problem whith some additional constraints, can be written has follows:

$$(MASTER) \tag{17}$$

$$\text{minimise} \sum_{q \in Q} c_q y_q (3) \tag{18}$$

$$\text{subject to}$$

$$\sum_{q \in Q} A_{aq} y_q \geq 1, \qquad \forall a \in \bar{A} (4) \tag{19}$$

$$\sum_{q|\bar{y}_q=1} (1 - y_q) + \sum_{q|\bar{y}_q=0} y_q \geq 1, \qquad \forall \bar{y}_q \in \mathcal{I} \tag{20}$$

$$y_q \in \{0, 1\}, \qquad \forall q \in Q \tag{21}$$

Let $\bar{y}_q$ be the current solution found by solving the master problem. The subproblems, which correspond for each vehicle to deciding if the partial solution can be scheduled as a tour, can be written has follows:

$$(SUB_k) \tag{22}$$

$$\text{minimise} 0 \tag{23}$$

$$\sum_{t \in D} z_q^t = \bar{y}_q, \qquad \forall q \in Q^k \tag{24}$$

$$\sum_{q \in Q^k} \sum_{a \in \delta^+(v)}^{a \in \underline{A}} S_{aq} z_q^{t+s} - \sum_{a \in \delta^-(v)}^{a \in \underline{A}} E_{aq} z_q^t = 0, \qquad \forall v \in \bar{V}, t \in D (5) \tag{25}$$

$$\sum_{q \in Q^k} z_q^t \leq 1, \qquad \forall t \in D (6) \tag{26}$$

$$z_q^t = 0, \qquad \forall t \notin H_q, q \in Q^k (7) \tag{27}$$

$$z_q^t \in \{0, 1\}, \qquad \forall q \in Q^k, q \in Q^k \tag{28}$$

$$\tag{29}$$

This decomposition has two main drawbacks. In the master problem, the infomration about how the tasks can be sequenced is missing. This can force the algorithm to generate a lot of combinatoiral Benders cut solely for satisying the

flow constraint. Secondly, the subproblems are special instances of the travelling salesman problem. Because this problem is known to be NP-hard to solve, they might be difficult to solve.

For these two reaons we proposed to incorporate in the master problem an aggegrated version of the conservation flow constraint defined in the time-space graphe. Secondly, we proposed not to solve every subproblems to optimality, but to try to solve relaxations instead. The idea is that since these relaxations are fastest, they can be used to detect quickly a set of interesting infeasibility cuts.

This enhanced master problem is presented above:

$$(MASTER') \tag{30}$$

$$\text{minimise} \sum_{q \in Q} c_q y_q (3) \tag{31}$$

$$\text{subject to}$$

$$\sum_{q \in Q} A_{aq} y_q \geq 1, \qquad \forall a \in \bar{A} (4) \tag{32}$$

$$\sum_{q \in Q^k} \sum_{a \in \delta^+(v)}^{a \in \underline{A}} S_{aq} y_q - \sum_{a \in \delta^-(v)}^{a \in \underline{A}} E_{aq} y_q = 0, \qquad \forall v \in \bar{V} \tag{33}$$

$$\sum_{q | \bar{y}_q = 1} (1 - y_q) + \sum_{q | \bar{y}_q = 0} y_q \geq 1, \qquad \forall \bar{y}_q \in \mathcal{I} \tag{34}$$

$$y_q \in \{0, 1\}, \qquad \forall q \in Q (21) \tag{35}$$

In the first relxation subproblem the time constraints are removed. The problem to solve becomes a travelling salesman problem without time windows. In the second relaxation subproblem, the space constraint is removed resulting in a simple assignement problem. A solution which is infeasible for one of these relaxation is surely infeasible for the original subproblem. Furthermore, the cut obtained from theses reformulations are stronger than the one generated after solving the original problem.

These two formulations are respectively has follows:

$$(SUB_k(assign)) \tag{36}$$

$$\text{minimise} 0 \tag{37}$$

$$\sum_{t \in D} z_q^t = \bar{y}_q, \qquad \forall q \in Q^k (44) \tag{38}$$

$$\sum_{q \in Q^k} z_q^t \leq 1, \qquad \forall t \in D (6) \tag{39}$$

$$z_q^t = 0, \qquad \forall t \notin H_q, q \in Q^k (7) \tag{40}$$

$$z_q^t \in \{0, 1\}, \qquad \forall q \in Q^k, q \in Q^k \tag{41}$$

$$(SUB_k(flow)) \tag{42}$$

$$\text{minimise0} \tag{43}$$

$$\sum_{t \in D} z_q^t = \bar{y}_q, \qquad \forall q \in Q^k \tag{44}$$

$$\sum_{q \in Q^k} \sum_{a \in \delta^+(v)}^{a \in \underline{A}} S_{aq} z_q^{t+s} - \sum_{a \in \delta^-(v)}^{a \in \underline{A}} E_{aq} z_q^t = 0, \qquad \forall v \in \bar{V}, t \in D(5) \tag{45}$$

$$z_q^t \in \{0,1\}, \qquad \forall q \in Q^k, q \in Q^k \tag{46}$$

Is should be noticed that the subproblem corresponding to an assignement problem is a linear program. This ensure that the obtained cut can be generated using the dual information like in the original Benders decomposition scheme.

5 Heuristical cut and column generation

The great number of columns in the mathematical problem does not permit us to generate them a priori. Furthermore, the use of an exact column generation scheme is not possible due to the presence of combinatorial Benders cuts which have no special structure. In order to speed up the resolution we designed an heuristic based on column and cut generation.

First of all, the algorithm determines a continuous solution to the Benders master problem by solving its linear relaxation with a simplex method. This solution is then used to warm start an heuristic inspired by a greedy algorithm proposed by [5]. It is this solution which is checked against feasibility in the different subproblems. If it is infeasible, a combinatorial benders cut is added to the master problem. Otherwise the current solution is returned as the best.

6 Column generation based heuristic - *AlgoColGen*

6.1 Overall view

The proposed algorithm is based on a mathematical decomposition which is heuristically solved in three steps. The first one is used to aggregate simple tasks into work shifts with the use of a column generation algorithm applied to the model $\mathcal{RM}$. It generates a continuous solution to the problem. In the second step, a rounding greedy heuristic is used to get an integer solution. This new integer candidate solution is tested against calendar day assignment to check if it is feasible according to task cover constraints (6). If it is not, a local pseudo cut is generated. If it is, the new candidate solution is used to generate a constraint program for the third stage. This last stage is used to check the feasibility of the set of work shifts. If this test fails, a local pseudo cut which can be added to

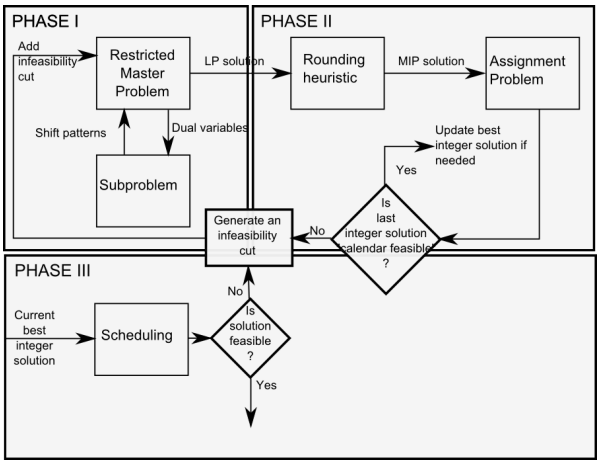


Figure 2: Scheme of the decomposition algorithm

$\mathcal{RM}$ is generated. Otherwise, a solution with minimum total deadhead traversal distance is approximated. The general scheme of this algorithm is given in the figure 2.

6.2 Stage 1 - Column generation

The master problem is a set covering problem (SCP) with additional constraints. The subproblems are to find elementary shortest paths between two refill stations with resource constraints.

Master problem - ($\mathcal{RM}$) The master problem of the column generation is the mathematical model $\mathcal{RM}$. It is a linear program solved by the simplex algorithm.

Subproblems - ($\mathcal{SP}$) Shift patterns are generated by solving elementary shortest path problems with two resource constraints (water, shift duration). The implemented procedure is a label setting algorithm inspired by the algorithm presented in [10].

The dual variable λ_a is associated to each constraint (**??**). The dual variable $\mu_{a\bar{k}}$ is associated to each constraint (**??**). The parameter $\bar{k}$ is the index of the vehicle for which a shortest path is to be computed and $\bar{t}$ is the first period of a shift.

$SP(\lambda, \mu, \bar{k}, \bar{t}) =$

$$\text{minimise} \sum_{a \in A} \sum_{t=\bar{t}}^{t \leq \bar{t}+s} (c_a \lambda_a + \mu^t_{a\bar{k}}) x^t_{a\bar{k}} \tag{47}$$

subject to

$$\sum_{a \in \bar{A}} \sum_{t=\bar{t}}^{\bar{t}+s} l_{ak} x^t_{a\bar{k}} \leq w_{\bar{k}}, \tag{48}$$

$$\sum_{a \in \delta^-(v)} x^{t-d_{a\bar{k}}}_{a\bar{k}} - \sum_{a \in \delta^+(v)} x^t_{a\bar{k}} = 0, \qquad t \in H, \bar{t} \leq t \leq \bar{t}+s \tag{49}$$

$$x^t_{ak} = 0 \qquad a \in A, t \in \bar{H}_{a,k} \tag{50}$$

$$x^t_{ak} \in \{0,1\}, \qquad a \in A, t \in H \tag{51}$$

Constraints (48) enforce length of shortest paths to not exceed vehicle's water capacity. Constraints (49) ensure flow conservation. Constraints (50) ensure that no arc are traversed during outages. Finally, constraints (51) ensure that vehicle can only move on the graph.

It should be noticed that it would be intractable to compute, at each column generation iteration, $K \cdot D$ constrained shortest paths. Our implementation enables finding valid shortest paths for multiple calendar days. It can be parameterised to generate from K to $K \cdot D$ subproblems. At the first extreme, the feasible solution space of each of the K subproblems is large. Solving one of them is very time consuming. In the other extreme, the feasible solution space of each of the $K \cdot D$ subproblems is narrow and solving one of them is fast.

6.3 Stage 2 - Early feasibility test

The column generation model becomes quickly degenerated with a lot of columns and few constraints. Getting an integer solution from $\mathcal{RM}$ with a general purpose branch-and-bound-and-cut is not a realistic choice because the solution space is far too wide. In order to quickly get an integer feasible solution, a rounding heuristic, named *AlgoGreedyCover*, inspired by the greedy algorithm proposed by Chvàtal [5] is applied to the set covering problem. This heuristic selects columns to be rounded up by computing a ratio of the column cost and the number of times it appears in the rows. It ensures the satisfaction of cover constraints (4). The selected shifts are tested against calendar day assignment with a max flow problem. The optimal flow gives an upper bound on the maximum number of shifts which can be scheduled. If it is lower than the number of shifts, a local pseudo cut is generated and added to the master problem, see Section **??**. Otherwise, a new candidate solution has been found. It is saved and will be tested against feasibility for model $\mathcal{M}$ in the third stage. It should be noticed that the rounding heuristic and the max flow algorithm are both polynomial algorithms [1].

Rounding heuristic - *AlgoGreedyCover* The rounding heuristic consists in computing, for each fractional variable, the ratio between the objective function coefficient and the number of times it appears in the rows. The value of the variable with lowest ratio is rounded up and removed from the list of selectable columns ($\bar{Q}$). The related cover constraints are marked as satisfied. Each variable which is selectable and for which every cover constraints are marked as satisfied is removed from $\bar{Q}$ and its value set to zero. The ratio of each column is updated and the algorithm iterates until all cover constraints are satisfied or no variable is selectable.

Calendar day assignment - ($\mathcal{M}^{\mathbf{cal}}$) The problem of flow maximisation in a graph is used for modeling possible assignment of shifts to calendar days. This problem has been proved to be polynomially solvable, [1].

Let B_{qt} be a parameter which equals one if shift q can be assigned to calendar day t. Let $\bar{Q}^k$ be the set of selected shift patterns of vehicle k. Let y_{qt} be a binary variable which equals one if shift pattern q is assigned to calendar day t.

The assignment problem is defined as follows: $\mathcal{M}^{\text{cal}}(k) =$

$$\text{maximise} \sum_{t \in D} \sum_{q \in \bar{Q}^k} B_{qt} y_{qt} \tag{52}$$

$$\text{subject to}$$

$$\sum_{t \in D} B_{qt} y_{qt} \leq 1 \qquad \forall q \in \bar{Q}^k \tag{53}$$

$$\sum_{q \in \bar{Q}^k} B_{qt} y_{qt} \leq 1 \qquad \forall t \in D \tag{54}$$

$$y_{qt} \in \{0,1\} \qquad \forall t \in D, q \in \bar{Q}^k \tag{55}$$

The objective function (52) ensures that among all feasible solutions, the one maximising the number of assigned shifts is to be chosen. Constraints (53) ensures that a shift pattern can be assigned to at most one calendar day. Constraints (54) ensures that a calendar day can not be assigned to more than one shift.

6.4 Stage 3 - Complete feasibility test

The above-described rounding heuristic does not take into account tasks sequencing constraints. Solutions found during stage 2 can still violate the flow conservation constraints between shifts (5). To overcome this situation, an extension of a Traveling Salesman Problem with Time Windows [3] is used to construct a feasible solution from the task groups selection of these solutions. This problem models at a macroscopic level the RTISP with a period duration of one shift. A list algorithm is presented to solve it. Each node of the TSP graph represents a shift pattern. For each node, a list of time windows, during which vehicles can go through, are defined. Arcs between nodes represents end of day deadhead moves. Their cost is the total distance between the end of the

shift and the start of the next shift. The duration needed to traverse each arc depends on the shift pattern duration and the vehicle deadhead speed.

This problem is solved with a heuristic named *AlgoSchedList*, based on a constraint propagation and list scheduling. It relies on depth first search without backtracking. Indeed, due to computational difficulty, we replace backtracking by a *guided multi start* framework. At the end of each search, each decision taken during the search (branch selection) is priced. These prices are used to update a transition cost matrix. Once matrix cost is fully updated, the search is restarted. The pricing mechanism is inspired by the Vickrey-Clarke-Groves mechanism, a well known externality measure described in [16]. At first, these prices are initialised with travel distance between tasks. They are further estimated after each *AlgoSchedList* run.

7 Enhanced greedy heuristic - *AlgoGreedy*

In order to evaluate our column generation heuristic, we additionally designed a greedy algorithm to solve the complete RTISP based on the dynamic programming method described in 6. Starting from a node, a period and a vehicle, a shortest path satisfying resource constraints is computed. The shortest path is appended to the schedule of the current vehicle. We let the vehicle go forward until the end of the schedule horizon is reached. Then, we continue with the next vehicle. If no task is reachable from the current node, then a deadhead move is performed to find the nearest node which enables performing a task.

In order for the tasks to be selected during shortest path computation, different weight update rules have been tested. The one used in this paper uses information about task time windows and tasks duration.

Let w_a^k denote the cost of performing task i 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 the earliest and latest start of task a. M should have a value such that the algorithm will always prefer performing a task rather than deadheading.

8 Computational tests

8.1 Real dataset

The test dataset contains mainly two distinct parts which are static data and dynamic data. Static data contains the network representation of the railway network and the vehicles outages which are likely to rarely change during the life cycle of the decision tool. Dynamic data contains the tasks time windows and tracks outages which can be updated at most every months. For the purpose of this article we present a dataset based on information acquired for 2009.

The infrastructure graph has 1000 arcs, 500 edges and 760 nodes. A total of 500 tasks must be performed per year. The generated graph has 1600 arcs, 500 edges and 770 nodes of which 90 are refill stations. Task time windows have a fixed size of 28 days and duration ranging from few minutes to six hours. The duration of a shift is fixed to seven hours. The horizon used for shortest paths computation is one month, which yields 12 subproblems per vehicle.

8.2 Computational tests

Based on the real dataset we derived three scenarios. In the first one, named *no outage*, we removed all track outages. In the second one, *small outages*, we divided by 10 the duration of each outage. The last one, named *full outages*, is the real dataset provided by the company. For each algorithm and each scenario, we show the task completion rate (r) and the performance ratio (p).

The performance ratio (p) is calculated to reflect the rate between the total inspected distance (d_i) and the total deadhead (d_d) moves :

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

The task completion rate (r) is used for getting information about the hardness of the instance. The real dataset is actually in constant evolution and is known not to be feasible. In fact, the information about whether outages can be traversed or not is not yet available. To cope with this situation, a slack variable with a prohibitive cost is added to each covering constraints of the model.

In the table in figure 3, it can be seen that the column generation heuristic outperforms the greedy algorithm in terms of task coverage and solution quality. In the table in figure 4, it can be seen that the performance of the column generation algorithm seems to be better when time windows are tight.

	No outages		small outages		full outages	
	r	p	r	p	r	p
AlgoGreedy	100%	18.82%	27%	9.77%	23%	9.06%
AlgoColGen	100%	30.50%	37%	25.54%	31%	22%

Figure 3: Task coverage and solution quality

	No outages	small outages	full outages
	t	t	t
AlgoGreedy	47	767	539
AlgoColGen	3434	180	61

Figure 4: Computation time (in seconds)

9 Conclusion

In this paper, a railway maintenance routing problem and a mixed integer formulation is presented. An original column generation heuristic is proposed to solve it. Cut generators based on model relaxation resolution are proposed and implemented. A comparison between this heuristic and an enhanced greedy algorithm is presented. The numerical tests show that the column generation heuristic performs better than the greedy heuristic. Furthermore, it highlights the difficulty for the greedy algorithm to tackle dataset with highly constrained time windows. The difficulty to perform all tasks is due to the presented dataset. It is an extreme situation in which it is forbidden to traverse during every outages and the minimum outage duration is one day.

This work on train units for ultrasonic inspection can be extended to other maintenance train units which also have a limited capacity. An extensive study of the pseudo local cut impact is also of interest.

A TSPTW - ($\mathcal{M}^{tsp}$)

Let $\bar{Q}$ be the set of selected task groups found during stage 2. Let $c^k_{qq'}$ be the shortest path distance between the last visited node of shift q and the first visited node of shift q'. Let $d^k_{qq'}$ be the shortest path duration between the end of q and the begin of q'. Let d^k_q be the duration of the shift pattern q if performed by vehicle k. Let $H\ P_q$ be the set of periods during which the shift pattern q can be started. Let K_q be the set of vehicles allowed to perform shift pattern q. Let s_q (e_q) be a variable containing the start (end) time of shift pattern q. Let v_q be the vehicle performing shift pattern q. Let p_q be a variable which contains the position of shift pattern q in the sequence of vehicle v_q. The artificial shift $\mathcal{E}$ models the end of the schedule.

Vehicle maintenances rendezvous are modeled by tasks with fixed start time and only one assignable vehicle.

$\mathcal{M}^{tsp} =$

$$\text{minimise } z_{\mathcal{M}^{tsp}} = \sum_{q\in\bar{Q}}\sum_{q'\in\bar{Q}} x^k_{qq'}c^k_{qq'} \tag{56}$$

subject to:

$$\sum_{k\in K} y^k_q = 1 \qquad \forall q \in \bar{Q} \tag{57}$$

$$\sum_{q'\in\bar{Q}} x^k_{qq'} - y^k_q = 0 \qquad \forall k \in K, q \in \bar{Q}^k \tag{58}$$

$$\sum_{q'\in\bar{Q}} x^k_{q'q} - \sum_{q'\in\bar{Q}} x^k_{qq'} = 0 \qquad \forall k \in K, q \in \bar{Q} \tag{59}$$

$$s_q + \sum_{q'\in\bar{Q}} x^k_{qq'}(d^k_q + d^k_{qq'}) - e_q = 0 \qquad \forall k \in K, (qq') \in \bar{Q}^2 \tag{60}$$

$$e_q - s_{q'} - \mathbb{M}(1 - x^k_{qq'}) \leq 0 \qquad \forall k \in K, (qq') \in \bar{Q}^2 \tag{61}$$

$$s_q + \mathbb{M}(1 - y^k_q) \geq D \text{ or } e_q - \mathbb{M}(1 - y^k_q) \leq S \qquad \forall k \in K, q \in \bar{Q}, [S, D] \in \bar{H}_k \tag{62}$$

$$y^k_q \in \{0, 1\} \qquad \forall k \in K, q \in \bar{Q} \tag{63}$$

$$x^k_{q,q'} \in \{0, 1\} \qquad \forall k \in K, (q, q') \in \bar{Q}^2 \tag{64}$$

$$s_q \in H_q, e_q \geq 0 \qquad \forall q \in \bar{Q} \tag{65}$$

The objective function (56) ensures that among all feasible solutions the one which minimizes total deadhead distance is selected. Constraints (57) ensure that all shift are performed. Constraints (58) ensure that each shift is assigned a vehicle. Flow conservation constraints (59) ensure task sequence continuity for each vehicle. Constraints (60) ensure that shift pattern end time is equal to shift pattern start time plus shift pattern duration and transition duration to next shift pattern. Constraints (61) ensure that on a given vehicle no task can start before the end of the previous one. Constraints (62) ensure that task can not start during vehicle outages.

B The list scheduling algorithm: *AlgoSchedList*

Algorithm 1: The list algorithm : *AlgoSchedList*

Input: s_q
Output: $\bar{z}$,$\mathcal{S}$
begin
  $\mathcal{L} \leftarrow \bar{Q}$; /* Set of unscheduled tasks */
  $\mathcal{C} \leftarrow \{(k, 0, 0, \Delta) | k \in K\}$; /* Set of latest vehicle task */
  while $\mathcal{L} \neq \emptyset$ **do**
    $(\bar{k}, \bar{s}, \bar{e}, \bar{q}) = \arg\min_{(k,s,e,q)\in\mathcal{C}} (e)$; /* Select a 'last task' */
    $\mathcal{C} \leftarrow \mathcal{C} - (\bar{k}, \bar{s}, \bar{e}, \bar{q})$; /* Remove candidate */
    $(\bar{\bar{s}}, \bar{\bar{c}}, \bar{\bar{q}}) = (\infty, \infty, \emptyset)$; /* Reset for loop */
    for $q \in \mathcal{L}$ **do**
      if $\bar{e} + d^{\bar{k}}_{\bar{q}q} < \max(s_q)$ **then**
        $t = \max(\bar{e} + d^{\bar{k}}_{\bar{q}q}, \min(s_q))$; /* Task earliest start time */
        if $t \leq \bar{\bar{s}}$ *and* $c^{\bar{k}}_{\bar{q}q} < \bar{\bar{c}}$ **then**
          $(\bar{\bar{s}}, \bar{\bar{c}}, \bar{\bar{q}}) = (t, c^{\bar{k}}_{\bar{q}q}, q)$; /* Set current best */
    if $\bar{\bar{q}} = \emptyset$ **then**
      if $\mathcal{C} = \emptyset$ **then**
        Stop: infeasible
    else
      $\mathcal{L} \leftarrow \mathcal{L} - \bar{\bar{q}}$; /* Remove from unscheduled */
      $\mathcal{S} \leftarrow \mathcal{S} + (\bar{k}, \bar{\bar{s}}, \bar{\bar{s}} + d^{\bar{k}}_{\bar{\bar{q}}}, \bar{\bar{q}})$; /* Add to schedule */
      $\mathcal{C} \leftarrow \mathcal{C} + (\bar{k}, \bar{\bar{s}}, \bar{\bar{s}} + d^{\bar{k}}_{\bar{\bar{q}}}, \bar{\bar{q}})$; /* Add new candidate */
end

C The ratio heuristic: *AlgoGreedyCover*

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., 1993. ISBN 0-13-617549-X.

[2] A. Amaya, A. Langevin, and M. Trépanier. The capacitated arc routing problem with refill points. *Operations Research Letters*, 35(1):45–53, 2007. ISSN 0167-6377.

[3] D.L. Applegate, R.E. Bixby, V. Chvàtal, and W.J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton Press, 2007.

Algorithm 2: The ratio heuristic : *AlgoGreedyCover*

Input: $\bar{z}^{LP}$,c,M,Q,$\bar{A}$
Output: $\bar{z}^{IP}$
Data: $\bar{Q}$, $\bar{\bar{A}}$, r
Result: An integer solution satisfying cover constraints
begin
  $\bar{Q} := \left\{q \in Q | \bar{z}_q^{LP} > 0 \text{ and } \bar{z}_q^{LP} < 1\right\}$; `/* list of variables to round up */`
  for $q \in Q \setminus \bar{Q}$ **do** $\bar{z}_q^{IP} := \bar{z}_q^{LP}$; `/* fix integer values */`
  for $q \in \bar{Q}$ **do** $r_q := c_q / \sum_{a \in \bar{A}} M_{qa}$; `/* ratio calculation */`
  $\bar{\bar{A}} := \left\{a \in \bar{A} | \nexists q, \bar{z}_q^{IP} = 1 \text{ and } M_{qa} = 1\right\}$; `/* list of uncovered elements */`
  while $\bar{Q}$ *is not empty* **do**
    $\bar{q} := \arg\min_{q \in \bar{Q}} r_q$; $\bar{z}_{\bar{q}}^{IP} := 1$; `/* select variable to round up */`
    $\bar{\bar{A}} := \bar{\bar{A}} - \left\{a \in \bar{A} \cap \bar{\bar{A}} | M_{\bar{q}a} = 1\right\}$; `/* update uncovered elements */`
    $\bar{Q} := \bar{Q} - \{\bar{q}\}$; `/* remove` $\bar{q}$ `from` $\bar{Q}$ `*/`
    for $q \in \bar{Q}$ **do**
      if $\sum_{a \in \bar{\bar{A}}} M_{qa} = 0$
      then $\bar{z}_q^{IP} := 0$, remove q from $\bar{Q}$; `/* unselectable set */`
      else $r_q = c_q / \sum_{a \in \bar{\bar{A}}} M_{qa}$; `/* update ratio */`
end

[4] S. Raghavan B.L. Golden and E.A. Wasil, editors. *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer US, 2008.

[5] Vasek Chvàtal. A Greedy Heuristic for the Set-Covering Problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[6] M. Dror, editor. *Arc Routing: Theory, Solutions and Applications*. Springer, 2000. ISBN 0-79237-898-9.

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

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

[9] J.L Ellis and S. Wohlk. 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.

[10] 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.

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

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

[13] G. Hasle and O. Kloster. *Geometric Modelling, Numerical Simulation, and Optimization*, chapter Industrial Vehicle Routing, pages 397–435. Springer Berlin Heidelberg, 2007.

[14] Stefan Irnich. Solution of real-world postman problems. *European Journal of Operational Research*, 190(1):52 – 67, 2008. ISSN 0377-2217. doi: DOI: 10.1016/j.ejor.2007.06.002. URL .

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

[16] A. Max-Colell, M.D. Whinston, and R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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. ISBN 978-3-540-71804-8.