Rolling stock rostering optimization under maintenance constraints

Giovanni Luca Giacco
Direzione Pianificazione Industriale, Trenitalia, Rome, Italy
Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre, Rome, Italy

Andrea D’Ariano, Dario Pacciarelli *
Dipartimento di Informatica e Automazione, Università degli Studi Roma Tre, Rome, Italy

ABSTRACT
This paper presents an integer programming formulation for integrating maintenance planning tasks in the railway rolling stock circulation problem. This is a key problem in railway rostering planning that requires to cover a given set of services and maintenance works with a minimum amount of rolling stock units. An additional objective is to limit the number of empty rides. The constraints of the rolling stock rostering problem require that different types of maintenance operations must be carried out for each train periodically. The various maintenance tasks can only be done at a limited number of dedicated sites. This is achieved by a graph theoretical approach that combines the scheduling tasks related to train services, maintenance operations and empty rides. Preliminary computational results on real-world scenarios show that this integrated approach can reduce significantly the number of trains and empty rides when compared with the current rolling stock circulation.

Keywords: Railway Planning, Rolling Stock Circulation and Maintenance, Mixed Integer Programming.

INTRODUCTION
A main challenge of railway undertakings is to reduce the overall cost of railway operations by means of a more efficient use of rolling stock and crew resources. Rolling stock circulation and maintenance are significant cost components for a railway company, which are typically managed by solving a number of interrelated problems [3, 4, 7]. Most of the scientific research considers the management of train rostering, empty rides balancing and rolling stock maintenance separately, even if these are parts of the same problem (see, e.g., the recent literature reviews in [1, 5, 6]).

In this work, a new mixed integer linear programming formulation is presented to combine the different aspects of the problem in an efficient way. The goal is to minimize the total number of trains, empty rides and maintenance tasks that are needed to execute a given timetable. Given the departure and arrival times of each train service, the problem is composed by three main tasks: (a) assign rolling stock units (i.e., trains) to the timetable services, (b) schedule the maintenance tasks, (c) limit the number of empty rides.

The proposed model is solved with a commercial MIP solver and tested on a number of practical instances based on timetable examples of the Italian railway company Trenitalia for year 2011. The real-life solutions are compared with those obtained by our solution method, in terms of the number of trains and empty rides needed to realize the given timetables.

The next sections describe the integrated problem of train rostering and maintenance scheduling, present the mathematical formulation of this problem, give preliminary computational experiments on real-world scenarios, discuss the obtained results and provided a description of on-going research directions.

PROBLEM DESCRIPTION
The rolling stock rostering (RRR) problem is to determine a circulation for given scheduled rides. The general goal is to minimize the cost of the rolling stock assignment. The problem has typically numerous requirements and constraints to be satisfied, that may differ for each railway company [2].

This paper studies a general problem version with special attention to the rolling stock and maintenance constraints of interest for Trenitalia. We consider a macroscopic description of the railway traffic flow. The network is composed by a number of tracks and stations. A train route is a path between two given stations, with a given travel time. A train service i is a route from a departure station di at departure time ti
to an arrival station $a_i$ at arrival time $t_i$ that must be covered by a specific train. A roster is a cycle spanning over several working days that covers all the services and the required maintenance tasks. We assume that the same timetable is repeated every day. In other words, we consider a cyclic timetable and do not study its variability e.g. in case of high/low demand days. With this assumption, finding a roster spanning over $k$ days allows to cover all services in a day with $k$ trains.

A maintenance site is where maintenance work is performed and can coincide with a station or not. Each maintenance site is dedicated to specific types of maintenance work, such as: interior or exterior cleaning, refuel (only for diesel units), regular inspection, repair (scheduled or not) and technical check-up. Each type of maintenance task must be performed regularly, i.e. within a maximum time limit or a maximum number of kilometers from the last maintenance of the same type. Since performing some maintenance task too often would cause an unnecessary cost for the company, each type of maintenance task should be performed in the proximity of its maximum limit.

The problem addressed in this paper consists of finding a shortest roster, i.e. a sequence of all services spanning over the minimum number of days, such that all required maintenance tasks are inserted in the roster. Empty rides can be added to the roster in order to connect services and/or to visit maintenance sites. Although empty rides cause a relevant cost (e.g. related to additional energy consumption, rolling stock and crew resources) for the company and increase the traffic in the network, their inclusion may help to reduce the maintenance cost and the roster length. For these reasons, optimizing the scheduling of maintenance tasks, trains services and empty rides is an important contribution to reduce the overall company costs.

We use the following data as problem input: the timetable with the scheduled train services; the maintenance sites; the maximum number of kilometers and the planned time of an empty ride; the minimum and maximum number of kilometers for each type of maintenance task.

**PROBLEM FORMULATION**

The RRR problem is represented by a graph $G = (V, A)$ in which the set of nodes $V$ contains all the train services to be included in the roster, while the set of arcs $A$ is associated to a feasible sequencing of train services in a roster, plus the possible inclusions of empty rides and maintenance tasks. There can be several types of arcs between any two nodes $i$ and $j$, and we denote by $z$ the type of arc $(i, j, z)$ and by $Z$ the set of arc types.

If the arrival station $a_i$ of service $i$ is equal to the departure station $d_j$ of service $j$, we add to $A$ a first arc of type $z = \text{waiting}$ between $i$ and $j$ plus an arc of type $z = \text{maintenance}$ for each type of maintenance task $m$ that can be executed in the proximity of station $a_i$, i.e., such that the distance between $a_i$ and the closest maintenance site enabled to perform $m$ is smaller than a pre-defined value.

If $a_i \neq d_j$, we add to $A$ a first arc $(i, j, z)$ of type $z = \text{empty ride}$ plus an arc for each maintenance task that can be processed in a maintenance site close to $a_i$, $d_j$ or along the route from $a_i$ to $d_j$.

Each arc $(i, j, z)$ has a cost $c_{ijz}$ that is the number of days required to process $j$ after the completion of $i$, i.e., zero if $i$ and $j$ can be performed consecutively in the same day. The rostering problem can be viewed as the problem of finding a minimum cost Hamiltonian cycle in $G$ with additional constraints related to the implementation of maintenance tasks.

To formulate the maintenance constraints, for each arc $(i, j, z)$ and for each maintenance type $m$, we introduce a real variable $g_{ijz}^m$ that counts the kilometers covered by each train since the last maintenance task of type $m$ was performed. A lower bound $\beta_m$ and an upper bound $\gamma_m$ on the kilometers are specified for each maintenance task $m$ and constraints $\beta_m \leq g_{ijz}^m \leq \gamma_m$ are added to the formulation to force each train to visit a maintenance site between $\beta_m$ and $\gamma_m$ kilometers.

Figure 2 shows a small graph to illustrate our problem representation. For each train service, there is a (red) node with labels indicating departure and arrival stations, plus the associated times. The green arcs indicate paired services, the (solid) black arcs the empty rides without maintenance, the (dotted) black arcs the empty rides with maintenance tasks, the blue arcs the maintenance tasks without empty rides. The numeric labels show arc costs, while non-numeric labels indicate maintenance types (M1 and M2). For simplicity, the maintenance costs are not shown in the graph.
In the example of Figure 2, we have three services (Napoli - Udine, Udine - Roma and Roma - Napoli) that require a number of trains, maintenance works and empty rides. A solution is to schedule the services without empty rides and maintenance works (i.e., to select the three green arcs). A more expensive solution is to include maintenance works in the schedule (i.e., to select two green arcs and one blue arc).

**List of Notations**

We now list the notation used in this paper.

- \(V\) is the set of train services (i.e., the set of nodes)
- \(n\) is the cardinality of the set \(V\)
- \(A_1\) is the set of empty ride arcs without maintenance tasks (i.e., the set of solid black arcs)
- \(A_2\) is the set of empty ride arcs with maintenance tasks (i.e., the set of dotted black arcs)
- \(A_3\) is the set of maintenance arcs without empty rides (i.e., the set of blue arcs)
- \(A_4\) is the set of service pairings (i.e., the set of green arcs)
- \(A\) is the set of all arcs: service pairings, empty rides and maintenance tasks \((A = A_1 \cup A_2 \cup A_3 \cup A_4)\)
- \(A_4^m\) is the set of service pairings, empty rides and maintenance tasks that do not include maintenance task \(m\)
- \(A_1^m\) (\(A_1^\neg m\)) is the set of empty ride arcs with maintenance tasks that (do not) include task \(m\) in a maintenance site in the middle of their route
- \(Z_{i,j}\) is the set of arc types between nodes \(i\) and \(j\)
- \((i, j, z)\) is an arc between start node \(i\) and end node \(j\) of type \(z \in Z\)
- \(K_i\) are the kilometers of train service \(i\)
- \(K_{ij}^1\) are the kilometers to be performed by a train (associated to arc \((i, j, z)\)) from \(a_i\) to a maintenance site in case of empty ride
- \(K_{ij}^2\) are the kilometers to be performed by a train (associated to arc \((i, j, z)\)) from a maintenance site to \(d_j\) in case of empty ride
- \(K_{ij}^3\) are the kilometers to be performed by a train (associated to arc \((i, j, z)\)) from \(a_i\) to \(d_j\) in case of empty ride
- \(\alpha\) is a bound related to the maximum number of empty rides allowed in a solution
- \(\beta_m\) is a lower bound on the kilometers traveled by a train between consecutive executions of task \(m\)
- \(\gamma_m\) is an upper bound on the kilometers traveled by a train between consecutive executions of task \(m\)

**Problem variables**

The proposed formulation considers three types of variables: \(X\) is a set of binary variables such that \(x_{ijz} \in X\) is equal to 1 if arc \((i, j, z)\) belongs to the Hamiltonian cycle and zero otherwise, \(Y\) is a set of integer variables that are used for sub-tour elimination, \(G\) is a set of real variables. Variable \(g_{ijz}^m \in G\) is used to force that if \(x_{ijz} = 1\) then the kilometers traveled by a train between two consecutive executions of task \(m\) is always between \(\beta_m\) and \(\gamma_m\). In a solution, the variables in \(Y\) and \(G\) can be derived from the variables in \(X\).

**Objective function**

The objective function of the problem is the minimization of the number of days included in the roster, i.e., the number of trains required to perform all services in a day:

\[
\sum_{(i,j,z) \in A} c_{ijz} x_{ijz}
\]

where \(c_{ijz}\) is the cost of arc \((i, j, z) \in A\).
Path constraints

The first set of constraints is:

(I) \[ \sum_{i \in V} \sum_{z \in Z_i} x_{ihz} = 1 \quad \forall h \in V \]

(II) \[ \sum_{j \in V} \sum_{z \in Z_j} x_{hjz} = 1 \quad \forall h \in V \]

The satisfaction of equation (I) forces exactly a predecessor and a successor for each node \( h \in V \).

Sub-tour elimination constraints

This set of constraints are introduced for modeling the roster solution as an Hamiltonian cycle. The basic idea is to have node labels that count the order of nodes in a roster solution, beginning from a first node \( n_0 \) randomly chosen. For each node \( n \neq n_0 \), the label value will be greater than 0 and two nodes cannot have the same label. An integer variable \( y_{ij} \in Y \) is associated to each pair of nodes \( k, j \in V \), with \( k \neq j \), such that:

(II) \[ \sum_{k \in V} y_{kj} = \sum_{k \in V} y_{jk} + 1 \quad \forall j \in V \setminus \{n_0\} \]

(III) \[ 0 \leq y_{ij} \leq n \sum_{(ij) \in A} x_{ij} \quad \forall y_{ij} \in Y \]

(IV) \[ \sum_{i \in V} y_{n_i} = 1 \]

Equation (II) constrains the sum of the arcs entering each node, but \( n_0 \), to be equal to the sum of the arcs leaving the same node plus 1. Equation (III) constrains the arc label values to be greater than 0 if and only if a variable \( x_{ij} \in X \) of type \( z \) exists between nodes \( i \) and \( j \) with value greater than 0. With these equations, there is just one arc leaving and one arc entering each node with \( y_{ij} > 0 \). If two services \( i \) and \( j \) are executed consecutively (i.e., if there is a variable \( x_{ij} = 1 \)), the label of \( j \) is equal to the one of \( i \) plus 1. Equation (IV) forces node \( n_0 \) to be numbered 1.

Maintenance constraints

This set of constraints prevents the execution of an excessive number of maintenance tasks. To this aim, we introduce a variable \( g^m_{ij} \) for each maintenance task \( m \) and for each arc \( (i, j, z) \in A \). The variable is increased at each train service and at each empty ride, and is set to 0 when the maintenance task \( m \) is performed. The set of maintenance constraints is the following:

(V) \[ \sum_{i \in V} \sum_{z \in Z_i} g^m_{ijz} = K_j \quad \forall j \in V \]

\[ \sum_{i \in V} \sum_{(i,j,z) \in A} g^m_{ijz} + \sum_{(j,i,z) \in A} g^m_{jiz} = K^2_{ijz} \quad \forall (i,j,z) \in A \]

\[ \sum_{(i,z) \in A} K^{3}_{ijz} = K_{ijz} \quad \forall (i,j,z) \in A \]

\[ \sum_{(j,z) \in A} K^{3}_{jiz} \quad \forall (j,i,z) \in A \]

Equation (V) counts the kilometers performed by each train, including the empty rides. Equation (VI) constrains the kilometers to be performed after a task of type \( m \) to be smaller than the upper bound, while equation (VII) constrains the kilometers to be performed before a task of type \( m \) to be at least equal to the lower bound.

Figure 3 shows a simple example with two services, a maintenance site and two empty ride possibilities: including (see dotted black arcs \( K^3 \) and \( K^2 \)) or not including the maintenance work (see black arc \( K^3 \)).

Empty ride bound constraints

This type of constraints defines the maximum number of empty rides permitted in a solution:

(VIII) \[ \sum_{(i,j,z) \in A} x_{ijz} \leq \alpha \quad \forall (i,j,z) \in A \]

where the bound \( \alpha \) is an input parameter.

Computational results

This section presents preliminary computational experiments on real-world cases from the Trenitalia timetable of year 2011. We consider practical rosters and solve the proposed MIP model with CPLEX 12.0.

We study three practical timetables (T1, T2 and T3). The first two (T1, T2) are based on real cases and compared with the practical solutions, while the third (T3) is hypothetical and has been created to test the computational limits of our approach.

Table 1 presents results on the three timetables (Column 1). Column 2–3 describe the number of train services and empty rides defined in each timetable. Columns 4–5 show the number of trains used in practical and our model solutions implemented in CPLEX.

To compare our solutions with the practical ones, we have fixed the maximum number of empty rides \( \alpha \) to the same value used in practice. For this first set of experiments, we average computation time of CPLEX is around 10 seconds.

<table>
<thead>
<tr>
<th>Timetable Scenario</th>
<th>Train Services</th>
<th>Empty Rides</th>
<th>Practical Solution</th>
<th>CPLEX Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>26</td>
<td>0</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>T2</td>
<td>78</td>
<td>8</td>
<td>40</td>
<td>38</td>
</tr>
<tr>
<td>T3</td>
<td>104</td>
<td>8</td>
<td>-</td>
<td>45</td>
</tr>
</tbody>
</table>

For timetable T2, our model solution compares favorably with the practical roster, with a 5% reduction in the number of trains needed to cover all services.
Figure 4 shows a second set of experiments in which our model considers the empty rides as additional problem variables that can be selected in a range of min-max values. The experiments are based on timetables T1, T2 and T3 using different settings of the maximum number of empty rides. We show on y-axis the number of trains needed for the roster and on the x-axis the maximum number of empty rides. On average, the computation time of CPLEX solutions is 0.59 minutes for T1, 1.1 minutes for T2 and 7.5 minutes for T3.

![Figure 4: Measuring the compromise between the required number of rolling stock units and empty rides](image)

From the results of Figure 4, we have the following observations. For T1, increasing the empty rides has no effect on the rolling stock required to run all services, while for T2 and T3 the rolling stock used is reduced down to 6 units (for T2: 41 to 35; for T3: 48 to 42).

For T2, the practical solution (with 40 trains and 8 empty rides) can be improved by two actions: reducing the number of trains and/or reducing the number of empty rides. When comparing the practical versus our model solutions, there is a trade-off between the two actions for the two cases with 9 and 10 empty rides. For smaller values of empty rides, our model gives always better solutions than the practical one for both objectives. In the solution with 8 empty rides, the number of trains can be reduced up to 10%.

Another observation for T2 is that the model with fixed number of empty rides performs worst than the model with min-max values for the number of empty rides. This is due to the additional flexibility added in the latter model that is able to reduce the necessary rolling stock from 38 to 36 units.

CONCLUSIONS AND NEXT RESEARCH DIRECTIONS

This paper presented a novel approach for the combined rolling stock and maintenance planning problem. The problem is solved by finding a minimal cost Hamiltonian cycle in a graph with service pairings, empty rides and maintenance tasks. Computational results show relevant potential application of the proposed formulation for improving the current practical solutions.

On-going research is focused on a thorough assessment of timetables and rosters. We are studying how to extend our model by including scheduling maintenance and platforming operations. Another issue is the definition of objective functions directly related to the costs of circulation, maintenance and empty rides. Open issues are the balanced use of resources and the limitation of extra working time for the operators. Additional research directions may focus on developing approaches for acyclic timetables and sophisticated algorithms for more complex and/or larger instances.

REFERENCES


