Rolling stock rostering optimization under maintenance constraints

Giovanni Luca Giacco\textsuperscript{1,2}, Andrea D’Ariano\textsuperscript{1}, Dario Pacciarelli\textsuperscript{1}

RT-DIA-195-2012 Giugno 2012

(1) Università degli Studi Roma Tre, Via della Vasca Navale, 79 00146 Roma, Italy.

(2) Direzione Pianificazione Industriale, Trenitalia S.p.A., Piazza della Croce Rossa, 1 00161 Roma, Italy.

This work is partially supported by the Italian Ministry of Research, project FIRB “Advanced tracking system in intermodal freight transportation”, Grant number RBIP06BZW8. We thank Trenitalia managers for their suggestions and support with railway data collection and analysis. A preliminary version of this work was published in Giacco et al. [15].
ABSTRACT

This paper presents a mixed-integer linear-programming formulation for integrating short-term maintenance planning in a network-wide 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. In our formulation, a rostering solution is viewed as a minimal cost Hamiltonian cycle in a graph with service pairings, empty runs and short-term maintenance tasks. We use a commercial MIP solver to compute efficient solutions in a short time. Experimental results on real-world scenarios from Trenitalia show that this integrated approach can reduce significantly the number of trains and empty runs when compared with the current rolling stock circulation plan.

Keywords: Railway Planning, Rolling Stock Circulation, Maintenance, Mixed-Integer Linear-Programming.
1 INTRODUCTION

Problem under study

A main challenge of railway undertakings is to reduce the overall cost of railway operations by means of a more efficient use of available resources, as well as rolling stock, crews and maintenance resources [17]. Rolling stock circulation and short-term maintenance planning are significant cost components for a railway company, which are typically managed by solving a number of interrelated problems [19, 21, 22].

The current practice of rolling stock rostering is focused on implementing slight modifications to the previous plan in presence of new requests. This is mainly due to the difficulty faced by planners when developing network-wide solutions manually. The computation of a globally feasible solution is already a very complex task and the planners have little concern of the gap between their solutions and the optimal solutions related to specific performance indicators. Research in this context is therefore worth to search for better quality solutions of practical interest.

Related literature

Most of the scientific work considers the management of train rostering, the balancing of empty runs and the cycles of rolling stock maintenance separately, even if these are parts of the same problem. In view of several recent reviews on this subject [1, 2, 8, 16, 18], we limit our literature discussion to research that deals with analytical approaches closely related to this work.

Erlebach et al. [13] discuss the NP-hardness of the train rostering problem, including the special cases in which empty movements are allowed and rolling stock maintenance is required. They work under the assumption that all trains are identical. They also simplify the maintenance scheduling by replacing an unmaintained train by a maintained one, once they are at the same station.

Eidenbenz et al. [14] study a so-called flexible train rostering problem in which two levels of flexibility are considered: the departure time of each route is free, but not the duration, and a delay threshold is given for each route. However, maintenance operations are not included in their approach and there is no computational evidence of the potential gain achievable by introducing flexible operations.

Mároti and Kroon [20] develop a multi-commodity flow model for preventive maintenance routing. Their basic idea is to improve an existing practical solution by implementing a limited number of changes to a macroscopic rolling stock plan in which rolling stock units move on lines between aggregate stations. The objective function is related to the minimization of shunting plan deviations. Alfieri et al. [3] propose a multi-commodity flow model for efficient rolling stock circulation on a single line of the Dutch railway network. Their objective is to minimize the distance run by train units of various types. Short-term maintenance requirements are not considered in their formulations.

Budai et al. [6] provide a mathematical formulation for the long-term planning of railway maintenance works. The objective is to minimize the time required for maintenance, expressed as a cost function. Heuristic algorithms compute nearly-optimal solutions by combining maintenance activities on each track.

Caprara et al. [9] study the train timetabling problem from an infrastructure manager
point of view: the objective is to improve the use of infrastructure resources. Maintenance operations are modeled as fixed constraints. An integer linear programming formulation is proposed and solved by a Lagrangian heuristic. Tests on a Italian test bed with different train types show that maintenance constraints may seriously affect the quality of the overall timetabling process.

Recently, Borndörfer et al. [5] study the assignment of rolling stock to timetable services. A hypergraph based integer programming formulation is proposed for a cyclic planning horizon of one week. In Cadarso and Marín [7], a more general rolling stock and train routing problem is addressed. The rolling stock subtask is to assign material to satisfy the timetable of a railway network, while the train routing subtask is to determine the best sequence for each material. Since the combined problem is not solvable by standard MIP solvers, they propose a new heuristic based on Benders decomposition. The objective function of the overall approach is to minimize a cost-based function related to commercial train services, empty movements, shunting and passengers in excess. Both the latter analytical approaches do not model short-term maintenance operations and do not evaluate their cost impact.

Paper contribution

This work addresses jointly the daily rolling stock and maintenance aspects of the rostering problem. We present a new decision support system based on a mixed-integer linear-programming formulation. The general goal is to cover all services of a given timetable, as well as maintenance operations, while minimizing the use of rolling stock units (i.e., trains). Given the departure and arrival times of each scheduled train service, the rostering problem is composed by three main tasks: (a) assign rolling stock units to the services, (b) schedule the maintenance tasks, (c) limit the number of empty runs. An empty run is a route between two stations with no assigned service, which is necessary when a train must cover in sequence two services and the arrival station of the first service is different from the departing station of the second.

The specific objectives of our research are to address the following questions: “How can railway company efficiently manage their rolling stock? Which is the maximal improvement that can be achieved?” In this work, we address these questions by performing an incremental assessment of key performance indicators related to tasks a–c.

Methodology employed

The rolling stock rostering problem is viewed as the problem of finding a minimal cost Hamiltonian cycle in a graph in which the nodes correspond to train services and the arcs correspond to scheduling decisions, short-term maintenance operations and empty runs. The objective function is to minimize the rolling stock units to cover all services. Additional performance indicators are improved by changing constraints on the bounds of some variables. The constraints of the rostering problem require that the 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 workshops.

The computational results are obtained by implementing and solving the proposed model with a commercial MIP solver. We tested a number of practical instances based on timetable examples from Trenitalia (the main Italian railway company) for year 2011.
We reconstructed the services scheduled in the practical timetables as a benchmark for performance evaluation. The real-life solutions are compared with those obtained by our solution method in terms of the number of rolling stock units, the empty runs needed to realize the given timetables, and the distance run by trains between consecutive maintenance operations.

Paper organization

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

2 PROBLEM DESCRIPTION

The rolling stock rostering (RSR) problem is to determine a circulation for given scheduled runs. 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 [4, 10].

This paper studies a RSR problem with rolling stock and maintenance constraints particularly relevant for Trenitalia. The proposed approach considers a macroscopic modeling 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 $d_i$ at departure time $t_d^i$ to an arrival station $a_i$ at arrival time $t_a^i$ 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 study a cyclic timetable and do not include, e.g., its variability in case of high/low demand days or the inclusion of acyclic services, like long distance services. However, cyclic rosters can be adapted to deal also with additional acyclic services. With our cyclic assumption, finding a roster spanning over $k$ days allows to cover all services in a day with $k$ trains.

A maintenance workshop is where maintenance work is performed and can coincide with a station or not. Each maintenance workshop 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 task of the same type. Since performing the same task too frequently would cause an unnecessary cost for the company, each type of maintenance task should be performed in the proximity of its maximum limit. It is known from the literature that maintenance tasks may impose severe constraints on the RSR problem and their effect on the line capacity may be difficult to analyze [9].

Figure 1 presents two rosters for a cyclic timetable with 14 train services (S1–S14). Each row shows the daily route to be covered by a specific train. The different colors identify the train services (green), the maintenance tasks (blue) and the empty runs (black). Both rosters can be executed with 6 trains. Specifically, Figure 1 (a) shows a roster without empty runs while Figure 1 (b) shows a roster with two empty runs (E1–
E2). The two schedules also differ in the number of maintenance tasks (3 in the first case and 2 in the second one).

In Figure 1 (b), the two empty runs are added for moving the train corresponding to services S9 and S10 from Salerno to Napoli and from Napoli to Salerno. In this way, the maintenance task M2’ can be performed at a maintenance workshop in Napoli. The maintenance tasks M2 and M3 are thus replaced by the maintenance task M2’.

Specifically, the differences between the two rosters are the following:

- M1 and M1’ have approximately the same duration (around 12 hours), they are both performed at a maintenance workshop in Rome, and their schedules are just shifted of a few minutes;
- M2 and M3 are both performed at a maintenance workshop in Rome and require (in total) around 19 hours, while M2’ is performed in Napoli and requires around 16 hours. So, there is a gain of around 3 maintenance hours plus one less visit to a maintenance workshop for the roster of Figure 1 (b) compared to the one of Figure 1 (a).

The comparison between the two rosters of Figure 1 shows how it is possible to reduce the number of maintenance tasks by using empty runs. From the one hand, the maintenance costs are potentially reduced since less maintenance operations are performed in the schedule of Figure 1 (b) compared to the one of Figure 1 (a). On the other hand, the cost of empty runs is increased since two empty runs are added in the schedule of Figure 1 (b).

The specific 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 runs can be added to the roster in order to connect train services and/or to visit maintenance workshops. Although empty runs 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 the above reasons, optimizing the scheduling of maintenance tasks, trains services and empty runs is an important contribution to reduce the overall company costs. However, we notice that the overall cost of a solution is difficult to evaluate. In fact, besides its monetary value, a solution is also characterized by the utilization of track capacity, maintenance workshops and other resources. Therefore, in this paper we analyze separately the main components affecting the cost of a solution, i.e., the number of rolling stock units, the amount of maintenance tasks and the number of empty runs.

The input data of the problem include: the rolling stock asset; the timetables and the scheduled train services; the maximum number allowed of empty runs; the railway infrastructure; the location and characteristics of maintenance workshops; the maintenance tasks to perform and time windows of [minimum, maximum] number of kilometers for each maintenance task.

Clearly, allowing a larger number of empty runs or larger time windows for the maintenance tasks corresponds to leaving greater flexibility to minimize the number of rolling stock units. The effects of flexibility on the efficient management of assets such as trains, infrastructure elements and staff have been assessed, e.g., by [9, 11, 12].

In the computational experiments presented in this paper, we investigate the interaction between the minimum number of trains needed to perform all services and different
Figure 1: Rolling stock rosters without (a) and with (b) empty runs
values for the maximum number of empty runs, and for the time windows of the distance covered between consecutive maintenance operations of the same type. We also evaluate the efficiency of the optimization-based solutions compared with the practical ones in terms of these three performance indicators.

3 PROBLEM FORMULATION

In this section we introduce the problem notation and the formulation of the RSR problem. The notation is also listed in the Appendix of the paper.

The RSR problem is represented by a graph $G = (V, A)$ in which the set of nodes $V$ contains the $n$ train services to be included in the roster, while each arc in $A$ models a feasible sequencing of train services in a roster, plus the possible inclusions of empty runs and/or maintenance tasks.

There can be several types of arcs between 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 $A$ an arc of type $z = \text{waiting}$ between $i$ and $j$ and, possibly, 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 workshop enabled to perform $m$ is smaller than a pre-defined value.

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

Each arc $(i, j, z)$ has a cost $c_{ijz}$ equal to the time lag (i.e., the number of days) required to process $j$ after the completion of $i$, which is zero if $i$ and $j$ can be performed consecutively in the same day. When $z = \text{maintenance}$ or $z = \text{empty run}$, other values associated to arc $(i, j, z)$ are: the distance $K_{ijz}^1$ from $a_i$ to the maintenance workshop, the distance $K_{ijz}^2$ from the maintenance workshop to $d_j$, and the distance $K_{ijz}^3$ from $a_i$ to $d_j$ (only in case of empty run). Figure 2 shows a simple example with two services (from Rome to Naples and from Udine to Rome), a maintenance workshop and two empty run possibilities: including (see the dotted black arcs) or not including a maintenance work (see the solid black arc) when moving from Naples to Udine with an empty run.

![Figure 2: Example of empty run and maintenance tasks](image)

In order to compute $c_{ijz}$, we must preliminarily compute the time lag between the two services $i$ and $j$, which depends on the arc type:

- If $d_i = a_j$ and $z = \text{waiting}$, the time lag must only take into account the minimum slack time between the two services;
• if \( d_i = a_j \) and \( z = \text{maintenance} \), the time lag must also take into account the maintenance time window needed to execute maintenance works for the rolling stock involved, plus the travel time to reach the maintenance workshop and come back;

• if \( d_i \neq a_j \) and \( z = \text{empty run} \), the time lag must take into account the time needed to reach \( a_j \) plus, possibly, the slack time and the maintenance time needed to perform the prescribed maintenance tasks. For the example in Figure 2, the cost of the solid arc is one, since the second service can start only the day after the arrival in Naples. The cost of the dotted arc can be even two or more, depending on the time needed to perform the maintenance task.

Note that the feasibility of a Hamiltonian cycle passing through arc \((i, j, z)\) also depends on the maintenance status of the rolling stock at node \(i\). In fact, the distance elapsed between \(i\) and \(j\) plus the rolling stock status at \(i\) must be compatible with the maintenance windows for all maintenance tasks. To compute the rolling stock maintenance status, for each arc \((i, j, z)\) and for each maintenance type \(m\), we introduce a real variable \(g_{ijz}^m\) that counts the distance covered by each train since the last maintenance task of type \(m\) was performed. A solution is feasible if \(g_{ijz}^m\) is bounded between a minimum \(\beta_m\) and a maximum \(\gamma_m\) on the distance (in kilometers) run by a train between consecutive executions of task \(m\), i.e., \(\beta_m \geq g_{ijz}^m \geq \gamma_m\).

In conclusions, 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.

**Illustrative example**

Figure 3 shows a small graph to illustrate the problem formulation. For each train service, there is a (red) node, with labels indicating departure and arrival stations plus the associated times. The solid black arcs (set \(A_1\)) indicate the empty runs without maintenance, the dotted black arcs (set \(A_2\)) the empty runs with maintenance tasks, the blue arcs (set \(A_3\)) the maintenance tasks without empty runs, the green arcs (set \(A_4\)) the service pairings. The numeric labels show arc costs, while non-numeric labels indicate maintenance types (M1, M2 and their combination M1+M2). For simplicity, the maintenance costs are not shown in the graph.

The three services (Napoli-Udine, Udine-Roma and Roma-Napoli) of Figure 3 require a number of trains, maintenance works and empty runs. A solution is a Hamiltonian path with maintenance operations constraints. In the solution the empty runs (black arcs) are optional.
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 that are used to count the distance run by each train between two consecutive executions the same maintenance task. If $x_{ijz} = 1$ then the distance between two consecutive executions of task $m$ must be always between a lower bound $\beta_m$ and an upper bound $\gamma_m$. In a solution, the variables in $Y$ and $G$ can be derived from the variables in $X$.

Objective function

The objective function is the minimization of the number of days included in the roster or, equivalently, 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:

$$(I) \quad \sum_{i \in V} \sum_{z: (i, h, z) \in A} x_{ihz} = 1 \quad (\forall h \in V)$$

Equation (I) prescribes that there must be exactly one predecessor and one successor for each node $h \in V$. 
Sub-tour elimination constraints

This set of constraints is introduced for modeling the roster as an Hamiltonian cycle. The basic idea to avoid sub-tours is the use of node labels that count the order of nodes in the solution, beginning from a first node \( n_0 = 1 \) randomly chosen. Along the path, the label of each visited node is increased by one unit compared with the previous node (except for \( n_0 \)). Hence, the value of labels is from 1 to \( n \) and two nodes cannot have the same label.

In the problem formulation, an integer variable \( y_{kj} \in Y \) is associated to each pair of nodes \( k, j \in V \), with \( k \neq j \), such that:

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

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

\[
(\text{IV}) \sum_{i \in V} y_{n_0 i} = 1
\]

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 \( x_{ijz} \in X \) 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 \( x_{ijz} = 1 \)), the label of \( j \) is equal to the one of \( i \) plus 1. Equation (IV) forces all arcs outgoing node \( n_0 \) to be numbered 1.

Figure 4 shows an example situation with two sub-tours in the graph. Only arcs with variables \( x = 1 \) are shown in the figure. This solution violates Equation II for sub-tour 4,5,6. In fact, \( y_{45} = y_{64} + 1 \), \( y_{56} = y_{45} + 1 \), and \( y_{64} = y_{56} + 1 \) must hold, thus implying \( y_{45} = y_{45} + 3 \), which is impossible.
Maintenance constraints

Maintenance tasks need to be performed within a given time window of maintenance. However, the intention is to prevent the execution of an excessive number of maintenance tasks. In general, the less are the maintenance tasks the more cost effective is the overall solution.

The formulation of maintenance tasks requires the introduction of a new variable $g^m_{ijz}$ for each maintenance task of type $m$ and for each arc $(i,j,z) \in A$. This variable counts the kilometers run by a train from the last maintenance task of type $m$, and it is set to 0 when the maintenance task $m$ is performed.

Let us introduce some further notation. $K_i$ is the distance run by train service $i$, $A^m$ is the set of service pairings, empty runs and maintenance tasks that include [do not include] maintenance task $m$. $A^m_i$ is the set of empty run arcs with maintenance tasks that include [do not include] task $m$ in a maintenance workshop at the beginning of their route, $A^m_{II}$ is the set of empty run arcs with maintenance tasks that include [do not include] task $m$ in a maintenance workshop at the end of their route. The maintenance constraints are:

\[
\begin{align*}
\sum_{l \in V} \sum_{z \in Z_{j,l}} g^m_{ljz} &= K_j + \sum_{i \in V} \sum_{z \in Z_{i,j}: (i,j,z) \in A^m} g^m_{ijz} + \\
V & \sum_{(i,j,z) \in A^m_{II}} K^2_{ijz} x_{ijz} + \sum_{(i,j,z) \in A^m} K^3_{ijz} x_{ijz} + \forall j \in V, \forall m \\
& \sum_{(j,l,z) \in A^m_{II}} K^4_{j lz} x_{j lz} + \sum_{(j,l,z) \in A^m \cup A^m_{II} \cup A^m_{I} \cup A^m_{II} \cup A_{II}} K^3_{j lz} x_{j lz} \forall j, l, z
\end{align*}
\]

(VI) $g^m_{ijz} \leq \gamma_m x_{ijz} \forall (i,j,z) \in A, \forall m$

(VII) $g^m_{ijz} \geq \beta_m x_{ijz} \forall (i,j,z) \in A_2 \cup A_3 : m \in Z_{i,j}$.

Equation (V) counts the distance covered by each train, including the empty runs, from the last execution of a maintenance task of type $m$ to the end of service $j$. Note that, since there is exactly one arc outgoing node $j$ such that $x_{jiz} = 1$, the first term $\sum_{l \in V} \sum_{z \in Z_{j,l}} g^m_{ljz}$ is equal to the single variable $G^m_{ljz} > 0$ and must take into account the distance covered at the end of service $j$. The following terms take into account the distance covered by the train related to service $j$: the distance $K_j$ run by the train when performing service $j$, the cumulative distance covered by the train, including the empty runs, at the end of the service $i$ performed immediately before $j$ if no maintenance task of type $m$ is performed between $i$ and $j$. The subsequent two terms of Equation (V) take into account the distance covered during a possible empty run between $i$ and $j$ including a maintenance task of type $m$ performed along the route from $a_i$ to $d_j$ or at station $a_i$, respectively. The last two terms of Equation (V) take into account the situation in which a maintenance task of type $m$ is performed between the end of service $j$ and the start of service $l$. In this case the quantity $K^4_{j lz}$ must be taken into account to ensure that the total distance covered before the maintenance task is in the window $(\beta_m, \gamma_m)$. The last case arises when the maintenance is performed at station $d_j$ before the starting of service $l$. Note that, if a
maintenance task of type \( m \) is performed during an empty run, only the distance covered after the maintenance must be considered.

With reference to node \( j \), Figure 5 shows the types of ingoing and outgoing arcs and the corresponding distance to be covered. For the ingoing arcs to node \( j \), each train covers \( g_{ijz}^m \) in case of no maintenance of type \( m \), \( K_{ijz}^3 \) in case of empty run with maintenance of type \( m \) at the beginning of its route, \( K_{ijz}^2 \) in case of empty run with maintenance of type \( m \) in the middle of its route, 0 otherwise. For the outgoing arcs from node \( j \), each train covers \( K_{jlz}^3 \) in case of empty run without maintenance tasks or with the maintenance task of type \( m \) at the end of its route, \( K_{jlz}^1 \) in case of empty run with maintenance of type \( m \) in the middle of its route, 0 otherwise.

![Figure 5: Computation of the distance covered before and after node \( j \)](image)

Equation (VI) constrains the distance to be covered after a task of type \( m \) to be smaller than the upper bound \( \gamma_m \), while equation (VII) constrains the distance to be covered before a task of type \( m \) to be at least equal to the lower bound \( \beta_m \). The deadline of each basic maintenance task is thus constrained, even if it is sometimes possible to combine basic maintenance operations in multifunctional workshops.

**Bound constraints on the empty runs**

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

\[
(VIII) \quad \sum_{(i,j,z) \in A_1 \cup A_2} x_{ijz} \leq \alpha
\]

where the bound \( \alpha \) is an input parameter related to the maximum number of empty runs allowed in a solution.

## 4 COMPUTATIONAL EXPERIMENTS

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

Table 1 presents five practical timetables (Column 1). The first four timetables (T1–T4) are based on real cases, and will be compared with the practical solutions, while the fifth (T5) is a realistic timetable, larger than the practical ones, and has been created to test the computational limits of our approach. Specifically, the services in T5 are defined by the union of those in timetables T3 and T4.

<table>
<thead>
<tr>
<th>Timetable Scenario</th>
<th>Rolling Stock Categories</th>
<th>Num Train Services</th>
<th>Num Railway Cars</th>
<th>Total Length [m]</th>
<th>Maintenance Deadline [km]</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>Loco E444</td>
<td>46</td>
<td>1</td>
<td>17</td>
<td>&gt;&gt;1500</td>
</tr>
<tr>
<td>T2</td>
<td>ETR 485</td>
<td>20</td>
<td>9</td>
<td>237</td>
<td>1500</td>
</tr>
<tr>
<td>T3</td>
<td>ETR 600</td>
<td>26</td>
<td>7</td>
<td>237</td>
<td>1500</td>
</tr>
<tr>
<td>T4</td>
<td>ETR 500</td>
<td>78</td>
<td>11</td>
<td>328</td>
<td>1500</td>
</tr>
<tr>
<td>T5</td>
<td>ETR 600+500</td>
<td>104</td>
<td>18</td>
<td>237/328</td>
<td>1500</td>
</tr>
</tbody>
</table>

For each timetable scenario in Table 1, Column 2 shows the rolling stock categories. Specifically, T1 uses locomotives only while the other four (T2-T5) utilize different types of high speed trains. For each category, Columns 3 shows the number of train services scheduled in the timetable, Column 4 the number of railway cars, Column 5 the total length of each car (in meters), Column 6 the deadline of its maintenance works (in kilometers). The latter value corresponds to the upper bound on the distance run by a train between consecutive maintenance tasks.

Comparison of CPLEX versus real-life rosters

Table 2 presents results on the five timetables (Column 1). Column 2–3 describe the practical solutions in terms of the number of empty runs (E. R.) and trains required to perform the services in each timetable. To compare our solutions with the practical ones, Column 5 shows the number of trains found by CPLEX when the empty runs are constrained to be the same of the practical solution (the number of empty runs is reported in Column 4). Comparing the results in Columns 3 and 5 we observe that even if the set of empty runs is the same, the solution found by CPLEX often outperforms the practical solution in terms of number of trains. Column 6 and 7 show the CPLEX solutions when the number of empty runs allowed can be selected in the range [0, 8]. The last two columns show the CPLEX solutions when the range is [0, 10]. Specifically, Columns 6 and 8 present the number of empty runs in the CPLEX solutions while Columns 7 and 9 show the number of trains. Since instance T5 is fictitious, we do not have the number of trains and empty runs of the practical solution. However, in order to perform the experiments with fixed empty runs for T5 also, we use the same set of empty runs of the practical timetable T4, whose number is reported in Column 4.

From the results in Table 2 we can observe that CPLEX clearly outperforms the practical solutions. In fact, even when the cost component associated to the empty runs is exactly the same (Columns 4 and 5) CPLEX is able to save one train for timetables T1, T2 and two trains for timetable T4 compared to the practical solutions. When the choice of the empty runs to select is left to CPLEX, in a window of min-max values, CPLEX is able to further reduce the number of trains needed to cover all services, the
Table 2: Assessment of practical and CPLEX solutions

<table>
<thead>
<tr>
<th>Timetable Scenario</th>
<th>Practical Solution</th>
<th>CPLEX: Fixed E. R.</th>
<th>CPLEX [0,8]: Flexible E. R.</th>
<th>CPLEX [0,10]: Flexible E. R.</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>T2</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>T3</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>T4</td>
<td>8</td>
<td>40</td>
<td>8</td>
<td>38</td>
</tr>
<tr>
<td>T5</td>
<td>-</td>
<td>-</td>
<td>8</td>
<td>48</td>
</tr>
</tbody>
</table>

reduction being larger for the window [0, 10]. When relaxing the constraint on the empty runs, the maximum gain is obtained for timetable T1 for which the CPLEX solution with window [0, 10] presents a 23% reduction in the number of trains needed to cover all services compared to the practical solution. Clearly, the smaller cost associated to the number of trains is partially compensated by the larger cost associated to the number of empty runs.

For the set of experiments with fixed empty runs, the average computation time of CPLEX is around 10 seconds. In case of flexible empty runs, the average computation time of CPLEX solver is around 1 minute for T1, T2, T3 and T4, while T5 requires around 7.5 minutes.

In conclusion, we observe that the flexible model proposed in Section 3 may be an effective tool for railway operators to find the best balance among the different cost components depending of the actual monetary value of each component, as well as on their preferences and needs. For example, even if the cost of an empty run is small, a solution with no empty runs may be preferred if the rail network is heavily used, since adding new empty runs might increase the risk of congestion in the network.

Maintenance cost optimization

In this set of experiments, a real life solution is compared with our model solution regarding maintenance operations. We measure the maintenance efficiency as the distance run by a train between consecutive executions of a maintenance task of type $m (\beta_m)$. In the computational experiments, we tested timetable T4 with a fixed number of rolling stock units in order to find a feasible solution that improves the maintenance efficiency. We consider timetable T4 since this is the one involving the largest number of train services and requiring more maintenance operations.

Table 3 shows a solution for T4 (first row, named “Original”) and three adjustments of its maintenance tasks (other rows, named “Adjust”). Columns 2 and 3 report the fixed minimum and maximum values of the maintenance window. For the three adjusted solutions, the minimum maintenance value is increased of 1000 km, 2000 km and 3000 km, respectively. Larger adjustments are not shown since no feasible solution was found after some hours of computation. Columns 4 presents the percentage of maintenance efficiency improvement, i.e., $[(\text{Min Value Original Solution} - \text{Min Value Adjusted Solution}) / \text{Min Value Original Solution}]$. Column 5 gives the number of maintenance tasks required in T4.

The real-life solution of T4 can be seriously increased. As shown in Table 2, we can reduce the number of rolling stock units needed to cover all timetable services. From the
Table 3: Optimization of the maintenance efficiency

<table>
<thead>
<tr>
<th>Instance</th>
<th>Min Value [Km]</th>
<th>Max Value [Km]</th>
<th>Improvement [%]</th>
<th>Maintenance tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>5688</td>
<td>13040</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>Adjust1</td>
<td>6688</td>
<td>13040</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>Adjust2</td>
<td>7688</td>
<td>13040</td>
<td>35</td>
<td>5</td>
</tr>
<tr>
<td>Adjust3</td>
<td>8688</td>
<td>13040</td>
<td>53</td>
<td>5</td>
</tr>
</tbody>
</table>

results of Table 3, there is also a considerable improvement of the maintenance efficiency (up to 53%) and a reduction of the required number of maintenance tasks. We believe that the monetary values of maintenance cost reduction can also be assessed by referring to the performance indicators in Table 3, in addition to limiting the number of empty runs.

Flexible values for the empty runs

Figure 6 shows a third set of experiments in which we consider the empty runs as additional variables that can be selected in a range of min-max values. The experiments are based on the five timetables and use different settings of the maximum number of empty runs. We show on y-axis the number of trains needed for the roster and on the x-axis the maximum number of empty runs.

![Figure 6: Measuring the compromise between rolling stock units and empty runs](image)

From the results of Figure 6, we have the following observations. For T2 and T3, increasing the empty runs has no effect on the rolling stock required to run all services, while for T1, T4 and T5 the rolling stock used is progressively reduced.

Considering T4, the practical solution (with 40 trains and 8 empty runs) can be improved by two actions: reducing the number of trains and/or limiting the number of empty runs. When comparing the practical solution versus the optimal solution of our
model, there is a trade-off between the two actions for the two cases with 9 and 10 empty runs. For smaller values of empty runs, our model gives always better solutions than the practical one for both performance indicators. In the solution with 8 empty runs, the number of trains can be reduced up to 10%.

5 CONCLUSIONS AND FURTHER RESEARCH

This paper presents a new approach for optimizing rolling stock rostering and short-term maintenance planning. The mathematical problem is to find a minimal cost Hamiltonian cycle in a graph with service pairings, empty runs and maintenance tasks. Computational experiments are performed on a commercial MIP solver and show a thorough assessment of timetables and rosters. The proposed approach is considerably effective in reducing the company costs compared to the practical solutions, both with and without considering flexibility of rail operations. Specifically, we tried to answer the question: Is it possible to improve the practical solutions firstly in terms of the number of trains needed to cover all services, and then in terms of the number of empty runs and the maintenance efficiency? In fact, this is achieved by improving one by one these performance indicators.

Future research will be dedicated to the development of even more sophisticated formulations. We are studying how define objective functions directly related to the monetary values of circulation, empty runs and maintenance, representing the preferences of the railway company. Another issue is the extension of our model to include detailed scheduling of maintenance operations in station areas. Open issues are related to balance the use of resources when routing trains in wide-networks and to the limit the workload for the operators at maintenance workshops. Further research directions should also be focused on developing methods for acyclic timetables and advanced algorithms for complex and large instances.

References


**APPENDIX: List of notations**

This appendix lists the notation used in the paper.

- $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ is a graph with $V$ nodes and $A$ arcs
- $\mathcal{V}$ is the set of train services (i.e., the set of nodes)
- $n$ is the cardinality of the set $\mathcal{V}$
- $d_i$ is the departure station of service $i$ with departure time $t_i^d$
- $a_i$ is the arrival station of service $i$ with arrival time $t_i^d$
- $\mathcal{A}_1$ is the set of empty run arcs without maintenance tasks (i.e., the set of solid black arcs)
- $\mathcal{A}_2$ is the set of empty run arcs with maintenance tasks (i.e., the set of dotted black arcs)
- $\mathcal{A}_3$ is the set of maintenance arcs without empty runs (i.e., the set of blue arcs)
- $\mathcal{A}_4$ is the set of service pairings (i.e., the set of green arcs)
- $\mathcal{A}$ is the set of all arcs: service pairings, empty runs and maintenance tasks ($\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$)
• $A^m_A (A^m_{\overline{A}})$ is the set of service pairings, empty runs and maintenance tasks that (do not) include maintenance task $m$

• $A^m_A (A^m_{\overline{A}})$ is the set of empty run arcs with maintenance tasks that (do not) include task $m$ in a maintenance workshop at the beginning of their route

• $A^m_{I\overline{I}} (A^m_{\overline{I}I})$ is the set of empty run arcs with maintenance tasks that (do not) include task $m$ in a maintenance workshop at the end of their route

• $A^m_{III} (A^m_{\overline{II}I})$ is the set of empty run arcs with maintenance tasks that (do not) include task $m$ in a maintenance workshop in the middle of their route

• $Z$ is the set of arc types

• $(i, j, z)$ is an arc between start node $i$ and end node $j$ of type $z \in Z$

• $c_{ijz}$ is the cost of arc $(i, j, z)$

• $K_i$ are the kilometers of train service $i$

• $K_{ijz}^1$ is the distance to be covered by a train (associated to arc $(i, j, z)$) from $a_i$ to a maintenance workshop in case of empty run

• $K_{ijz}^2$ is the distance to be covered by a train (associated to arc $(i, j, z)$) from a maintenance workshop to $d_j$ in case of empty run

• $K_{ijz}^3$ is the distance to be covered by a train (associated to arc $(i, j, z)$) from $a_i$ to $d_j$ in case of empty run

• $\alpha$ is a bound related to the maximum number of empty runs allowed in a solution

• $\beta_m$ is a lower bound on the distance run by a train between consecutive executions of task $m$

• $\gamma_m$ is an upper bound on the distance run by a train between consecutive executions of task $m$

• $X$ is a set of binary variables

• $Y$ is a set of integer variables

• $G$ is a set of real variables

• $g_{ijz}^m \in G$ counts the kilometers run by a train from the last maintenance task of type $m$

• $x_{ijz} \in X$ is equal to 1 if arc $(i, j, z)$ belongs to the Hamiltonian cycle and zero otherwise

• $y_{kj} \in Y$ is associated to each pair of nodes $k, j \in V$ such that sub-tours can be recognized and eliminated