A Cut Separation Approach for the Rolling Stock Rotation Problem with Vehicle Maintenance

Boris Grimm
Zuse Institute Berlin, Germany
http://www.zib.de
grimm@zib.de

Ralf Borndörfer
Zuse Institute Berlin, Germany
http://www.zib.de
borndorfer@zib.de

Markus Reuther
LBW Optimization GmbH, Berlin, Germany
http://www.lbw-optimization.com
reuther@lbw-optimization.com

Thomas Schlechte
LBW Optimization GmbH, Berlin, Germany
http://www.lbw-optimization.com
schlechte@lbw-optimization.com

Abstract

For providing railway services the company's railway rolling stock is one if not the most important ingredient. It decides about the number of passenger or cargo trips the company can offer, about the quality a passenger experiences the train ride and it is often related to the image of the company itself. Thus, it is highly desired to have the available rolling stock in the best shape possible. Moreover, in many countries, as Germany where our industrial partner DB Fernverkehr AG (DBF) is located, laws enforce regular vehicle inspections to ensure the safety of the passengers. This leads to rolling stock optimization problems with complex rules for vehicle maintenance. This problem is well studied in the literature for example see [8, 9], or [5] for applications including vehicle maintenance. The contribution of this paper is a new algorithmic approach to solve the Rolling Stock Rotation Problem for the ICE high speed train fleet of DBF with included vehicle maintenance. It is based on a relaxation of a mixed integer linear programming model with an iterative cut generation to enforce the feasibility of a solution of the relaxation in the solution space of the original problem. The resulting mixed integer linear programming model is based on a hypergraph approach presented in [3]. The new approach is tested on real world instances modeling different scenarios for the ICE high speed train network in Germany and compared to the approaches of [10] that are in operation at DB Fernverkehr AG. The approach shows a significant reduction of the run time to produce solutions with comparable or even better objective function values.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization

Keywords and phrases Railway Operations Research, Integer Programming, Infeasible Path Cuts, Cut Separation, Rolling Stock Rotation Problem

Digital Object Identifier 10.4230/OASIcs.ATMOS.2019.1

Funding This work has been supported by the Research Campus MODAL Mathematical Optimization and Data Analysis Laboratories funded by the Federal Ministry of Education and Research (BMBF Grant 05M14ZAM). All responsibility for the content of this publication is assumed by the authors.

1 Introduction

In a liberalized market companies that offer products or services have to compete with competitors for market shares. In an increasing number of countries the railway sector is one of these markets and becomes liberalized more and more. Thus railway companies have to face the challenging problem to offer the best product possible, i.e., punctual, reliable, fast, and comfortable train rides for a reasonable price requiring being as cost efficient as possible. This leads to a wide variety of closely connected optimization problems. Typically these problems differ in the grade of detail and scope of the planning horizon for which decisions have to be made or optimized. One of these problems is the Rolling Stock Rotation Problem (RSRP) where the offered passenger trips of the timetable have to be covered by a set of available rolling stock vehicles in a most cost efficient way. This assignment has to be made with respect to a lot of operational requirements, i.e., the assigned vehicle should not exceed the platform length along a trip's path, electrical powered vehicles require electrified tracks, or the assigned number of coaches should match the expected number of passengers. Although the railway market is liberalized and open for competitors there are many laws the companies have to comply with. Some of them rule mandatory maintenance checks the rolling stock fleets have to pass to be able to transport passengers. A regular vehicle maintenance scheme has also direct and indirect effects on the value of the train journey a customer perceives. They result in a more reliable level of services and in a better shape of the vehicles which is directly recognized by the passengers and linked to the company's image. In Germany, at DB Fernverkehr AG (DBF) our industrial partner, exists a complex schedule of increasing maintenance schedules beginning with short checks of parts of the vehicle and ending in a more or less complete reassembly of the vehicle. Thus, integrating maintenance constraints in the rolling stock rotation problem is a natural choice. The main contribution of this paper is a novel solution approach to the rolling stock rotation problem with vehicle maintenance that differs from the one currently in operation at DBF. It relies completely on infeasible path cuts to take care of maintenance constraints instead of modelling a linked resource flow as it is done in the current approach at DBF. Adding these resource constraints to the arc based RSRP model used at DBF results in a significant increase of problem complexity and run time thus sophisticated algorithms that reduce that increase are very welcome at DBF. Another advantage of the new approach is that it produces several rather different incumbent solutions that are close to optimality which is a feature that planners like. The drawback of this feature and a disadvantage of the approach is that there is more a step wise than a monotone improvement of the solution quality during the solution process.

Rolling stock rotation problems were studied extensively under different names, in various level of detail, and with varying focus in literature the last decades. Since the focus of this paper is on vehicle maintenance we restrict the literature review to papers that consider vehicle maintenance rules to some extend. One of the earliest works that apply to this was done by [5] where locomotives and cars were assigned to passenger trains for scenarios of VIA Rail in Canada. Cyclic timetables were covered with detailed schedules. Vehicle maintenance was considered by a time discretization approach to schedule the maintenance service stops. In [8, 9] the authors proposed solution approaches to re-optimize vehicle schedules for so called urgent trains that require maintenance services within the next 1 to 3 days. Mixed integer programming models based on multi commodity flows are used to tackle the problems for real world scenarios of the Dutch railway operator NS Reizigers. A heuristic solution approach based on an integer linear programming formulation for the Rolling Stock Rotation Problem with integrated optimization of seat capacity of the assigned vehicle configurations is presented in [4]. Maintenance rules were added in a mileage based fashion and the approach is evaluated on scenarios of a regional train operator from Italy. The authors of [6] report a significant reduction of deadheading trips, respectively empty train runs, for their solutions computed by a mixed integer linear program using a commercial MILP-solver to find maintenance feasible Hamiltonian cycles in cyclic network wide instances of Trenitalia. The same approach was later used by [13] to compute maintenance feasible rosters for a case study in Japan. In [11] a nice overview of the research done in this field is presented. Moreover the authors describe heuristic solution approaches to solve the RSRP with integrated optimization of seat capacity for the DSB S-tog network in Copenhagen. Compared to the operated rolling stock schedules the heuristically constructed solutions were proven to be economically more attractive. A solution approach for a re-scheduling version of the RSRP is presented in [12]. After a disruption has happened re-optimized vehicle schedules have to be computed that have to consider maintenance appointments of the effected trains to be feasible. The authors evaluate their mixed integer programming formulation on instances of the Netherlands Railways. A path based branch-and-price algorithm to solve a re-scheduling variant of the RSRP is presented in [7]. In this approach maintenance constraints could be applied naturally to the vehicles due to the model's path based structure. The algorithm is designed for applications at on DSB S-tog network in Copenhagen. All this shows that algorithmic approaches to solve rolling stock scheduling problems with maintenance rules for real world applications is a growing area of interest for the scientific community as well as for the railway industry.

The paper is organized as follows, we define the Rolling Stock Rotation Problem with vehicle maintenance the way it is addressed in this paper in Section 2. This is followed by a solution approach for this problem based on an iterative cut separation procedure using infeasible path constraints in Section 3.2. We evaluate the performance of the presented approach in Section 4 with a comparison to the actual solution approach that is in operation at DBF. Finally, we conclude our results in Section 5.

2 Problem Definition

In this section we consider the *Rolling Stock Rotation Problem* as it is modeled in [3] and refer to the paper for additional technical details. In the following we shortly recapitulate the main modeling ideas.

Let T be the set of all passenger trips of a given timetable and V be a set of *nodes* representing departure and arrival events of dedicated vehicles operating passenger trips of T. Trips that could be operated with two or more vehicles have the appropriate number of arrival and departure nodes. Let further $A \subseteq V \times V$ be a set of directed standard arcs, and $H \subseteq 2^A$ a set of *hyperarcs*. Thus, a hyperarc $h \in H$ is a set of standard arcs and includes always an equal number of tail and head nodes, i.e., arrival and departure nodes. A hyperarc $h \in H$ *covers* $t \in T$ if each standard arc $a \in h$ represents an arc between departure and arrival of t. Each of the standard arcs a represents an individual vehicle that is required to operate t as part of the chosen vehicle configuration the hyperarc models. We define the set of all hyperarcs that cover $t \in T$ by $H(t) \subseteq H$. By defining hyperarcs appropriately, vehicle composition rules and regularity aspects can be directly handled by the model. In more detail for a single trip there are multiple different hyperarcs to chose from with different operational costs, i.e., for a trip that could be operated with one or two vehicles there exist hyperarcs that contain one, respectively, two directed arcs and thus a larger cost coefficient if two arcs, respectively, vehicles are involved. Moreover there are

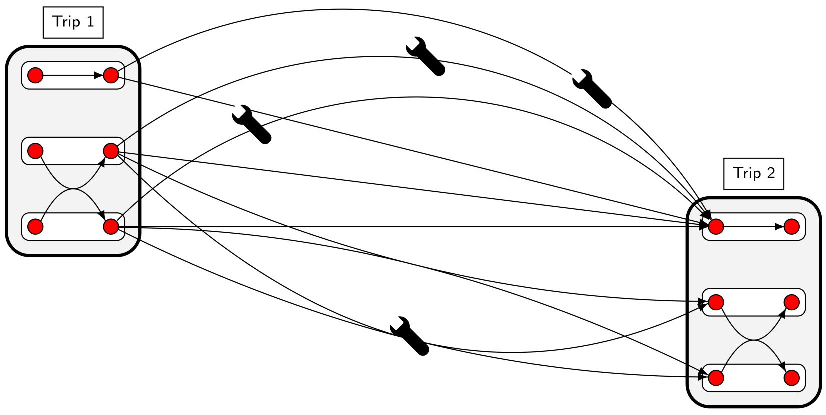

Figure 1 An example of hyperarcs to model two trips and maintenance services between them.

hyperarcs between two trips containing more than one directed arc if the vehicle composition does not change between the two trips. If the time between two trips is large enough to couple or decouple vehicles there are hyperarcs to model this as well. At DBF regularity is an important aspect during optimization so there are additional hyperarcs that contain hyperarcs that model exactly the same trip that is operated on multiple days of the week with exactly the same vehicle configuration. This regularity hyperarc is slightly cheaper than choosing the individual hyperarcs. Hyperarcs that contain arrival and departure nodes of different trips are used to model deadhead trips between the operation of two trips. With this construction it is possible to set up a cost function $c : H \mapsto \mathbb{Q}^+$ for the hyperarcs that includes a wide spectrum of different operational costs that have to be addressed by the model, i.e., costs for energy consumption, vehicle usage, coupling and combining of train units, short turn penalties, or even artificial cost for modelling regular vehicle movements. The RSRP *hypergraph* is denoted by $G = (V, A, H)$. We define sets of hyperarcs coming into and going out of $v \in V$ in the RSRP hypergraph G as $H(v)^- := \{h \in H \,|\, \exists\, a \in h \,:\, a = (u, v)\}$ and $H(v)^+ := \{h \in H \,|\, \exists\, a \in h \,:\, a = (v, w)\}$, respectively. One major challenge in rolling stock planning and optimization is vehicle maintenance. At DB Fernverkehr AG there are several different maintenance rules for the different ICE high speed train fleets that all have to be considered. In this paper we focus on maintenance rules that are based on the accumulated kilometers a vehicle is operated between two maintenance services. We denote the upper bound on the total mileage between two maintenance services by R. Maintenance services could only be performed at special maintenance locations $m \in M$. The kilometers a vehicle is moved during an operation modelled by a chosen hyperarc is given by a function $r : H \mapsto [0, R]$. By $|h|$ the number of standard arcs $a \in h$ required to model h is defined. Thus $|h| \cdot r(h)$ gives the aggregated kilometers of all vehicles modelled by h. This includes necessary deadhead trips to reach maintenance facilities or turn around trips to change the orientation of the vehicle. To model maintenance services in the RSRP *hypergraph* additional maintenance service hyperarcs were defined for each pair of trips if it is possible to visit a maintenance facility and perform a service between the operation of the two trips. The cost for the additional deadhead trip and the cost for the maintenance service is added to the cost of the hyperarc. In this sense, a cycle in G is called maintenance feasible, if and only if the accumulated kilometers of all trips and deadhead trips along the sub-paths between each two hyperarcs with a maintenance service of a cycle is smaller or equal than R.

In Figure 1 two trips are shown that could be operated by either one or two (red) vehicles. Thus, there is a red node for each arrival and departure event of a single vehicle. These are connected by either a single arc hyperarc or a double arc hyperarc to model the operation with one, respectively, two vehicles. Between the two trips there are hyperarcs that model maintenance events. These are marked by the wrench symbol containing single or double vehicle configurations. Additionally there are single arc hyperarcs to model single vehicle transitions between the two trips. The double arc hyperarc models a transition without changing the vehicle configuration.

▶ **Definition 1.** *Let G be a graph based hypergraph, c its associated cost function, and r a maintenance resource function with its upper bound R. The* `Rolling Stock Rotation Problem (RSRP)` *is to find a cost minimal, maintenance feasible set of hyperarcs $H_x \subseteq H$ such that H_x is a set of cycles that covers all trips $t \in T$ by a hyperarc $h \in H_x$.*

3 Solution Approaches to the RSRP

In the following section, we show how the RSRP is modelled and solved in the current application of RotOR, which is the optimization software used at DBF. After that the new approach, which is also implemented in RotOR, using infeasible path constraints in order to iteratively separate maintenance infeasible sub-paths, is presented.

3.1 Resource Flow Based Solution Approach

The current solution approach to the RSRP implemented in RotOR solves a mixed integer programming formulation of a hyperflow model with linked resource flow to model the vehicle maintenances. All details and sophisticated algorithms to solve this model can be found in [3, 10]. Using a binary decision variable x_h for each hyperarc and continuous variables w_a for the linked resource flow, the resource flow based MILP-formulation of the RSRP can be stated follows:

$$\min \sum_{h\in H} c_h x_h, \tag{Flow}$$

$$s.t. \quad \sum_{h\in H(t)} x_h = 1 \qquad \forall t \in T, \tag{1}$$

$$\sum_{h\in H(v)^-} x_h = \sum_{h\in H(v)^+} x_h \qquad \forall v \in V, \tag{2}$$

$$w_a \leq \sum_{h\in H(a)} R\, x_h \qquad \forall a \in A, \tag{3}$$

$$\sum_{a\in A(v)^+} w_a - \sum_{a\in A(v)^-} w_a = \sum_{h\in H(v)^+} r(h)\, x_h \qquad \forall v \in V, \tag{4}$$

$$\sum_{a\in A(m)^+} w_a = \sum_{h\in H(m)^+} r(h)\, x_h \qquad \forall m \in M, \tag{5}$$

$$w_a \in [0, R] \subset \mathbb{Q}_+ \qquad \forall a \in A, \tag{6}$$

$$x_h \in \{0,1\} \qquad \forall h \in H. \tag{7}$$

The objective function of (Flow) minimizes the sum of the operational cost of all chosen hyperarcs. This includes all cost for operating a trip, deadhead trips, performing maintenances, and costs to penalize irregularities. The first constraints (1) ensure the covering of each trip. Equations (2) take care about the (hyper)flow conservation. The following four sets of constraints deal with the vehicle maintenance. First, the maintenance variables w were coupled to the hyperarc variables allowing only those to be used for which a hyperarc was chosen. Followed by equations (4) which ensure the correct summation of the maintenance resource consumption. The constraints (5) state the possibility to reset the resource flow at maintenance service locations. Finally, the variable domains are given by (6) and (7). The presence of the constraints (3)-(6) makes the model way more difficult to solve as it implicitly implies a tracing of each individual vehicle while the other parts of the model can be seen as a vehicle type or fleet based formulation. This is one of the reasons to be interested in novel powerful algorithmic approaches to solve these kinds of problems.

3.2 Infeasible Path Cut Separation Approach

Besides the mentioned reason for a different approach to tackle the RSRP with included vehicle maintenance, the fact that maintenance service locations are often closely located to overnight parking depots is another one. This fact leads to the situation that solutions of the RSRP without consideration of the maintenance are often trivially maintenance feasible, respectively could be easily made maintenance feasible by replacing overnight parking hyperarcs with maintenance hyperarcs. Therefore we developed a new integer programming model that replaces the continuous resource flow by a rough bound on the overall resource consumption and a set of infeasible path constraints to forbid maintenance infeasible sub-paths in the chosen cycles. This class of cuts is well known and studied in the (asymmetric) travelling salesman community as for example in [1]. However, we are not aware of any publication applying this technique to rolling stock scheduling. Thus, the following integer program relies completely on the binary variables for choosing hyperarcs to be part of the solution. To set up the model, we define by P the set of all maintenance infeasible sub-paths in the underlying directed graph $D = (V, A)$ of the hypergraph G.

$$\min \sum_{h \in H} c_h x_h, \tag{Cut}$$

$$s.t. \quad \sum_{h \in H(t)} x_h = 1 \qquad \forall t \in T, \tag{8}$$

$$\sum_{h \in H(v)^-} x_h = \sum_{h \in H(v)^+} x_h \qquad \forall\, v \in V, \tag{9}$$

$$\sum_{h \in H \setminus M} |h|\, r(h)\, x_h \leq \sum_{h \in M} |h| R\, x_h, \tag{10}$$

$$\sum_{h \in P_i} x_h \leq |P_i| - 1 \qquad \forall\, P_i \in P, \tag{11}$$

$$x_h \in \{0, 1\} \qquad \forall\, h \in H. \tag{12}$$

The objective function of (Cut) and the constraints (8), (9), and (12) are completely identical to the ones in the (Flow) formulation and model all technical aspects of the problem with the exception of the vehicle maintenance rules. These rules are implied by the other two sets of constraints. The first set (10) forces the solution to contain a sufficient number of maintenance service hyperarcs. Utilizing that the total resource consumption of the vehicles, i.e., the left hand side of constraints (10), is bounded from above by the number of maintenance arcs chosen times the product of the upper bound of the resource consumption and the number of maintained vehicles modelled by the chosen hyperarc. The obvious argument for this becomes clear if we divide (10) by R. Note that the total resource consumption of the vehicles is not fixed a priori due to deadhead trips. The second set (11) then forbids to include maintenance infeasible sub-paths into the cycles of hyperarcs of the solution.

Trivially, one critical aspect of this model formulation is that it might contain an exponential number of infeasible path constraints (11). Therefore we solve this model by adding the infeasible path cuts dynamically to the model during the solution process. Nevertheless, it is easy to see that for the formulation containing all infeasible path constraints the following lemma holds.

▶ **Lemma 2.** *Let X^{Flow}, X^{Cut} be the sets of all integer feasible solutions of* Cut*, respectively* Flow *restricted to the x variables. It holds*

$$conv(X^{Flow}) = conv(X^{Cut}).$$

3.2.1 Dynamic Infeasible Path Constraint Separation

As mentioned before, the idea behind the algorithm is to generate the infeasible path constraints 11, respectively to separate maintenance infeasible solutions of the Cut approach dynamically. Algorithm 1 provides the respective pseudo code. It works as follow: The algorithm starts with a solution H_x of the Cut formulation, without any infeasible path constraint so far, computed by a commercial mixed integer solver with a limit on the optimal IP tolerance of $\frac{\epsilon}{2}$. We denote the problem Cut_P with $P := \emptyset$ in the following as *maintenance relaxation* of the original problem. It gives the algorithm a feasible solution to the RSRP without considering the maintenance constraints and therefore a valid lower bound on the objective function (trivially, since each maintenance feasible solution is still contained in the solution space). After that, the cycles of the directed arcs contained in the chosen hyperarc variables were constructed. By tracking each cycle once, beginning at an arbitrarily chosen maintenance arc, the algorithm checks the maintenance feasibility of the solution by summing up the resource consumption along the cycle resetting it every time a maintenance arc is passed. If the sum of the aggregated mileage of the arcs exceeds the upper bound R of the resource, the chosen cycle is proven to be infeasible and a valid infeasible (sub-)path constraint is generated automatically by the set of hyperarcs passed since the last maintenance arc, i.e., including the arc itself. We denote this set by $P_i \subseteq H$. In a maintenance feasible solution it is not possible to chose all $|P_i|$ hyperarcs from this set. The algorithm collects all of these constraints, denoted by $\hat{P}$ that could be generated from H_x. If no infeasible path constraint is generated, the solution is maintenance feasible and optimal since the objective function value equals the lower bound of the maintenance relaxation. In the opposite case where H_x was proven to be maintenance infeasible a neighborhood search is applied to substitute hyperarcs of the solution with their counterparts that include additional maintenance services to construct a maintenance feasible solution $\hat{H}_x$. Therefore we define two hyperarcs to be maintenance equivalent by the definition given in 3. In a nutshell, two hyperarcs are maintenance equivalent, denoted by $h \simeq g$, if they model the same hyperarc with and without performing a maintenance service in between. With this definition we can set up the so called *Maintenance Assignment Model (MAM)* as shown in MAM. This model is then solved by a commercial mixed integer solver. If it is feasible the algorithm constructs a solution $\hat{H}_x$ that is feasible to Cut_P and Cut as well. This solution is then checked whether its quality in sense of the gap between its objective function value and the best known lower bound is small enough, i.e., smaller than ϵ. If this is the case the algorithm terminates returning $\hat{H}_x$ as found solution. Otherwise Cut_P, respectively P is updated by $P = P \cup \hat{P}$. The algorithm then restarts the path separation loop by solving Cut_P until a fixed number of iterations are passed or a sufficiently good solution was found.

```
Input : Hypergraph $G$, resource function $r : H \mapsto \mathbb{Q}$, resource limit $R$
Output: Maintenance feasible solution $x \subseteq H$
{
  $x := \emptyset$
  $i := 0$; $P := \emptyset$
  do
  {
    $H_x :=$ MILPSolve($Cut_P$);
    $\hat{P} :=$ generateInfeasiblePathCuts($H_x$);
    $P := P \cup \hat{P}$; % update infeasible path cut set
    if($\hat{P} = \emptyset$ or $H_x = \emptyset$)
    {
      return $H_x$ % maintenance feasible and optimal or infeasible
    }
    else
    {
      $\hat{H}_x$ := solveMAM($H_x$);
      if($\hat{H}_x \neq \emptyset$)
      {
        %Solution $\hat{H}_x$ is a maintenance feasible solution for $Cut$
        $x := \hat{H}_x$
      }
    }
    $i++$;
  }
  while( $\frac{c(x)-c(H_x)}{c(H_x)} < \epsilon$ or $i < I$ )

  return $x$
}
```

Algorithm 1 Infeasible Path Constraint Separation Algorithm.

3.2.2 The Maintenance Assignment Model

To perform the neighborhood search for a maintenance infeasible solution H_x of the Cut_P model in the cut separation loop, the mixed integer programming model defined in MAM is solved. To set up the model, we formally define the maintenance equivalence relation for two hyperarcs as follows.

▶ **Definition 3.** *Two hyperarcs $h, g \in G$ are* `maintenance equivalent` *($h \simeq g$) if and only if $A(h) = A(g)$.*

The MAM-model contains a binary decision variable x_h for each hyperarc $h \in H$ that is either included ($h \in H_x$) or maintenance equivalent to a hyperarc included in the actual solution ($h \simeq g \in H_x$). It also contains a continuous resource flow variable $w_v \in [0, R]$ for each node of V that is part of a chosen hyperarc $h \in H_x$.

$$\min \sum_{h \simeq h_x \in H_x} c_h x_h, \tag{MAM}$$

$$s.t.: \sum_{h \simeq h_x} x_h = 1 \qquad \forall h_x \in H_h, \tag{13}$$

$$w_a + \sum_{h \simeq h_x} r(h)\, x_h \leq w_b \qquad \forall\, (a,b) \in h_x, h_x \in H_x, \tag{14}$$

$$x_h \in \{0,1\} \quad \forall\, h \in H_x, \tag{15}$$

$$w_a \in [0,R] \quad \forall\, (a,b) \in h, h \in H_x. \tag{16}$$

The objective function of (MAM) minimizes the sum of the operational cost of all chosen hyperarcs in exactly the same way it is done in (Flow) or (Cut). The constraints (13) define that either a hyperarc that does or does not perform a maintenance service for each arc contained in a cycle of the solution H_x is chosen. If a hyperarc that contains a maintenance service is chosen, the respective constraint (14) ensures that the associated values for the w values are reset. In the opposite case they ensure the correct propagation of the resource consumption values to the next w-variables along the cycle. The last two sets of constraints (15) and (16) define the variable domains. We remark that solutions of this model may contain an increased number of maintenance arcs than the original solution H_x. But, if the model is feasible the computed solution is maintenance feasible and therefore a feasible solution for *Cut*. For our practical instances these models are very small and easy to solve by a commercial state of the art mixed integer solver.

4 Computational Results

This section presents the computational results for the presented solution approaches on a set of real world instances of DB Fernverkehr AG. These instances contain different scenarios for rolling stock rotation problems for the ICE high speed train vehicles operated by DBF. They differ in the number of contained timetable trips, operated fleets, and characteristics of the maintenance rules. All instances contain between 200 and 400 timetabled trips and a maximum of two coupled vehicles to operate a trip. All computations were performed on an Intel® Xeon(R) E3-1245 v5 @ 3.50GHz CPU with eight cores and Gurobi 8.1 as LP and sub-MILP solver.

Table 1 compares the solution process of the LP-relaxation of Flow and Cut, both implemented in RotOR. The LP-relaxation is solved with an algorithm called Coarse-2-Fine Column Generation which is described in detail in [2, 10] and out of scope of the paper to be explained in detail here. At this stage the only difference between the two solution approaches is the set of constraints that is given to the solver, i.e, the LP model formulation of (Cut) with no infeasible path cuts respectively the LP model formulation of (Flow). The first column of Table 1 identifies the instance while the second column shows the total number of hyperarcs required to model the instance. The following two blocks of three columns each headlined with Cut, respectively Flow show the solution characteristics of the associated approach. In detail, *Obj.* columns show the objective function values of the two approaches (times a factor of $10^{-z}, z \in \mathbb{N}$), *CPU* columns give the total computation time of each approach in seconds, and the columns headlined with *Columns* mark the number of generated columns, respectively variables required to solve the LP-relaxation. The Cut approach shows a significant speed up and a lower number of generated columns for each of the instances, but for the price of a slightly weaker LP-bound.

Table 1 Computational results for the LP-relaxation of Cut and Flow.

		Cut			Flow		
Id	$\|H\|$	Obj.	CPU(s)	Columns	Obj.	CPU(s)	Columns
1	229292	7062	5	16683	7064	16	30487
2	116206	6598	7	10881	6600	14	19496
3	1813512	7853	15	56118	7779	55	109303
4	1686124	8170	20	61276	8164	52	82726
5	275356	9340	8	16301	9288	13	21959
6	275356	8886	6	15009	8793	12	22359
7	363132	8976	9	18898	8994	13	25052
8	1452528	9011	30	76346	9007	47	80599
9	471056	10907	11	20883	10868	43	42830
10	471056	11229	15	23027	11216	40	40297
11	229634	7181	13	19829	7181	19	30109
12	226340	7204	6	14661	7212	17	31758
13	229634	7120	7	19501	7104	18	32360
14	226340	7044	5	16631	6977	18	33647

Table 2 Computational results for Cut, Flow solved with RotOR, and Flow solved with Gurobi.

		Cut		RotOR		Gurobi	
Id	$\|H\|$	Obj.	CPU(s)	Obj.	CPU(s)	Obj.	CPU(s)
1	229292	7087	301	7105	45	7090	48
2	116206	6622	6	6619	1	6610	17
3	1813512	7879	22	7937	190	7866	84
4	1686124	8220	78	8230	181	8221	352
5	275356	9405	9	9439	146	9401	18
6	275356	8961	17	8981	199	8939	195
7	363132	9079	19	9055	122	9048	145
8	1452528	9068	292	9091	255	9038	3390
9	471056	11099	23	11006	210	10977	327
10	471056	11350	102	11388	50	11336	47
11	229634	7182	7	7182	20	7182	20
12	226340	7212	6	7212	20	7212	18
13	229634	7183	40	7146	112	7147	26
14	226340	7081	16	7059	72	7116	46
$\sum$			939		1635		4734
geometric mean			27.68		81.9		77.5

In the following three different solution approaches to compute an integer solution for the RSRP were compared. The first approach is the Infeasible Path Constraint Separation Algorithm described in Section 3.2.1. The second one is the currently used algorithm in RotOR at DBF which is explained in detail in [2, 10]. In a nutshell, this algorithm makes use of a heuristic start solution and sophisticated problem specific branching rules fixing different types of variables at different times of the solution process to solve the remaining sub-MILPs by Gurobi. The third approach is the default version of Gurobi to solve the Flow-MILP-formulation. The three different approaches to solve the integer formulation of Cut or Flow were restricted to the variables generated during the column generation process. The numbers of generated variables are shown in Table 1. The RotOR and the Gurobi approach solve exactly the same MIP model containing the variables generated for the Flow model.

Table 2 shows the characteristics of the solutions computed by the three different approaches. The first column identifies the instance while the second column shows the total number of hyperarcs required to model the instance. The next two columns show the objective function values and computation times for each instance using the Infeasible Path Constraint Separation Approach, followed by the same values ordered in two columns for RotOR's solution approach. Finally the objective function values and computation times of the default version of Gurobi are shown in the last two columns. For all solution approaches a limit of 1% on the gap between the best known upper and lower bounds was applied. All objective function values are shown times a factor of $10^{-z}, z \in \mathbb{N}$. The values for the objective function values of all three approaches show that each of them is able to find solutions within the desired bounds. Although, struggeling on two instances the Infeasible Path Constraint Separation Algorithm shows promising run times for the set of instances, especially for the largest instances $3, 4$ and 8. The two instances 1 and 13 are instances with a rather small upper bound on the maintenance resource which leads to an increased number of maintenances in the final IP-solution compared to the LP-relaxation.

5 Conclusion

In this paper we presented an optimization algorithm based on infeasible path constraints to deal with the Rolling Stock Rotation Problem with integrated vehicle maintenance. The algorithm is capable to deal with practical sized real world instances. It shows promising results in terms of solution quality and computation time. For future research it might be interesting to generate a set of cuts in beforehand of the solution process or to couple cut generation to certain aspects of the different maintenance rules.

References

1 Norbert Ascheuer, Matteo Fischetti, and Martin Grötschel. A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks: An International Journal*, 36(2):69–79, 2000.

2 Ralf Borndörfer, Markus Reuther, and Thomas Schlechte. A Coarse-To-Fine Approach to the Railway Rolling Stock Rotation Problem. In *14th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 42, pages 79–91, 2014. doi:10.4230/OASIcs.ATMOS.2014.79.

3 Ralf Borndörfer, Markus Reuther, Thomas Schlechte, Kerstin Waas, and Steffen Weider. Integrated Optimization of Rolling Stock Rotations for Intercity Railways. *Transportation Science*, 50(3):863–877, 2015. doi:10.1287/trsc.2015.0633.

4 Valentina Cacchiani, Alberto Caprara, and Paolo Toth. Solving a real-world train-unit assignment problem. *Mathematical Programming*, 124(1):207–231, July 2010. doi:10.1007/s10107-010-0361-y.

5 Jean-François Cordeau, François Soumis, and Jacques Desrosiers. Simultaneous Assignment of Locomotives and Cars to Passenger Trains. *Oper. Res.*, 49:531–548, July 2001. doi:10.1287/opre.49.4.531.11226.

6 Giovanni Luca Giacco, Andrea D'Ariano, and Dario Pacciarelli. Rolling stock rostering optimization under maintenance constraints. *Journal of Intelligent Transportation Systems*, 18(1):95–105, 2014.

7 Richard M. Lusby, Jørgen Thorlund Haahr, Jesper Larsen, and David Pisinger. A Branch-and-Price algorithm for railway rolling stock rescheduling. *Transportation Research Part B: Methodological*, 99:228–250, 2017. doi:10.1016/j.trb.2017.03.003.

8 Gábor Maróti and Leo Kroon. Maintenance Routing for Train Units: The Transition Model. *Transportation Science*, 39:518–525, November 2005. doi:10.1287/trsc.1050.0116.

9 Gábor Maróti and Leo G. Kroon. Maintenance routing for train units: The interchange model. *Computers & OR*, pages 1121–1140, 2007.

10 Markus Reuther. *Mathematical optimization of rolling stock rotations*. PhD thesis, TU Berlin, 2017. doi:10.14279/depositonce-5865.

11 Per Thorlacius, Jesper Larsen, and Marco Laumanns. An integrated rolling stock planning model for the Copenhagen suburban passenger railway. *Journal of Rail Transport Planning & Management*, 5(4):240–262, 2015.

12 Joris C. Wagenaar, Leo G. Kroon, and Marie Schmidt. Maintenance Appointments in Railway Rolling Stock Rescheduling. *Transportation Science*, 51(4):1138–1160, 2017. doi:10.1287/trsc.2016.0701.

13 Takuya Shiina Jun Imaizumi Yuuta Morooka, Naoto Fukumura and Susumu Morito. Rolling Stock Rostering Optimization Based on the Model of Giacco et al.: Computational. Evaluation and Model Extensions. In *7th International Conference on railway operations modelling and Analysis (RailLille 2017)*, pages 709–725, 2017.