Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

RALF BORNDÖRFER MARKUS REUTHER
THOMAS SCHLECHTE STEFFEN WEIDER

Integrated Optimization of Rolling Stock Rotations for Intercity Railways

This work was funded by DB Fernverkehr AG.

ZIB-Report 12-39 (November 2012)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Abstract

This paper provides a highly integrated solution approach for rolling stock planning problems in the context of intercity passenger traffic. The main contributions are a generic hypergraph based mixed integer programming model and an integrated algorithm for the considered rolling stock rotation planning problem. The new developed approach is able to handle a very large set of industrial railway requirements, such as vehicle composition, maintenance constraints, infrastructure capacity, and regularity aspects. By the integration of this large bundle of technical railway aspects, we show that our approach has the power to produce implementable rolling stock rotations for our industrial cooperation partner DB Fernverkehr. This is the first time that the rolling stock rotations at DB Fernverkehr could be optimized by an automated system utilizing advanced mathematical programming techniques.

1 Introduction

Deutsche Bahn Fernverkehr AG provides the largest intercity railway service in Europe. In order to implement their timetables *rolling stock rotations* are built to operate passenger trips by rail vehicles, which are among the most expensive and limited assets of a railway company. The main challenge arising in planning the operations of the rolling stock is to integrate the treatment of different technical aspects.

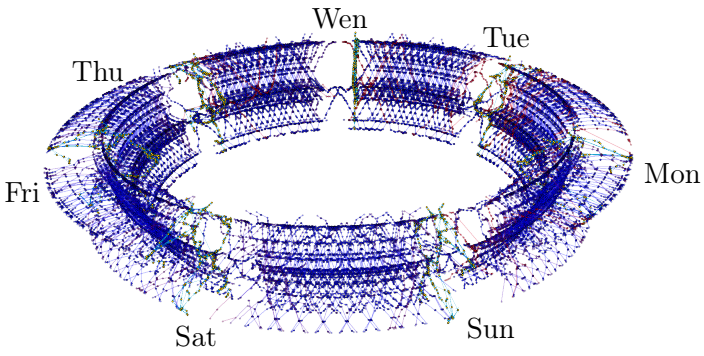

Figure 1: Almost periodic timetable.

The major requirements can be summarized as *vehicle composition rules*, *maintenance constraints*, *infrastructure capacities*, and *regularity stipulations*. Each of these requirements is already complex in its own right. Moreover, it is almost impossible to treat them sequentially, *i.e.,* a step by step

approach leads to infeasibilities or inefficient results for most of all real-world scenarios with consider. The most important requirements are the following:

- Figure 1 shows a cyclic timetable that is valid on seven operating days. For each day there are plotted all given passenger trips as blue time expanded paths. A profile at a specific time of this torus represents the current location of all vehicles operating the timetable. As one can see in this picture a common structure of railway timetables is that they are almost periodic. Only a few of the given passenger trips differ from day to day. That implies another requirement for the rolling stock rotations: They should utilize the periodicity of the timetable; this objective is called *regularity*.

- Another main characteristic of nearly all railway systems is that rail vehicles can be combined to form *vehicle compositions*. Therefore our model provides generic structures to formulate those detailed requirements in an integrated manner.

- The rolling stock has to be maintained frequently. This leads to several maintenance constraints, with different technical backgrounds. We consider cumulative time and distance resources which are constrained by predefined bounds. To comply to those bounds rail vehicles must be maintained in periodical intervals. We show a novel Mixed Integer Programming formulation especially for this type of constraints and new algorithmic method to solve the resulting model.

- Maintenance and also parking activities usually consume infrastructure and crew capacity. This capacity is limited and therefore those considerations have to be integrated.

The paper contributes a new generic Mixed Integer Programming approach to optimize rolling stock rotations. We present a hypergraph based formulation modeling all technical requirements as well as new algorithmic methods to solve very hard problem instances of large scale.
Comprehensive computational results prove that our model and algorithm produces high quality and implementable results. Rotation planners of Deutsche Bahn validated the resulting rolling stock rotations from a detailed technical and operational point of view.
The paper is organized as follows. Section 2 gives an overview of the known approaches for rolling stock optimization. Section 3 defines the considered problem from an abstract mathematical point of view by introducing a formal description of the RSRPP. We embed the rolling stock optimization

problem of our industrial partner into our modeling framework in Section 4. Section 5 presents our algorithmic Mixed Integer Programming formulation for the RSRPP. In Section 6 we present or algorithmic procedure to solve the RSRPP. Computational results for instances given by our industrial partner DB Fernverkehr AG are presented in the final Section 7.

2 Related work

Vehicle scheduling is extensively discussed in the literature, see [9] for a survey. The paper [1] present a mixed integer programming formulation for a locomotive scheduling problem. The model is solved by a very large-scale neighborhood search technique but does not include any maintenance constraints. Savings of over 400 locomotives resulting in over one hundred million dollars annually are reported. In [13] is developed a large-scale non-linear integer programming formulation for the integrated optimization of locomotive schedules including maintenance constraints. The proposed model is solved by a Dantzig-Wolfe decomposition within a Branch-and-Bound framework. The authors of [6] propose an integer programming model based on a multi commodity flow formulation for the integrated assignment of locomotives and passenger cars to passenger trips. Maintenance constraints are taken into account by using a time-expanded graph model. Various decomposition techniques embedded in a Branch-and-Bound-and-Cut framework are utilized to solve the problem. A three stage heuristic approach to incorporate maintenance tasks in precomputed rolling stock rosters is described in [2]. Furthermore two integer programming formulations which can be used to pre-optimize rolling stock rosters to incorporate maintenance tasks can be found in [10] and [11]. In duty rostering problems there are several constraints on the maximal working time per week and the maximal number of successive working days of the drivers. These constraints are very similar to the maintenance constraints in the RSRPP. Behrendt introduced several mixed integer programming formulations for the duty rostering problem in [3] including constraints on cumulative resources, *e.g.,* working time. We will present an adaption of one of those models for the treatment of maintenance constraints for the RSRPP. Our adaptation of the formulation for the maintenance requirements proposed in [3] is mathematically equivalent to the model developed independently in [8]. The authors of [8] reported preliminary computational results for rather small and simplified scenarios of the Italian railway company Trenitalia. They assume that the railway timetables are repeat on every operation day, they do not consider a realistic

objective function (only the number of vehicles is minimized), and moreover they do not integrate vehicle composition, regularity, and infrastructure capacity aspects in their model.

3 The Rolling Stock Rotation Planning Problem

The Rolling Stock Rotation Planning Problem comes from one of the most complex railway applications and therefore it is necessary to describe the main mathematical aspects of this problem as short as possible. Therefore, one main contribution of our work is a generic description of all needed mathematical structures and relations to be able to handle all technical details arising in industrial use cases. In this section we will provide an abstract mathematical description of the RSRPP and we are showing some well known problem reductions to illustrate the generic manner of our considerations.
The set of timetabled passenger trips is denoted by T. A trip $t \in T$ has a departure and arrival time in our standard week. In a solution of the RSRPP one has to perform *maintenance tasks* on vehicles. Let M be the set of all possible maintenance tasks. A *service* $s \in S$ is a non-empty set of maintenance tasks. We say that $s \in S$ *implements* $m \in M$ if $m \in s$.
Let V be a set of *nodes* representing departures and arrivals of vehicles operating passenger trips of T and let $A \subseteq (V \cup S)^2$ be a set of directed standard arcs. We define a set $H \subseteq 2^A$, called *hyperarcs*. The RSRPP *hypergraph* is denoted by $G = (V \cup S, A, H)$. From a high level point of view, one could say that A models what is possible to do for vehicles while H models what is possible to decide for the RSRPP at once.
The standard arc $a = (u, v) \in A$ *operates* a trip $t \in T$ if $u \in V$ represents the departure of t and $v \in V$ represents the arrival of t. We say that the hyperarc $h \in H$ *covers* $t \in T$, if each arc $a \in h \subseteq A$ operates t. We define the set of all hyperarcs covering t as $H(t) := \{h \in H \mid h \text{ covers } t\}$. A *maintenance constraint* l is represented by a resource function $r_l : S \cup A \mapsto \mathbb{Q}_+$, a *resource upper bound* $U_l \in \mathbb{Q}+$, and a set of maintenance tasks $m_l \subseteq M$. A maintenance task of m_l must be performed to *reset* the resource r_l to fulfill the bound U_l. A *feasible path* $P \subseteq A$ in G w.r.t. the maintenance constraint l is a simple path starting and ending at nodes of S (which implement an appropriate maintenance task of $m_l \subseteq L$) resetting the resource of l such that

$$\sum_{S(P)} r_l(v) + \sum_{a \in P} r_l(a) \leq U_l.$$

The inequality (1) states that the sum of all consumed resources on a feasible path P has to be smaller then or equal to the bound of a maintenance constraint. Note that even a service $s \in S(P) \subset S$ can consume resources if it does not implement maintenance tasks for each constraint. A *feasible rotation* w.r.t. the maintenance constraint l is a cycle $C \subseteq A$ such that each node covered by C is contained in a feasible path w.r.t. l. Performing a maintenance tasks consumes some commodities, *i.e.,* crew, machines, and infrastructure. Those commodities have usually a limited availability. A *base constraint* b is represented by a resource function $r_b : H \mapsto \mathbb{Q}$ and a *capacity bound* $U_b \in \mathbb{Q}$. The set of all base constraints is denoted by B. We say that the set of hyperarcs $H_0 \subseteq H$ fulfills the base constraint $b \in B$ if $\sum_{h \in H_0} r_b \leq U_b$. Now we have defined all needed terms to state the RSRPP problem.

Definition 1 *(Rolling Stock Rotation Planning Problem (RSRPP))*
Let T be a set of timetabled passenger trips and let $G = (V \cup S, A, H)$ be a RSRPP hypergraph with a cost function $\mathbf{c} : H \mapsto \mathbb{Q}_+$. Furthermore let L be a set of maintenance constraints and let B be a set of base constraints.
The RSRPP is to find a cost minimal set of hyperarcs $H_0 \subseteq H$ such that

- *Each timetabled trip $t \in T$ is covered by exactly one hyperarc $a \in H_0$.*
- *The set $\bigcup_{a \in H_0} a$ is a set of feasible rotations w.r.t. all maintenance constraints of L.*
- *The set H_0 fulfills all base constraints of B.*

The RSRPP has some interesting relations to well known combinatorial optimization problems from the literature. If we assume that all arcs of the set H have cardinality one, we call the resulting problem the *non-hyper restriction* of the RSRPP and if we assume that the RSRPP does not have any maintenance constraints and also does not have any base constraints, we call the relaxed problem the *non-maintenance* relaxation of the RSRPP. If we consider the non-hyper restriction which is also a non-maintenance relaxation of the RSRPP, the problem reduces to an integer multi commodity flow problem, which is known to be $\mathcal{NP}$-hard, see [9]. In this reduction the commodities are represented by the sets $H(t)$. Therefore, the RSRPP is also a $\mathcal{NP}$-hard combinatorial optimization problem. In addition, if we assume that each set $H(t)$ has cardinality one, the problem reduces to the standard assignment problem. The non-hyper restriction of the RSRPP with exactly one maintenance constraint is closely related to a variant of the vehicle routing problem, *i.e.,* a route can be seen as a fleet, see [7].

4 The RSRPP for intercity railway planning

In this section we motivate our formalism introduced in Section 3 and how it is related to real-world industrial instances for intercity railway planning.

4.1 Vehicle composition

The types of basic vehicle units are called *fleets*. Let F be the set of fleets and C be the set of vehicle compositions. A *vehicle composition* $\texttt{config} \in C$ is a multiset of fleets of F. The set $C(t) \subseteq C$ denotes the set of feasible vehicle compositions to cover the timetabled trip $t \in T$.
The relation of fleets and vehicle compositions plays one of the most important role in intercity planning. This is because rail vehicles can be coupled together. In intercity problems even *on the fly*. This means that no technical equipment or crew is needed for a coupling. There exist fleets at Deutsche Bahn for which the coupling time does not exceed ten minutes. Coupling activities create a huge number of degrees of freedom in intercity planning. There are a lot of technical rules regarding to this coupling activities, *e.g.,* rules for the position of vehicles in a composition. In this paper we do not explicitly consider rules for, *e.g.,* positions of vehicles in vehicle compositions. But we point out that since our model is directly based on decisions for coupled vehicles (*i.e.,* hyperarcs) it is able to handle all technical requirements arising in this context in an integrated manner.
Figure 2 shows how fleets and vehicle compositions are related and how they are modeled in the RSRPP. The picture shows three timetabled trips $t_1, t_2, t_3 \in T$. All red and blue circles are nodes of the node set V of the RSRPP hypergraph $G = (V, A, H)$, *i.e.,* departures or arrivals of physical vehicles of the three timetabled trips. The colors of the circles indicate two fleets – a red and a blue one. As one can imagine, in intercity rotation planning it is not allowed to connect nodes of different fleets in the RSRPP hypergraph and therefore this picture illustrates also the reduction of the RSRPP to an integer multi commodity flow problem which was described in Section 3. Here the fleets are the commodities.
The hyperarcs $h_1, h_2, h_3, h_4 \in H$ form the set $H(t_1)$, *i.e.,* the set of alternative hyperarcs which can be used to cover the timetabled trip t_1, *i.e.,* $H(t_1)$ represents the set $C(t_1)$ of feasible vehicle compositions to operate t_1. From a practical point of view this can be seen as follows: It is feasible to cover t_1 by a single vehicle of the red or blue fleet, see arcs h_1 and h_2. But it is also feasible to *haul* up to two vehicles by operating t_1, see arcs h_3 and h_4. As mentioned above, *e.g.,* the position and orientation of vehicles in vehicle compositions

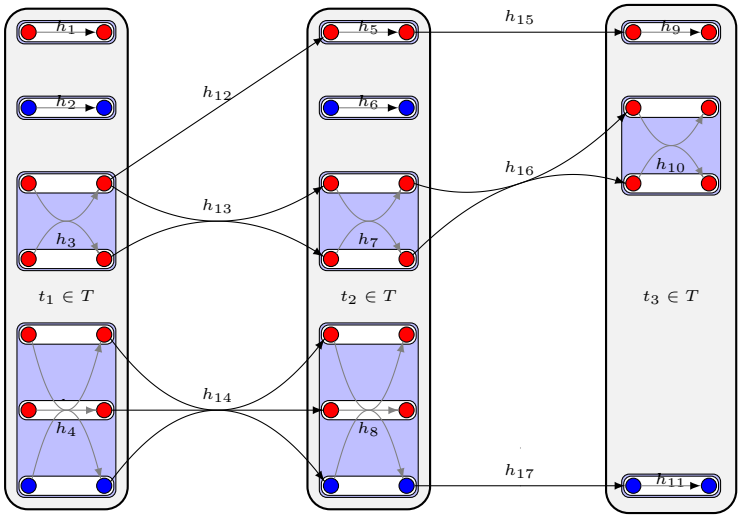

Figure 2: Hypergraph model.

must be also taken into account. This can be easily done by extending this example, *i.e.,* declaring the nodes as states w.r.t. position of vehicles at departures and arrivals of trips. While the arcs $h_1, h_2, \ldots, h_{11}$ model how the trips can be covered, the arcs $h_{12}, h_{13}, \ldots, h_{17}$ model how the trips can be connected to build a set of feasible rotations. Arc $h_{12} \in H$ implements a coupling activity after the arrival of t_1. The hyperarcs $h_{13}, h_{14}, h_{15}, h_{16} \in H$ model connections between trips without coupling activities. For intercity planning is very important to separately consider the decisions whether to connection All hyperarcs connecting trips can also be interpreted as hyperarcs modelling service paths (see Sub-section 4.2). To consider explicitly hyperarcs that model a connection makes use able to model that allocating a track for two coupled vehicles is much cheaper than allocating a track for two vehicles individually and also to integrate costs for coupling and decoupling of rail vehicles.
Note that it is important for the RSRPP that hyperarcs are defined as sets of standard directed arcs because the set of rotations must be well defined for the exact treatment of the maintenance constraints, which we consider in the next sub-section.

4.2 Maintenance constraints

To model these requirements, we consider each possible path of length two of the form $\{(v,s),(s,w)\} \subseteq A$ with $s \in S$ and $v, w \in V$, called *service path*, as a single hyperarc $h = \{(v,s),(s,w)\} \in H$. We call h a *replenishment arc*. Because we assume that our objective function is non-negative, we do not have to consider any other structures where service nodes appear, *e.g.,* cycles, paths, or loops of service nodes.

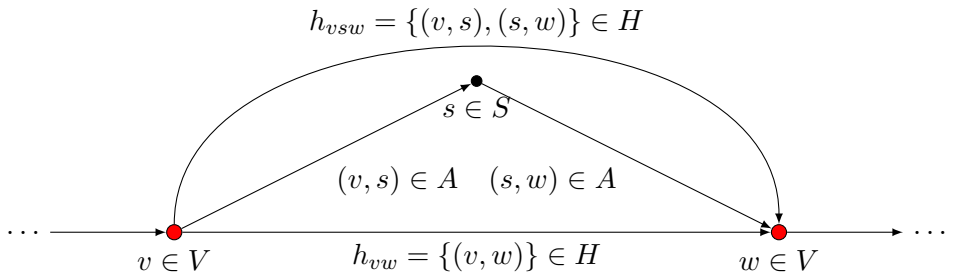

Figure 3: Representation of service paths.

Figure 3 illustrates the treatment of service paths in our RSRPP hypergraph. The hyperarc h_{vw} models a direct connection of the arrival node $v \in V$ and the departure node $w \in V$. These connections can include deadhead trips (*i.e.,* empty rides) if the involved locations are different. The service path $\{(v,s),(s,w)\} \subseteq A$ is represented by the hyperarc $h_{vsw} \in H$. It models that a vehicle composition arrives at v, traverses all maintenance tasks, which are implemented in the service $s \in S$, and finally departs on w. Also h_{vsw} can include several deadhead trips.

4.3 Regularity

As mentioned in Section 1, we focus in this paper on a cyclic planning horizon over one *standard week*. The structure of the given timetable is *almost* periodic. Only few trips of the timetable differ over the single week days of the standard week. In view of this structure, it is desirable to construct a vehicle rotation plan which utilizes this periodicity. We call such a plan a *regular* vehicle rotation plan.

Imagine that we are given a timetable for that each trip repeats every day in the standard week as it was considered in [8]. We call such a timetable a *periodic timetable* and we call a set of repeating trips in the standard week a *train*. The set of trains is denoted by $\mathfrak{T}$. A periodic timetable can

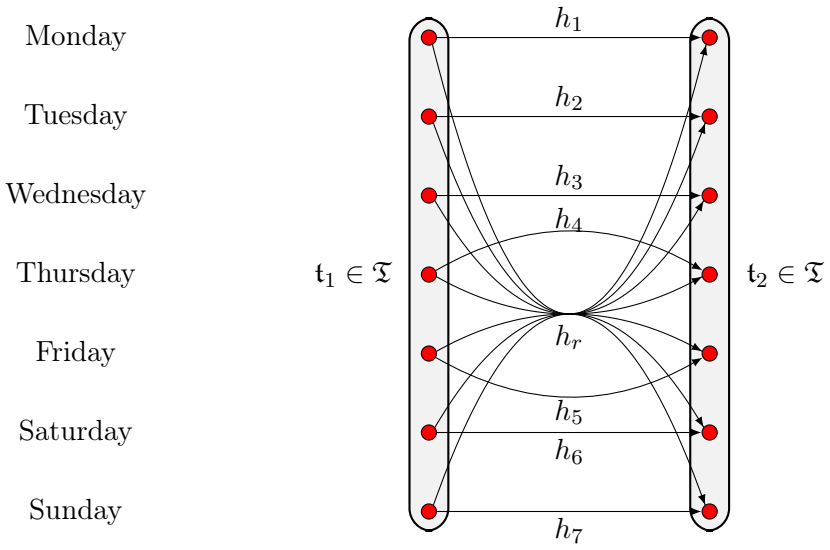

Figure 4: Hyperarc model for regularity.

also be considered as input data for the RSRPP. But in this case the set of trains $\mathfrak{T}$ can be viewed as the set of trips T (this is not quite accurate w.r.t. maintenance constraints). In case of a standard week with seven week days, this reinterpretation leads to a RSRPP hypergraph where the number of nodes is reduced by a divisor of seven. Since an instance of the RSRPP hypergraph is very dense (almost complete), the number of arcs reduces by a divisor of 49. A hyperarc in the RSRPP hypergraph for a periodic timetable can be seen as a set of hyperarcs if the trips are individually considered. This motivates our approach for integrating regularity aspects to the RSRPP. We easily construct a set of hyperarcs which are sets of other (individual) hyperarcs.

Figure 4 illustrates our regularity approach. The red circles for train $\mathfrak{t}_1 \in \mathfrak{T}$ can be seen as *equal* arrivals of each trip $t \in \mathfrak{t}_1 \subseteq T$. Equal means equal arrival locations and equal points in time in the standard week. The set of circles for train $\mathfrak{t}_2 \in \mathfrak{T}$ represents equal departures of all trips of $\mathfrak{t}_2$. The individual hyperarcs $h_1, h_2, \ldots, h_7$ may not be simultaneously chosen in a solution of the RSRPP. To express that this is desired, we create the hyperarc $h_r \in H$ as $h_r = \bigcup_{i=1}^{7} h_i \subseteq A$. From an applied point of view, the purpose of this regularity policy is to create rolling stock plans that are compactly representable, easy to communicate, and easy to operate. A more detailed description of the technical aspects of vehicle composition and regularity can

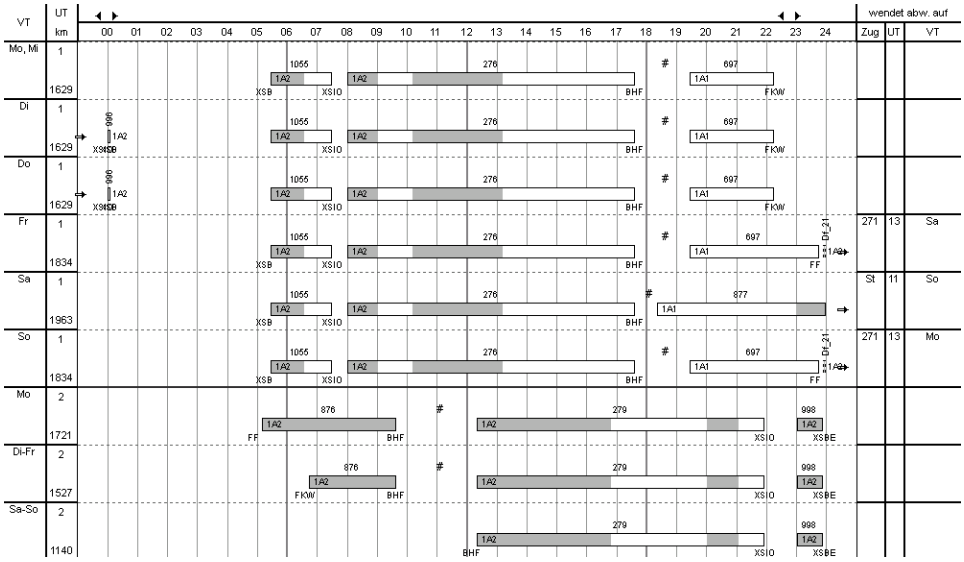

Figure 5: Solution representation at Deutsche Bahn Fernverkehr

be found in [5].

Figure 5 illustrates the regularity stipulations from the industrial point of view. For example, the vehicles operating trip 1055 is is assigned to the trip 276 at every day of operation. But some of the other connections are not regular. Those connections are chosen because of other parts of the objective function, *e.g.,* maintenance cost, and also because of timetabled trips which appear non-periodic. The pictured shows how a real rolling stock rotation plan looks like at Deutsche Bahn Fernverkehr AG.

4.4 Infrastructure capacity

On the left of Figure 6 one can see an example of a set of tracks of a parking area. We enumerate all feasible assignments of vehicle compositions for a single track by comparing the length of the vehicle compositions and the length of the tracks. This results in a capacity for each track which must not be exceeded at every date in our standard week. It can be observed, that bottlenecks of infrastructure capacity appear only at specific dates, *e.g.,* parking areas are almost empty during the day and almost completely exhausted at night. Therefore we integrate the consideration of infrastructure capacity by defining base constraints for specific dates. The right of Figure 6 illustrates how we deal with different fleets of rail vehicles. We define one base con-

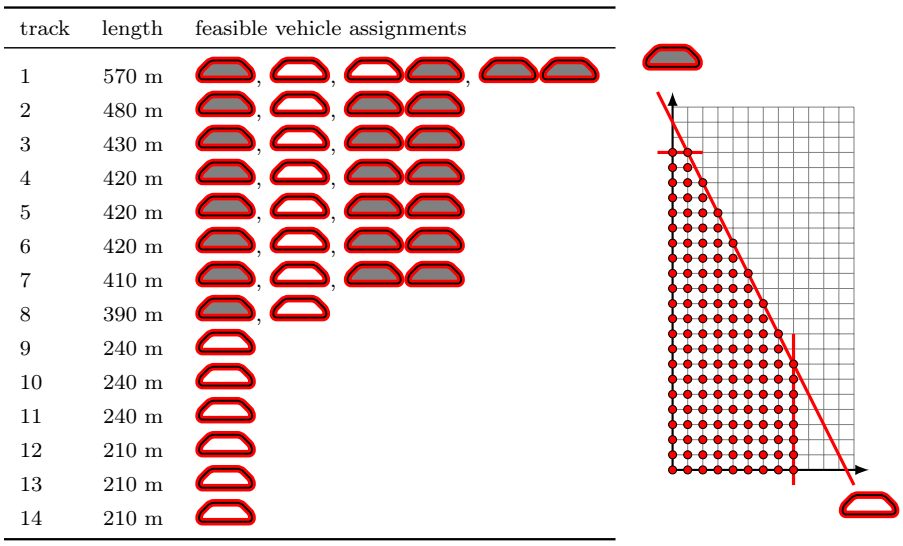

Figure 6: Capacity constraints for infrastructure capacity.

straint for each fleet independently and also one base constraint as a linear combination for all fleets, what is illustrated by the three red lines on the right.

4.5 Objective function

This sub-section is about the objective function $\mathbf{c} : H \mapsto \mathbb{Q}^+$ especially for intercity railway planning. Let $h \in H$ be a hyperarc, the cost $\mathbf{c}_h$ the sum of the following terms: $\mathbf{c}_h := r_h + l_h + k_h + \mathfrak{v}_h + t_h$. The first part of the objective function states a cost term for the number of irregularities, *i.e.,* r_h is higher for a hyperarc h that models only one connection than for a hyperarc that models several connections as explained in Sub-section 4.3. The term l_h denotes cost for possible deadhead distances modeled by the arc. Deadheads can be necessary to connect trips with different arrival and departure locations as well as trips though parking areas or service paths. If a hyperarc implements a service path, we integrate the cost for executing the service by k_h. The most important, and thus largest term, is denoted by $\mathfrak{v}_h$ that states the vehicle cost of a hyperarc. Monetary vehicle cost are given per fleet and we transform those cost by computing the *duration* of a hyperarc and calculating the amount of vehicles w.r.t. to the duration

of the standard week. Therefore the vehicle cost are distributed over all hyperarcs of a solution. For connections of timetabled trips we are given a so called *planned duration*. If the total duration of a connection is lower than it's planned duration we have to penalize this in our objective function. This very important for the German intercity systems. The data for those durations is very detailed and realistic, because Deutsche Bahn knows some bottlenecks in their operations which influence in particular the propagation of delays. We penalize those *planned turn time deviations* per minute by t_h.

5 Mixed Integer Programming formulation

Let $G = (V \cup S, A, H)$ be a given RSRPP hypergraph with a cost function $\mathbf{c} : H \mapsto \mathbb{Q}_+$, $\mathbf{c}(h) := \mathbf{c}_h$ as introduced in Section 3. Further, let L be a set of maintenance constraints with a resource function $r_l : S \cup A \mapsto \mathbb{Q}_+$ and a resource upper bound $U_l \in \mathbb{Q}+$ for each constraint $l \in L$. In addition, let B be a set of capacity constraints with a resource functions $r_b : H \mapsto \mathbb{Q}_+$ and capacity bounds $U_b \in \mathbb{Q}+$. A *RSRPP instance* is defined by all the previous defined data.
Let $t \in T$ be a trip and $a \in A$ be a set of standard arcs. We define the set $H(t)$ of arcs covering t and the set $H(a)$ of hyperarcs of a as $H(t) := \{h \in H \,|\, h \text{ covers } t\}$ and $H(a) := \{h \in H \,|\, a \in h\}$.
W.l.o.g. we assume that $H(a) \neq \emptyset$ for each arc $a \in A$. If $H(a) = \emptyset$, the standard arc a can never be contained in a feasible set of rotations. For a node $v \in V$ we define sets of incoming and outgoing (hyper-) arcs of v in the RSRPP hypergraph G as:

$$\begin{aligned} H(v)^{\text{in}} &:= \{h \in H \,|\, \exists\, a \in h \,:\, a = (u, v)\}, \quad A(v)^{\text{in}} := \{(u, v) \in A\}, \\ H(v)^{\text{out}} &:= \{h \in H \,|\, \exists\, a \in h \,:\, a = (v, w)\}, \quad A(v)^{\text{out}} := \{(v, w) \in A\}. \end{aligned}$$

We introduce a binary decision variable x_h for each hyperarc $h \in H$. In addition we define a non-negative continuous variable w_a^l for each standard arc $a \in A$ and each maintenance constraint $l \in L$ fulfilling the upper bound U_l.
Let $h \in H$ be a hyperarc and $a = (u, v) \in h$ a standard arc. To permit simple notation we define the resource consumption $r_l(v) := 0$ for a node $v \in V$. Further, we introduce $r_l^v(h) := r_l(\,(u, v) \in h\,) + r_l(v)$ as the sum of the resource consumption of the in v incoming standard arc $(u, v) \in h$ and the resource consumption of the node v. W.l.o.g. we assume that the

standard arc (u, v) is unique in h, because we the set of standard arcs of a hyperarc must forms a perfect matching between the tail and head nodes of this arc set. If this is not the case, there must be a node with two or more incoming/outgoing standard arcs and this can not be contained in a set of rotations.
The RSRPP can now be stated as a mixed integer program as follows:

$$\min \sum_{h \in H} \mathbf{c}_h x_h, \tag{MP}$$

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

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

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

$$\sum_{a \in A(v)^{\text{out}}} w_a^l - \sum_{a \in A(v)^{\text{in}}} w_a^l = \sum_{h \in H(v)^{\text{out}}} r_l^v(h) \, x_h \qquad \forall v \in V,\, l \in L \tag{4}$$

$$\sum_{a \in A(s)^{\text{out}}} w_a^l = \sum_{h \in H(s)^{\text{out}}} r_l^s(h) \, x_h \qquad \forall s \in S,\, l \in L \tag{5}$$

$$\sum_{h \in H} r_b(h) \, x_h \leq U_b \qquad \forall b \in B, \tag{6}$$

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

$$w_a^l \in [0, U_l] \subset \mathbb{Q}_+ \qquad \forall a \in A,\, l \in L. \tag{8}$$

The linear objective function minimizes the total cost and is directly related to the cost of operating a timetable. For each trip $t \in T$ the covering constraints (1) assign exactly one hyperarc of $H(t)$ to t. The equalities (2) are flow conservation constraints for each node $v \in V$ that imply the set of rotations in the arc set A. The subset of constraints (1), (2), and (7) state the non-maintenance relaxation of the RSRPP. This problem was already considered in [5]. The constraints (3), (4), and (5) ensure that the hyper-assignment is feasible w.r.t. all maintenance constraints $l \in L$.
Let the x variables be fixed to a feasible solution such that they imply a set of rotations. Inequalities (3) imply that a w_a^l can only be non-zero if a corresponding hyperarc of H consists of a standard arc which corresponds to a. Therefore the w-flow traverses the same set of cycles that the x-flow implies. Let $v \in V$ be a node and let $l \in L$ be a maintenance constraint.

Suppose that $a^{\mathtt{out}} \in A(v)^{\text{out}}$, $a^{\mathtt{in}} \in A(v)^{\text{in}}$, and $h \in H(v)^{\text{out}}$ are the in v outgoing and incoming (hyper-) arcs in the considered fixed x. Since $x_h = 1$ equation (4) for $v \in V$ and $l \in L$ reduces to:

$$w^l_{a^{\mathtt{out}}} = w^l_{a^{\mathtt{in}}} + r^v_l(h). \tag{9}$$

Equation (9) states that the resource flow value of an arc, namely $w^l_{a^{\mathtt{out}}}$, is always the sum of the flow value of the predecessor arc $w^l_{a^{\mathtt{in}}}$ and the actual resource consumption $r^v_l(h)$. In addition, equations (5) model the resource flow for service nodes. Differently to (4) we do not cumulate the in-flow for service nodes such that the cumulative flow for a maintenance constraint is replenished at service nodes.

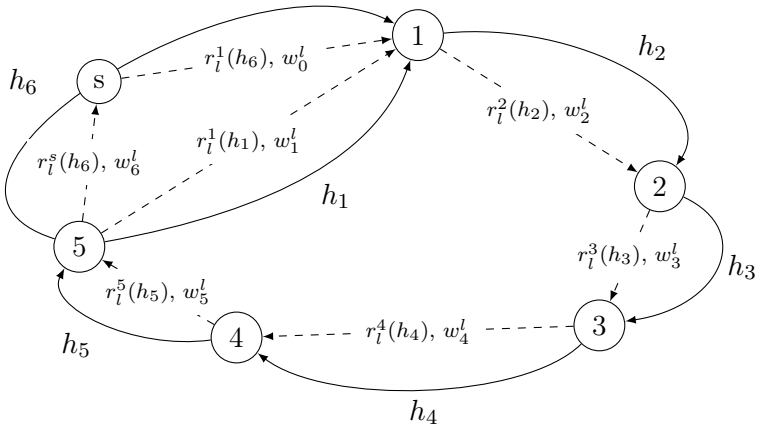

Figure 7: Example for resource flow constraints.

Figure 7 illustrates this considerations. Suppose we are given a hypergraph with five nodes and one service node as well as one hyperarc h_6 that implements a maintenance task for a maintenance constraint $l \in L$ at service node s. The dashed arcs illustrate the set of standard arcs and the not dashed arcs the set of considered hyperarcs. For this example the constraints (4)

and (5) are instanciated as follows:

$$
\begin{array}{lcll}
w_2^l & = & w_1^l + w_0 + & r_l^2(h_2) \cdot x_{h_2} \qquad ((4) \text{ for node } 1) \\
w_3^l & = & w_2^l + & r_l^3(h_3) \cdot x_{h_3} \qquad ((4) \text{ for node } 2) \\
w_4^l & = & w_3^l + & r_l^4(h_4) \cdot x_{h_4} \qquad ((4) \text{ for node } 3) \\
w_5^l & = & w_4^l + & r_l^5(h_5) \cdot x_{h_5} \qquad ((4) \text{ for node } 4) \\
w_1^l + w_6^l & = & w_5^l + & r_l^1(h_1) \cdot x_{h_1} + r^s(h_6) \cdot x_{h_6} \qquad ((4) \text{ for node } 5) \\
w_0^l & = & & r_l^1(h_6) \cdot x_{h_6} \qquad ((5) \text{ for node } s)
\end{array}
$$

Finally, the linear inequalities (6) are the canonical formulation of the base constraints.

6 Solving the RSRPP

In this section we describe a new developed algorithm to solve the mixed integer programming formulation of the RSRPP from Section 5. Our method consists of three main algorithmic features: A column generation approach, a problem specific local search heuristic, and an adaption of the rapid branching search scheme. In the following subsections we describe our algorithmic procedure with the goal to provide the main algorithmic ideas.
To permit consistent notation we introduce the following notation. The symbol $(\text{MP})_{\texttt{MIP}}(\overline{H}, \underline{H})$ for $\overline{H}, \underline{H} \subseteq H$ and $\overline{H} \cap \underline{H} = \emptyset$ denotes our mixed integer programming formulation from Section 5 for a given RSRPP instance with the additional bound constraints $x_h = 1$ for all $h \in \overline{H}$ and $x_h = 0$ for all $h \in \underline{H}$. Therefore $\overline{H}$ defines the set of fixed to one variables, while $\underline{H}$ defines the set of fixed to zero variables. Similarly, $(\text{MP})_{\texttt{LP}}(\overline{H}, \underline{H})$ denotes the LP relaxation of $(\text{MP})_{\texttt{MIP}}(\overline{H}, \underline{H})$, *i.e.,* the problem that results if we relax all integrality constraints. In the following, we assume that all variables defined by $\underline{H}$ are not contained in the denoted models, what is exactly the same as fixing the variables to zero. Using the introduced notation we aim to solve $(\text{MP})_{\texttt{MIP}}(\emptyset, \emptyset)$.

6.1 Hyperarc generation

Since the number of variables and rows of $(\text{MP})_{\texttt{MIP}}(\emptyset, \emptyset)$ is very large, even the solution of $(\text{MP})_{\texttt{LP}}(\emptyset, \emptyset)$ can not be computed for most of all large industrial problem instances using standard static LP solvers in a reasonable amount of time. Therefore we developed a column generation algorithm, called *hyperarc generation*.

The standard column generation approach to solve the LP relaxation of problems with very many columns, is to devise the overall model into a *master problem* and a sub-problem, called *pricing problem*. Then iteratively solve a restricted version (*i.e.,* not all variables are included) of the master problem, and decide depending on the dual solution vector, which variables to add to the master problem by solving the pricing problem. The method stops if it can be proven that there is no variable left which can improve the value of the objective function. We apply this scheme in the following way. The master problem is $(\text{MP})_{\texttt{LP}}(\emptyset, \emptyset)$ itself. We denote the restricted master problem by $(\text{MP})_{\texttt{LP}}(\emptyset, \underline{H})$. The pricing problem is to find at least one hyperarc $h \in \underline{H}$ with negative reduced cost. This problem can be solved by enumeration. But as known in the literature it heavily depends on the set of variables found in each pricing round how many iterations the overall procedure takes and how many columns are generated.
The main idea of our column generation approach is to combinatorial solve a reduction of our model with an adjusted objective function to clever decide which set of variables to add to the restricted master problem in each pricing round. The huge impact of this method can be explained by the fact, that the set of priced variables in each iteration contains also variables with positive reduced costs. This results in very less iterations, almost no stalling effect, and a small set of priced variables.
Consider the non-zero coefficients of the binary variable x_h for the hyperarc $h = \{a_1, a_2\} \subseteq A$ in the flow conservation constraints (2):

$$\underbrace{\begin{pmatrix} \cdots \\ 1 \\ \cdots \\ -1 \\ \cdots \\ 1 \\ \cdots \\ -1 \\ \cdots \end{pmatrix}}_{h} = \underbrace{\begin{pmatrix} \cdots \\ 0 \\ \cdots \\ 0 \\ \cdots \\ 1 \\ \cdots \\ -1 \\ \cdots \end{pmatrix}}_{a_1} + \underbrace{\begin{pmatrix} \cdots \\ 1 \\ \cdots \\ -1 \\ \cdots \\ 0 \\ \cdots \\ 0 \\ \cdots \end{pmatrix}}_{a_2}.$$

This indicates that a column vector of a hyperarc variable is the sum of the column vectors of the standard arcs of which it consists. Let $m \in \mathbb{Z}_+$ be the number of rows of our original model. For the hyperarc $h \in H$ let $A_h \in \mathbb{Q}^m$ be the column vector of the hyperarc variable x_h. As described above this column vector is the sum of the column vectors of the standard

arcs, *i.e.,* $A_h = \sum_{a \in h} A_a$. Further let $\pi \in \mathbb{Q}^m$ be an optimal dual solution vector of the restricted master problem. The reduced cost of x_h are given by $d_h := \mathbf{c}_h - \pi^T A_h$. We define the *partial reduced cost* $d_h(a) \in \mathbb{Q}$ of $a \in h \in H$ as

$$d_h(a) := \frac{\mathbf{c}_h}{|h|} - \pi^T A_a \tag{10}$$

The partial reduced cost $d_h(a)$ can be interpreted as the from the standard a in x_h induced reduced costs. We introduce binary decision variables x_a for each standard arc of A, compute the *minimal partial reduced cost* d_a for each standard arc a, *i.e.,* $d_a := \min_{h \in H(a)} d_h(a)$ for all standard arcs and solve the following problem:

$$\begin{aligned} \min \sum_{a \in A} d_a x_a, & & & \text{(APP)} \\ \sum_{a \in A(v)^{\text{in}}} x_a &= 1 & \forall v \in V, \\ \sum_{a \in A(v)^{\text{out}}} x_a &= 1 & \forall v \in V, \\ x_a &\geq 0 & \forall a \in A. \end{aligned}$$

This is an assignment problem (we make the underlying graph complete by introducing all missing arcs with an objective coefficient of zero) which can be solved by the Hungarian method. Note that since we formulated t his problem over the set of standard arcs, the problem (APP) usually decomposes into several smaller sub-problems. This can be seen by considering Figure 2 in Subsection 4.1. For this example, one has to solve the pricing problem (APP) for the red and blue fleets independently.

Algorithm 1 gives an overview of our column generation method. We are starting with a set of starting variables that consists of all looping hyperarcs of the hypergraph. We have to ensure that we are always solving a feasible model. That can easily ensured by introducing slack variables with sufficient high objective coefficients for the covering constraints (1).

Let $A_0 \subseteq A$ be the set of standard arcs which represent the optimal solution of our pricing problem (APP) w.r.t. a specific iteration. The set of variables we add, *i.e.,* removing those variables from $\underline{H}$, to the restricted master problem is $\bigcup_{a \in A_0} H(a) \subseteq H$. Thus, we add all hyperarcs that contain at least one in the solution of (APP) assigned standard arc. The column generation algorithm is exact, because we add also a small subset of all outgoing

Algorithm 1: `hyperarcGeneration()`

Data: A RSRPP problem instance.
Result: A set $\underline{H}$ s.t. $(\text{MP})_{\text{LP}}(\emptyset, \underline{H})$ and $(\text{MP})_{\text{LP}}(\emptyset, \emptyset)$ have equal optimal objective value.

1 **set** $\underline{H} := H \setminus \{h \in H \,|\, \forall a \in h \,:\, a = (v, v),\, v \in V\}$;
2 **while** $\{h \in \underline{H} \,|\, d_h < 0\} \neq \emptyset$ **do**
3 **compute** optimal dual solution vector π for $(\text{MP})_{\text{LP}}(\emptyset, \underline{H})$;
4 **compute** partial reduced cost d_a depending on π for each $a \in A$;
5 **compute** optimal solution $A_0 \subseteq A$ of (APP);
6 **set** $\underline{H} := \underline{H} \setminus \bigcup_{a \in A_0} H(a)$;
7 **foreach** $v \in V$ **do**
8 **compute** $h_1, h_2, \ldots, h_n, \ldots, h_{|H(v)^{\text{out}}|}$ such that $d_{h_i} \leq d_{h_j} < 0$ for $i < j < n$;
9 **set** $\underline{H} := \underline{H} \setminus \{h_1, \ldots, h_{\lceil \sqrt[3]{n} \rceil}\}$;

hyperarcs for a specific node with negative reduced costs to the restricted master problem in each iteration (see line 9). Therefore we can not miss any hyperarc with negative reduced cost.
Let $(\text{MP})_{\text{LP}}(\emptyset, \underline{H})$ be our restricted master problem. We only have to price x-variables, if we add the corresponding w-variables simultaneously. More precisely, if we found a hyperarc $h \in \underline{H}$ with negative reduced cost to be removed from $\underline{H}$, we add also all w_a^l for all $a \in h$ and for all $l \in L$. This is valid because constraints (3) state that the w-variables can only be non-zero if the corresponding x-variables are non-zero. Moreover we add the coupling constraints (3) dynamically, *i.e.,* if the corresponding w-variable is not included in $(\text{MP})_{\text{LP}}(\emptyset, \underline{H})$, we do not have to consider the corresponding constraint. In this case the constraint is empty, *i.e.,* has no non-zeros, and therefore the value of the corresponding dual variable does not affect in the computation of the reduced cost.

6.2 Cut date heuristic

It turned out that it can be very challenging to construct high quality feasible solutions for the RSRPP by pure tree based search approaches like branch and bound or even rapid branching. To overcome this, we developed a problem specific local search heuristic. The general procedure of a local search algorithm is to compute an initial (possibly infeasible) solution and

to iteratively improve the solution by exploring a neighborhood structure. In the case of a infeasible solution one tries to reduce the infeasibility and in the case of a feasible solution one tries to decrease the value of the objective function. The non-maintenance relaxation of the RSRPP for our industrial problem instances is not that hard to solve. What makes the problem really hard are the maintenance constraints. The main idea of our cut date heuristic is to iteratively remove the infeasibilities of an infeasible solution by exploring the *cut date neighborhood*. After becoming feasible we try to improve the value of the objective function also by iteratively exploring the cut date neighborhood.

Let $h \in H$ be a hyperarc and $a = (u, v) \in h$ be a standard arc of h. We introduce an artificial service node s_{art} implementing all maintenance tasks that exists and extend a to an artificial service path $\{(u, s_{\text{art}}), (s_{\text{art}}, v)\}$. With this simple construction we are able to construct *artificial hyperarcs* that consist only of artificial service paths. Let H_{art} be the set of all artificial hyperarcs that result if we transform every hyperarc of H to an artificial hyperarc, *i.e.,* $|H| = |H_{\text{art}}|$. It easily can be seen that an infeasible solution $H_0 \subset H$ of a given RSRPP instance that only violates maintenance constraints, can made "feasible" if we extend the set of hyperarcs by the set of artificial hyperarcs. We can insert artificial services, *i.e.,* by replacing hyperarcs with artificial hyperarcs, just before any appearance of maintenance constraint violations along the rotations. By introducing artificial replenishment hyperarcs which have a sufficient large objective coefficient, *i.e.,* $\mathbf{c}_h := M \gg \max_{h \in H} \mathbf{c}_h$ for $h \in H_{\text{art}}$, we are able to tackle maintenance constraint violations by minimizing the objective function.

Each timetabled trip $t \in T$ has a departure and arrival date in our standard week. A node $v \in V$ represents the departure or arrival of a vehicle operating a trip. We define the symbol $\texttt{date}(v)$ stating the date of a node in our standard week. Further, we define the function $\texttt{isBetween}(u, v, w)$ for the nodes $u, v, w \in V$ that returns `true` if $\texttt{date}(v)$ *is between* $\texttt{date}(u)$ and $\texttt{date}(w)$ and `false` otherwise. More precisely, this means that $\texttt{date}(v)$ is between $\texttt{date}(u)$ and $\texttt{date}(w)$, if one starts to wait at $\texttt{date}(u)$ and $\texttt{date}(v)$ is approached before $\texttt{date}(w)$ is approached. We define $\texttt{tail}(h) := \{u \in V \mid \exists\, v \in V \cup S \,:\, a = (u, v) \in h\}$ as the *tail nodes* of h and $\texttt{head}(h) := \{u \in V \mid \exists\, v \in V \cup S \,:\, a = (u, v) \in h\}$ as the *head nodes* of h. Let $H_0 \subset H$ be a solution of a given RSRPP instance and $v \in V$ a node. We define the *cut date neighborhood* $\mathcal{N}(H_0, v)$ as:

$$\mathcal{N}(H_0, v) := H_0 \cup \{h \in H \mid \exists\, u \in \texttt{tail}(h),\, w \in \texttt{head}(h) \,:\, \texttt{isBetween}(u, v, w) = \texttt{true}\}\,.$$

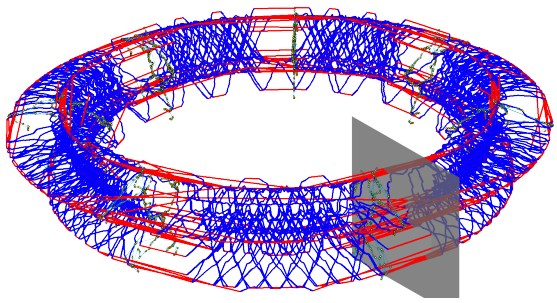

Figure 8: Idea of cut heuristic.

Figure 8 illustrates the main idea of the cut date neighborhood $\mathcal{N}(H_0, v)$. It includes all hyperarcs of a given solution as well as all hyperarcs covering a date defined by a specific node. Algorithm 2 gives an overview of our local search procedure. We start with a possibly infeasible solution, such that this solution only violates maintenance constraints. We assume that M is sufficient large such that any infeasible solution, *i.e.,* a solution that contains artificial replenishment hyperarcs, is much more expensive than any feasible solution. Therefore algorithm 2 runs in two phases: First it tries to remove all infeasibilities, *i.e.,* all artificial replenishment hyperarcs, and second it tries to improve the value of the objective function of the incumbent solution. The algorithm terminates with a local optimal solution H_0, *i.e.,* a solution which can not be improved by exploring any cut date neighborhood, if a feasible solution has been found.

6.3 Rapid Branching

The authors of [4], see also the thesis [12], proposed in the context of integrated vehicle and duty scheduling a search scheme that tries to overcome this problem by a combination of cost perturbation to "make the LP more integer", a selective branching scheme to fix large sets of variables, and an associated backtracking mechanism to correct wrong decisions. This method is called *rapid branching*. The main idea is, is that fixing a single variable to zero or one has most of the time almost no effect on the value of the LP relaxation. Rapid branching belongs to the class of objective diving heuristics guided by solutions of the LP relaxation for the construction of high-quality integer solutions for very large scale mixed integer programs.
Algorithm 3 gives a simplified overview of our adaption of the rapid branching algorithm. After applying the fixations defined by $\overline{H}$ and $\underline{H}$ we solve the

Algorithm 2: `cutDateHeuristic`$(\widetilde{H})$

Data: A (possibly infeasible) solution $\widetilde{H}$ of a RSRPP problem instance.

Result: A local optimal solution $H_0 \subseteq H \cup H_{\text{art}}$.

```
1 set H := H ∪ H_art;
2 FIND_IMPROVEMENTS:
3 foreach v ∈ V do
4     compute solution Ĥ of (MP)_MIP(∅, N(H̃, v) ∪ H_art);
5     if Σ_{h∈H̃} c_h > Σ_{h∈Ĥ} c_h then
6         set H̃ := Ĥ;
7         goto line 2;
8 set H := H \ H_art;
```

current linear relaxation (see line 1). After that we call our local search algorithm. In our implementation, we use the objective coefficients $\mathbf{c}_h := -x_h$ for the computation of the solution of the non-maintenance relaxation in line 3. This can be seen as heuristic method to round the fractional solution of the current LP relaxation to an integer feasible solution which is usually infeasible w.r.t. maintenance constraints. Nevertheless, this provides a starting solution for our local search method in line 4. If our local search method finds a feasible solution we update H_0.

A main feature of the rapid branching search scheme is the perturbation of the objective function in line 6. We do this iteratively by decreasing the current objective coefficient of each hyperarc by a linear perturbation function $\alpha : \mathbb{Q} \mapsto \mathbb{Q}$. We do not describe the detailed implementation of α since it is much technical. The most important fact is, that it is quadratically related to the value of the corresponding hyperarc variable in the current LP solution. The idea behind this quadratic perturbation is that variables with values close to 1 are driven towards 1. In each perturbation round i we determine the set $\overline{H}_i$ of hyperarc variables which are close to 1 and not fixed yet, called *candidates*, see line 8. We compute a score for each of the candidate sets. This score states how many of the variables have become integer and how much the original objective function was increased by the perturbation. If we have not found any candidate set, we solve the remaining sub-problem by traditional branch and bound with the original objective function (line 11). Those sub-problems are in most of all cases very small.

Algorithm 3: `rapidBranching`$(\overline{H}, \underline{H})$

Data: A set of hyperarcs $\overline{H}$ to fix to one and a set of hyperarcs $\underline{H}$ to fix to zero.

Result: A feasible solution H_0 of the given RSRPP instance.

1 **compute** optimal solution vector $x \in [0,1]^H$ of $(\text{MP})_{\texttt{LP}}(\overline{H}, \underline{H})$;
2 **if** $H_0 = \emptyset$ **then**
3 **compute** solution $\widetilde{H}$ of non-maintenance relaxation of $(\text{MP})_{\texttt{MIP}}(\overline{H}, \underline{H})$;
4 **compute** `cutDateHeuristic`$(\widetilde{H})$;
5 **for** $i := 1$ **to** 10 **do**
6 **set** $\mathbf{c}_h := \mathbf{c}_h - \alpha(x_h^2) \quad \forall\, h \in H \setminus (\overline{H} \cup \underline{H})$;
7 **compute** optimal solution vector $x \in [0,1]^H$ of $(\text{MP})_{\texttt{LP}}(\overline{H}, \underline{H})$;
8 **set** $\overline{H}_i := \{h \in H \setminus \overline{H} \,|\, x_h \geq 0.9\}$;
9 **compute** `score`$(\overline{H}_i)$;
10 **if** $\overline{H}_i = \emptyset \;\forall\, i = 1, \ldots, 10$ **then**
11 **compute** $(\text{MP})_{\texttt{MIP}}(\overline{H}, \underline{H})$;
12 **else**
13 reorder $\overline{H}_1, \ldots, \overline{H}_{10}$ by `score`();
14 **for** $i := 1$ **to** 10 **do**
15 **set** $\underline{H}_i := \emptyset$;
16 **if** $i > 1$ *and* $\overline{H}_{i-1} \setminus \overline{H}_i \neq \emptyset$ **then**
17 choose $\underline{h}_i \in \overline{H}_{i-1} \setminus \overline{H}_i$;
18 **set** $\overline{H} := \overline{H} \cup \overline{H}_i$; and $\underline{H} := \underline{H} \cup \{\underline{H}_i\}$;
19 `rapidBranching`$(\overline{H}, \underline{H})$;
20 **set** $\overline{H} := \overline{H} \setminus \overline{H}_i$; and $\underline{H} := \underline{H} \setminus \underline{H}_i$;

If we found candidate sets we reorder this sets (only the non-empty ones) by the computed score to explore the most promising candidate sets first in line 13. To avoid cycling of the algorithm (at least from the practial point of view) we choose for two succeeding candidate sets one variable which was fixed in the previous set $\overline{H}_{i-1}$ and should not be fixed in the current set $\overline{H}_i$ to fix to zero (see lines 15 to 17).

We induce the search tree by a recursion, *i.e.,* by calling the rapid branching function again. Since the recursion is called inside the loop over the candidate sets, we execute a depth first search over those candidate sets. One

Algorithm 4: RSRPP algorithm

Data: A RSRPP problem instance.
Result: A feasible solution $H_0 \subseteq H$ of the RSRPP instance.

1 **set** $\overline{H} := \emptyset$;
2 **set** $\underline{H} :=$ `hyperarcGeneration()`;
3 **set** $H_0 :=$ `rapidBranching`$(\overline{H}, \underline{H})$;

can see the reordered candiate sets as backtracking alternatives. In [4] this backtracking sets were constructed by another strategic, that is not based on the perturbation rounds. This is the most important difference we made in our rapid branching adaption.
Finally, algorithm 4 describes our overall algorithmic procedure. Our two step approach generates a set of columns with or hyperarc gerneration algorithm and a lower bound for optimal objective function value as well. In the second phase we try to compute an integer feasible solution for the RSRPP by using our adaption of the rapid branching search scheme. We do not generate additional hyperarcs and we frequently call our local search heuristic within the second phase.

6.4 Summary of the algorithm

We use rapid branching because the LP relaxation of our model is too hard to tackle within a standard branching search tree. Our column generation method is needed to compute the inital LP relaxation and moreover to shrink the size of the model. Note that this is a heuristic procedure, since we do not generate additional columns within the search tree. Our local search heuristic is based on cut dates. Thus, the number of vehicles is almost fixed by the intital solution during this heuristic. To diversify the number of vehicles we call the heuristic within the rapid branching search tree. Therefore all algorithmic aspects work together as a good team.
Note that we did not explain the concept of bounding and also not the relations of rapid branching to branch and bound. For further reading see [4] and [12].

7 Computational results

We implemented our model and algorithm in a computer program, called `VR-OPT`. This implementation takes use of the commercial mixed integer

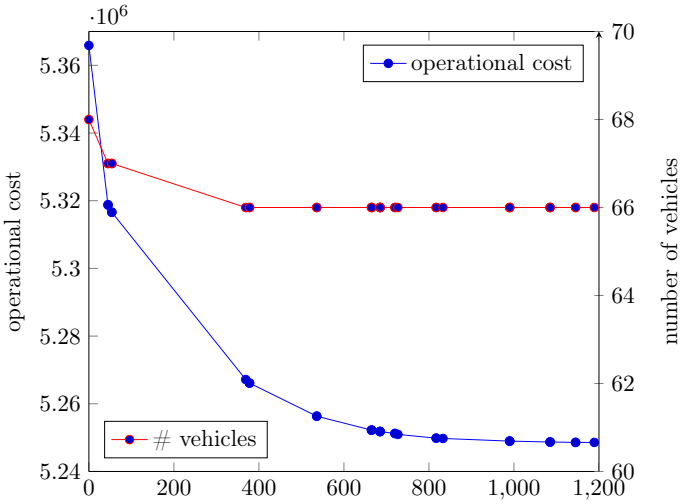

Figure 9: Cost analysis of deviations from planned durations for bottleneck connections.

programming solver `Cplex 12.2`. We use `Cplex` barrier without crossover to compute all arising linear programs and we also use the branch and cut algorithm of `Cplex` to compute all mixed integer subproblems arising during the algorithm runs. `VR-OPT` is integrated in the IT system of Deutsche Bahn and is evaluated and used by planners of Deutsche Bahn. All our computations were performed on computers with an Intel(R) Xeon(R) CPU X5672 with 3.20 GHz, 12 MB cache, and 48 GB of RAM in single thread mode. The considered instances include scenarios for the currently operated high speed intercity vehicles (ICE) as well as studies for future rolling stock fleets.
To give examples of use cases of our model and algorithm at Deutsche Bahn, we formulate a set of questions for that our model provides to ability to tackle them: Which vehicles of which fleets in which vehicle compositon are the best ones to operate a given timetable? How to operate a given timetable in detail? What are the influences of parts of the objective function? What are the cost of regularity? What could be the profit of an investment in infrastructure capacity?
Figure 9 illustrates a real-world study of the devation from planned durations of bottlenck connections as intoduced in Subsection 4.5. This was done by a multi criteria optimization with two objective functions: *(1) Minimize the total operational cost (all terms without t_h of the objective function).* and *(2) Minimize the total deviation from planned durations for connections between*

timetabled trips.

Each blue circle represents a solution for the RSRPP for a given timetable which was computed by our implementation of our approach. On the x-axis we denote the total deviation in minutes that appears in each solution, thus it states the second objective function. On the left y-axis we denote the total operational cost of the solutions, *i.e.,* the first objective function. Therefore the blue line states the pareto front of the bicriteria optimization. In addition we denoted at the right y-axis the number of vehicles (which affects the largest amount in the operational cost). A decision maker could now very detailed analyze the 17 very different computed rolling stock rotation plans and make a decision by a compromise of the two objective functions. This results prove that our approach is highly robust in computing high-quality solutions for real-world RSRPP scenarios.

Table 1 provides results for some very different instances w.r.t. the size and hardness. The first five columns of describe 14 chosen instances that a representable for different use cases arising at Deutsche Bahn. The second and third column report about the number of nodes V and the number of hyperarcs H of our hypergraph. The next two columns denote the number of maintenance constraints $|L|$ as well as the number of base constraints $|B|$. The last three columns report about the computed results, such as the number of vehicles, the worst case optimality gap, and the computation time. We observed that the running time and worst case integrality gap is mainly related to the hardness of the problem and has less relations to the size of the problem. Problems of very large scale that have no maintenance constraints can be solved to proven optimality. Problems that are much constrained by maintenance and base constraints are much harder to solve, but we still produce very high-quality solutions. This proves that our algorithm is highly efficient and much tuned in computing rolling stock rotations.

Our best result so far, is that we could compute a high-quality solution for the last RSRPP instance. This is a complete timetable of the year 2010 that is operated with all fleets of the German ICE high speed fleet. It has less degrees of freedom (in comparison to some of the other instances) but is much constrained since the result should be directly implementable. Almost 4 days of computation time for such a large and complex instance is an outstanding result.

Our computational study demonstrate that our solution approach can be used to produce high quality solutions for large-scale rolling stock rotation planning problems.

instance	$\|V\|$	$\|H\|$	$\|B\|$	L	$\mathfrak{v}$	gap [%]	dd:hh:mm:ss
RSRPP_01	310	149573	0	2	17	2.65	00:00:11:43
RSRPP_02	915	748609	0	2	56	5.28	00:05:51:57
RSRPP_03	884	1139453	0	2	55	5.62	00:09:40:18
RSRPP_04	490	68338	0	2	13	0.74	00:00:12:20
RSRPP_05	490	125647	0	2	13	0.12	00:00:06:32
RSRPP_06	490	73681	196	2	13	4.97	00:00:18:12
RSRPP_07	1879	3203619	49	2	63	5.78	01:22:33:30
RSRPP_08	1864	1782565	147	2	64	3.56	00:16:36:23
RSRPP_09	1379	1333665	0	2	65	2.53	00:19:33:14
RSRPP_10	336	198454	0	2	22	4.33	00:00:04:07
RSRPP_11	26396	34446093	0	0	116	0.00	00:01:54:50
RSRPP_12	9896	17003142	0	1	116	0.55	01:01:40:34
RSRPP_13	30446	61678335	0	0	188	0.00	00:04:00:29
RSRPP_14	6198	7008168	322	2	283	5.01	03:22:49:10

Table 1: Computational results.

References

[1] Ravindra K. Ahuja, Jian Liu, James B. Orlin, Dushyant Sharma, and Larry A. Shughart. Solving Real-Life Locomotive-Scheduling Problems. *Transportation Science*, 39:503–517, November 2005.

[2] Luzi Anderegg, Stephan Eidenbenz, Martin Gantenbein, Christoph Stamm, David Scot Taylor, Birgitta Weber, and Peter Widmayer. Train Routing Algorithms: Concepts, Design Choises, and Practical Considerations. In *ALENEX*, pages 106–118, 2003.

[3] Sebastian Behrendt. Dienstreihenfolgeplanung mit ganzzahliger Optimierung. Master's thesis, TU Berlin, July 2008.

[4] Ralf Borndörfer, Andreas Löbel, and Steffen Weider. A bundle method for integrated multi-depot vehicle and duty scheduling in public transit. In Mark Hickman, Pitu Mirchandani, and Stefan Voß, editors, *Computer-aided Systems in Public Transport*, volume 600 of *Lecture Notes in Economics and Mathematical Systems*, pages 3–24. Springer-Verlag, 2008.

[5] Ralf Borndörfer, Markus Reuther, Thomas Schlechte, and Steffen Weider. A Hypergraph Model for Railway Vehicle Rotation Planning. In Alberto Caprara and Spyros Kontogiannis, editors, *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization,*

and Systems, volume 20 of *OpenAccess Series in Informatics (OASIcs)*, pages 146–155, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[6] Jean-François Cordeau, François Soumis, and Jacques Desrosiers. Simultaneous Assignment of Locomotives and Cars to Passenger Trains. *Oper. Res.*, 49:531–548, July 2001.

[7] Benoit Crevier, Jean-Francois Cordeau, and Gilbert Laporte. The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756–773, January 2007.

[8] Giovanni Luca Giacco, Andrea D'Ariano, and Dario Pacciarelli. Rolling stock rostering optimization under maintenance constraints. In *Proceedings of the 2nd International Conference on Models and Technology for Intelligent Transportation Systems*, pages 1–5, Leuven, Belgium, 2011. MT-ITS 2011.

[9] Andreas Löbel. *Optimal Vehicle Scheduling in Public Transit*. Shaker Verlag, Aachen, 1997. Ph.D. thesis, Technische Universität Berlin.

[10] Gábor Maróti and Leo Kroon. Maintenance Routing for Train Units: The Transition Model. *Transportation Science*, 39:518–525, November 2005.

[11] Gábor Maróti and Leo G. Kroon. Maintenance routing for train units: The interchange model. *Computers & OR*, 1:1121–1140, 2007.

[12] Steffen Weider. *Integration of Vehicle and Duty Scheduling in Public Transport*. PhD thesis, TU Berlin, 2007.

[13] Koorush Ziarati, François Soumis, Jacques Desrosiers, Sylvie Gélinas, and André Saintonge. Locomotive assignment with heterogeneous consists at CN North America. *Eur. J. Oper. Res.*, 97(2):281–292, 1997.