# A Coarse-To-Fine Approach to the Railway Rolling Stock Rotation Problem*

**Ralf Borndörfer, Markus Reuther, and Thomas Schlechte**

**Zuse Institute Berlin**
**Takustrasse 7, 14195 Berlin, Germany**
reuther@zib.de

## Abstract

We propose a new coarse-to-fine approach to solve certain linear programs by column generation. The problems that we address contain layers corresponding to different levels of detail, i.e., coarse layers as well as fine layers. These layers are utilized to design efficient pricing rules. In a nutshell, the method shifts the pricing of a fine linear program to a coarse counterpart. In this way, major decisions are taken in the coarse layer, while minor details are tackled within the fine layer. We elucidate our methodology by an application to a complex railway rolling stock rotation problem. We provide comprehensive computational results that demonstrate the benefit of this new technique for the solution of large scale problems.

**1998 ACM Subject Classification** G.1.6 Optimization

**Keywords and phrases** column generation, coarse-to-fine approach, multi-layer approach, rolling stock rotation problem

**Digital Object Identifier** 10.4230/OASIcs.ATMOS.2014.79

## 1 Introduction

This paper is motivated by an application in railway optimization, namely the rolling stock rotation problem (RSRP). This problem consists of several "layers" that address different levels of detail. The major decisions of the RSRP deal with covering timetabled trips by rolling stock rotations. This is a coarse layer of the problem. At the same time minor decisions, for example, about the detailed arrival of a multi-traction vehicle composition at some station, must be considered for technical reasons. This defines a fine layer. Suppose there is a solution for the coarse layer that has been found by ignoring the details of the fine layer. Then it is often possible to extend this coarse solution to a solution for the fine layer, but not always. In this situation one can try to refine the coarse model locally at the critical parts. This leads to an iterative refinement approach with a model that mixes coarse and fine parts and is therefore difficult to handle. The idea of this paper is different. We propose to work with a version of the fine model that is restricted to a small subset of variables. This restricted model is iteratively extended using information from the coarse model. In other words, the coarse model is used to identify the relevant parts of the fine model, (hopefully) focusing the attention exactly to where it is needed.

Technically, the variable selection process is handled by column generation. Our idea is to work with two linear programs (LPs), one for the coarse and one for the fine layer. The coarse LP is constructed by aggregating suitable rows of the fine LP and sometimes turns out to be a combinatorial optimization problem of low complexity, e.g., a network flow problem.

* This work was partially supported by DB Fernverkehr AG.

14th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'14).
Editors: Stefan Funke and Matúš Mihalák; pp. 79–91

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Variables for the fine LP are generated using the coarse LP until convergence. This method aims at a rapid solution progress and at a complete elimination of stalling and tailing-off effects that are due to the fine layer.

Row aggregation techniques for column generation algorithms are a topical research area. Elhallaoui et al. [3] present a multi-phase dynamic constraint aggregation approach to solve large scale set partitioning type models. Desrosiers and Lübbecke [2] use row aggregation to utilize degeneracy in linear programming to improve the convergence characteristics of column generation algorithms. Coarse-to-fine ideas have also been studied to solve optimization problems on graphs. Raphael [6] describes an algorithm for solving a dynamic program (DP) on a large graph corresponding to a state space. A sequence of coarse DPs is solved, and the level of detail in the fine DP increases gradually. Schlechte et al. [10] used a two level micro-macro approach to solve railway track allocation models. An exact iterative graph aggregation procedure for solving network design problems is considered in Bärmann [1]. For a survey on aggregation and disaggregation techniques for optimization problems, see Rogers et al. [9].

In contrast to our approach, all these methods mix the coarse and the fine layer within one model, while our approach separates the coarse and the fine layer. This approach turns out to be easier. Of course, the layers have to be defined in a meaningful way and the success of the method depends on the quality of the layering. While we offer no general theory how to do this, in many applications the layers are evident, e.g., for the RSRP. These are the applications that we have in mind. We remark that a somehow similar idea of separated layers is used by multi-grid methods to solve linear equation systems, see [11]. Here, the preconditioner plays the role of the coarse layer which improves the tractability of the fine layer.

The paper is organized as follows. In Section 2 we describe our general coarse-to-fine column generation approach for linear programming. Section 3 introduces the RSRP application. Three layers for the RSRP that are motivated by combinatorial vehicle composition requirements for rolling stock are introduced and motivated in Section 4. We present in Section 5 our instantiation of the coarse-to-fine method for the RSRP. Finally, we provide comprehensive computational results for real-world instances of the RSRP given by our industrial partner DB Fernverkehr AG. We assume that the reader is familiar with column generation methods for linear programming, see [5] for an introduction.

## 2 A Coarse-To-Fine Approach to Column Generation

Given index sets $I = \{1, \dots, m\}$ and $J = \{1, \dots, n\}$, a matrix $A \in \mathbb{R}^{I \times J}$, and vectors $b \in \mathbb{R}^I$ and $c \in \mathbb{R}^J$, consider a linear program $\{A, b, c\}(J)$

$$\text{(MP)}(J) \quad \text{s.t.} \quad \begin{array}{l} \min\ c^T x \\ Ax = b \\ x \in \mathbb{R}^J_+, \end{array} \qquad \text{and its dual} \qquad \text{s.t.} \quad \begin{array}{l} \max\ b^T \pi \\ A^T \pi \leq c \\ \pi \in \mathbb{R}^I. \end{array}$$

We call (MP) = (MP)$(J)$ the *master LP*. If $|J|$ is very large, the *column generation algorithm* (CGA) is the method of choice to solve the master problem. By using the CGA one restricts $J$ to a sub-set $J' \subseteq J$ of columns to solve the *restricted master problem* (RMP). We assume $x_i$ to be zero for $i \in J \setminus J'$. In each iteration of the CGA we try to *price* columns (i.e., to find at least one column that is added to (RMP)) $j \in J \setminus J'$ by solving the *pricing problem*. The pricing problem is to solve $\bar{c} := \min \{c_j - \pi^T a_j \mid j \in J\}$ where $a_j \in \mathbb{R}^m$ is the column vector of $A$ for column $j \in J$ and $c_j \in \mathbb{R}$ is the objective coefficient for column $j$. If $\bar{c} \geq 0$ we have a proof that an optimal solution $x^*$ for (MP)$(J')$ is also an optimal solution

for (MP)($J$). Otherwise, we select a set of columns $J^* \subseteq J$ such that at least one $j \in J^*$ has negative *reduced cost* $d_j := c_j - \pi^T a_j$, add the columns associated with $J^*$ to (RMP), and continue with re-optimizing (RMP).

We are free in selecting columns for the set $J^*$ by a *column selection rule* as long as at least one element of $J^*$ has negative reduced cost. But, it is obvious that a better column selection rule improves the efficiency of the CGA. In particular, it can be beneficial to add also columns with positive reduced cost as we will see. We address applications where $J$ is enumerated to check every $j \in J$ whether $d_j$ is negative, e.g., the simplex method. We call this enumeration *pricing loop*. For a survey on column generation techniques see [5].

Our main idea is to introduce *layers* (precise definition follows) that are utilized to improve two aspects of the column generation method. The first one is to speed-up the pricing loop in each iteration of the CGA. The second one is to refine the column selection rule. The latter, aims at reducing the total number of iterations performed by the column generation algorithm and to reduce the total number of columns generated.

We restrict our considerations for general linear programs to two layers, namely the *coarse layer* and the *fine layer*. The *fine layer* is equal to (RMP). The coarse layer appears by the following considerations.

Let $[\cdot] : I \mapsto [I]$ be a *coarsening projection* that maps the index set $I$ of the equations of (MP) to a smaller *coarse index set* $[I]$ of size $|[I]| \leq |I|$. We use this notation because $[\cdot]$ induces an equivalence relation on the row indices $I$, namely, $i \sim j \iff [i] = [j]$. Let $v \in \mathbb{R}^I$ be a (column) vector with index set $I$, let $v_i$ be the element of $v$ with index $i \in I$, and let $\tau(v, i)$ be the cardinality of the set $\{v_k \neq 0 \,|\, [k] = [i]\}$, i.e., $\tau(v, i)$ is the number of non-zero coefficients in $v$ supported by rows equivalent to row $i$. We define $[v] \in \mathbb{R}^{[I]}$ to be the *coarse vector* or *coarsening* of $v$ using *coarse coefficients*

$$[v]_{[i]} := ([v]_{[i]1}, [v]_{[i]2}) := (\min\{v_k \,|\, k \in I \,:\, [k] = [i]\}, \max\{v_l \,|\, l \in I \,:\, [l] = [i]\}) \cdot \tau(v, i).$$

Note that $[v]_{[i]}$ is a pair of numbers, namely, the minimal and the maximal coefficient in the set of rows equivalent to row $i$, multiplied by the number of non-zeros. Let $([A_{\cdot j}])_{j=1,\ldots,|J|}$ be the bimatrix of coarse column vectors of $A$. Typically, this bimatrix contains identical columns caused by the coarsening projection, see Example 3. We chose exactly one representative for a set of identical columns and denote the resulting bimatrix by $[A]$ with columns $[J]$. Further, we define the coarse objective coefficient $[c_j] := \min_{i \in J}\{c_i \,|\, [i] = [j]\}$ for column $j \in J$.

Let $\pi \in \mathbb{R}^I$ be an optimal dual solution vector of (MP) and let $a_j$, $j \in J$, be a column vector with objective coefficient $c_j$. For ease of notation, the *coarse reduced cost* $[d]$ is defined via coefficients $[d_j] := [c_j] - [\pi]^T \cdot [a_j]$, $j \in J$, where we define the multiplication of pairs as $(a_1, b_1) \cdot (a_2, b_2) := \max\{a_1 b_1, a_1 b_2, a_2 b_1, a_2 b_2\}$ for two pairs $(a_1, a_2) \in \mathbb{R}^2$ and $(b_1, b_2) \in \mathbb{R}^2$. Note that the coarse reduced cost is *not* the coarsening of the reduced cost vector $d$. The *coarse reduction* of the master (MP) is

$$\text{(R)} \quad \min\ [d]^T x \quad \text{s.t.} [A]x[=][b], x \in \mathbb{R}_+^{[J]},$$

where we define

$$[A]x[=][b] \quad :\Leftrightarrow \quad [b]_{[i]1} \leq \sum_{j \in J} [A_{\cdot j}]_{[i]2} x_j, \quad \sum_{j \in J} [A_{\cdot j}]_{[i]1} x_j \leq [b]_{[i]2} \quad \forall [i] \in [I].$$

That is, the coarse reduction (R) approximates every equation of the master LP by two extreme case constraints arising from the minimum and maximum coefficients in equivalent rows. Note that the objective function of the coarse reduction is to minimize $[d]$ (and not $c$); the reason for this will become clear in the sequel. Let $R^\star \subseteq [J]$ be all coarse columns that

**Algorithm 1:** Coarse-To-Fine column generation iteration for linear programs.

**Data**: (RMP) given by $\{A, b, c\}$ and coarsening projection $[\cdot]$
**Result**: a set of columns $J^*$ to be added to (RMP)

1 **compute** optimal solution of (RMP) with optimal dual solution vector $\pi^* \in \mathbb{R}^m$;
2 **compute** coarse dual solution vector $[\pi^*]$ defined by $[\cdot]$;
3 **compute** $[J^*] := \{[j] \in [J] \,|\, [d_j] < 0\}$ ; `/* pricing loop in coarse layer */`
4 **compute** $J^\star \subseteq \{j \in J \,|\, [j] \in [J^*], d_j < 0\}$;
5 **compute** optimal solution of (R) and $R^\star$;
6 **compute** $J^\star := J^\star \cup \{j \in J \,|\, [j] \in R^\star\}$ ; `/* column selection rule */`

have a non-zero primal solution value in the optimal solution of the coarse reduction (R). We also address the coarse reduction as *coarse LP* and the master LP as *fine LP*.

The polytope associated with (MP)($J$) is denoted by $P_{(\mathrm{MP})(J)}$. Coarsening has the following simple but important properties.

▶ **Lemma 1.** *The coarse polytope associated with* ($R$) *includes the fine polytope associated with* ($MP$)*, i.e.,* $P_{(R)} \supseteq P_{(MP)}$.

**Proof.** Every row in (R) is a relaxation of an original row of (MP). ◀

▶ **Lemma 2.** *The coarse reduced cost can be used to underestimates the reduced cost, i.e.,*

$$[d_j] = [c_j] - [\pi]^T \cdot [a_j] \leq c_j - \pi^T \cdot a_j = d_j.$$

**Proof.** By definition we have $[c_j] \leq c_j$ and each summand in $\pi^T \cdot a_j$ is overestimated by a summand of $[\pi]^T \cdot [a_j]$. ◀

Lemma 1 shows that the coarse reduction (R) provides an approximation of the fine master LP which has fewer rows and thus is probably easier to solve. We want to take advantage of this approximation in a column generation algorithm (CGA) for the fine master LP by shifting the pricing loop to the coarse reduction. A naive way to do this is to solve the coarse reduction by a CGA in a first step, producing a set of columns $J^\star \subseteq J$, and then to solve the fine master LP in a second step, starting from the restriction (MP)($J^\star$) to the set of columns $J^\star$. However, this simplistic procedure is unlikely to work well because of a lack of information exchange between the coarse and the fine linear programs. Also the quality of the polyhedral approximation of the coarse reduction is unclear.

Lemma 2 proposes an alternative to simply price in the coarse reduction using the coarsened reduced cost from the fine master LP. This generic idea is formalized in Algorithm 1 that illustrates one iteration within a CGA.

The *coarse-to-fine column generation algorithm* solves the fine master LP by a CGA that iterates though a coarse-to-fine pricing loop. In this loop an optimal dual solution (step 2) of the restricted fine master LP is computed and coarsened. Afterwards, we compute the coarse reduced cost in the coarse layer that defines the set $J^*$ of coarse columns with negative coarse reduced cost and select some of them in step 4. By Lemma 2 we can not miss any columns in the fine layer with negative reduced cost. That shows that the preselection by $J^*$ is exact. There is one more ingredient that is crucial for the performance of our coarse-to-fine approach, namely, a column selection rule to restrict the set of coarsely priced columns. We propose to compute a reasonable combination of (hopefully) improving columns by solving the coarse reduction in step 5 and 6. Using the coarse reduced cost as an objective aims at a

"good combination" of improving columns of negative reduced cost and further columns of positive reduced cost that are "necessary" to complete the construction of the solution. This iteration is performed until convergence. This is the general method that we propose. It works particularly well when the coarse reduction turns out to be a simple combinatorial optimization problem such as a network flow problem. We will discuss an example of this type in the context of our RSRP application in Section 4.

▶ **Example 3.** Consider the following matrix and coarsening projection:

$$A = \begin{pmatrix} 1 & 0 & 0 & -4 \\ 0 & 1 & 2 & 0 \end{pmatrix} \text{ and } [i] := \left\lfloor \frac{i}{2} \right\rfloor .$$

Then we have $[A] = ((0,1)\,(0,2)\,(-4,0))$.

Example 3 shows that coarsening typically produces many identical columns, in particular, for matrices arising from combinatorial optimization problems. As defined, identical columns are reduced, keeping only the copy with the smallest objective coefficient. This a desirable effect that can produce a substantial speed-up of the coarse-to-fine pricing loop.

## 3 The Rolling Stock Rotation Problem

In this section we consider the *Rolling Stock Rotation Problem* (RSRP) and state a hypergraph based integer programming formulation, see [7]. We apply the ideas of Section 2 to the LP-relaxation of this formulation. We focus here on the main modeling ideas and refer the reader to our paper [7] for technical details including the treatment of maintenance and capacity constraints. The extension of the following problem description and model to include maintenance constraints is straight forward and does not affect the content nor the contribution of the paper.

We consider a cyclic planning horizon of one *standard week*. The set of timetabled passenger trips is denoted by $T$. Let $V$ be a set of *nodes* representing timetabled departures and arrivals of vehicles operating passenger trips of $T$, let $A \subseteq V \times V$ 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. The RSRP *hypergraph* is denoted by $G = (V, A, H)$. The hyperarc $h \in H$ *covers* $t \in T$ if each standard arc $a \in h$ represents an arc between the departure and arrival of $t$. 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 our model. We define sets of hyperarcs coming into and going out of $v \in V$ in the RSRP hypergraph $G$ as $H(v)^{\text{in}} := \{h \in H \mid \exists\, a \in h \,:\, a = (u, v)\}$ and $H(v)^{\text{out}} := \{h \in H \mid \exists\, a \in h \,:\, a = (v, w)\}$, respectively.

The RSRP 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 $h \in H_0$ and $\bigcup_{h \in H_0} h \subseteq A$ is a set of *rotations*, *i.e.,* a packing of cycles (each node is covered at most once).

Using a binary decision variable for each hyperarc, the RSRP can be stated as an integer program as follows:

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

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

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

$$x_h \in \{0,1\} \qquad \forall h \in H. \tag{3}$$

The objective function of model (MP) minimizes the total cost of the chosen hyperarcs. For each trip $t \in T$ the covering constraints (1) assign one hyperarc of $H(t)$ to $t$. The equations (2) are flow conservation constraints for each node $v \in V$ that define a set of cycles of arcs of $A$. Finally, (3) states the integrality constraints for our decision variables.

The RSRP is $\mathcal{NP}$-hard, even without maintenance and base constraints and if constraints (1) are trivially fulfilled, i.e., $|H(t)| = 1$ for all trips $t \in T$, see [4].

## 4 Three Layers for the RSRP

The mixed integer programming formulation for the RSRP defined in Section 3 only depends on a hypergraph and a cost function. It is therefore natural to define the layers to be used in our coarse-to-fine approach as projections of node sets. Such projections induce hypergraphs themselves. The layers, namely, a *composition layer* $G = (V, A, H)$, a *configuration layer* $[G] = ([V], [A], [H])$, and a *vehicle layer* $[[G]] = ([[V]], [[A]])$, are motivated by our application at Deutsche Bahn Fernverkehr AG. In this application the RSRP must be solved for the composition layer, but many technical rules only apply to the configuration layer, which is much smaller w.r.t. the size of the set of hyperarcs. In addition, we define a vehicle layer to set up a super-coarse RSRP that provides a reasonable description of the major problem characteristics (i.e., the total number of rolling stock vehicles used in a solution) and that is solvable in polynomial time. We discuss in the following the detailed combinatorial aspects of vehicle composition that motivate our layers.

A *fleet* is a basic type of rail vehicles. For example, the slightly more than 220 Intercity-Express rail vehicles of Deutsche Bahn Fernverkehr AG are partitioned into several structurally identical sets of vehicles named fleets. Let $F$ be the set of fleets.

An *orientation* is an element of the set $O = \{Tick, Tack\}$. Orientation describes the two options of how vehicles can be placed on a railway track. At Deutsche Bahn Fernverkehr AG this is distinguished by the position of the first class carriage of the vehicle w.r.t. the driving direction. Tick (Tack) means that the first class carriage is located at the head (tail) of the vehicle w.r.t. the driving direction.

A *(vehicle) composition* $c$ of size $n \in \mathbb{N}_+$ is an $n$-tuple of the form

$$c = ((f_1, o_1), (f_2, o_2), ..., (f_n, o_n)) \in (F \times O)^n.$$

A sub-index $p \in \{1, \ldots, n\}$ of $c$ is called a *position* of an individual vehicle in a vehicle composition. In rolling stock rotation planning, a vehicle composition has to be chosen for each departure of a timetabled trip.

For example, if we consider the set of fleets $F = \{Red, Blue\}$ we get the following vehicle compositions of size one: $(Red, Tick)$, $(Red, Tack)$, $(Blue, Tick)$, $(Blue, Tack)$. Figure 1 illustrates the 16 possibilities for such vehicle compositions of size two. The fleet Red is

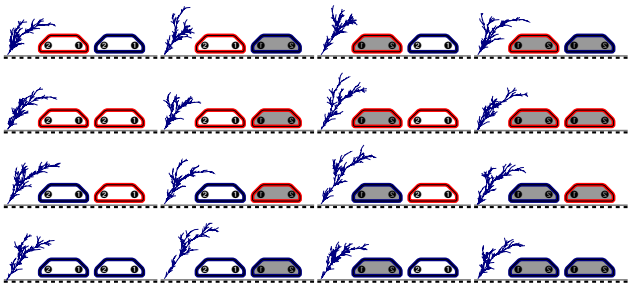

**Figure 1** Vehicle compositions of size two for two fleets. The trees indicate the driving directions.

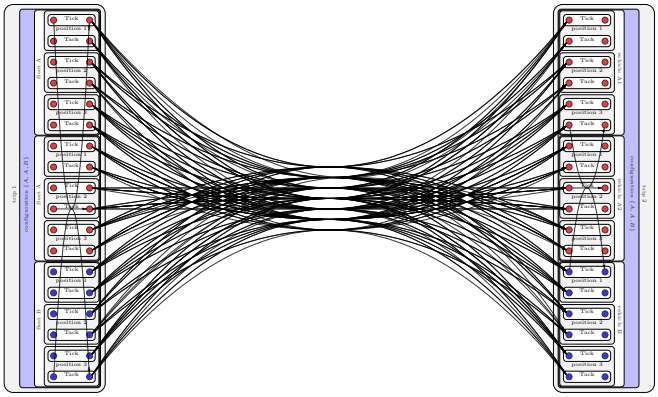

**Figure 2** Possible hyperarcs for vehicle compositions of two trips operated with vehicle configuration $\{A, A, B\}$.

represented by the red vehicle, while the blue vehicles represent the fleet Blue. Each gray vehicle has orientation Tack and each white vehicle has orientation Tick w.r.t. the driving direction indicated by the blue tree.

A *(vehicle) configuration* is a multiset of fleets. We say that the configuration $k$ is *realized* by the vehicle composition $c = ((f_1, o_1), (f_2, o_2), ..., (f_n, o_n))$ if $k = \{f_1, ..., f_n\}$, i.e., if the multi-set of fleets used in the composition $c$ is equal to the configuration $k$. In the above example the configurations $\{Red\}$, $\{Blue\}$, $\{Red, Red\}$, $\{Red, Blue\}$, and $\{Blue, Blue\}$ are realized by the 20 compositions.

We define an *event* as a triple $e = (\{d, a\}, t, p)$ defining the departure $(d)$ or the arrival $(a)$ of an individual vehicle at position $p \in \mathbb{N}_+$ in a vehicle composition operating trip $t \in T$.

We define the *composition layer* as the hypergraph $G = (V, A, H)$; here, each hyperarc $h \in H$ identifies a vehicle composition, as shown in Figure 2. A node $v \in V$ is a four-tuple $v = (e, k, f, o)$ defining an event $e$, the vehicle configuration $k$, the fleet $f$, and an orientation $o \in O$.

A discussion of the detailed reasons for defining the composition layer on the proposed form is out of the scope of this paper. It relies on experience of how the arising requirements in rotation planning for rolling stock can be handled.

Consider the following projections:

$$\begin{aligned} [v] &:= (e,k) \text{ for } v=(e,k,f,o)\in V,\\ [a] &:= ([v],[w]) \text{ for } a=(v,w)\in A,\\ [h] &:= \{[a]\,|\,a\in h\} \text{ for } h\in H,\\ [V]:=\{[v]\,|\,v\in V\},\quad [A]&:=\{[a]\,|\,a\in A\},\text{ and }\quad [H]:=\{[h]\,|\,h\in H\}. \end{aligned}$$

Given a composition layer with $G=(V,A,H)$, we define the *configuration layer* as the hypergraph $[G]:=([V],[A],[H])$. The projection omits the orientation and the fleet and therefore the hyperarcs of $[H]$ can be interpreted as connections of timetabled trips with vehicle configurations.

Consider the following further projections:

$$\begin{aligned} [[v]] &:= e \text{ for } [v]=(e,k)\in[V],\\ [[a]] &:= ([v],[w]) \text{ for } a=(v,w)\in A,\\ [[V]]:=\{[v]\,|\,v\in V\},&\text{ and }\ [[A]]:=\{[a]\,|\,a\in A\}. \end{aligned}$$

For the sake of a uniform notation, we also define a set of hyperarcs $[[H]]$ as follows. Let $t\in T$ and let $[H](t)$ be the set of hyperarcs that cover $t$ in $[G]$. We define $h(t):=\{[[a]]\in[[A]]\,|\,\exists[h]\in[H](t)\,:\,[a]\in[h]\}$ as the *unique* hyperarc that covers $t$ in the vehicle layer. Finally, we denote by $[[H]]:=\bigcup_{t\in T}h(t)\cup\{\{[[a]]\}\,|\,[[a]]\in[[A]]\})$ the set of unique hyperarcs that cover the trips combined with all standard directed arcs of $[[A]]$ denoted as hyperarcs.

Given a configuration layer with $[G]=([V],[A],[H])$, we define the *vehicle layer* as $[[G]]:=([[V]],[[A]])$; note that $[[G]]$ is a standard directed graph. Moreover, the coarse reduction w.r.t. the vehicle layer $[[G]]$ is solvable in polynomial time. In fact, each timetabled trip is uniquely covered by the hyperarcs of $[[H]]$. Therefore, constraints (1) are trivially fulfilled. The remaining problem is defined by the flow conservation constraints (2) and the integrality constraints (3) for a standard directed graph, i.e., this problem is a standard network flow problem. In our application the total number of rail vehicles used is of major importance. In our computations, we observed that it can be approximated reasonably well by considering only the RSRP on the vehicle layer.

With respect to the applicatiom, our layers are motivated as follows. Rail vehicles are not very flexible w.r.t. shunting operations, e.g., it is difficult to change the orientation. In addition, there are technical constraints stipulating dedicated orientations at locations. One reason for these constraints are the indicator tables that are used in Germany at passenger platforms; they show the position and orientation of individual carriages to provide informations w.r.t. seat reservations to the passengers. Those tables can not be changed easily in operation. Hence, these tables imply a lot of constraints w.r.t. position and orientation of individual vehicles within vehicle compositions. Moreover, some vehicle compositions are forbidden. For a dedicated fleet $f$ the single vehicle composition $((f,Tack),(f,Tick))$ results in a reduction of the maximal speed to 80 km/h. Because of these (and many other) detailed technical requirements we need to consider the composition layer in our application.

Nevertheless, the concept of vehicle configurations plays an essential role in our application. Most of the time-dependent constraints, e.g., the minimal time needed for cleaning or refueling, refer "only" to the configuration layer, i.e., they are independent of the concrete vehicle composition that is realized.

**Algorithm 2:** Coarse-To-Fine column generation iteration for the RSRP.

**Data**: (RMP) given by (MP) from Section 4 for $G = (V, A, \overline{H})$,

$$\begin{aligned} G &= (V, A, H) && \text{as composition layer,} \\ [G] &= ([V], [A], [H]) && \text{as configuration layer, and} \\ [[G]] &= ([[V]], [[A]], [[H]]) && \text{as vehicle layer} \end{aligned}$$

**Result**: a set of hyperarcs $H^* \subseteq H \backslash \overline{H}$ to be added to (RMP)

**set** $H^* := \emptyset$;
**compute** optimal solution of (RMP) with optimal dual solution vector $\pi^* \in \mathbb{R}^m$;
**compute** $[\pi^*]$ defined by model (MP) for $[G]$;
**compute** $[d]$ as reduced cost defined by model (MP) for $[G]$ and $[\pi^*]$;

`/* PRICE by enumeration in COMPOSITION LAYER and */`
`/* PRUNE enumeration by` $[d]$ `of CONFIGURATION LAYER */`

**foreach** $v \in V$ **do**
  **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 $i < j < n$;
  **set** $H^* := H^* \cup \{h_1, \ldots, h_{\lceil \sqrt[3]{n} \rceil}\}$;

`/* PRICE by solving the flow problem in VEHICLE LAYER */`

**set** (FP) as flow problem defined by model (MP) for $[[G]] = ([[V]], [[A]], [[H]])$ with objective function

$$[[c]] : [[A]] \mapsto \mathbb{R} \; : \; [[c]]([[a]]) := \min \left\{ \frac{[d_h]}{|h|} \;\middle|\; [a] \in [h] \in [H] \right\}$$

**compute** optimal solution $[[A]]^* \subseteq [[A]]$ of (FP);
**set** $H^* := H^* \cup \{h \in H \mid \exists a \in h \; : \; [[a]] \in [[A]]^*\}$;

To compare the size of the composition and configuration layer we consider a vehicle configuration $k$ that consists of the fleets $\{f_1, ..., f_l\} \subseteq F$ such that fleet $f_i$ appears $m_i \in \mathbb{N}_+$ times in $k$. Let $C$ be the set of all possible vehicle compositions that realize $k$. Each composition of $c \in C$ must be of size $n := \sum_{i=1}^{l} m_i$. We have $2^n$ possibilities of different combinations of orientations in $C$. Furthermore, we have $n!$ possible permutations of fleets. A fleet that appears $m$ times reduces this number by $m!$ equal permutations. In summary we have $|C| = 2^n \cdot n! / (\prod_{i=1}^{n} m_i!)$. For one fleet we have $|C| = 4$, for two different fleets we get $|C| = 8$, for three different fleets we get $|C| = 48$. Hence, the cardinality of the set of hyperarcs in the composition layer $G$ is exponential in the size of the set of hyperarcs in the configuration layer.

## 5 Application and Computational Study

We study the integer programming formulation for the RSRP of Section 3 as a prototype application for our coarse-to-fine approach proposed in Section 2 using the three layers introduced in Section 4.

Algorithm 2 summarizes our specialization of the general coarse-to-fine method for the RSRP. We are given a restricted master problem (RMP) that only includes columns for a sub-set $\overline{H}$ of hyperarcs that are already priced. The set $H^*$ of new hyperarc variables

is found by two strategies. First, we enumerate hyperarcs of the composition layer with negative reduced cost. If a node has $n$ outgoing hyperarcs with negative reduced cost we add the $\lceil \sqrt[3]{n} \rceil$ "best" ones to $H^*$, see line 7. This enumeration, i.e., the pricing loop, is performed by using a *pruning strategy*, i.e., we only have to consider hyperarcs $h \in H$ of the composition layer that have negative reduced cost $[d_h]$ (denoting the reduced cost of the column that corresponds to $h$ in model (MP)) in the configuration layer, see Lemma 2. The second strategy is to solve the flow problem (see Section 4) defined by the vehicle layer and the objective function $[[c]]$ (line 8 of Algorithm 2). This is a canonical way to approximate the reduced cost of the configuration layer to be used in the vehicle layer. We add all hyperarcs to $H^*$ that correspond to an arc of the optimal solution of the flow problem, see line 10 of Algorithm 2. This strategy is our interpretation of the coarse reduction (R) introduced in Section 2 for the RSRP and acts as an efficient column selection strategy.

In our computational study we "only" focus on the linear relaxation of model (MP) to highlight the impact of the coarse-to-fine feature. The interior point solver (without crossover) of the commercial software `Cplex 12.1` is used to solve the linear programs arising during our CGA. 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. We remark that we could have reported results for the algorithm proposed in [7] to generate integer feasible solutions for the RSRP as well, because our method clearly also applies to integer programming. This algorithm, however, is not completely exact. Therefore, the effect of our approach can become blurred.

A notable implementation detail is how we handle the hypergraphs. We only store the hypergraph associated with the configuration layer in memory. Given a hyperarc $[h] \in [H]$ we can enumerate all fine hyperarcs that map to $[h]$ by an iterator routine for the composition layer on the stack of the computer program. This can be seen as a dynamic graph generation approach, since by using our pruning strategy we do not have to handle or enumerate the whole fine hypergraph at any time (but we do this once to count the total number of hyperarcs).

We run four different algorithmic variants for each instance of our test set to show the relevance of all algorithmic ingredients we introduced:

**Variant 1**: The first variant is exactly as described in Algorithm 2.

**Variant 2**: This variant is defined by Algorithm 2 excluding lines 8 to 10, i.e., we omit our column selection strategy.

**Variant 3**: This variant is defined by lines 5 to 7 Algorithm 2 without our column selection strategy and without our pruning strategy by $[d]$.

**Variant 4**: We solve the RSRP for the composition layer from scratch, i.e., without any column generation.

Table 1 reports major characteristics of the considered instances of the RSRP, namely the number $|T|$ of trips to cover, the number $|V|$ of nodes, and the number $|H|$ of hyperarcs for 14 of our 147 test instances for the RSRP. These instances were chosen to form a representative test set; the remaining results can be found in the Appendix of the corresponding technical report [8]. The columns of Table 2 in the appendix denote the number of columns, rows, and non-zeros that were generated as well as the maximal memory usage in Megabytes, that was allocated by the executing process of the algorithm. The last two columns report the running time of the algorithm and the time to resolve the generated model from scratch (which is essential when the algorithm is used within an integer programming method). The rows of Table 2 in the appendix correspond to each run of the four variants for a single RSRP instance in the canonical order (the first row corresponds to variant 1 for `RSRP_010`, the last one to variant 4 for `RSRP_140`). A row showing no results indicates an "out of memory"-run.

**Table 1** Characteristics of instances.

| instance | $\|T\|$ | $\|V\|$ | $\|H\|$ |
|---|---|---|---|
| RSRP_010 | 884 | 1768 | 6508938 |
| RSRP_020 | 277 | 1464 | 390110 |
| RSRP_030 | 310 | 620 | 805482 |
| RSRP_040 | 2030 | 4910 | 8464864 |
| RSRP_050 | 1126 | 4696 | 20963280 |
| RSRP_060 | 174 | 898 | 271794 |
| RSRP_070 | 277 | 1443 | 377056 |
| RSRP_080 | 4216 | 13354 | 25521577 |
| RSRP_090 | 277 | 1464 | 1347270 |
| RSRP_100 | 1126 | 4696 | 19234364 |
| RSRP_110 | 73 | 146 | 21796 |
| RSRP_120 | 1033 | 3106 | 8407556 |
| RSRP_130 | 1488 | 2976 | 11670716 |
| RSRP_140 | 987 | 16790 | 75274348 |

Our results show that the running time of our algorithm, namely variant 1, is competitive with the running time of variant 4. Moreover, the size of the generated model, i.e., the set of generated columns indicated by column 2 to 5 is dramatically reduced by variant 1 in comparison to variant 4. The most drastic improvement was achieved for the resolving time (that is equal to the solving time for variant 4), since an integer programming algorithm often resolves the linear program that is slightly changed by perturbation (for heuristics) and branching.

The results for variant 2 and variant 3 demonstrate that each of our two additional layers for the RSRP is needed to be competitive to a "from scratch" approach if we only restrict to the linear programming relaxation. Nevertheless, some of the instances, e.g., RSRP_140 with more than $7 \cdot 10^7$ hyperarcs could only be solved using the new technique.

**Acknowledgments.** We want to thank three anonymous referees for improving this paper by their valuable comments.

## References

1 Andreas Bärmann, Frauke Liers, Alexander Martin, Maximilian Merkert, Christoph Thurner, and Dieter Weninger. Solving network design problems via iterative aggregation. Technical report, Department Mathematik, 2013.

2 Jacques Desrosiers, Jean Bertrand Gauthier, and Marco E. Lübbecke. Row-reduced column generation for degenerate master problems. *European Journal of Operational Research*, 236(2):453 – 460, 2014.

3 Issmail Elhallaoui, Abdelmoutalib Metrane, François Soumis, and Guy Desaulniers. Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming*, 123(2):345–370, 2010.

4 Olga Heismann. *The Hypergraph Assignment Problem*. PhD thesis, Technische Universität Berlin, 2014.

5 M.E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Oper. Res.*, 53(6):1007–1023, 2005.

6 C. Raphael. Coarse-to-fine dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1379–1390, 2001.

7 Markus Reuther, Ralf Borndoerfer, Thomas Schlechte, and Steffen Weider. Integrated optimization of rolling stock rotations for intercity railways. In *Proceedings of the 5th International Seminar on Railway Operations Modelling and Analysis (RailCopenhagen)*, Copenhagen, Denmark, May 2013.

8 Markus Reuther, Ralf Borndörfer, and Thomas Schlechte. A coarse-to-fine approach to the railway rolling stock rotation problem. Technical Report 14-26, ZIB, Takustr.7, 14195 Berlin, 2014.

9 David F. Rogers, Robert D. Plante, Richard T. Wong, and James R. Evans. Aggregation and disaggregation techniques and methodology in optimization. *Operations Research*, 39(4):553–582, 1991.

10 Thomas Schlechte, Ralf Borndörfer, Berkan Erol, Thomas Graffagnino, and Elmar Swarat. Micro-Macro Transformation of Railway Networks. *Journal of Rail Transport Planning & Management*, 1(1):38–48, 2011.

11 J. Tang, S. MacLachlan, R. Nabben, and C. Vuik. A comparison of two-level preconditioners based on multigrid and deflation. *SIAM Journal on Matrix Analysis and Applications*, 31(4):1715–1739, 2010.


## A Computational results

**Table 2** Computational results (time format is `dd:hh:mm:ss`).

| instance | columns | rows | non-zeros | memory | time | resolving time |
|---|---|---|---|---|---|---|
| RSRP_010 | 309538 | 42208 | 1691612 | 1032 | 00:00:23:56 | 00:00:01:06 |
| RSRP_010 | 1451027 | 151613 | 7917223 | 2869 | 00:04:51:01 | 00:00:04:02 |
| RSRP_010 | 1451027 | 151613 | 7917223 | 2849 | 00:05:29:36 | 00:00:03:30 |
| RSRP_010 | 7272961 | 767255 | 41617693 | 9354 | 00:00:16:51 | |
| RSRP_020 | 49359 | 3245 | 170109 | 96 | 00:00:00:56 | 00:00:00:01 |
| RSRP_020 | 129539 | 3245 | 394285 | 185 | 00:00:02:25 | 00:00:00:03 |
| RSRP_020 | 129539 | 3245 | 394285 | 188 | 00:00:03:19 | 00:00:00:02 |
| RSRP_020 | 391991 | 3245 | 1170461 | 380 | 00:00:00:38 | |
| RSRP_030 | 86385 | 12523 | 485291 | 188 | 00:00:02:57 | 00:00:00:04 |
| RSRP_030 | 266931 | 29065 | 1493851 | 542 | 00:00:21:05 | 00:00:00:17 |
| RSRP_030 | 266931 | 29065 | 1493851 | 515 | 00:00:24:13 | 00:00:00:16 |
| RSRP_030 | 901805 | 97443 | 5265727 | 1223 | 00:00:01:34 | |
| RSRP_040 | 412359 | 13080 | 1433948 | 995 | 00:00:08:21 | 00:00:00:32 |
| RSRP_040 | 1648117 | 13080 | 5449094 | 2212 | 00:01:05:24 | 00:00:01:08 |
| RSRP_040 | 1648117 | 13080 | 5449094 | 2258 | 00:01:35:45 | 00:00:01:11 |
| RSRP_040 | 8473291 | 13080 | 26999090 | 7321 | 00:00:07:07 | |
| RSRP_050 | 1002933 | 180804 | 7369383 | 2605 | 00:05:25:32 | 00:00:13:56 |
| RSRP_050 | 2981197 | 440050 | 21510641 | 6561 | 01:19:15:26 | 00:00:43:16 |
| RSRP_050 | 3232294 | 463437 | 23487090 | 7263 | 02:00:47:44 | 00:00:42:45 |
| RSRP_050 | - | - | - | - | - | - |
| RSRP_060 | 46462 | 9000 | 239702 | 110 | 00:00:01:40 | 00:00:00:03 |
| RSRP_060 | 116000 | 15810 | 579422 | 208 | 00:00:04:08 | 00:00:00:06 |
| RSRP_060 | 116000 | 15810 | 579422 | 213 | 00:00:04:43 | 00:00:00:06 |

Continued on next page

**Table 2 – continued from previous page**

| instance | columns | rows | non-zeros | memory | time | resolving time |
|---|---|---|---|---|---|---|
| RSRP_060 | 305740 | 35630 | 1521254 | 421 | 00:00:00:48 | |
| RSRP_070 | 48698 | 3364 | 168309 | 98 | 00:00:01:01 | 00:00:00:01 |
| RSRP_070 | 119940 | 3364 | 364469 | 183 | 00:00:02:23 | 00:00:00:03 |
| RSRP_070 | 121100 | 3364 | 370291 | 185 | 00:00:03:23 | 00:00:00:02 |
| RSRP_070 | 379026 | 3364 | 1133345 | 386 | 00:00:00:40 | |
| RSRP_080 | 1303150 | 34288 | 4305082 | 2838 | 00:01:01:08 | 00:00:03:19 |
| RSRP_080 | 3910682 | 34288 | 12948542 | 5716 | 00:05:27:09 | 00:00:06:13 |
| RSRP_080 | 3910682 | 34288 | 12948542 | 5687 | 00:07:48:10 | 00:00:05:54 |
| RSRP_080 | - | - | - | - | - | - |
| RSRP_090 | 121974 | 26232 | 943030 | 302 | 00:00:09:43 | 00:00:00:25 |
| RSRP_090 | 380082 | 55912 | 2951626 | 775 | 00:00:46:06 | 00:00:01:11 |
| RSRP_090 | 380082 | 55912 | 2951626 | 754 | 00:00:52:15 | 00:00:01:02 |
| RSRP_090 | 1525138 | 181872 | 12410680 | 2307 | 00:00:05:08 | |
| RSRP_100 | 749426 | 91325 | 4988934 | 1876 | 00:03:03:44 | 00:00:07:01 |
| RSRP_100 | 2819827 | 255646 | 18306921 | 5801 | 01:04:35:17 | 00:00:28:35 |
| RSRP_100 | 2717973 | 248122 | 17483355 | 5735 | 01:05:01:23 | 00:00:22:33 |
| RSRP_100 | 20680283 | 1454616 | 140691067 | 26817 | 00:03:15:58 | |
| RSRP_110 | 7284 | 366 | 25694 | 67 | 00:00:00:39 | 00:00:00:00 |
| RSRP_110 | 13046 | 366 | 43554 | 67 | 00:00:00:42 | 00:00:00:00 |
| RSRP_110 | 13046 | 366 | 43554 | 67 | 00:00:00:34 | 00:00:00:00 |
| RSRP_110 | 22050 | 366 | 81276 | 67 | 00:00:00:38 | |
| RSRP_120 | 491219 | 76767 | 2754706 | 1058 | 00:00:40:04 | 00:00:02:23 |
| RSRP_120 | 1079795 | 144509 | 5931178 | 2251 | 00:03:01:27 | 00:00:04:44 |
| RSRP_120 | 1079795 | 144509 | 5931178 | 2208 | 00:03:21:02 | 00:00:04:25 |
| RSRP_120 | 9414973 | 1012971 | 46583640 | 11235 | 00:00:36:03 | |
| RSRP_130 | 753109 | 131693 | 5721831 | 1921 | 00:01:08:04 | 00:00:03:27 |
| RSRP_130 | 2348603 | 329385 | 17919703 | 5024 | 00:11:58:58 | 00:00:13:14 |
| RSRP_130 | 2348603 | 329385 | 17919703 | 5032 | 00:12:41:50 | 00:00:13:00 |
| RSRP_130 | 13556953 | 1894535 | 104249717 | 19983 | 00:01:02:15 | |
| RSRP_140 | 3639659 | 103609 | 31758855 | 8721 | 01:17:04:26 | 00:02:50:51 |
| RSRP_140 | - | - | - | - | - | - |
| RSRP_140 | - | - | - | - | - | - |
| RSRP_140 | - | - | - | - | - | - |