Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

RALF BORNDÖRFER
MARKUS REUTHER
THOMAS SCHLECHTE

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

Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany {borndoerfer, reuther, schlechte}@zib.de
This work was partially supported by DB Fernverkehr AG.

ZIB-Report 14-26 (July 2014)

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

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

Ralf Borndörfer[1], Markus Reuther[1], and Thomas Schlechte[1]

1 **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.xxx.yyy.p

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.

* This work was partially supported by DB Fernverkehr AG.

Conference/workshop/symposium title on which this volume is based on.
Editors: Billy Editor and Bill Editors; pp. 1–30

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

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. 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 $\{A, b, c\}(J)$

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

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 $\bar{c} := \min \{c_j - \pi^T a_j \,|\, j \in J\}$ 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

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 */`

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 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 $P_{(\text{MP})(J)}$. 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

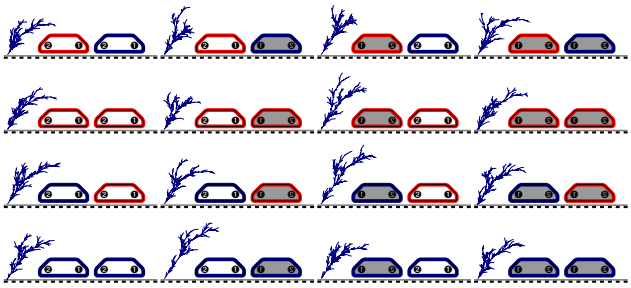

Figure 1 Vehicle compositions of size two for two fleets. The trees indicate the driving directions.

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 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:

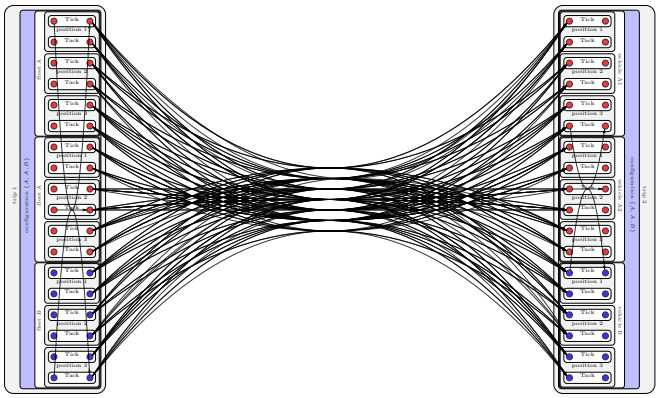

Figure 2 Possible hyperarcs for vehicle compositions of two trips operated with vehicle configuration $\{A, A, B\}$.

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

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:

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

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 $h(t) := \{[[a]] \in [[A]] \mid \exists [h] \in [H](t) : [a] \in [h]\}$ as the *unique* hyperarc that covers t in the vehicle layer. Finally, we denote by $[[H]] := \bigcup_{t \in T} h(t) \cup \{\{[[a]]\} \mid [[a]] \in [[A]]\})$ 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.

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

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)

1 **set** $H^* := \emptyset$;
2 **compute** optimal solution of (RMP) with optimal dual solution vector $\pi^* \in \mathbb{R}^m$;
3 **compute** $[\pi^*]$ defined by model (MP) for $[G]$;
4 **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 */`

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

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

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

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

9 **compute** optimal solution $[[A]]^* \subseteq [[A]]$ of (FP);
10 **set** $H^* := H^* \cup \left\{h \in H \,|\, \exists a \in h \,:\, [[a]] \in [[A]]^*\right\}$;

$[[c]]$ (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 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 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_001	310	620	805482
RSRP_002	310	620	1163370
RSRP_003	310	620	436534
RSRP_004	884	1830	4085542
RSRP_005	884	1830	5868584
RSRP_006	884	1830	2212292
RSRP_007	884	1830	4085386
RSRP_008	884	1830	5868428
RSRP_009	884	1830	2212136
RSRP_010	884	1768	6508938
RSRP_011	884	1768	9385210
RSRP_012	884	1768	3521040
RSRP_013	884	1830	4100410
RSRP_014	884	1830	2212292
RSRP_015	277	1464	757340
RSRP_016	277	1464	1110078
RSRP_017	277	1464	389980
RSRP_018	277	1464	757514

Continued on next page

Table 1 – continued from previous page

instance	$\|T\|$	$\|V\|$	$\|H\|$
RSRP_019	277	1464	1110208
RSRP_020	277	1464	390110
RSRP_021	277	980	549264
RSRP_022	277	980	1079526
RSRP_023	277	980	1589436
RSRP_024	277	980	549442
RSRP_025	174	898	139442
RSRP_026	146	828	114058
RSRP_027	277	980	1079526
RSRP_028	277	980	1589436
RSRP_029	277	980	549442
RSRP_030	310	620	805482
RSRP_031	310	620	1163370
RSRP_032	310	620	436534
RSRP_033	310	620	805482
RSRP_034	310	620	436534
RSRP_035	2030	4910	16101438
RSRP_036	2030	4910	23413966
RSRP_037	2030	4910	8464864
RSRP_038	2030	4910	16101438
RSRP_039	2030	4910	23413966
RSRP_040	2030	4910	8464864
RSRP_041	1126	4696	14335774
RSRP_042	1126	4696	7507040
RSRP_043	1126	4696	14335774
RSRP_044	1126	4696	7507040
RSRP_045	1126	4696	14335774
RSRP_046	1126	4696	7507040
RSRP_047	1126	4696	14335774
RSRP_048	1126	4696	7507040
RSRP_049	1126	4696	14335774
RSRP_050	1126	4696	20963280
RSRP_051	1126	4696	7507040
RSRP_052	1126	4696	14335774
RSRP_053	1126	4696	7507040
RSRP_054	277	1464	757340
RSRP_055	277	1464	1110078
RSRP_056	277	1464	389980
RSRP_057	277	980	1079104
RSRP_058	277	980	1588922
RSRP_059	277	980	549264
RSRP_060	174	898	271794
RSRP_061	174	898	398192
RSRP_062	174	898	139442
RSRP_063	146	828	222958

Continued on next page

Table 1 – continued from previous page

instance	$\|T\|$	$\|V\|$	$\|H\|$
RSRP_064	146	828	326288
RSRP_065	146	828	114058
RSRP_066	1126	4593	13572721
RSRP_067	1126	4593	7102630
RSRP_068	277	1443	732152
RSRP_069	277	1443	1072828
RSRP_070	277	1443	377056
RSRP_071	2186	8630	17525972
RSRP_072	1060	3934	18612816
RSRP_073	1060	3934	10018932
RSRP_074	1126	4696	14335774
RSRP_075	1126	4696	7507040
RSRP_076	4194	13222	48530881
RSRP_077	4194	13222	26124420
RSRP_078	4216	13354	48182502
RSRP_079	4216	13354	70055918
RSRP_080	4216	13354	25521577
RSRP_081	4216	13354	49069226
RSRP_082	4216	13354	70942642
RSRP_083	4216	13354	26408301
RSRP_084	277	1464	994532
RSRP_085	277	1464	1347270
RSRP_086	277	1464	627172
RSRP_087	277	1464	1347270
RSRP_088	277	1464	627172
RSRP_089	277	1464	994532
RSRP_090	277	1464	1347270
RSRP_091	277	1464	627172
RSRP_092	277	1464	993868
RSRP_093	277	1464	1346606
RSRP_094	277	1464	626508
RSRP_095	277	1464	994532
RSRP_096	277	1464	1347270
RSRP_097	277	1464	627172
RSRP_098	1126	4696	19234394
RSRP_099	1126	4696	12405660
RSRP_100	1126	4696	19234364
RSRP_101	1126	4696	12405642
RSRP_102	77	154	47214
RSRP_103	77	154	69594
RSRP_104	77	154	24370
RSRP_105	77	154	47926
RSRP_106	77	154	70880
RSRP_107	77	154	24746
RSRP_108	73	146	42692

Continued on next page

Table 1 – continued from previous page

instance	$\|T\|$	$\|V\|$	$\|H\|$
RSRP_109	73	146	63412
RSRP_110	73	146	21796
RSRP_111	75	150	44718
RSRP_112	75	150	66506
RSRP_113	75	150	22730
RSRP_114	336	672	934018
RSRP_115	336	672	1363482
RSRP_116	336	672	498576
RSRP_117	854	1708	3597182
RSRP_118	854	1708	5155568
RSRP_119	854	1708	1961760
RSRP_120	1033	3106	8407556
RSRP_121	1033	3106	12213320
RSRP_122	1033	3106	4487388
RSRP_123	5600	14278	39649900
RSRP_124	5600	14278	57480718
RSRP_125	5600	14278	21205982
RSRP_126	5593	14250	21218266
RSRP_127	1488	2976	8049988
RSRP_128	1488	2976	4362772
RSRP_129	1488	2976	8050512
RSRP_130	1488	2976	11670716
RSRP_131	1488	2976	4363296
RSRP_132	1806	4610	13937042
RSRP_133	1806	4610	20209896
RSRP_134	1806	4610	7421090
RSRP_135	167	486	355108
RSRP_136	167	486	459034
RSRP_137	167	486	191534
RSRP_138	805	9810	29049302
RSRP_139	805	9810	15819548
RSRP_140	987	16790	75274348
RSRP_141	987	16790	48378460
RSRP_142	805	2928	9081282
RSRP_143	805	2928	4944614
RSRP_144	21	126	11292
RSRP_145	21	126	5720
RSRP_146	183	586	328922
RSRP_147	183	586	175398

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.

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

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_001	85383	12345	470571	187	00:00:02:48	00:00:00:05
RSRP_001	257135	27787	1420861	468	00:00:13:50	00:00:00:13
RSRP_001	257135	27787	1420861	476	00:00:17:50	00:00:00:12
RSRP_001	901749	97387	5243971	1206	00:00:01:19	
RSRP_002	119975	22735	898973	292	00:00:04:57	00:00:00:09
RSRP_002	345387	51611	2613647	775	00:00:41:06	00:00:00:28
RSRP_002	345387	51611	2613647	736	00:00:47:48	00:00:00:30
RSRP_002	1354853	193223	10618161	2069	00:00:02:32	
RSRP_003	48359	1551	165511	101	00:00:00:53	00:00:00:00
RSRP_003	141743	1551	468095	208	00:00:02:58	00:00:00:01
RSRP_003	141743	1551	468095	210	00:00:04:48	00:00:00:01
RSRP_003	437585	1551	1659099	443	00:00:00:40	
RSRP_004	290764	40568	1558757	737	00:00:17:46	00:00:00:35
RSRP_004	1144115	122273	6271640	2324	00:02:23:37	00:00:01:52
RSRP_004	1144115	122273	6271640	2324	00:02:53:52	00:00:01:53
RSRP_004	4568917	486701	25820552	5905	00:00:11:25	
RSRP_005	405696	75100	2909432	1050	00:00:33:36	00:00:01:09
RSRP_005	1234284	180082	8944070	2724	00:05:04:51	00:00:04:50
RSRP_005	1234284	180082	8944070	2780	00:05:25:29	00:00:04:07
RSRP_005	6832284	968856	51216464	10257	00:00:32:15	
RSRP_006	162908	4546	576982	416	00:00:02:25	00:00:00:07
RSRP_006	574678	4546	1978986	778	00:00:13:23	00:00:00:14
RSRP_006	574678	4546	1978986	800	00:00:22:17	00:00:00:16
RSRP_006	2215342	4546	8470520	2089	00:00:02:05	
RSRP_007	356045	46805	1989130	881	00:00:33:47	00:00:01:03
RSRP_007	527917	60539	2806022	1165	00:01:03:37	00:00:01:33
RSRP_007	527917	60539	2806022	1178	00:01:24:21	00:00:01:26
RSRP_007	4568709	486649	25819328	5998	00:00:14:08	
RSRP_008	516261	91183	3850315	1243	00:01:10:10	00:00:01:49
RSRP_008	711975	114347	5176105	1623	00:02:54:23	00:00:03:28
RSRP_008	711975	114347	5176105	1652	00:03:07:24	00:00:03:09
RSRP_008	6832024	968752	51214536	10379	00:01:01:33	
RSRP_009	179794	4546	651944	439	00:00:03:22	00:00:00:11
RSRP_009	340458	4546	1099354	605	00:00:11:19	00:00:00:18
RSRP_009	340458	4546	1099354	601	00:00:20:06	00:00:00:18
RSRP_009	2215186	4546	8470000	2325	00:00:03:19	
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_011	452530	81240	3295538	1494	00:01:06:29	00:00:01:51
RSRP_011	1761390	247774	12875538	3948	00:11:16:46	00:00:09:01
RSRP_011	1761390	247774	12875538	4069	00:12:25:37	00:00:08:15
RSRP_011	10910299	1530089	82603147	16105	00:00:45:22	

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_012	163153	4421	573519	600	00:00:02:52	00:00:00:08
RSRP_012	773527	4421	2680837	1047	00:00:19:59	00:00:00:23
RSRP_012	773527	4421	2680837	1035	00:00:31:03	00:00:00:25
RSRP_012	3523997	4421	13589537	3303	00:00:03:09	
RSRP_013	286903	39909	1540812	726	00:00:14:40	00:00:00:31
RSRP_013	1114914	119420	6077741	2118	00:02:12:39	00:00:01:41
RSRP_013	1114914	119420	6077741	2057	00:02:45:55	00:00:01:57
RSRP_013	4584355	487271	26119048	5938	00:00:10:30	
RSRP_014	162908	4546	576982	418	00:00:02:27	00:00:00:07
RSRP_014	574678	4546	1978986	786	00:00:13:17	00:00:00:15
RSRP_014	574678	4546	1978986	790	00:00:22:15	00:00:00:15
RSRP_014	2215342	4546	8470520	2095	00:00:02:04	
RSRP_015	99251	16873	530734	230	00:00:03:27	00:00:00:10
RSRP_015	332480	40098	1695561	604	00:00:26:12	00:00:00:28
RSRP_015	332480	40098	1695561	630	00:00:25:14	00:00:00:23
RSRP_015	847139	92457	4244930	1085	00:00:01:32	
RSRP_016	145887	30633	1040655	342	00:00:11:12	00:00:00:28
RSRP_016	447287	68645	3163729	864	00:01:00:56	00:00:01:02
RSRP_016	447287	68645	3163729	937	00:01:05:26	00:00:01:03
RSRP_016	1287795	181669	8968791	1906	00:00:04:45	
RSRP_017	52525	3245	181041	99	00:00:00:54	00:00:00:01
RSRP_017	135365	3245	408177	200	00:00:02:29	00:00:00:03
RSRP_017	135365	3245	408177	203	00:00:03:33	00:00:00:03
RSRP_017	391861	3245	1170085	376	00:00:00:43	
RSRP_018	106044	17696	570473	235	00:00:03:32	00:00:00:10
RSRP_018	319773	38649	1643900	602	00:00:21:42	00:00:00:29
RSRP_018	319773	38649	1643900	573	00:00:22:27	00:00:00:25
RSRP_018	847332	92476	4245967	1086	00:00:01:43	
RSRP_019	135288	28912	976326	358	00:00:07:56	00:00:00:24
RSRP_019	409801	65721	2881935	812	00:00:51:09	00:00:00:55
RSRP_019	409801	65721	2881935	833	00:00:53:28	00:00:00:58
RSRP_019	1287963	181707	8969857	1902	00:00:03:56	
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_021	65101	2825	191401	106	00:00:01:12	00:00:00:01
RSRP_021	130541	2825	380281	197	00:00:02:44	00:00:00:03
RSRP_021	130541	2825	380281	191	00:00:04:07	00:00:00:02
RSRP_021	551053	2825	1537641	538	00:00:00:44	
RSRP_022	100678	14586	506079	204	00:00:03:05	00:00:00:08
RSRP_022	271444	30032	1399113	513	00:00:18:41	00:00:00:20
RSRP_022	271444	30032	1399113	531	00:00:21:06	00:00:00:18
RSRP_022	1200569	122889	5879184	1495	00:00:02:08	
RSRP_023	137587	25007	945879	309	00:00:06:32	00:00:00:17

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_023	357519	51231	2543771	763	00:00:36:09	00:00:00:43
RSRP_023	357519	51231	2543771	764	00:00:41:41	00:00:00:48
RSRP_023	1829733	242953	12617863	2578	00:00:04:44	
RSRP_024	64039	2825	187013	109	00:00:01:09	00:00:00:01
RSRP_024	129211	2825	376021	195	00:00:02:44	00:00:00:03
RSRP_024	129211	2825	376021	189	00:00:04:03	00:00:00:02
RSRP_024	551231	2825	1538113	541	00:00:00:47	
RSRP_025	21982	2010	70430	67	00:00:00:42	00:00:00:00
RSRP_025	46534	2010	128846	78	00:00:00:53	00:00:00:00
RSRP_025	46534	2010	128846	76	00:00:01:07	00:00:00:00
RSRP_025	140594	2010	408602	152	00:00:00:31	
RSRP_026	14155	1821	45501	66	00:00:00:39	00:00:00:00
RSRP_026	33831	1821	93105	66	00:00:00:51	00:00:00:00
RSRP_026	35931	1821	99125	66	00:00:00:59	00:00:00:00
RSRP_026	115083	1821	330217	116	00:00:00:31	
RSRP_027	105428	14760	531571	212	00:00:03:15	00:00:00:07
RSRP_027	290178	30646	1493229	558	00:00:20:23	00:00:00:20
RSRP_027	290178	30646	1493229	548	00:00:24:03	00:00:00:22
RSRP_027	1200575	122895	5879214	1501	00:00:02:02	
RSRP_028	132525	24361	915111	302	00:00:05:43	00:00:00:17
RSRP_028	347214	49550	2489330	763	00:00:35:41	00:00:00:41
RSRP_028	347214	49550	2489330	745	00:00:40:59	00:00:00:39
RSRP_028	1829739	242959	12617893	2580	00:00:04:50	
RSRP_029	68085	2831	199491	117	00:00:01:16	00:00:00:01
RSRP_029	140061	2831	405815	214	00:00:02:52	00:00:00:02
RSRP_029	140061	2831	405815	207	00:00:04:31	00:00:00:02
RSRP_029	551237	2831	1538143	525	00:00:00:41	
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_031	124000	23420	935526	287	00:00:05:59	00:00:00:11
RSRP_031	370940	54000	2826618	732	00:00:41:25	00:00:00:37
RSRP_031	370940	54000	2826618	770	00:00:48:26	00:00:00:41
RSRP_031	1354909	193279	10639917	2097	00:00:02:33	
RSRP_032	41419	1607	141859	99	00:00:00:51	00:00:00:00
RSRP_032	131411	1607	435607	206	00:00:02:31	00:00:00:01
RSRP_032	131411	1607	435607	198	00:00:04:15	00:00:00:01
RSRP_032	437641	1607	1659155	449	00:00:00:43	
RSRP_033	86047	12485	483285	188	00:00:03:04	00:00:00:05
RSRP_033	233396	25530	1301546	430	00:00:12:20	00:00:00:15
RSRP_033	233396	25530	1301546	441	00:00:15:38	00:00:00:16
RSRP_033	901805	97443	5265727	1209	00:00:01:38	
RSRP_034	41419	1607	141859	97	00:00:00:57	00:00:00:00
RSRP_034	131411	1607	435607	194	00:00:02:37	00:00:00:01

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_034	131411	1607	435607	198	00:00:04:25	00:00:00:01
RSRP_034	437641	1607	1659155	457	00:00:00:51	
RSRP_035	820487	122610	4353957	2054	00:01:42:10	00:00:04:14
RSRP_035	2736604	334345	15052302	5652	00:23:18:51	00:00:16:10
RSRP_035	2736604	334345	15052302	5559	01:00:53:54	00:00:12:16
RSRP_035	18061706	1969831	91908216	22623	00:02:42:53	
RSRP_036	1066358	214189	7592519	2983	00:03:40:24	00:00:08:34
RSRP_036	2937722	505270	21429752	6926	01:20:28:17	00:00:24:19
RSRP_036	2937722	505270	21429752	6765	02:00:29:56	00:00:24:54
RSRP_036	-	-	-	-	-	-
RSRP_037	412359	13080	1433948	1001	00:00:08:34	00:00:00:34
RSRP_037	1648117	13080	5449094	2251	00:01:04:22	00:00:01:09
RSRP_037	1648117	13080	5449094	2271	00:01:38:02	00:00:01:10
RSRP_037	8473291	13080	26999090	7288	00:00:06:14	
RSRP_038	821692	121907	4381962	2062	00:01:44:54	00:00:03:46
RSRP_038	2791331	334418	15095147	5564	00:20:46:28	00:00:11:31
RSRP_038	2791331	334418	15095147	5572	00:23:21:28	00:00:13:07
RSRP_038	18061706	1969831	91908216	22621	00:02:36:38	
RSRP_039	1093157	219206	7770222	2984	00:04:35:03	00:00:10:29
RSRP_039	2982385	516429	21689739	7086	01:16:27:16	00:00:33:40
RSRP_039	2982385	516429	21689739	6931	01:18:52:52	00:00:34:09
RSRP_039	-	-	-	-	-	-
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_041	737691	101282	4008161	1737	00:02:45:11	00:00:07:24
RSRP_041	2409872	262217	12833058	4809	00:20:37:50	00:00:17:44
RSRP_041	2475704	264501	13134132	4956	00:23:13:11	00:00:20:45
RSRP_041	15782220	1454449	78231950	18988	00:02:10:52	
RSRP_042	433263	11284	1462369	822	00:00:15:40	00:00:01:11
RSRP_042	1267417	11284	3960081	1926	00:01:46:01	00:00:02:45
RSRP_042	1188727	11284	3725575	1786	00:01:45:53	00:00:02:07
RSRP_042	7514485	11284	22174685	7178	00:00:11:05	
RSRP_043	746813	101500	4032833	1740	00:02:45:38	00:00:07:58
RSRP_043	2326587	253120	12265965	4699	00:19:31:32	00:00:20:55
RSRP_043	2519261	268068	13409995	4869	00:23:17:27	00:00:21:01
RSRP_043	15782220	1454449	78231950	18830	00:02:29:09	
RSRP_044	444511	11284	1448959	828	00:00:16:57	00:00:01:07
RSRP_044	1128653	11284	3603929	1679	00:01:07:39	00:00:02:21
RSRP_044	1176097	11284	3666079	1744	00:01:43:01	00:00:02:13
RSRP_044	7514485	11284	22174685	7125	00:00:12:28	
RSRP_045	746970	101705	4023396	1712	00:02:32:36	00:00:06:55
RSRP_045	2298944	242577	12050002	4626	00:18:17:13	00:00:22:22
RSRP_045	2298944	242577	12050002	4534	00:17:23:53	00:00:18:12

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_045	15782220	1454449	78231950	18988	00:02:37:09	
RSRP_046	444363	11284	1446351	858	00:00:16:39	00:00:01:11
RSRP_046	1208035	11284	3799205	1866	00:01:18:01	00:00:02:02
RSRP_046	1208035	11284	3799205	1826	00:01:47:34	00:00:02:06
RSRP_046	7514485	11284	22174685	7153	00:00:10:55	
RSRP_047	726568	99633	3935016	1699	00:02:41:22	00:00:07:16
RSRP_047	2468176	264953	12978446	4760	00:20:25:45	00:00:28:46
RSRP_047	2468176	264953	12978446	4988	00:21:30:38	00:00:26:39
RSRP_047	15782220	1454449	78231950	18974	00:02:22:15	
RSRP_048	456313	11284	1515217	841	00:00:17:22	00:00:01:13
RSRP_048	1148581	11284	3611229	1861	00:01:17:45	00:00:02:08
RSRP_048	1148581	11284	3611229	1758	00:01:46:26	00:00:02:17
RSRP_048	7514485	11284	22174685	7134	00:00:11:01	
RSRP_049	745333	101264	3957631	1744	00:02:30:19	00:00:07:42
RSRP_049	2235730	253395	11760276	4352	00:16:20:03	00:00:16:50
RSRP_049	2160922	245241	11294526	4387	00:16:59:09	00:00:18:33
RSRP_049	15782073	1454302	77811783	18900	00:02:21:01	
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_051	463396	11137	1525026	859	00:00:17:18	00:00:01:14
RSRP_051	1176802	11137	3701010	1752	00:01:16:49	00:00:02:15
RSRP_051	1148434	11137	3611082	1788	00:01:49:16	00:00:02:10
RSRP_051	7514338	11137	22174538	7345	00:00:11:14	
RSRP_052	676677	93120	3627671	1649	00:02:27:43	00:00:06:54
RSRP_052	2374729	255070	12498829	4734	00:19:47:54	00:00:18:18
RSRP_052	2519795	263706	13224155	5099	00:23:17:14	00:00:20:50
RSRP_052	15781552	1454475	78210996	18825	00:02:04:27	
RSRP_053	409317	11310	1335819	818	00:00:17:40	00:00:01:19
RSRP_053	1141361	11310	3552699	1746	00:01:25:54	00:00:02:18
RSRP_053	1141361	11310	3552699	1776	00:01:55:53	00:00:02:23
RSRP_053	7513817	11310	22162781	7232	00:00:11:27	
RSRP_054	86089	15659	452662	208	00:00:03:59	00:00:00:11
RSRP_054	270079	33993	1397476	523	00:00:18:40	00:00:00:26
RSRP_054	276311	34649	1421048	505	00:00:21:31	00:00:00:27
RSRP_054	847248	92618	4273807	1091	00:00:01:42	
RSRP_055	109402	24904	787212	278	00:00:06:53	00:00:00:24
RSRP_055	305420	50194	2175698	640	00:00:35:25	00:00:01:02
RSRP_055	317953	51211	2248775	659	00:00:39:36	00:00:00:50
RSRP_055	1287904	181830	8996604	1902	00:00:04:23	
RSRP_056	34378	3406	112818	76	00:00:00:51	00:00:00:01
RSRP_056	102774	3406	309282	158	00:00:02:11	00:00:00:02
RSRP_056	94346	3406	284238	152	00:00:02:54	00:00:00:02
RSRP_056	391970	3406	1168850	377	00:00:00:43	

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_057	93264	14182	473327	193	00:00:03:51	00:00:00:09
RSRP_057	273935	30541	1411708	543	00:00:23:53	00:00:00:24
RSRP_057	273935	30541	1411708	547	00:00:28:09	00:00:00:27
RSRP_057	1200195	122977	5915962	1510	00:00:03:14	
RSRP_058	120808	22926	847472	280	00:00:06:34	00:00:00:18
RSRP_058	348184	49078	2504134	730	00:00:51:50	00:00:01:00
RSRP_058	348184	49078	2504134	740	00:00:52:40	00:00:00:48
RSRP_058	1829236	243010	12652228	2570	00:00:05:40	
RSRP_059	59624	2944	178680	101	00:00:01:09	00:00:00:01
RSRP_059	119820	2944	354032	175	00:00:02:27	00:00:00:02
RSRP_059	119820	2944	354032	178	00:00:03:38	00:00:00:02
RSRP_059	551132	2944	1536764	493	00:00:00:44	
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
RSRP_060	305740	35630	1521254	421	00:00:00:48	
RSRP_061	56338	14394	390724	151	00:00:02:20	00:00:00:08
RSRP_061	173121	29289	1201993	359	00:00:09:41	00:00:00:17
RSRP_061	173121	29289	1201993	351	00:00:10:06	00:00:00:16
RSRP_061	464797	69089	3213253	690	00:00:01:16	
RSRP_062	20049	2171	66197	67	00:00:00:37	00:00:00:00
RSRP_062	42825	2171	119205	74	00:00:00:54	00:00:00:00
RSRP_062	42825	2171	119205	74	00:00:01:04	00:00:00:00
RSRP_062	140729	2171	408073	153	00:00:00:32	
RSRP_063	36642	7766	189201	88	00:00:01:13	00:00:00:02
RSRP_063	99219	13783	490770	193	00:00:03:52	00:00:00:06
RSRP_063	99219	13783	490770	192	00:00:04:07	00:00:00:06
RSRP_063	251423	30011	1245694	356	00:00:00:45	
RSRP_064	45390	12222	318136	131	00:00:01:57	00:00:00:06
RSRP_064	134092	23660	930172	281	00:00:07:11	00:00:00:14
RSRP_064	134092	23660	930172	280	00:00:07:22	00:00:00:14
RSRP_064	382059	58047	2631215	591	00:00:01:11	
RSRP_065	16249	1975	55177	67	00:00:00:37	00:00:00:00
RSRP_065	32017	1975	87825	67	00:00:00:47	00:00:00:00
RSRP_065	32017	1975	87825	67	00:00:00:58	00:00:00:00
RSRP_065	115217	1975	329849	121	00:00:00:31	
RSRP_066	643292	93147	3462409	1618	00:01:55:22	00:00:05:59
RSRP_066	2443982	280537	12795215	4906	00:23:14:29	00:00:21:10
RSRP_066	2448736	270464	12812521	4899	00:22:25:16	00:00:18:51
RSRP_066	15018374	1454269	74250269	18009	00:02:00:56	
RSRP_067	398986	11104	1301225	802	00:00:19:14	00:00:01:00
RSRP_067	1234496	11104	3837206	1913	00:01:31:12	00:00:02:11
RSRP_067	1234496	11104	3837206	1924	00:02:00:53	00:00:02:07
RSRP_067	7109282	11104	20969742	6913	00:00:09:33	
RSRP_068	87420	16252	473534	210	00:00:03:33	00:00:00:11

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_068	272324	35284	1407469	514	00:00:17:01	00:00:00:27
RSRP_068	320975	40691	1650592	596	00:00:24:05	00:00:00:31
RSRP_068	822040	92576	4148099	1063	00:00:01:39	
RSRP_069	138231	29711	991303	360	00:00:08:08	00:00:00:25
RSRP_069	409227	65467	2896716	798	00:00:45:38	00:00:01:01
RSRP_069	460820	72230	3250242	900	00:00:53:22	00:00:01:08
RSRP_069	1250634	181788	8723238	1847	00:00:03:57	
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_071	859228	21188	2853217	1747	00:00:43:14	00:00:02:42
RSRP_071	2364752	21188	7757783	3443	00:03:19:07	00:00:04:49
RSRP_071	2364752	21188	7757783	3364	00:04:19:12	00:00:04:42
RSRP_071	17538956	21188	53333011	16299	00:00:27:57	
RSRP_072	652058	91161	3591004	1713	00:01:58:45	00:00:05:03
RSRP_072	2048934	250919	11376228	4099	00:10:30:08	00:00:11:02
RSRP_072	2092519	258292	11584461	4176	00:12:45:01	00:00:12:00
RSRP_072	20393573	1787866	102713615	24340	00:01:36:16	
RSRP_073	400640	9913	1376955	858	00:00:12:35	00:00:00:49
RSRP_073	1105266	9913	3842375	1520	00:00:59:48	00:00:01:19
RSRP_073	1164588	9913	4064191	1604	00:01:43:55	00:00:01:15
RSRP_073	10025174	9913	31170265	8397	00:00:08:25	
RSRP_074	710008	96181	3817396	1680	00:02:44:52	00:00:06:07
RSRP_074	2440348	261033	12774648	4845	00:20:18:45	00:00:15:37
RSRP_074	2342453	250966	12246455	4742	00:19:05:46	00:00:19:39
RSRP_074	15781526	1454449	78210922	18828	00:02:12:38	
RSRP_075	422453	11284	1354517	816	00:00:18:09	00:00:01:15
RSRP_075	1140007	11284	3518927	1797	00:01:20:24	00:00:02:15
RSRP_075	1204827	11284	3685315	1885	00:01:46:40	00:00:02:15
RSRP_075	7513791	11284	22162705	7114	00:00:11:51	
RSRP_076	2101712	306563	11568458	5487	00:08:43:22	00:00:21:57
RSRP_076	-	-	-	-	-	-
RSRP_076	-	-	-	-	-	-
RSRP_076	-	-	-	-	-	-
RSRP_077	1228243	33975	4339601	2894	00:00:57:08	00:00:03:16
RSRP_077	3442746	33975	11976036	5171	00:04:59:23	00:00:06:45
RSRP_077	3442746	33975	11976036	5286	00:06:46:28	00:00:05:53
RSRP_077	26145192	33975	89351525	25583	00:00:58:34	
RSRP_078	1993105	294444	10816502	5346	00:10:16:59	00:00:24:46
RSRP_078	-	-	-	-	-	-
RSRP_078	-	-	-	-	-	-
RSRP_078	-	-	-	-	-	-
RSRP_079	2740552	542018	19993611	7893	00:23:07:53	00:00:56:48
RSRP_079	-	-	-	-	-	-

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_079	-	-	-	-	-	-
RSRP_079	-	-	-	-	-	-
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_081	2035475	297927	11246994	5435	00:09:12:58	00:00:20:26
RSRP_081	-	-	-	-	-	-
RSRP_081	-	-	-	-	-	-
RSRP_081	-	-	-	-	-	-
RSRP_082	2758776	541800	20281403	8040	00:23:42:09	00:01:12:16
RSRP_082	-	-	-	-	-	-
RSRP_082	-	-	-	-	-	-
RSRP_082	-	-	-	-	-	-
RSRP_083	1233754	34309	4332590	2860	00:00:54:05	00:00:03:21
RSRP_083	3566001	34309	12261824	5459	00:05:38:45	00:00:07:31
RSRP_083	3566001	34309	12261824	5429	00:07:20:07	00:00:06:29
RSRP_083	-	-	-	-	-	-
RSRP_084	107536	16638	659109	249	00:00:05:30	00:00:00:12
RSRP_084	357661	36907	2227860	644	00:00:28:29	00:00:00:30
RSRP_084	357661	36907	2227860	670	00:00:35:52	00:00:00:28
RSRP_084	1084482	92660	7201243	1465	00:00:02:32	
RSRP_085	133194	26936	1027288	344	00:00:10:22	00:00:00:25
RSRP_085	410714	58876	3199104	842	00:00:58:48	00:00:01:00
RSRP_085	410714	58876	3199104	846	00:01:07:56	00:00:01:03
RSRP_085	1525138	181872	12410680	2308	00:00:04:50	
RSRP_086	42916	3448	200846	94	00:00:01:11	00:00:00:01
RSRP_086	128708	3448	626426	209	00:00:03:33	00:00:00:02
RSRP_086	128708	3448	626426	213	00:00:06:35	00:00:00:03
RSRP_086	629204	3448	3609646	695	00:00:01:01	
RSRP_087	121974	26232	943030	302	00:00:09:27	00:00:00:23
RSRP_087	380082	55912	2951626	777	00:00:45:04	00:00:01:06
RSRP_087	380082	55912	2951626	753	00:00:53:00	00:00:01:09
RSRP_087	1525138	181872	12410680	2302	00:00:04:32	
RSRP_088	41400	3448	197718	94	00:00:01:07	00:00:00:01
RSRP_088	126116	3448	588550	196	00:00:03:20	00:00:00:03
RSRP_088	126116	3448	588550	209	00:00:06:25	00:00:00:03
RSRP_088	629204	3448	3609646	700	00:00:00:55	
RSRP_089	84754	14932	509923	212	00:00:04:21	00:00:00:11
RSRP_089	332923	34197	2070604	628	00:00:24:03	00:00:00:28
RSRP_089	332923	34197	2070604	617	00:00:31:43	00:00:00:29
RSRP_089	1084482	92660	7201243	1466	00:00:02:28	
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

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_090	1525138	181872	12410680	2307	00:00:05:08	
RSRP_091	41400	3448	197718	94	00:00:01:12	00:00:00:01
RSRP_091	126116	3448	588550	196	00:00:03:27	00:00:00:03
RSRP_091	126116	3448	588550	209	00:00:06:26	00:00:00:03
RSRP_091	629204	3448	3609646	695	00:00:01:03	
RSRP_092	105125	16535	644280	241	00:00:04:22	00:00:00:13
RSRP_092	351198	37240	2157259	674	00:00:29:10	00:00:00:34
RSRP_092	351198	37240	2157259	656	00:00:36:35	00:00:00:35
RSRP_092	1083818	92660	7192995	1460	00:00:02:28	
RSRP_093	131454	27220	995558	326	00:00:09:43	00:00:00:26
RSRP_093	407461	61603	3116841	826	00:00:50:06	00:00:01:16
RSRP_093	407461	61603	3116841	815	00:00:56:45	00:00:01:09
RSRP_093	1524474	181872	12401104	2365	00:00:05:30	
RSRP_094	51104	3448	257274	100	00:00:01:25	00:00:00:01
RSRP_094	131320	3448	631362	206	00:00:03:29	00:00:00:03
RSRP_094	131320	3448	631362	215	00:00:06:47	00:00:00:03
RSRP_094	628540	3448	3602726	690	00:00:01:05	
RSRP_095	95930	15480	590971	231	00:00:04:45	00:00:00:12
RSRP_095	323587	35093	1920904	594	00:00:25:32	00:00:00:29
RSRP_095	287186	31928	1693425	535	00:00:26:55	00:00:00:26
RSRP_095	1084482	92660	7201243	1466	00:00:02:35	
RSRP_096	127144	26134	980270	346	00:00:08:47	00:00:00:32
RSRP_096	381077	57659	2895407	753	00:01:00:11	00:00:01:13
RSRP_096	335687	51493	2556523	686	00:00:56:42	00:00:01:07
RSRP_096	1525138	181872	12410680	2294	00:00:05:09	
RSRP_097	52596	3448	260126	104	00:00:01:20	00:00:00:02
RSRP_097	124808	3448	574914	200	00:00:03:28	00:00:00:03
RSRP_097	124808	3448	574914	205	00:00:06:39	00:00:00:03
RSRP_097	629204	3448	3609646	697	00:00:01:01	
RSRP_098	786951	93798	5153133	1968	00:03:14:27	00:00:08:05
RSRP_098	2667454	241215	17391668	5433	00:21:32:06	00:00:22:51
RSRP_098	2667454	241215	17391668	5723	01:04:47:41	00:00:27:34
RSRP_098	20680319	1454622	140691311	26634	00:02:49:11	
RSRP_099	528808	11457	2798718	1108	00:00:32:04	00:00:01:28
RSRP_099	1384206	11457	7302492	2341	00:02:15:41	00:00:02:36
RSRP_099	1293454	11457	6935478	2199	00:03:46:50	00:00:03:07
RSRP_099	12412584	11457	74508432	14163	00:00:22:41	
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_101	518362	11457	2746542	1084	00:00:30:28	00:00:01:18
RSRP_101	1349658	11457	7292312	2273	00:02:13:16	00:00:02:42
RSRP_101	1349658	11457	7292312	2322	00:04:06:19	00:00:02:43
RSRP_101	12412566	11457	74508320	14220	00:00:23:59	

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_102	17263	2727	95812	68	00:00:00:37	00:00:00:00
RSRP_102	29276	3880	162671	71	00:00:00:54	00:00:00:00
RSRP_102	29276	3880	162671	72	00:00:01:02	00:00:00:00
RSRP_102	53553	6613	304506	87	00:00:00:30	
RSRP_103	25262	4874	187656	68	00:00:00:55	00:00:00:00
RSRP_103	36570	6112	273818	88	00:00:01:25	00:00:00:01
RSRP_103	36570	6112	273818	88	00:00:01:22	00:00:00:01
RSRP_103	82006	12840	624188	148	00:00:00:44	
RSRP_104	8170	386	28522	68	00:00:00:32	00:00:00:00
RSRP_104	13864	386	46626	68	00:00:00:39	00:00:00:00
RSRP_104	13864	386	46626	68	00:00:00:42	00:00:00:00
RSRP_104	24636	386	91088	67	00:00:00:29	
RSRP_105	16799	2665	92678	67	00:00:00:38	00:00:00:00
RSRP_105	31296	4088	173229	72	00:00:00:57	00:00:00:00
RSRP_105	31296	4088	173229	73	00:00:01:01	00:00:00:00
RSRP_105	54342	6690	308201	90	00:00:00:31	
RSRP_106	27295	5127	203501	73	00:00:00:49	00:00:00:00
RSRP_106	39950	6558	299838	92	00:00:01:15	00:00:00:01
RSRP_106	39950	6558	299838	93	00:00:01:24	00:00:00:01
RSRP_106	83446	12994	633190	152	00:00:00:36	
RSRP_107	7760	386	27450	67	00:00:00:32	00:00:00:00
RSRP_107	14152	386	47444	67	00:00:00:36	00:00:00:00
RSRP_107	14152	386	47444	67	00:00:00:41	00:00:00:00
RSRP_107	25012	386	92364	67	00:00:00:30	
RSRP_108	16100	2534	88875	67	00:00:00:36	00:00:00:00
RSRP_108	29000	3852	159513	71	00:00:00:56	00:00:00:00
RSRP_108	29000	3852	159513	69	00:00:00:59	00:00:00:00
RSRP_108	48501	6067	275348	84	00:00:00:29	
RSRP_109	24997	4745	185671	67	00:00:00:49	00:00:00:00
RSRP_109	36955	6037	277809	88	00:00:01:10	00:00:00:01
RSRP_109	36955	6037	277809	90	00:00:01:20	00:00:00:01
RSRP_109	74776	11768	568806	140	00:00:00:33	
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_111	17052	2686	94827	67	00:00:00:40	00:00:00:00
RSRP_111	29862	3914	168115	73	00:00:00:57	00:00:00:00
RSRP_111	29862	3914	168115	72	00:00:01:04	00:00:00:00
RSRP_111	50821	6371	293730	85	00:00:00:32	
RSRP_112	25807	4897	192097	71	00:00:00:49	00:00:00:00
RSRP_112	44132	7050	333734	104	00:00:01:27	00:00:00:01
RSRP_112	44132	7050	333734	104	00:00:01:39	00:00:00:01
RSRP_112	78454	12366	603484	139	00:00:00:31	
RSRP_113	8068	376	29058	67	00:00:00:32	00:00:00:00

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_113	13424	376	45904	67	00:00:00:34	00:00:00:00
RSRP_113	13424	376	45904	67	00:00:00:39	00:00:00:00
RSRP_113	22988	376	87978	67	00:00:00:30	
RSRP_114	124985	17057	721339	260	00:00:04:11	00:00:00:06
RSRP_114	370297	40891	2082439	766	00:00:26:20	00:00:00:18
RSRP_114	370297	40891	2082439	747	00:00:29:19	00:00:00:14
RSRP_114	1047083	114311	6242293	1417	00:00:01:31	
RSRP_115	199024	34156	1508358	435	00:00:07:55	00:00:00:16
RSRP_115	578242	83066	4353940	1290	00:01:18:47	00:00:00:45
RSRP_115	578242	83066	4353940	1269	00:01:29:16	00:00:00:44
RSRP_115	1588505	226941	12383177	2464	00:00:02:32	
RSRP_116	59193	1681	231433	122	00:00:01:06	00:00:00:01
RSRP_116	156299	1681	507035	269	00:00:04:42	00:00:00:02
RSRP_116	156299	1681	507035	250	00:00:08:12	00:00:00:02
RSRP_116	499683	1681	2052117	473	00:00:00:43	
RSRP_117	278166	38488	1502250	668	00:00:13:27	00:00:00:25
RSRP_117	869820	89826	4668662	1678	00:01:31:58	00:00:01:14
RSRP_117	869820	89826	4668662	1675	00:01:44:20	00:00:01:14
RSRP_117	4019632	425538	22487680	5144	00:00:06:14	
RSRP_118	390611	71475	2845889	963	00:00:29:26	00:00:00:56
RSRP_118	1028853	147469	7486497	2190	00:02:43:29	00:00:02:21
RSRP_118	1028853	147469	7486497	2173	00:03:02:27	00:00:02:38
RSRP_118	5997576	846804	45004718	8889	00:00:19:07	
RSRP_119	145780	4272	499960	361	00:00:02:02	00:00:00:05
RSRP_119	526890	4272	1773436	730	00:00:11:21	00:00:00:13
RSRP_119	526890	4272	1773436	739	00:00:18:22	00:00:00:13
RSRP_119	1964652	4272	7305604	1853	00:00:01:59	
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_121	633184	134790	4712924	1572	00:01:27:36	00:00:04:29
RSRP_121	1552020	303838	11547114	3326	00:10:16:47	00:00:11:19
RSRP_121	1552020	303838	11547114	3421	00:10:06:08	00:00:10:09
RSRP_121	14222924	2018264	96853296	19599	00:01:22:11	
RSRP_122	264140	7678	998446	478	00:00:04:51	00:00:00:19
RSRP_122	759500	7678	2619430	1017	00:00:27:11	00:00:00:40
RSRP_122	703134	7678	2458620	935	00:00:36:32	00:00:00:31
RSRP_122	4492618	7678	13826816	3845	00:00:04:16	
RSRP_123	1930201	302840	11487017	5271	00:03:10:40	00:00:07:42
RSRP_123	-	-	-	-	-	-
RSRP_123	-	-	-	-	-	-
RSRP_123	-	-	-	-	-	-
RSRP_124	2749024	566930	21739579	7810	00:06:20:10	00:00:17:14
RSRP_124	-	-	-	-	-	-

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_124	-	-	-	-	-	-
RSRP_124	-	-	-	-	-	-
RSRP_125	1068522	36097	4101705	2963	00:00:28:00	00:00:01:06
RSRP_125	3695301	36097	13285947	5195	00:03:05:06	00:00:02:23
RSRP_125	3695301	36097	13285947	5144	00:05:23:13	00:00:02:43
RSRP_125	21229194	36097	74930609	18364	00:00:15:07	
RSRP_126	1059202	36013	4051951	2962	00:00:29:10	00:00:01:04
RSRP_126	3554657	36013	12870029	5117	00:03:01:16	00:00:02:33
RSRP_126	3554657	36013	12870029	5030	00:05:08:38	00:00:02:48
RSRP_126	21241420	36013	74965569	18375	00:00:16:45	
RSRP_127	497295	67727	2761327	1370	00:00:34:12	00:00:01:29
RSRP_127	2253795	233513	12689019	4170	00:07:34:13	00:00:07:22
RSRP_127	2253795	233513	12689019	4436	00:08:53:35	00:00:07:29
RSRP_127	8995503	950837	51398533	11484	00:00:24:49	
RSRP_128	276947	7445	940049	748	00:00:04:51	00:00:00:18
RSRP_128	1122935	7445	3668453	1625	00:00:51:06	00:00:01:01
RSRP_128	1122935	7445	3668453	1658	00:01:15:25	00:00:00:57
RSRP_128	4367871	7445	15888297	4082	00:00:05:48	
RSRP_129	514961	69863	2866831	1332	00:00:37:49	00:00:01:31
RSRP_129	2153442	220964	12157414	3961	00:07:39:17	00:00:06:21
RSRP_129	2153442	220964	12157414	4327	00:08:49:10	00:00:08:20
RSRP_129	8996180	950990	51401912	11509	00:00:25:55	
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_131	269791	7445	912631	743	00:00:04:56	00:00:00:18
RSRP_131	1193183	7445	4001539	1641	00:00:47:30	00:00:01:05
RSRP_131	1193183	7445	4001539	1611	00:01:08:59	00:00:01:02
RSRP_131	4368395	7445	15890053	4076	00:00:05:44	
RSRP_132	782929	109247	4359000	2132	00:02:02:09	00:00:05:36
RSRP_132	3236915	344271	17630182	6355	00:23:59:38	00:00:21:17
RSRP_132	3236915	344271	17630182	6143	01:02:01:29	00:00:21:14
RSRP_132	15607844	1679214	84464781	19415	00:01:44:24	
RSRP_133	1056449	196397	8004125	2993	00:04:22:19	00:00:11:51
RSRP_133	3718902	538010	27889234	7898	01:16:44:40	00:00:32:21
RSRP_133	3718902	538010	27889234	8179	01:18:27:16	00:00:33:19
RSRP_133	-	-	-	-	-	-
RSRP_134	413532	11252	1395266	1044	00:00:13:24	00:00:00:56
RSRP_134	1776942	11252	5676828	2530	00:01:33:54	00:00:02:39
RSRP_134	1776942	11252	5676828	2550	00:02:04:32	00:00:02:29
RSRP_134	7428540	11252	24569568	6602	00:00:10:42	
RSRP_135	51019	8541	290876	114	00:00:01:18	00:00:00:02
RSRP_135	85476	12322	480371	163	00:00:03:07	00:00:00:03
RSRP_135	85476	12322	480371	173	00:00:03:33	00:00:00:03

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_135	399562	45392	2088657	514	00:00:00:42	
RSRP_136	74055	17531	562573	166	00:00:02:27	00:00:00:06
RSRP_136	96744	23190	723134	212	00:00:04:42	00:00:00:09
RSRP_136	96744	23190	723134	207	00:00:05:07	00:00:00:08
RSRP_136	547069	89459	3923685	816	00:00:01:10	
RSRP_137	31383	1325	115433	75	00:00:00:38	00:00:00:00
RSRP_137	54755	1325	183293	90	00:00:01:02	00:00:00:00
RSRP_137	54755	1325	183293	89	00:00:01:17	00:00:00:00
RSRP_137	192407	1325	600533	197	00:00:00:33	
RSRP_138	1570890	87038	8904560	4934	00:12:22:26	00:00:33:23
RSRP_138	-	-	-	-	-	-
RSRP_138	-	-	-	-	-	-
RSRP_138	-	-	-	-	-	-
RSRP_139	826057	20427	2898185	2672	00:00:21:48	00:00:01:02
RSRP_139	2618183	20427	8739899	3847	00:01:32:48	00:00:01:55
RSRP_139	2618183	20427	8739899	3668	00:02:39:31	00:00:01:56
RSRP_139	15831937	20427	60454981	14291	00:00:15:18	
RSRP_140	3639659	103609	31758855	8721	01:17:04:26	00:02:50:51
RSRP_140	-	-	-	-	-	-
RSRP_140	-	-	-	-	-	-
RSRP_140	-	-	-	-	-	-
RSRP_141	3377672	35114	16993798	6519	00:12:59:52	00:00:56:15
RSRP_141	-	-	-	-	-	-
RSRP_141	-	-	-	-	-	-
RSRP_141	-	-	-	-	-	-
RSRP_142	477436	65106	2647128	1563	00:02:03:05	00:00:03:36
RSRP_142	1653420	168196	9056088	3172	00:07:15:27	00:00:06:59
RSRP_142	1653420	168196	9056088	3311	00:07:57:09	00:00:07:03
RSRP_142	10140411	1064455	58089051	13173	00:00:43:41	
RSRP_143	231149	6663	809849	833	00:00:06:00	00:00:00:31
RSRP_143	973537	6663	3330957	1368	00:00:39:04	00:00:01:06
RSRP_143	973537	6663	3330957	1381	00:00:56:58	00:00:01:02
RSRP_143	4948879	6663	18926709	4618	00:00:05:53	
RSRP_144	5562	1024	27351	68	00:00:00:32	00:00:00:00
RSRP_144	8852	1238	44505	68	00:00:00:34	00:00:00:00
RSRP_144	8852	1238	44505	68	00:00:00:34	00:00:00:00
RSRP_144	12688	1610	63509	67	00:00:00:29	
RSRP_145	2809	297	7859	68	00:00:00:32	00:00:00:00
RSRP_145	3965	297	10887	67	00:00:00:30	00:00:00:00
RSRP_145	3965	297	10887	67	00:00:00:29	00:00:00:00
RSRP_145	5901	297	16295	67	00:00:00:30	
RSRP_146	71458	10210	403617	158	00:00:02:10	00:00:00:03
RSRP_146	104750	12916	578923	209	00:00:03:34	00:00:00:05
RSRP_146	104750	12916	578923	205	00:00:04:37	00:00:00:05
RSRP_146	367857	39989	2093736	502	00:00:00:50	

Continued on next page

Table 2 – continued from previous page

instance	columns	rows	non-zeros	memory	time	resolving time
RSRP_147	37148	1394	130814	71	00:00:00:52	00:00:00:00
RSRP_147	59000	1394	195166	93	00:00:01:12	00:00:00:00
RSRP_147	59000	1394	195166	95	00:00:01:39	00:00:00:00
RSRP_147	176296	1394	630006	191	00:00:00:46	

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.