



Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

RALF BORNDÖRFER
BORIS GRIMM
MARKUS REUTHER
THOMAS SCHLECHTE

Optimization of Handouts for Rolling Stock Rotation Visualization

This work has been developed within the Research Campus MODAL (Mathematical Optimization and Data Analysis Laboratories) funded by the German Ministry of Education and Research (BMBF).

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

Optimization of Handouts for Rolling Stock Rotations Visualization

Ralf Borndörfer, Boris Grimm, Markus Reuther, Thomas Schlechte¹


1 Zuse Institute Berlin
Takustrasse 7, 14195 Berlin, Germany
{borndoerfer, grimm, reuther, schlechte}@zib.de

Abstract

A railway operator creates (*rolling stock*) *rotations* in order to have a precise master plan for the operation of a timetable by railway vehicles. A rotation is considered as a cycle that multiply traverses a set of *operational days* while covering trips of the timetable. As it is well known, the proper creation of rolling stock rotations by, e.g., optimization algorithms is challenging and still a topical research subject. Nevertheless, we study a completely different but strongly related question in this paper, i.e.: How to visualize a rotation? For this purpose, we introduce a basic *handout concept*, which directly leads to the visualization, i.e., handout of a rotation. In our industrial application at DB Fernverkehr AG, the handout is exactly as important as the rotation itself. Moreover, it turns out that also other European railway operators use exactly the same methodology (but not terminology). Since a rotation can have many handouts of different quality, we show how to compute optimal ones through an integer program (IP) by standard software. In addition, a construction as well as an improvement heuristic are presented. Our computational results show that the heuristics are a very reliable standalone approach to quickly find near-optimal and even optimal handouts. The efficiency of the heuristics is shown via a computational comparison to the IP approach.

1 Introduction

Railway companies that earn their money with passenger trips have to offer a timetable of planned passenger trips. Usually, these trips repeat from week to week, i.e., a trip offered on Monday is offered in exactly the same way each following Monday. Moreover, trips often repeat for each day of the week or each working day. Hence, the timetable could be completely defined as a set of trips for a set of seven consecutive operational days, which is called the *standard week*. This week is repeated as long the timetable is valid. To operate these trips rolling stock vehicles have to be assigned to the trips. In order to have a master plan for the validity period of the timetable *rolling stock rotations* are developed by a railway operator in a certain point of preparation. The rolling stock rotations are cycles in a graph containing a vertex for each trip and arcs for operating two trips consecutively by a rolling stock vehicle. Each cycle covers timetabled trips for the purpose of deciding what happens to a dedicated railway vehicle after the operation of a timetabled trip. Each of these decisions is crucial for the operational efficiency and must absolutely agree with several intricate conditions: vehicle composition rules, maintenance constraints, and infrastructure capacities. This variety of requirements gives rise to a very challenging competition on rolling stock rotation planning. Our productive optimization software ROTOR participates in this competition for one of the leading railway operators in Europe: DB Fernverkehr AG (DBF). A distinguished feature of ROTOR is the generation of handouts for rolling stock rotations. In fact, even when ROTOR finds rolling stock rotations that can be proved to be optimal ROTOR does not stop computing! That's not a bug, that's a feature. In reality, it turns out that it is just not enough to compute a solution. We also have to think about how to make it easy to manage. In simple words, we ask how to print a rolling stock



rotation on a physical paper? A good visualization of a rolling stock rotation is mandatory and can not be neglected during the complex planning process that leads to the rotation's operation. We assume that a rolling stock rotation is given and is not allowed to be modified in this paper. For this rotation, we introduce a function, i.e., *handout* that directly leads to a visualization, see Section 2. A dedicated rotation can have many visualizations of different qualities. To this end, we propose an optimization approach in Section 3. This approach is composed of an integer program, a construction heuristic, and an improvement heuristic. These components are described in detail in Section 3. We evaluate these algorithms for real-world rotations of DBF, which we present in Section 4 of the paper. Finally, we provide a conclusion.

2 Rolling Stock Rotation Handouts

Visualizations of rolling stock rotations are made in order to conveniently communicate the rotations in further planning steps. The methodology of these visualizations in terms of the standard week is standardized across many railway companies from different (European) countries. For example, NS Reizigers from the Netherlands, the Österreichische Bundesbahn from Austria, Trenitalia from Italy, and, of course, Deutsche Bahn are using the visualization concept that we call *handout concept*. The vocabulary among these companies is not standardized, but the methodology is exactly the same.

Imagine a directed graph D with a vertex for each trip of a cyclic timetable and an arc (t, t') between two trips t, t' if a rolling stock vehicle operates t' after t . The sequence of operated trips of a rolling stock vehicle forms a cycle in D . We assume that each rolling stock rotation becomes visualized separately. This assumption does not completely hold in industry, i.e., sometimes a whole set of cycles is meant by a rotation and also the whole set of cycles is visualized together. If several cycles are identified as a rotation in industry, a single cycle is called *sub-rotation*. The explicit consideration of sub-rotations complicates notation and does not contribute any benefit here. Therefore, we assume that we are given a single rolling stock rotation in this paper.

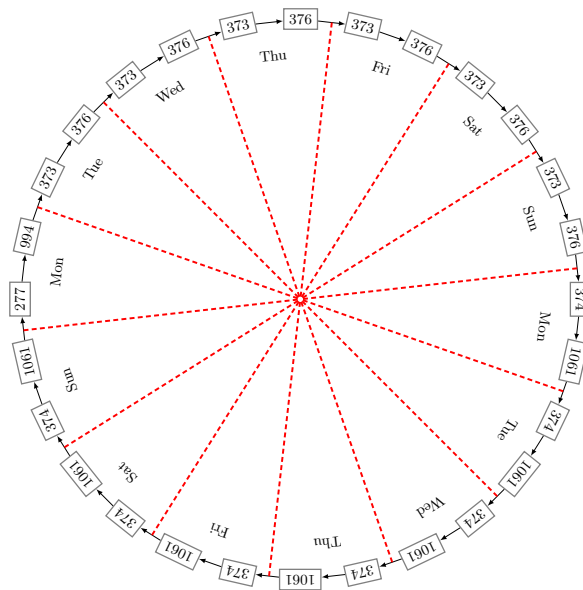
2.1 Handout Segments

By construction, each rolling stock rotation runs an integral number of times through the set of operational days \mathbb{D} until it reaches again its first trip. We denote this number by $\mathfrak{v} \in \mathbb{Z}_+$ for the rolling stock rotation that we want to visualize. The number of rolling stock vehicles needed to operate the rotation is also equal to \mathfrak{v} , i.e., \mathfrak{v} rolling stock vehicles run through the rotation one by one.

For the visualization, we imagine a rolling stock rotation as a cycle that runs through timetabled trips of a standard week, i.e., $|\mathbb{D}| = 7$, as illustrated in Figure 1. Each number inside of a box of Figure 1 describes a train number of a passenger train run of the timetable. In order to find a convenient visualization, the rotation is split into $|\mathbb{D}| \cdot \mathfrak{v}$ segments. A segment represents the operation of the rotation on a single day of operation. In this way, the rolling stock rotation with $\mathfrak{v} = 2$ in Figure 1 decomposes into $|\mathbb{D}| \cdot \mathfrak{v} = 14$ segments. A segment may be empty, i.e., does not contain any timetabled trip at all.

Note that the assumption $|\mathbb{D}| = 7$ is motivated by the fact that we consider the standard week in most cases of our application. Nevertheless, all considerations in this paper also apply to other non-trivial planning horizons. For example, five operational days could be considered if one only plans for weekdays or $|\mathbb{D}| = 28$ is applicable for handouts that visualize the operational days of four weeks.

Let S be the set of segments of the rolling stock rotation to be visualized and let $\mathbb{D}(s) \in \{\text{Mon}, \dots, \text{Sun}\} =: \mathbb{D}$ denote the day of operation of the segment $s \in S$ and the set of days of operation, respectively. In addition, we denote by $[\mathfrak{v}] := \{k \in \mathbb{N} \mid k \leq \mathfrak{v}\}$ the set of the first \mathfrak{v}



■ **Figure 1** Splitting of a rolling stock rotation with $\mathfrak{v} = 2$ into 14 segments.

natural numbers. A *handout* is a function $\Omega : S \mapsto [\mathfrak{v}]$ such that

$$\mathbb{D}(s) = \mathbb{D}(t) \quad \Rightarrow \quad \Omega(s) \neq \Omega(t) \quad \forall s, t \in S.$$

That is, Ω assigns different values of $[\mathfrak{v}]$ to each pair of segments that are both associated with the same day of operation. By definition, always exactly \mathfrak{v} segments of a rolling stock rotation are associated with the same day of operation. Thus, if a *handout* Ω is at hand each segment $s \in S$ can be precisely identified by $\Omega(s)$ and $\mathbb{D}(s)$. This is an evident motivation for the concept of *handouts*. For most of the rolling stock rotations in industry \mathfrak{v} is much greater than one. Indeed, rolling stock rotations with $\mathfrak{v} > 40$ are not an exception at DBF. For those rotations a *handout* obviously provides a significant gain: Segments can now be precisely distinguished during all further planning steps. The major objective of a *handout* Ω is to create the standardized visualization for rolling stock rotations that we mentioned above. Indeed, Ω completely defines this visualization. The visualization appears by printing all segments one below the other such that they are lexicographically ordered according to Ω and the day of operation. Figure 2 provides two different visualizations (i.e., the two tables) derived from two different *handout* functions for the rolling stock rotation of Figure 1. In both tables the first three columns state the value $\Omega(s)$ for a segment $s \in S$ associated with $\mathbb{D}(s)$. The underlying rolling stock rotation is *not* modified by those visualizations. Therefore, the successor relations of the segments that are defined by the rolling stock rotation remain. We denote by $\text{next}(s) \in S$ that segment that directly follows $s \in S$ in the rotation. These successor relations are stated in the last columns of the two tables. The rolling stock rotation that is visualized by a *handout* Ω can now be followed in the tables. For example, for the direct successor $t \in S$ of the segment $s \in S$ (i.e., $\text{next}(s) = t$) with $\Omega(s) = 2$ and $\mathbb{D}(s) = \text{Sun}$ on the left of Figure 2 we know that $\Omega(\text{next}(s)) = \Omega(t) = 1$ and that $\mathbb{D}(t) = \text{Mon}$ from the visualization. In this way, the successor segment t can be easily found and we are able to double-check that both *handouts* in Figure 2 visualize the rolling stock rotation of Figure 1.

2.2 Handout Quality

The *handout* function Ω is extensively distributed as a planning tool in the railway industry. In particular, the manual planning of rolling stock rotations is based on similar functions. There,

$\Omega(s)$	$\mathbb{D}(s)$	$s \in \mathcal{S}$	$\Omega(\text{next}(s))$
1	Mon	374 → 1061	2
1	Tue	373 → 376	2
1	Wed	374 → 1061	2
1	Thu	373 → 376	2
1	Fri	374 → 1061	2
1	Sat	373 → 376	2
1	Sun	374 → 1061	2
2	Mon	277 → 994	1
2	Tue	374 → 1061	1
2	Wed	373 → 376	1
2	Thu	374 → 1061	1
2	Fri	373 → 376	1
2	Sat	374 → 1061	1
2	Sun	373 → 376	1

$\Omega(s)$	$\mathbb{D}(s)$	$s \in \mathcal{S}$	$\Omega(\text{next}(s))$
1	Mon	374 → 1061	1
1	Tue	374 → 1061	1
1	Wed	374 → 1061	1
1	Thu	374 → 1061	1
1	Fri	374 → 1061	1
1	Sat	374 → 1061	1
1	Sun	374 → 1061	2
2	Mon	277 → 994	2
2	Tue	373 → 376	2
2	Wed	373 → 376	2
2	Thu	373 → 376	2
2	Fri	373 → 376	2
2	Sat	373 → 376	2
2	Sun	373 → 376	1

■ **Figure 2** Two different handouts for the rolling stock rotation of Figure 1.

timetabled trips of a planned rotation are equipped with values of \mathfrak{v} . This is comprehensive because such functions can be made visible (as we have seen for segments) and determine a large part of the rolling stock rotations as well.

Thus, it is not surprising that the function Ω has some expectations in the railway industry. For example, for all segments $s \in \mathcal{S}$ arranged in the left of Figure 2

$$\Omega(\text{next}(s)) = (\Omega(s) \bmod \mathfrak{v}) + 1 \quad (1)$$

holds. If a successor relation follows equation (1) we call it *logical turn*. It is desired to have as much logical turns as possible in a handout in order to obtain a visualization in that the rolling stock rotations can be easily followed. Imagine that the segments are printed on paper in the order given Ω . In case of a logical turn preceding and succeeding segments are printed very close to each other. It is notable that it is not always possible to find a handout such that all successor relations are logical turns. If we consider a handout that has the maximal number of logical turns (i.e., $|\mathbb{D}| \cdot \mathfrak{v}$) and assume $\mathfrak{v} = |\mathbb{D}| \cdot k$ for $k \in \mathbb{Z}_+$, we recognize (by following the logical turns) that the handout visualizes seven individual sub-rotations. In this way, a single rotation (without sub-rotations) with $\mathfrak{v} = |\mathbb{D}| \cdot k$ for $k \in \mathbb{Z}_+$ can not have the maximal number of logical turns.

By definition, Ω induces a natural partition of the segments of the rotation into *blocks*. All segments $s \in B$ of a block $B \subseteq \mathcal{S}$ have the same $\Omega(s)$, but a different day of operation. Therefore, each block is of cardinality $|\mathbb{D}|$ and there are always exactly \mathfrak{v} blocks induced by a handout. On the left hand side of Figure 2 the first seven segments $B \subset \mathcal{S}$ with $\Omega(s) = 1$ for all $s \in B$ form a block¹.

¹ At DB Fernverkehr AG a block is called *Umlauftag*. This translates to the English term “rotation day” or “day of rotation” which we do not use here.

Another desired property (if not the most important) is related to the blocks of a handout. For example, on the right hand side of Figure 2 we recognize that the first seven segments, (i.e., the segments of the first block) are all equal. We say that two segments $u, v \in S$ with $\Omega(u) = \Omega(v)$ have a *difference* if they cover trips with different train numbers. In reality some more details are taken into account for the quantification of differences but those details do not affect the presentation here. A handout is desired to have as few differences in its blocks as possible. If the number of differences is low, patterns can be easily remembered if they are arranged appropriately in blocks. For example, the connection of trips with train number 374 to trips with train number 1061 is reflected in the handout on the right of Figure 2. This points out that a handout can only contain fewer differences if appropriate patterns along the rotation exist. This is an important requirement during rolling stock rotation planning. It is called *regularity* and is tackled by an integrated hypergraph-based modeling approach for rolling stock rotations, see [2].

2.3 Handout Optimization Problem

Finding Ω is an optimization problem because many handouts of different quality w.r.t. logical turns and differences exist for a given rolling stock rotation. We call this problem *handout optimization problem* (HOP). In order to have a formal reference for this problem, we formulate it as a quadratic assignment problem in this section. To this end, let $x_s^\omega \in \{0, 1\}$ be a binary decision variable that takes value one if and only if $\Omega(s) = \omega$ for the segment $s \in S$. In order to qualify handouts we denote by $\text{diff}(s, t) \in \mathbb{Z}_+$ the number of different train numbers in $s \in S$ and $t \in S$ with $s \neq t$. A straight-forward formulation of the HOP as a special quadratic assignment problem (HOP_{QAP}) reads as follows:

$$\min \sum_{\omega \in [\mathbf{v}]} \left(\sum_{\substack{s, t \in S \\ s \neq t}} \text{diff}(s, t) x_s^\omega x_t^\omega - \alpha \sum_{s \in S} x_s^\omega x_{\text{next}(s)}^{\text{next}(\omega)} \right) \quad (\text{HOP}_{\text{QAP}})$$

$$\sum_{\omega \in [\mathbf{v}]} x_s^\omega = 1 \quad \forall s \in S, \quad (2)$$

$$\sum_{s \in S(d)} x_s^\omega = 1 \quad \forall d \in \mathbb{D}, \omega \in [\mathbf{v}], \quad (3)$$

$$x_s^\omega \in \{0, 1\} \quad \forall s \in S, \omega \in [\mathbf{v}].$$

Equalities (2) and (3) of program (HOP_{QAP}) constrain the binary x -variables to perfectly match all segments of S to pairs of $\mathbb{D} \times [\mathbf{v}]$, i.e., to form a perfect matching (i.e., an assignment) in a bipartite graph that is composed of the two disjoint node parts S and $\mathbb{D} \times [\mathbf{v}]$. Thus, any feasible solution to program (HOP_{QAP}) precisely defines a handout Ω where $\Omega(s) = \omega$ if and only if $x_s^\omega = 1$. By the objective function of program (HOP_{QAP}) we model both the minimization of differences as well as the maximization of logical turns. To this end, quadratic terms are used. Note that these two desired properties compete with each other. A handout that minimizes differences may not maximize the number of logic turns and vice versa, see Figure 2. In general, this leads to a bi-criteria optimization problem, but we only consider a single objective function in model (HOP_{QAP}) in this paper. In order to adjust the relationship between logical turns and differences to a desired level, the parameter $\alpha \in \mathbb{Q}$ is introduced. Note that the setting of α does not lead to a proper prioritization of logical turns over differences or vice versa in our industrial application at DBF.

► **Theorem 1.** *The handout optimization problem is \mathcal{NP} -hard even for $\alpha = 0$.*

Proof. For the 3-partition problem with $n = 3k$ for $n, k \in \mathbb{N}$ the integer numbers $a_1, \dots, a_n \in \mathbb{N}$ fulfilling $\sum_{i=1}^n a_i = ka, a \in \mathbb{N}$ are given. The problem is to decide whether there is a block

partition of the numbers a_1, \dots, a_n into triples such that the sum over the three integers of one triple sums up to a . We model an instance of the 3-partition problem as an instance of the HOP (assuming $\alpha = 0$) by introducing k blocks over a time horizon of a days as follows. For each integer a_i of the 3-partition instance a_i identical segments are considered as segments of the HOP instance. Two segments $s, t \in S$ that originate from different integer values are considered to have no differences, i.e., $\text{diff}(s, t) = 0$. Otherwise, i.e., if s and t belong to the same integer value a_i we define $\text{diff}(s, t) = -\frac{2a_i}{(n-1)n}$. It is easy to see that if and only if the objective value of the respective HOP instance is $-ka$, a feasible 3-partition is at hand. This shows that the HOP is \mathcal{NP} -hard because the 3-partition problem is well known to be \mathcal{NP} -hard as well, see [1] where a similar argumentation is made for the (k, v) -balanced graph partitioning problem. ◀

3 Handout Optimization

In this section we show how the handout optimization problem (HOP) can be modeled and solved via a mixed integer linear program. In addition, we provide fast construction and improvement heuristics with which near-optimal solutions can be obtained very quickly. The intention of the heuristics is to prevent the generation of the handouts from consuming more computation time than the proper optimization of the rolling stock rotations themselves (ROTOR reports all intermediate solutions to the user). Nevertheless, we consider the IP model in this paper in order to evaluate the quality of the solutions obtained from the heuristics. In addition, we will see that program (HOP_{QAP}) is not always trivially solvable and that the solution obtained from the heuristics can significantly contribute to standard integer programming algorithms. In fact, when ROTOR is used with default settings the MIP model is not used for the creation of handouts. Instead, the heuristic procedure is called, which is described in detail in Sections 3.2 and 3.3. For the sake of simplicity, we assume $\alpha = 1$ for the parameter that weighs logical turns and differences in the objective function of the HOP from now on.

3.1 Handouts via Integer Programming

In this section we present an integer programming (IP) model for the HOP. An IP formulation for the HOP derives from program (HOP_{QAP}) as follows. The main idea is to linearize its quadratic objective function by introducing two types of additional variables. We use the first type, namely the y -variables in order to model the minimization of differences. The second variable type, i.e., the z -variables are introduced in order to linearize the objective function in terms of logical turns. Program (HOP_{MIP}) states our linear IP formulation for the HOP:

$$\min \sum_{\omega \in [\mathbf{v}]} \sum_{\substack{s, t \in S \\ s \neq t}} \text{diff}(s, t) y_{s,t}^\omega - \alpha \sum_{s \in S} z_s \quad (\text{HOP}_{\text{MIP}})$$

$$\sum_{\omega \in [\mathbf{v}]} x_s^\omega = 1 \quad \forall s \in S, \quad (4)$$

$$\sum_{s \in S(d)} x_s^\omega = 1 \quad \forall d \in \mathbb{D}, \omega \in [\mathbf{v}], \quad (5)$$

$$\sum_{t \in S(d)} y_{s,t}^\omega = x_s^\omega \quad \forall s \in S, \omega \in [\mathbf{v}], d \in \mathbb{D} \setminus \{\mathbb{D}(s)\}, \quad (6)$$

$$x_s^\omega - x_{\text{next}(s)}^{\text{next}(\omega)} \leq 1 - z_s \quad \forall s \in S, \omega \in [\mathbf{v}], \quad (7)$$

$$x_s^\omega \in \{0, 1\} \quad \forall s \in S, \omega \in [\mathbf{v}], \quad (8)$$

$$y_{s,t}^\omega \in \mathbb{Q}_+ \quad \forall s, t \in S, \omega \in [\mathbf{v}], \quad (9)$$

$$z_s \in \mathbb{Q}_+ \quad \forall s \in S. \quad (10)$$

In program (HOP_{MIP}) the x -variables and constraints (4) as well as (5) are exactly taken over from program (HOP_{QAP}) of Section 2. The y - and the z -variables are continuous, but they automatically take binary values if the x -variables are all binary in a solution. This is easy to see if their organization is understood via the following explanation.

y -variables.

Assume a solution (i.e., handout) with $\Omega(s) = \omega$, i.e., $x_s^\omega = 1$ holds for the segment $s \in S$. Since s is assigned to the block ω the solution contains $|\mathbb{D}| - 1$ other segments in block ω . In order to measure the differences of s to the other segments of the block we use the y -variables. The variable $y_{s,t}^\omega \in \{0, 1\}$ with $s, t \in S$ and $\omega \in [\mathbf{v}]$ is a binary decision variable such that: If and only if $y_{s,t}^\omega = 1$ we have $\Omega(s) = \Omega(t) = \omega$ for $s \in S$ and $t \in S$, i.e., the segments s and t belong to the same block ω if and only if $y_{s,t}^\omega = 1$. In order to force the correct configuration of the y -variables w.r.t. the x -variables we introduce the constraints (6), which work as follows. Again, assume a solution with $x_s^\omega = 1$ for the segment $s \in S$. Now consider one dedicated constraint of type (6) associated with the segment s , the block ω , and a day d that is different to $\mathbb{D}(s)$. This constraint configures exactly one y -variable to take value one. This variable is associated with one other segment $t \in S(d)$ such that $\Omega(t) = \omega$ and $\mathbb{D}(t) = d$ holds. Under this construction it is easy to linearize the first part of the objective function of program (HOP_{QAP}) by the y -variables as it is denoted in program (HOP_{MIP}). The combinatorial structure that is implied by the y -variables in a solution can be interpreted as follows. It is a set of \mathbf{v} cliques with $|\mathbb{D}|$ nodes each in the undirected graph (V, E) with $V := S \times [\mathbf{v}]$ and $E = V \times V$. Each clique corresponds to a block where each two segments of one block are connected by an edge of the clique. Each of the clique's edges is used to measure the differences of the connected segments.

z -variables.

As already mentioned, the z -variables are responsible for the contribution to the objective function value in terms of logical turns. The variable z_s that is associated with the segment $s \in S$ equals one if and only if s is followed by a logical turn in the handout. More precisely, if $z_s = 1$ equation (1) holds for $s \in S$ and its succeeding segment $\text{next}(s) \in S$. This is accomplished via constraints (7) of program (HOP_{MIP}). If $z_s = 1$, the solution needs to fulfill $x_s^\omega \leq x_{\text{next}(s)}^{\text{next}(\omega)}$ for all $\omega \in [\mathbf{v}]$, which is precisely the case if s is followed by a logical turn.

3.2 Construction Heuristic

In this section, we present an algorithm that creates a feasible handout, which is not necessarily optimal, i.e., we introduce a construction heuristic. The procedure is subdivided into two stages as it is outlined by Algorithm 1. There, the segments are partitioned into blocks but without assigning a value to them. This is the first stage. In the second stage a value of $[v]$ is assigned to each of these blocks. Consequently, the segments take over the values assigned to each block in that they are contained. The sum of differences of the segments within the blocks is minimized in the first stage, while the number of logical turns is maximized in the second stage.

```

1 HANDOUTCONSTRUCTIONHEURISTIC( S ) // S is the set of segments
2 {
3   // partition S into blocks  $\mathcal{B} = \{B_1, \dots, B_v\}$ ,  $B_i \subseteq S$ ,  $|B_i| = |\mathbb{D}|$ 
4    $\mathcal{B} := \text{MINIMIZEDIFFERENCES}(S)$ ;
5
6   // compute bijection  $\Omega_{\mathcal{B}}: \mathcal{B} \mapsto [v]$ 
7    $\Omega_{\mathcal{B}} := \text{MAXIMIZELOGICALTURNS}(\mathcal{B})$ ;
8
9   // finally create  $\Omega: S \mapsto [v]$ 
10  for (  $B \in \mathcal{B}$  ) { for (  $s \in B$  ) {  $\Omega(s) := \Omega_{\mathcal{B}}(B)$ ; } }
11 }

```

■ **Algorithm 1** Two-stage construction heuristic for HOP

The first stage of the handout construction heuristic is denoted in Algorithm 2. Up to line 5 the set of blocks is initialized such that each block contains exactly one segment associated with Monday (where Monday is arbitrary). Afterwards, the blocks are iteratively increased. This iteration is made for each day of operation independently. In one iteration an assignment problem is set up and solved with an $\mathcal{O}(|V|^3)$ implementation of the classical Hungarian method, see [4]. The denoted program in line 12 can be seen as a standard assignment problem in a bipartite graph in that the two node parts are formed by the set $S(d)$ (all segments associated with the day of operation in the current iteration) and the set of blocks \mathcal{B} . A (binary) variable z_{sB} decides if the segment $s \in S(d)$ is assigned to the block $B \in \mathcal{B}$. Note that, these z -variables are different from those in (HOP_{MIP}). The objective function of the assignment problem is configured for the minimization of differences.

The second stage of the handout construction heuristic is denoted in Algorithm 3. The blocks $\mathcal{B} = \{B_1, \dots, B_v\}$ with $|B| = |\mathbb{D}|$ for all $B \in \mathcal{B}$ that are created in the first stage serve as input data for this procedure. It remains to assign a value of $[v]$ to each block of \mathcal{B} . In other words, a permutation of the blocks \mathcal{B} has to be found. This is an Asymmetric Travelling Salesman Problem (ATSP) in that the blocks \mathcal{B} take over the role of the cities and the weight of a connection between blocks corresponds to the number of logical turns between them. Note that this ATSP instance has a slightly special objective function, which makes it non-obvious to derive an alternative \mathcal{NP} -hardness proof if one assumes that the blocks are already built. We also solve this ATSP instance heuristically as denoted in Algorithm 3. At first, the assignment relaxation of the ATSP is set up and solved. There, each binary variable $z_{B_1 B_2}$ is defined to be equal to one if and only if block $B_2 \in \mathcal{B}$ is the directed successor of block $B_1 \in \mathcal{B}$ in the handout (assuming no subtours). Possibly appearing subtours are patched in order to form a proper Hamiltonian cycle through the blocks \mathcal{B} . The objective function that is taken into account during this procedure corresponds to the maximization of logical turns, see line 6 of Algorithm 3.

```

1 MINIMIZEDIFFERENCES( S ) // S is the set of segments
2 {
3   B := ∅;
4   // initialize blocks
5   for( s ∈ S : D(s) = Mon ) { B := B ∪ {s}; }
6
7   for( d ∈ D \ {Mon} )
8   {
9     S(d) := {s ∈ S | D(s) = d}; // |S(d)| = v
10
11    // assign segments of S(d) to blocks B by solving
12    min {
13      ∑_{s ∈ S(d)} ∑_{B ∈ B} ∑_{s_B ∈ B} diff(s, 1, s_B) z_{sB}
14      |
15      ∑_{s ∈ S(d)} z_{sB} = 1   ∀ B ∈ B,
16      ∑_{B ∈ B} z_{sB} = 1   ∀ s ∈ S(d),
17      z_{sB} ∈ {0, 1}   ∀ s ∈ S(d), B ∈ B
18    }
19
20    // update blocks B according to the optimal assignment
21    for( B ∈ B ) { for( s ∈ S : z_{sB} = 1 ) { B := B ∪ {s}; } }
22  }
23 }

```

■ **Algorithm 2** First stage of handout construction heuristic

```

1 MAXIMIZELOGICALTURNS( B ) // B are v blocks of segments
2 {
3   // compute assignment relaxation of
4   // ATSP over the graph (B, B × B)
5
6   max {
7     ∑_{B_1 ∈ B} ∑_{B_2 ∈ B} ∑_{\substack{s_1 ∈ B_1, \\ s_2 ∈ B_2: \\ s_2 = \text{next}(s_1)}} z_{B_1 B_2}
8     |
9     ∑_{B_1 ∈ B} z_{B_1 B_2} = 1   ∀ B_2 ∈ B,
10    ∑_{B_2 ∈ B} z_{B_1 B_2} = 1   ∀ B_1 ∈ B,
11    z_{B_1 B_2} ∈ {0, 1}   ∀ B_1, B_2 ∈ B
12  }
13
14  // compute Ω_B : B → [v] by patching subtours:
15  i := 1;
16
17  for( B_1 ∈ B )
18  {
19    if( Ω_B(B_1) is not computed yet )
20    {
21      Ω_B(B_1) := i;
22      B_1 := B_2; // s.t. z_{B_1 B_2} = 1
23      i := i + 1;
24      goto 13;
25    }
26  }
27 }

```

■ **Algorithm 3** Second stage of handout heuristic

Finally, the following is easy to see but not natural in general for construction heuristics and, therefore, remarkable:

► **Corollary 2.** *Algorithm 1 always produces a feasible handout.*

3.3 Improvement Heuristic

Since the handout construction heuristic from the previous section may not compute an optimal handout, it is worth considering ways to improve the quality of the constructed handout. To this end, we present an adaptation of the famous Kernighan-Lin heuristic for the graph partitioning problem, see [3]. Let Ω be a feasible handout and let $\text{quality}(\Omega) \in \mathbb{Q}$ be the *negative* of the objective function value of program (HOP_{QAP}) for $x_s^\omega = 1$ if and only if $\Omega(s) = \omega$ for all segments $s \in S$. Further, let $\Omega_{s,t}$ for $s, t \in S$ with $\mathbb{D}(s) = \mathbb{D}(t)$ be the resulting handout that derives from Ω by interchanging s and t , i.e., $\Omega_{s,t}(s) := \Omega(t)$, $\Omega_{s,t}(t) := \Omega(s)$, and $\Omega_{s,t}(o) := \Omega(o)$ for $o \in S \setminus \{s, t\}$. We define

$$\text{gain}(\Omega, s, t) := \text{quality}(\Omega_{s,t}) - \text{quality}(\Omega)$$

as the quality gain that we obtain if we interchange $s \in S$ and $t \in S$ with $\mathbb{D}(s) = \mathbb{D}(t)$ in Ω . The interchange operation can be seen as an 1-opt move. The idea of the Kernighan-Lin procedure is to find beneficial k -opt moves that are iteratively composed by chaining 1-opt moves. This is illustrated in Algorithm 4, which is always initiated with the best known handout and $G := 0$. The procedure always chooses an interchange that maximizes the quality gain, see line 3. If such an interchange already gives an improvement, the incumbent handout Ω^* is updated. The main idea of the partitioning procedure that was introduced by Kernighan and Lin is to increase the k -opt move by further 1-opt moves, but only if the gain over all previous moves is positive, see line 7. This criterion is known as *positive gain criterion* and leads to a significant increase of the look-ahead since it allows investigating 1-opt moves with $\text{gain}(\Omega, s, t) < 0$. This criterion was latter also successfully applied to the symmetric TSP, see [5]. Within ROTOR we call Algorithm 4 as long as no further improvement is found.

```

1 HANDOUTIMPROVEMENTHEURISTIC(  $\Omega, G$  ) //  $\Omega: S \mapsto [p]$ ,  $G$  is the current gain
2 {
3    $(s, t) := \text{argmax}\{\text{gain}(\Omega, s, t) \mid s, t \in S : \mathbb{D}(s) = \mathbb{D}(t)\}$ ; // choose if not unique
4
5   if(  $\text{quality}(\Omega_{s,t}) > \text{quality}(\Omega^*)$  ) {  $\Omega^* := \Omega_{s,t}$ ; } // update incumbent handout
6
7   if(  $G + \text{gain}(\Omega, s, t) > 0$  ) // go ahead whenever gain is positiv
8   {
9     HANDOUTIMPROVEMENTHEURISTIC(  $\Omega_{s,t}, G + \text{gain}(\Omega, s, t)$  );
10  }
11 }
```

■ **Algorithm 4** Handout Improvement Heuristic: Ω^* is considered as a global variable that stores the incumbent handout. The recursion is initialized by calling `HANDOUTIMPROVEMENTHEURISTIC($\Omega, 0$)`.

4 Computational Results

In this section we present computational results for the optimization of handouts. We implemented and integrated the proposed algorithms within ROTOR. The considered HOP instances are based on optimized rolling stock rotations also computed by ROTOR for real world instances provided by DB Fernverkehr AG. They differ in characteristics as number of trips, fleet sizes to cover the trips, or possible connections of trips. In particular, these instances represent a wide spectrum of problem sizes of interest in practice. All computations were performed on CPU with 3.50 GHz, 16 GB of RAM in multi thread mode with four cores using `Gurobi 6.0` with a runtime limit of one hour (3600 seconds).

We consider the following three solution approaches to the HOP:

■ **Table 1** Computational results for handout optimization problems.

\mathbf{v}	columns	rows	δ_1	δ_3	t_1	t_2	t_3	c_1	c_2	c_3
2	182	208	0	0	0.01	0.99	0.01	36	36	36
3	588	426	0	0	0.16	0.61	0.15	111	111	111
4	1372	720	0	0	0.87	0	0.86	100	100	100
5	2660	1090	1	0	5.43	0.99	2.50	160	165	160
6	4578	1536	0	0	5.61	0.71	4.98	220	221	220
7	7252	2058	0	0	13.25	0.03	14.02	212	214	212
8	10808	2656	0	1	115.42	0.99	291.24	195	197	195
9	15372	3330	1	2	2621.30	1.25	3600.03	286	292	286
10	21070	4080	1	1	228.52	0.49	447.97	375	377	375
11	28028	4906	13	13	3600.01	1.25	3600.01	463	478	464
12	36372	5808	1	1	1839.42	0.42	1113.20	492	498	492
12	36372	5808	9	8	3600.01	1.25	3600.01	736	743	728
13	46228	6786	1	1	3600.01	0.61	3600.01	546	546	546
13	46228	6786	12	12	3600.01	0.99	3600.01	590	593	588
15	70980	8970	11	9	3600.01	0.55	3600.01	368	372	367
17	103292	11458	62	17	3600.01	0.55	3600.02	1141	522	520
18	122598	12816	7	7	3600.02	0.42	3600.06	952	959	951
23	255668	20746	8	7	3600.03	0.61	3600.05	1294	1286	1281
25	328300	24450	44	9	3600.12	1.25	3600.16	3233	1998	1990
29	512372	32770	∞	7	3600.00	0.49	3600.00	-	1470	1468
31	625828	37386	7	7	3600.11	0.71	3600.07	1528	1530	1527
31	625828	37386	46	9	3600.08	0.99	3600.07	3371	2006	1991

1. static solving of the MIP formulation (HOP_{MIP}) with `Gurobi 6.0`,
2. a sequential heuristic approach of Algorithm 1 and Algorithm 4, and
3. a MIP approach using the solution of approach 2 as warmstart.

Table 1 provides the results of these three solution approaches for our test set. The number of vehicles \mathbf{v} range from 2 to 31, see the first column of Table 1. As a consequence, we have to solve different models of HOP_{MIP} . The corresponding number of columns and rows of (HOP_{MIP}) are given in columns two and three. Columns δ_1 and δ_3 provide the final relative gap for approach 1 and 3, respectively. Note that we explicitly refuse to include δ_2 because without approach 1 it would not be possible to provide a quality measure for approach 2. The relative gap is defined between the best integer objective UB and the objective of the best lower bound LB as $100 \cdot \frac{UB-LB}{UB}$. The last six columns show the computation time t_i and the final cost value c_i for all three approaches $i \in \{1, 2, 3\}$.

In all cases the heuristic finds solutions within at most two seconds, see column t_2 . By using the MIP approach we are able to benchmark the quality of the heuristic solutions, see the values in columns, c_1 to c_3 . This impressively demonstrates the high quality of the solutions provided by approach 2. In addition, for the larger instances, e.g., $\mathbf{v} > 15$, the warmstart approach is beneficial in comparison to static MIP approach, see column δ_1 and δ_2 . For example the static MIP approach was not able to provide any solution for the scenario with 29 vehicles after one hour. In contrast to that the approach 3 provides a solution with a gap of at most 7%. Note that the MIP was only able to improve the heuristic solution with value 1470 to 1468 after one hour. For the complete set of instances one could observe that approach 3 improves the solution quality after one hour of at most 1%. Thus, we conclude that approach 2 is a very powerful heuristic to solve real world instances of the HOP.

5 Conclusion

The construction of handouts for given rolling stock rotations is a crucial task which is widespread and common in practice. In contrast, the problem is not at all investigated in the operations research literature. We introduced the handout visualization problem for rolling stock rotations and developed an optimization approach, i.e., an IP formulation. Moreover, we designed a heuristic solution approach based on an assignment-based construction heuristic and a classical combinatorial Kernighan-Lin procedure in order to construct high quality handouts fast. By means of the IP formulation, and the resulting lower bounds, we are able to benchmark the heuristic approach. The results for the test set provided by our project partner DB Fernverkehr AG demonstrate that the proposed heuristic is a very fast and powerful approach.

References

- 1 K. Andreev and H. Räcke. Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939, 2006.
- 2 R. Borndörfer, M. Reuther, T. Schlechte, and S. Weider. A Hypergraph Model for Railway Vehicle Rotation Planning. In Alberto Caprara and Spyros Kontogiannis, editors, *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 20 of *OpenAccess Series in Informatics (OASICs)*, pages 146–155, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 3 B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.
- 4 H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.
- 5 S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.