Shunting operations at flat yards: Retrieving freight railcars from storage tracks

WORKING PAPER
May 2016

Florian Jaehn
Sustainable Operations and Logistics, University of Augsburg, Germany
florian.jaehn@wiwi.uni-augsburg.de

Alena Otto
Management Information Science, University of Siegen, Germany
alena.otto@uni-siegen.de

Kilian Seifried
Logistics and Supply Chain Management, University of Mannheim, Germany
kilian.seifried@bwl.uni-mannheim.de

Abstract

In this paper, we study the railcar retrieval problem (RRT) where specified numbers of certain types of railcars have to be withdrawn from the storage tracks of a flat yard. This task arises in the daily operations of workshop yards for railcar maintenance. The objective is to minimize the total cost of shunting via methods such as minimizing the usage of shunting engines.

We describe the RRT formally, present a mixed-integer program formulation, and prove the general case to be NP-hard. For some special cases, exact algorithms with polynomial runtimes are proposed. We also analyze several intuitive heuristic solution approaches motivated by observed real-world planning routines. We evaluate their average performances in simulations with different scenarios and provide their worst-case performance guarantee. We show that although the analyzed heuristics result in much better solutions than the naive planning approach, they are still on average 30%-50% from the optimal objective value and may result in up to 14 times higher costs in the worst case. Therefore, we conclude that optimization should be implemented in practice in order to save valuable resources. Furthermore, we analyze the impacts of yard layout and the widespread organizational routine of presorting on the railcar retrieval cost.

1 Introduction

One of the current priorities in transport policy is the promotion of freight rail transportation to reduce carbon dioxide emissions and to relieve congested roads [7]. However, rail companies still have to reduce the costs and travel times of customer freight to increase the attractiveness of rail transportation. One of the important cost drivers is railcar maintenance. Annual railcar operation, routine maintenance and repair costs amount to about $800 and $10,000 per railcar depending on its type, age and annual mileage [4]. Recently, maintenance has received high visibility due to the large-scale modernization of freight railcars with modern K- and LL-blocks, so-called low-noise blocks made of more protective materials that reduce the wearing-off of railcar wheels, as a part of the European noise-reduction program.

Railcar maintenance is not a trivial task for companies: it takes a lot of time and resources. A railcar in operation visits a repair shop several times a year, and approximately two of these visits are unplanned [12]. For relatively new (three- to seven-year-old) intermodal double pocket railcars, the time to recovery usually lasts more than 10 days, and in 20% of cases, it even exceeds 40 days [18]. These numbers are even higher for a typical railcar in Germany because the average freight railcar age is approximately 21–30 years [21].

Railcars spend significant portions of their downtimes associated with maintenance and repair on storage tracks awaiting the opening of a suitable workshop lane and the arrival of ordered parts at the workshop. If the waiting time is rather long, railcars are put on storage tracks at low-cost sites, which are as a rule far from the workshops. The storage tracks in Hamm/Germany, for example, mainly host railcars from the workshop in Paderborn/Germany. Periodically, usually once a day, railcars of certain types ordered by the workshop are retrieved from the storage tracks, assembled into a train and delivered to the workshop. Because storage tracks represent a *flat yard* with no hump or hill that facilitate shunting, the retrieval of railcars is a very time consuming and expensive operation. Moreover, due to the variability inherent in maintenance and repair, the planning time horizon of the repair workshops is rather short. Therefore, faster retrieval will increase the utilization of the workshop resources and speed up the time to recovery of the railcars.

In this paper, we are looking for the most efficient policies for the retrieval of railcars of given types from storage tracks. In other words, for the specified parking positions and types of railcars on the storage tracks, we decide which railcars have to be withdrawn so that the shunting cost and time are minimized.

Considering the large amount of academic literature already published on shunting operations, see Hansmann and Zimmermann [11], Gatto et al [9] and Boysen et al [2] for surveys and literature reviews, there is astonishingly little

research that might assist practitioners in the outlined planning problem. The majority of the literature deals with the re-ordering of railcars according to a specified sequence. The objective is usually to minimize the number of classification tracks or the number of classification stages. In contrast, we investigate the retrieval of a limited number of railcars of certain types, thereby the sequence of the retrieved railcars can be arbitrary.

Another core theme in the shunting literature is devoted to operations in depots of self-propelled vehicles, as a rule passenger train units or trams (see, e.g., Di Stefano and Koči [5], Winter and Zimmermann [22], and Blasum et al [1]). A common objective here is to position the arriving vehicles on storage tracks so that they do not block each other while departing from the depot. This problem setting is different from ours because when the railcars arrive, we do not know when they will be transported to the workshop or which railcars will be a part of the same request.

The work that comes closest to the problem in our paper is conducted by Lübbecke and Zimmermann [17]. The authors analyze the allocation of railcars within in-plant rail networks, specifically which yards have to supply railcars of the required types to the points of demand so that transportation and shunting costs are minimized. Two sub-problems arise: how many railcars of each type will be supplied from each yard for each request, and how to retrieve the railcars of the requested types from a given yard. The latter problem is similar to ours, except for the assumption that the railcar retrieval cost depends linearly on the order number of the railcars on the track. In other words, it is about three times more expensive to retrieve the ninth railcar on the track than the third railcar. This assumption does not hold in our problem, where the engine can dislocate a *block* of railcars in one move (see Section 2).

To the best of our knowledge, only Hall [10] examines operations at railcar maintenance centers. He performs strategic layout analysis, and as a part of the analysis, Hall [10] develops analytical formulas on the expected number of shunting moves under some restrictive assumptions (e.g., retrieval of only one railcar at a time, random distribution of railcars on tracks and a retrieval cost that is linear with the order number of the railcar).

Several articles investigate operations at rail maintenance centers in general, without taking railcars or storage tracks into account explicitly [19, 20, 13]. They develop models for scheduling operations to critical resources of limited capacities. These articles do not treat shunting as a part of optimization, instead they assume certain retrieval policies, such as FIFO or LIFO.

Note that we are not dealing with a general problem of scheduling maintenance and repair operations in this paper. We refer the interested reader to, e.g., Doganay and Bohlin [6] and Budai et al [3] and to the general reviews of Kobbacy and Murthy [15] and Lidén [16].

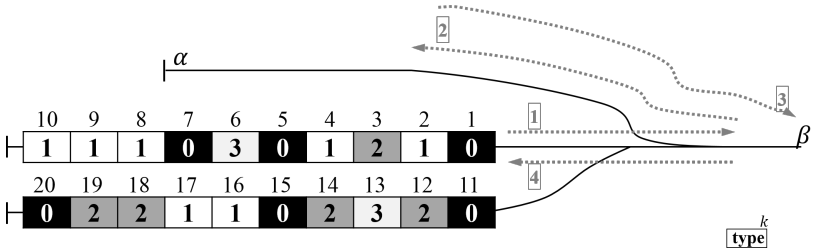


Figure 1: Example of $K = 20$ railcars of types $t = 0, 1, 2, 3$ on storage tracks. The dotted line visualizes the movement of the engine transporting railcars 3 and 4 to the collecting track α

In the following, we set up the problem and analyze its complexity in Sections 2 and 3, respectively. In Section 4, we provide the performance guarantee for several heuristic solution approaches, motivated by the planning routines observed in practice. We proceed with extensive computational experiments and managerial insights in Section 5. We conclude with final remarks and an outlook in Section 6.

2 The railcar retrieval problem

We can summarize the *railcar retrieval problem (RRT)* as follows. Given railcars of different types (marked with different colors in the example in Figure 1) parked on storage tracks, we have to retrieve the specified number of railcars of certain types in a way such that the retrieval cost is minimized.

In the following (Section 2.1), we describe the practical motivation of the RRT first in order to understand the underlying assumptions. Afterwards, in Section 2.2, we state the RRT formally.

2.1 Practical motivation for the RRT

The length and the number of storage tracks highly depend on the size of the allotment. Often these are about 800–900 meter long tracks. Although some tracks are accessible from both sides, only one of them, which we call the *head* of the track (cf. Lübbecke and Zimmermann [17]), is used for the retrieval of railcars.

The workshop order specifies how many railcars of certain railcar types are requested. As a rule for each type t, there are many more railcars stored on the tracks than requested in the order. Indeed, although dozens of railcar categories exist (such as refrigerator boxcars, flat logging railcars, and intermodal double pocket railcars), about 90% of railcars belong to only a handful (about five) categories. Moreover, the railcars in each category often require the same kind of re-

pair or maintenance. The most widespread repairs include the wheels of all kinds of railcars, the carriage bodies of gondolas transporting scrap that often become deformed, the valves of tank railcars, etc.

The railcars are retrieved by a shunting engine and assembled on a special track α into a train (see Figure 1). Let us look at a retrieval operation that involves moving railcars 3 (type 2) and 4 (type 1) to track α (the movements of the engine are depicted with gray dotted arrows in Figure 1). First, the engine picks up railcars 1, 2, 3 and 4 and moves them to the intermediate position β. After a member of the shunting team has switched the track, the engine transports the railcars backwards to the collecting track α. Then, railcars 3 and 4 are decoupled, and the engine returns to position β with the rest of the railcars. Afterwards, the switches are repositioned, the engine pushes railcars 1 and 2 to their track, railcars 1 and 2 are coupled to the rest of the railcars, and if needed, they are decoupled from the shunting engine. Now, the shunting engine is ready for the next retrieval operation. This kind of yard in which no hill or hump assists the shunting process is called a *flat yard*. As one can see, shunting in a flat yard is a very time consuming process because of the many manual steps (switching, sometimes coupling and decoupling of the railcars), the stop-and-go operation of the engine, the low speed limits and the long distances for shunting unit workers. It takes several hours to assemble a train of 20–40 railcars, which is the typical size of a workshop order.

If we retrieve several consecutive railcars together as a *block*, we incur about the same cost as for the retrieval of a single railcar. In our example, the same engine moves, the same switching and similar further supporting procedures will be incurred if we would retrieve railcar 3 alone versus railcar 3 together with railcar 4. Moreover, the operation of the critical resource, the engine, is usually the bottleneck and determines the duration of the retrieval.

The retrieval cost is lower if we can retrieve a block of railcars whose first railcar is positioned at the head of the track. Indeed, we can attach this block to the outgoing train in the last step, so that the shunting engine does not need to visit collecting track α.

The sequence of the retrieved railcars can be arbitrary because they are classified upon arrival at the maintenance center. The railcars are assigned to free workshop lanes or to on-site storage tracks for short waiting times.

Our aim is to retrieve as few blocks of consecutive railcars as possible. Specifically, we aim at minimizing the total time of retrieval operations. Another possible objective is to reduce the operation costs of the critical resource, which is the shunting engine.

2.2 Formal description of the RRT

The RRT can be described as follows.

Instance: Given the following data

- a set $\{1, 2, \ldots, K\}$ (of railcars),
- a set $\{0, \ldots, T\}$, $T < K$ (of railcar types),
- a surjective mapping $u : \{1, 2, \ldots, K\} \to \{0, 1, \ldots, T\}$ (defining the railcars' types),
- a vector $(n_1, ..., n_T) \in \mathbb{N}^T$ with $N := \sum_{t=1}^{T} n_t < K$ (demand vector, or workshop's order),
- a subset $K' \subseteq \{1, 2, \ldots, K\}$ with $1 \in K'$ (of first railcars on each track),
- and cost parameters $0 \leq z_0 \leq z_1$.

Question: Determine $X \subseteq \{1, 2, \ldots, K\}$ with $|\{x \in X | u(x) = t\}| = n_t$, $\forall t \in \{1, \ldots, T\}$ and $\{x \in X | u(x) = 0\} = \emptyset$ such that the following objective is minimized:

$$z_0 \cdot |\{x \in X | x \in K'\}| + z_1 \cdot |\{x \in X | x \notin K', x - 1 \notin X\}|$$

Here, the K railcars are positioned on parallel storage tracks such that the railcar numbers are increasing on each track. The first railcar on each track is defined by set K'. For each railcar, its type is defined by the function u. We may assume w.l.o.g. that all railcar types that are not demanded are summarized as dummy railcar type zero, and thus, all other railcar types have a demand of at least one. Moreover, we assume w.l.o.g. that there is at least one railcar of type 0, which could – if it did not exist – be added to the ends of the tracks without influencing the solutions. Therefore, we assume u to be surjective.

In the RRT, we find a subset X of the set of railcars that denotes the railcars to be sent to the workshop. In X, the number of railcars of each type has to match the demand exactly. We call a set X with this property a *feasible solution*. It can easily be determined whether a feasible solution exists by verifying for each railcar type whether a sufficient number of railcars is available. The objective is then determined by the number of blocks being removed from the tracks with blocks at the head of the track being weighted less (z_0) than the other blocks (z_1). To calculate the number of blocks, we only count the railcar in each block with the lowest index. Thus, each railcar that is in X and at the head of a track defines one block at the head of the track. All other blocks are defined by $x \in X$ with x

not being at the head of the track and with the previous railcar $x-1$ not being in X.

The RRT can be modeled as mixed-integer program using the following parameters and variables. Binary parameters f_k specify whether railcar k, $k \in \{1, ..., K\}$ is the first on its track ($f_k = 1$) or not ($f_k = 0$). In a similar way, binary parameters u_{kt} provide information on the type of railcar k:

$$u_{kt} = \begin{cases} 1 & \text{if railcar } k \text{ is of type } t \\ 0 & \text{otherwise} \end{cases}.$$

The binary variables x_k show whether railcar k will be retrieved ($x_k = 1$) or not ($x_k = 0$). We also introduce nonnegative decision variables $y_k \geq 0$ to trace the incurred retrieval costs associated with railcar k.

We formulate a mixed-integer program as follows:

$$\text{minimize} \quad RRT(x, y) = \sum_{k=1}^{K} y_k \tag{1}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} x_k \cdot u_{kt} = n_t \qquad \forall t \in \{0, \ldots, T\} \tag{2}$$

$$y_k \geq z_0 \cdot f_k \cdot x_k + z_1 \cdot (1 - f_k) \cdot (x_k - x_{k-1}) \qquad \forall k \in \{2, \ldots, K\} \tag{3}$$

$$y_1 \geq z_0 \cdot x_1 \tag{4}$$

$$x_k \in \{0, 1\} \qquad \forall k \in \{1, \ldots, K\} \tag{5}$$

$$y_k \geq 0 \qquad \forall k \in \{1, \ldots, K\} \tag{6}$$

The objective function (1) is to minimize the total retrieval cost. (2) ensures that the demand for each railcar type is satisfied. Note that we also enforce this equality for the dummy type $t = 0$ to prohibit the retrieval of superfluous railcars. Constraints (3) and (4) define the auxiliary variables y_k. Because a block of consecutive railcars can be retrieved together, only the cost of the first railcar of the block is taken into account. Finally, we define the domains of the decision variables in (5) and (6).

Consider the example in Figure 1, and let the demand vector be $(3, 2, 1)$, i.e., three railcars of type 1, two railcars of type 2 and one railcar of type 3 have to be retrieved. An optimal solution is to pull railcars 8 to 10 and 12 to 14 as two blocks, respectively. This leads to the cost of $2z_1$.

3 Complexity analysis of the RRT

We will show that, in the general case, the RRT is NP-hard in the strong sense by a reduction from exact cover by 3-sets in Section 3.1. This reduction holds true even if all railcars are located on one storage track if the demand for each railcar type is one, and if z_0 equals zero. In Section 3.2, we feature some polynomially solvable special cases, some of which are very relevant for practice.

3.1 Computational complexity of the RRT

We will show that an instance of the problem "exact cover by 3-sets" (X3C in Garey and Johnson [8]) can be transfered in polynomial time to an appropriate decision problem of the RRT in such way that each YES-instance and only YES-instances of X3C are mapped to YES-instances of the particular RRT decision problem. Problem X3C is known to be NP-complete in the strong sense (Karp [14]) and can be stated as follows.

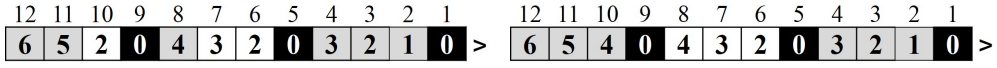
Instance: Let A be a set with $|A| = 3q$ and let $\mathcal{P}(A)$ denote the power set of A. Let $B \subseteq \mathcal{P}(A)$ be a set of subsets of A such that each element of B is a set with exactly three elements.

Question: Is there a $B' \subseteq B$ such that every element of A appears in exactly one element of B'?

Property 1. *The RRT problem is NP-hard in the strong sense.*

Proof. Given an instance of X3C, we show that this instance can be transformed to an instance of the RRT, the optimal objective function score of which is qz_1 if and only if the original instance is a YES-instance. The instance of the RRT has $K = 4|B|$ railcars, which are all stored on one track. There are $3q + 1$ different railcar types with railcar type 0 representing the dummy type and all other railcar types represent one element of set A. Except for railcars of type 0, which are not demanded, one unit is required from each type, i.e. $n_0 = 0$ and $n_1 = \ldots = n_{3q+1} = 1$. The railcars are arranged on the storage track as follows. Railcars 1, 5, 9,..., $4|B| - 3$ are of type 0. Railcars 2, 3, and 4 are of the types representing the elements of the first set of B (the exact sequence of these three railcars can be arbitrarily chosen). The next three railcars 6, 7, and 8 are of the types representing the elements of the second set of B, and so on.

Note that a lower bound for the objective function score for such an instance of the RRT is given by qz_1: At most three railcars can be retrieved in a block, as each larger block would include a dummy type, which is not feasible. As $N = 3q$ and the very first railcar is a dummy railcar, this leads to the lower bound of qz_1.



(a) ...NO-instance of X3C (b) ...YES-instance of X3C

Figure 2: Illustration: the RRT instance constructed from a...

Assume that we are given a YES-instance of X3C. Then, the particular instance of the RRT has an optimal objective function score of qz_1. In this case, B' has q elements, and each element of A appears once in B'. If the particular railcars of B' are retrieved in the RRT, i.e. the three railcars representing the particular types of each set of B' are retrieved, then this instance has an objective function value of qz_1, and because each type is retrieved exactly once, we have a feasible solution.

Now, assume that the instance of the RRT has an optimal objective function score of qz_1. In this case, each block being retrieved must be of length three. As three railcars, which are not of type 0, are always surrounded by railcars of type 0, each block corresponds to one element of set B. As each railcar type only appears once in all blocks, the set of blocks correspond to a subset of B that denotes a feasible subset B', and thus, the original instance of X3C is a YES-instance. □

The transformation of the proof is illustrated by the following example. Let $q = 2$, then $|A| = 6$ and let $A = \{1, 2, 3, 4, 5, 6\}$. At first we suppose that B has three members: $b_1 = \{1, 2, 3\}$, $b_2 = \{2, 3, 4\}$ and $b_3 = \{2, 5, 6\}$. This instance of X3C transforms to the shunting problem as shown in Figure 2a. This is a NO-instance for the X3C problem. There is a solution to the shunting problem, e.g. by pulling railcars 2, 3, 4, 8, 11 and 12 (shown in light gray). However, this solution has costs of $3z_1 > 2z_1$ and accordingly also leads to a NO-instance.

Now assume that $b_3 = \{4, 5, 6\}$. The transformation is now shown in Figure 2b. X3C now has a YES-instance with $B' = \{b_1, b_3\}$ and also the shunting problem has a solution with pulling railcars 2, 3, 4, 10, 11 and 12, leading to costs of $2z_1$, which is also a YES-instance.

3.2 Polynomially solvable variants of the RRT

First, let us mention that if the demand for each railcar type corresponds to its supply, then selection decisions of the single railcars are not relevant and the decision variables $(x_1, \ldots, x_K)$ in problem (1) to (6) can easily be determined. Thus, the RRT becomes trivial and can be solved in $O(K)$.

The other special case appears when the railcars in the yard are pre-sorted by their type. With this, we mean that if two railcars with index i and j, $i < j$ are of the same type $t \neq 0$ ($u(i) = u(j) \neq 0$), then railcars $i + 1, \ldots, j - 1$ must

be of the same type as well ($u(i) = u(i+1) = \ldots = u(j-1) = u(j)$). This case is especially interesting for answering the question whether pre-sorting is beneficial, a procedure which is frequently applied at storage tracks. Note that the dummy type may be located at various places in between groups of railcars of the same type. This arises from the fact that the dummy type actually represents various railcar types with zero demand. Without loss of generality, we will assume that with increasing railcar index, the railcar type (except for the dummy type) is increasing as well. In Section 3.2.1, we will show that this problem can be transformed to a maximum weighted matching problem in a bipartite graph.

We examine another practically important special case in Section 3.2.2. Notice that the length of the input for an instance of the RRT is (linearly) bounded by K, as the order volume N is bounded by K, and the number of different order types T is bounded by N. From a practical point of view, K, which is the number of railcars available for maintenance, can hardly be limited. However, due to the limited capacity of the workshops, the order size commonly does not vary a lot. Thus, the parameterized problem with a fixed N (and thus a fixed T) could be of high relevance in practice. Indeed, we are able to show that for a fixed N, the RRT is polynomial solvable in the input size.

3.2.1 The RRT with railcars sorted by types

Lemma 1. *The RRT with presorted railcars, i.e. for all* $1 \leq i < j \leq K$ *with* $u(i) = u(j) \neq 0$, $u(i) = u(i+1) = \ldots = u(j-1) = u(j)$ *must hold, is solvable in* $O(K^3)$.

Proof. Without loss of generality, we assume that no two railcar types exist that are next to each other on the tracks and for both of which supply equals demand. If two such types exist, they could be modeled as a single railcar type. We will denote a railcar type *breaking* if its railcars are located on two storage tracks on which other railcars are located as well (note that if its railcars are located on more than two storage tracks or if the railcars of this type are the only railcars on one of the tracks, this case can be reduced to the one at hand using a case-by-case analysis). Let $a(t)$ denote the number of railcars of a breaking type that are available at the end of the storage track, and let $b(t)$ be the number of railcars at the beginning of the next track. Let $\mathcal{B}$ be the number of railcar types that are breaking.

We now analyze the costs of retrieving the railcars of each type, considering the fact that only the very first railcar of a block induces costs. We start our deliberations from the "worst-case solution" in which each non-breaking railcar type and, whenever possible, each breaking railcar type is pulled in a single block and the remaining breaking railcar types are pulled in two blocks so that each

block induces costs of z_1 or, whenever possible, of z_0. Based on this solution, we estimate all possible cost savings ζ that can be achieved if two blocks are combined.

We cluster the railcar types $t = 1, \ldots, T$ into ten categories, and for each category, we define whether the particular type has one of the following properties.

leftopen(ζ) The railcars of type t can be withdrawn together with railcars of type $t+1$ (if $t+1$ is rightopen), reducing the objective by at most ζ.

rightopen(ζ) The railcars of type t can be withdrawn together with railcars of type $t-1$ (if $t-1$ is leftopen), reducing the objective by at most ζ.

I. For type t, the demand equals the supply, and it does not start at the head of the track. In this case, the railcars of type t to be retrieved are predetermined. If they are retrieved together with railcars of type $t-1$, no additional pulling costs emerge. Still, they allow for being pulled together with railcars of type $t+1$, reducing their retrieving costs. Thus, railcar types of category I are *leftopen(z_1)* and *rightopen(z_1)*.

II. For type t, the demand equals the supply, and it starts at the head of the track. Again, railcars of type t to be retrieved are predetermined. Pulling costs are z_0. The railcars can possibly be retrieved together with railcars of type $t+1$. Railcar types of category II are *leftopen(z_1)*.

III. For type t, the demand is lower than the supply, it is not breaking, and it does not start at the head of the track. In this case, retrieving either the first n_t railcars or the last n_t railcars is the dominant strategy. In the first case, building a block together with railcars of type $t-1$ could be possible; in the latter case, the blocks would be built with $t+1$. In the first case, there are no retrieval costs; in the second case, the costs are z_1, but the costs of $t+1$ may be reduced. Railcars of type III are *leftopen(z_1)* exclusive or *rightopen(z_1)*.

IV. For type t, the demand is lower than the supply, it is not breaking, and it starts at the head of the track. Again, it is a dominant strategy to either retrieve the first n_t railcars or the last n_t railcars. The retrieving costs are either z_0 or z_1. Railcars of type IV are *leftopen(z_0)*.

V. For type t, the demand is lower than the supply, it is breaking, and $n_t > a(t)$, $n_t > b(t)$. These railcars can only be retrieved in two blocks, one of which costs only z_0. Here, the dominant strategy is pulling all $b(t)$ railcars at the start of one track and taking the remaining railcars with the lowest railcar index. Similar to type I, railcars can possibly be retrieved together with railcars of type $t-1$, type $t+1$, or both types, which would lead to reductions

in the costs of z_1, z_1 or $z_1 + z_1$, respectively, compared to the worst case. Railcars of type V are *leftopen(z_1)* and *rightopen(z_1)*.

VI. For type t, the demand is lower than the supply, it is breaking, and $n_t > a(t)$, $n_t = b(t)$. These railcars can only be retrieved in one block if they are taken from the $b(t)$ on a storage track. Therefore, retrieving no railcars from the end of the other storage track is the dominant strategy. The pulling costs are z_0. Similar to type II, the railcars can possibly be retrieved together with railcars of type $t + 1$. Railcar types of category VI are *leftopen(z_1)*.

VII. For type t, the demand is lower than the supply, it is breaking, and $n_t > a(t)$, $n_t < b(t)$. These railcars can only be retrieved in one block if they are taken from the $b(t)$ on a storage track. Therefore, retrieving no railcars from the end of the other storage track is the dominant strategy. Similar to type IV, retrieving either the first n_t railcars or the last n_t railcars is the dominant strategy. The retrieving costs are either z_0 or z_1. Railcars of type VII are *leftopen(z_0)*.

VIII. For type t, the demand is lower than the supply, it is breaking, and $n_t \leq a(t)$, $n_t > b(t)$. Some railcars certainly must be retrieved from the $a(t)$ last railcars on one track. The costs are z_1, or if they are retrieved together with railcars of type $t - 1$, there are no costs. However, it might even be beneficial to retrieve railcars of type t from two tracks because retrieving the $b(t)$ railcars at the head of the track (inducing z_0 further retrieving costs) could reduce the retrieving costs of type $t + 1$ by z_1. Thus, railcars of type VIII are *leftopen($z_1 - z_0$)* and *rightopen(z_1)*.

IX. For type t, the demand is lower than the supply, it is breaking, and $n_t \leq a(t)$, $n_t = b(t)$. Railcars could exclusively be retrieved from the head of the track, inducing costs of z_0, and still allowing for railcars of type $t + 1$ to be retrieved together. However, if railcars of type t could be retrieved together with railcars of type $t - 1$, no retrieving costs would emerge. Railcars of type IX are *leftopen(z_1)* exclusive or *rightopen(z_0)*.

X. For type t, the demand is lower than the supply, it is breaking, and $n_t \leq a(t)$, $n_t < b(t)$. If the n_t railcars were retrieved from the head of the track, costs of z_0 would emerge. However, the railcars could be retrieved together either with railcars of type $t - 1$ or of type $t + 1$ so that no retrieving costs emerge or the retrieving costs of the $t + 1$ are reduced. Railcars of type X are *leftopen(z_0)* exclusive or *rightopen(z_0)*.

We now have to guarantee that the logical operators *and* and *exclusive or* are considered when trying to retrieve railcars of different types together. We will ap-

proach this via a simple maximum weighted matching problem in which vertices represent railcar types and each edge of the matching means that the respective railcar types are retrieved together.

The construction of the graph is as follows:

1. For each railcar type that is only leftopen (categories II, IV, VI, VII), one vertex is generated, which is considered to be leftopen.

2. For each railcar type that is leftopen exclusive or rightopen (categories III, IX, X), one vertex is generated, which is considered to be both leftopen and rightopen.

3. For each railcar type that is leftopen and rightopen (categories I, V, VIII), two vertices are generated. One vertex is considered to be leftopen, and one is considered to be rightopen.

4. Two vertices i and j are connected by an edge (i, j) if and only if the following properties hold true.

   - i represents type $t+1$ and j represents type t for some $t \in \{1, \ldots, T-1\}$,
   - no railcars of type 0 are stored between types t and $t+1$, and
   - i is rightopen and j is leftopen.

5. The edge weight of an edge (i, j), with i being rightopen(ζ) and j being leftopen(ζ') is $\min\{\zeta, \zeta'\}$.

This describes the graph on which we are to find a maximum weighted matching. The solution can be re-transformed to the shunting problem. Every edge that is part of the maximum weighted matching implies that the corresponding types of the incident vertices are pulled together. Isolated vertices represent types that can be pulled in an arbitrary single block.

The number of vertices in this graph is limited by $2T \leq 2K$, and the number of edges is linearly limited by the number of vertices. The maximum weighted matching problem can therefore be solved in $O(K^3)$ using the Hungarian Method. □

Consider an example in Figure 3. Let the demand vector be $(1, 2, 2, 1)$, i.e. $N = 6$ railcars of $T = 4$ types are requested. Categories of types 1, 2, 3 and 4 are IV, I, X and III, respectively. Therefore, for example, type 2 is depicted with two nodes, one of which is leftopen(z_1) and another one is rightopen(z_1). We connect the rightopen node of type 2 with the node of the neighboring type 1, which is

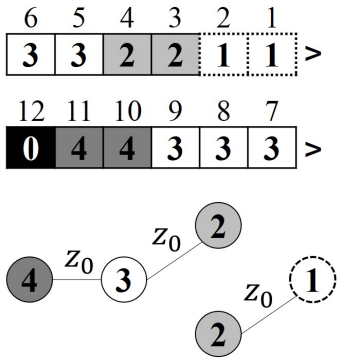


Figure 3: Illustration of the transformation to a maximum weighted matching problem

leftopen(z_0), and label the edge with the cost of $\min\{z_0, z_1\} = z_0$. An optimal solution of the resulting maximum weighted matching problem is to withdraw railcars of types 1 and 2 and of 2 and 3 together, which leads to savings of $(z_0 + z_0)$. Therefore, an optimal solution of the RRT-instance is $X^* = \{-2, 3, 4, 5, 6-10-\}$. Because the cost in the "worst-case solution" is $(z_0 + z_1 + z_0 + z_1)$, as explained above, the optimal objective value equals to $(z_0 + z_1 + z_0 + z_1) - (z_0 + z_0) = 2 \cdot z_1$, which is the worst-case objective value reduced by the savings.

3.2.2 The RRT with a fixed order size N

Property 2. *Let N be some (fixed) positive integer value. The RRT can then be solved in $O(K)$, i.e. the RRT is fixed parameter tractable in N.*

Proof. Notice that the retrieval costs can never exceed $N \cdot z_1$. Because N is a constant, we can solve the RRT optimization problem in polynomial time if we have a polynomial time algorithm for solving the decision problem "Is there a solution to the RRT with objective function equal to φ?" for some threshold $1 \leq \varphi \leq N \cdot z_1$. We will show this under the assumption (with losing generality) that the first railcar on each track is of the dummy type. However, this is only for the sake of clarity, and the idea of the proof can easily be extended to the general case. Thus, we may assume that φ is some multiple of z_1, i.e. $\tau := \varphi / z_1$ is an integer, and we ask for a solution in which the railcars are retrieved in τ blocks.

Given the set of all YES-instances for all $1 \leq \tau \leq N$. A set of τ blocks, i.e. τ sequences of railcar types, is called a feasible solution for a YES-instance, if the set of blocks exactly satisfies the order. Note that by this definition, we only care about the railcar types and their sequence within a block, but we disregard from what part of the available railcars this block is retrieved. Thus, we consider the set that contains all solutions that are feasible solutions for at least one YES-instance.

We will search for a bound on the cardinal number of this set. There can be at most N railcar types and the order is bounded by N. The N railcar types (worst case) can be split into blocks in at most B_N different ways, with B_N representing the $N-$th Bell number. For a constant input N, the $N-$th Bell number is a constant as well. Because the Bell number ignores the sequence of the railcar types within the blocks, at most $B_N \cdot N!$ possibilities exist. Thus, there is a constant number (in K) of feasible solutions for all YES-instances.

Therefore, we still need to determine whether a feasible solution (i.e. the arrangement of the blocks but not the positions from where they are retrieved) that can be applied to a given arrangement of K railcars on one or more storage tracks can be found in polynomial time. This can be done in $O(K)$, although it has to be mentioned that it is not sufficient if we simply search for the blocks. The railcar type sequence of a block may also appear within some other block or the sequences may overlap. However, there is a constant number of blocks, and thus, conflicts of block pairs, block triples, etc. can still be resolved via a common examination. □

Even though formally, the RRT is fixed parameter tractable, we hardly dare to say that it is solvable in practice even for a fixed N. Even with high-performance computers, the procedure described in the proof cannot be computed in a reasonable time unless the values of N are very small. From our experience, we suggest that for values greater than 10 (not to mention for realistic values of N), it is not applicable.

4 Heuristic solution approaches

Instead of using optimization software, planners often rely on experience-based heuristics. The retrieval procedure without any support may resemble the naive heuristic (Section 4.1). The largest block heuristic (Section 4.2) has been worked-out at our discussion partner, a major rail company and revealed a significant improvement potential with respect to the status quo in preliminary studies. We introduce the *weighted* largest block heuristic as an alternative to the largest block heuristic in Section 4.3. In the current section, we assume that $z_1, z_0 > 0$ and $N \geq 2$.

4.1 The naive heuristic

In the *naive heuristic (Naive)*, planners sequentially examine the railcars stored on the tracks. If the railcar type is in the order and its demand has not yet been satisfied, then this railcar is selected for retrieval. In the example in Figure 1 and

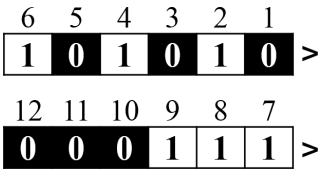


Figure 4: Illustration: For demand vector (3), the Naive solution is $N \cdot \frac{z_1}{z_0}$ times more expensive than the optimum

the workshop's order of $(3, 2, 1)$, the Naive heuristic selects railcars with numbers 2, 3, 4, 6, 8 and 12. These railcars are pulled out in four blocks $\{-2, 3, 4-6-8-12-\}$.

The computational complexity of the Naive heuristic is $O(K)$.

However, in the worst case, the solution found by the Naive heuristic is $N \cdot \frac{z_1}{z_0}$ times larger than the optimal objective function value. Indeed, for the example in Figure 4 and the order of (3) with $T = 1$ and $N = 3$, the Naive heuristic pulls out three blocks $\{-2-4-6-\}$ at a cost of $3 \cdot z_1$, whereas the optimal solution consists of just one block of railcars $\{-7, 8, 9-\}$ and has a cost of z_0. We can construct examples for such poor performance of the Naive heuristic for any $N \geq 2$ in a similar way. Obviously, the ratio of the objective value found by the Naive heuristic to the optimum cannot be larger than $N \cdot \frac{z_1}{z_0}$. Let I be some RRT instance, $Opt(I)$ be its optimal objective function value and Naive(I) be the objective value received by application of the Naive heuristic. We formulate the following absolute performance ratio, which is the tightest upper bound on the $\frac{\text{Naive}(I)}{Opt(I)}$:

Property 3. *The absolute performance ratio for the Naive is* $N \cdot \frac{z_1}{z_0}$.

4.2 The largest block heuristic

In the *largest block heuristic (LBH)*, the planner sequentially examines the railcars, selects the largest possible block of the ordered railcars and updates the quantities of still required railcar types in each iteration. The planner repeats iterations until all of the ordered railcars are retrieved. In the example in Figure 1 and for order $(3, 2, 1)$, the LBH will select railcars $\{16, 17, 18, 19\}$ forming a block of four in the first iteration. In the next iteration, the LBH will look for one railcar of type 1 and for one railcar of type 3. They are available only as single-car blocks. Thus, the LBH solution contains three blocks, for example, $\{-16, 17, 18, 19-2-6-\}$. The computational complexity of the LBH is $O(K \cdot N)$ because we repeat at most $O(N)$ iterations and examine up to $O(K)$ railcars in each iteration.

The LBH is intuitively very appealing. Indeed, if it is possible to retrieve

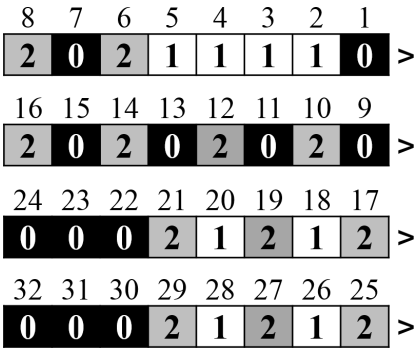


Figure 5: Illustration: For demand vector $(4, 6)$, the LBH solution is $\left(\frac{N}{4} + \frac{1}{2}\right) \cdot \frac{z_1}{z_0}$ times more expensive than the optimum

all the ordered railcars as a single block, the LBH will find an optimal solution. Moreover, if only one type $T = 1$ of railcar is ordered, the LBH will find a best possible solution as well. However, the tight performance guarantee for the LBH, as we show below, is $\left(\frac{N}{4} + \frac{1}{2}\right) \cdot \frac{z_1}{z_0}$, which means that even in case $z_1 = z_0$ for an order of 20-40 railcars, which is a common size of an order in practice, the LBH may construct solutions five to ten times more expensive than the optimum.

Property 4. *There is an approximation ratio for the LBH of* $\left(\frac{N}{4} + \frac{1}{2}\right) \cdot \frac{z_1}{z_0}$.

Proof. Assume that the optimum solution of some instance I requires $b \geq 1$ blocks. Then, the optimal objective function score is $Opt(I) \geq b \cdot z_0$. This value can indeed be reached if the blocks can be withdrawn from the head of the tracks. In an optimal solution, the *minimal* size of the *largest* block is $\frac{N}{b}$. Hence, the LBH selects a block with at least $\frac{N}{b}$ railcars during the first iteration. In the worst case, the LBH can select only blocks of size 1 at each of the following iterations. Thus, the solution of the LBH is not worse than $LBH(I) \leq \left(1 + \left(N - \frac{N}{b}\right)\right) \cdot z_1$. The upper bound for the ratio amounts to $\frac{LBH(I)}{Opt(I)} \leq \frac{N \cdot b - N + b}{b^2} \cdot \frac{z_1}{z_0}$. By taking taking the first and the second derivatives in $b > 0$, we get that the right side achieves its maximum if b equals a fractional number $\frac{2 \cdot N}{N+1}$. Because b takes only integer values, we get $\frac{LBH(I)}{Opt(I)} \leq \left(\frac{N}{4} + \frac{1}{2}\right) \cdot \frac{z_1}{z_0}$. □

In the following example, we illustrate that $\frac{LBH(I)}{Opt(I)} = \left(\frac{N}{4} + \frac{1}{2}\right) \cdot \frac{z_1}{z_0}$ holds for some instances. Let $N = 10$, $T = 2$ (two types), the demand vector be $(4, 6)$, and the positioning of railcars at tracks be as shown in Figure 5. If LBH breaks ties by selecting a block with the lowest number of its first railcar, then the LBH constructs a solution with 6 blocks $\{-2, 3, ..., 6 - 8 - 10 - 12 - 14 - 16 -\}$. In the optimal solution, two bottom blocks have to be retrieved. Thus, $\frac{LBH(I)}{Opt(I)} = 3 \cdot \frac{z_1}{z_0} = \left(\frac{N}{4} + \frac{1}{2}\right) \cdot \frac{z_1}{z_0}$.

A similar example may be constructed with any number of types $T \leq N$. In addition, we can construct an example with poor performance by the LBH for arbitrary large N with $(N \bmod 4 = 2)$. For example, we set an optimal solution to consist of two blocks with $\frac{N}{2}$ railcars each and number the types in each optimal block intermittently as 2 and 1. We set the demand vector to $\left(\frac{N-2}{2}, \frac{N+2}{2}\right)$ and introduce an additional block located earlier at the track. This additional block consists of $\frac{N}{2}$ railcars, $\frac{N-2}{2}$ of them are of type 1, and the remaining railcar is of type two. In this case, the LBH constructs a solution with the worst possible performance consisting of $\frac{N+2}{2}$ railcar blocks. This example shows that there is no constant approximation guarantee for the LBH, which is independent of N.

4.3 The weighted largest block heuristic

The *weighted largest block heuristic (WLBH)* is very similar to the LBH. However, we also have to determine the *most critical* ordered type $t^* = \max_t \frac{n'_t}{K'_t}$, where $n'_t > 0$ is the number of still required railcars of type t and K'_t is the number of railcars of type t currently available on the tracks. In each iteration, we discard all the blocks that do not contain the currently most critical ordered railcar type. All the other algorithmic steps remain the same as in the LBH. In the example in Figure 1 and for demand vector $(3, 2, 1)$, the ratios $\frac{n'_t}{K'_t}$ in the first iteration are $\frac{3}{7}$, $\frac{2}{5}$ and $\frac{1}{2}$, so that $t^* = 3$ is the most critical type. We retrieve the largest block containing type three, which is $\{12, 13, 14\}$. In the second iteration, the only type that is still required is type one; therefore, $t^* = 1$, and we pull out $\{8, 9, 10\}$. For this instance, the WLBH computes an optimal solution with a cost of $2 \cdot z_1$.

The WLBH combines the strengths of the LBH with the following intuition. The higher the ratio $\frac{n'_t}{K'_t}$, *ceteris paribus*, the higher the probability of selecting a block that is part of an optimal solution. Indeed, if this ratio equals 1 for type t^*, then all the railcars of type t^* belong to an optimal solution.

The computational complexity of the WLBH is $O\left(K \cdot (N + T)\right)$. Because the number of ordered types cannot exceed the number of ordered railcars, the computational complexity of the WLBH is the same as that of the LBH. However, surprisingly, the WLBH has a worse performance guarantee than the LBH. For real-world instances with $20 \leq N \leq 40$ railcars, the WLBH may find seven to fourteen times more expensive solutions than the optimum.

Property 5. *There is an approximation ratio for the WLBH of* $\frac{N+1}{3} \cdot \frac{z_1}{z_0}$.

Proof. We notate the objective function value found by the WLBH for instance I as $WLBH(I)$. In the following, we provide the main idea of the proof that $\frac{WLBH(I)}{Opt(I)}$ cannot be greater than $\frac{N+1}{3} \cdot \frac{z_1}{z_0}$ for any instance I. The detailed proof can be found in the Appendix.

The ratio $\frac{WLBH(I)}{Opt(I)}$ cannot be greater than or equal to $\frac{N+1}{3} \cdot \frac{z_1}{z_0}$ for all instances with $Opt(I) \neq 2 \cdot z_0$, $Opt(I) \neq 2 \cdot z_1$, or $Opt(I) \neq z_0 + z_1$. Indeed, if an optimal solution consists only of one block, i.e. $Opt(I) = z_0$ or $Opt(I) = z_1$, then the WLBH will retrieve all the railcars in one block as well. Further, it is straightforward to see that the $WLBH(I)$ cannot be larger than $N \cdot z_1$. Hence, if an optimal solution consists of three or more blocks, i.e. $Opt(I) \geq 3 \cdot z_0$, $\frac{WLBH(I)}{Opt(I)}$ is necessarily lower than $\frac{N+1}{3} \cdot \frac{z_1}{z_0}$. Therefore, we have to examine only the cases where an optimal solution consists of two blocks to prove that $\frac{N+1}{3} \cdot \frac{z_1}{z_0}$ is indeed an approximation ratio for the WLBH.

The main idea of the proof is the following: Let X^* be some optimal solution for instance I, and run the WLBH on this instance. At each iteration, the WLBH withdraws a railcar of some type t^* that is currently critical. There is necessarily a railcar of this type t^* belonging to the not yet withdrawn railcars of X^*. The WLBH always retrieves a largest block of railcars containing the currently critical type. Thus, the size of the blocks of still requested railcars in the solution X^* prevents the WLBH from withdrawing a railcar block of a smaller size.

□

The performance ratio is tight, as can be seen in the example in Figure 6, which has the demand vector $(1, 2, 1, 2, 2, 2, 2, 2)$ of $N = 14$ railcars. The optimal solution with two blocks is $\{-52, 53, 54-69, 70, 71, \ldots, 79-\}$. Let the WLBH break the ties in selecting the critical type by the lowest number of the first railcar of this type on the track. The first iteration ratios for types 1 to 8 are $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{2}$, $\frac{1}{3}$, $\frac{2}{7}$, $\frac{2}{7}$, $\frac{2}{7}$ and $\frac{2}{7}$, respectively. The critical type is $t^* = 1$, and provided the WLBH breaks ties by taking the first encountered largest block containing type 1, we retrieve railcars $\{2, 3, 4\}$. In the second iteration, we pull out block $\{6, 7, 8\}$ with critical type $t^* = 3$. In the next eight iterations, we retrieve eight single-car blocks of types 5, 6, 7 and 8. Overall, for this instance I, we have $\frac{WLBH(I)}{Opt(I)} = \frac{10}{2} \cdot \frac{z_1}{z_0} = \frac{N+1}{3} \cdot \frac{z_1}{z_0}$.

We can construct such poorly-performing instances for any $N > 2$ and $(N \bmod 6) = 2$. We set the optimal solution to consist of two blocks with 3 and $N - 3$ railcars, respectively. We number the type of the second railcar in the first block as 1. We proceed by numbering the railcar types of the second block (with each type consisting of one railcar) in the following way. We first number the types of the fourth and eighth railcars in the block followed by the second railcar in the block. We proceed by assigning types intermittently to two railcars at the next available $4 \cdot l$-th positions, where $l > 0$ is some integer, then to a railcar at the next available even position. Afterwards, we number the rest of the railcars at the even positions in the block and conclude by numbering the railcars at the odd positions. Finally, we number the remaining two railcars of the three-railcar optimal block as types $N - 1$ and N. We set the number of requested types to $T = N$

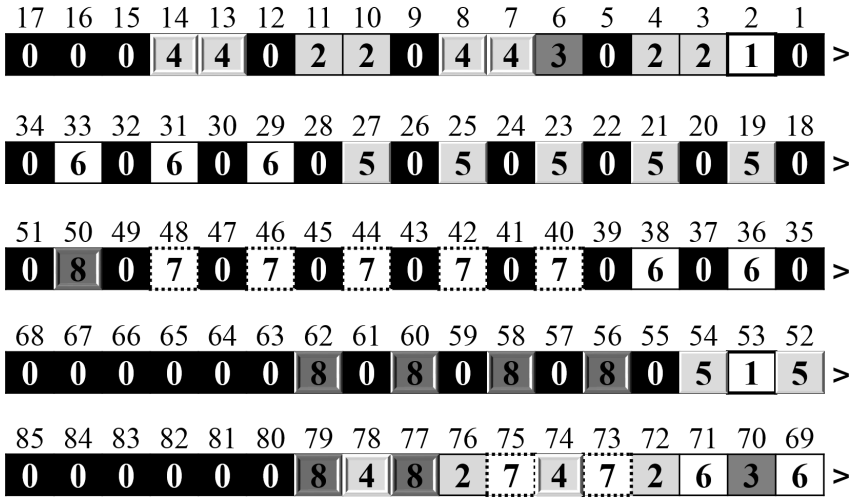


Figure 6: Illustration: For demand vector $(1, 2, 1, 2, 2, 2, 2, 2)$, the WLBH solution is $\frac{N+1}{3} \cdot \frac{z_1}{z_0}$ times more expensive than the optimum.

and the demand vector to $(\underbrace{1, 1 \ldots 1}_{N \text{ times}})$. Additionally, we introduce $\frac{N-2}{6}$ three-railcar blocks that are located earlier on the track than the optimal blocks. These additional blocks consist of railcars of types $(1, 2, 3)$, $(4, 5, 6)$ and so on. We also introduce enough single-car blocks to enforce the following priority order of railcar types in the first iteration of the WLBH: $1, 2, 3, \ldots, N$. The WLBH retrieves $\frac{N-2}{6}$ additional three-railcar blocks first and then proceeds with the remaining requested railcars as single-car blocks, which makes the number of railcar blocks retrieved $\frac{N-2}{6} + N - \frac{N-2}{2} = \frac{2 \cdot N+2}{3}$. If we place the optimal blocks at the heads of the tracks and ensure that the WLBH only withdraws blocks from the middle or the end of the track, then for the constructed instance I, the performance ratio equals $\frac{WLBH(I)}{Opt(I)} = \frac{N+1}{3} \cdot \frac{z_1}{z_0}$.

There are no dominance relations between the examined heuristics, and each of them may perform better than the others for some problem instances. Thus, in the example in Figure 1 for demand vector $(3, 2, 1)$, the WLBH finds a better solution than the LBH, and the LBH is better than the Naive heuristic. On the other hand, for Figure 6 with demand vector $(1, 2, 1, 2, 2, 2, 2, 2)$, the LBH is better than the WLBH. If we set railcars 21 and 29 to type 0, set the demand vector to (4,4) and renumber the railcars in Figure 5 from the bottom track to the upper track (so that railcar 25 becomes number 1, railcar 32 becomes number 8 and railcar 17 becomes railcar 9), the Naive heuristic would perform better than the LBH.

5 Managerial insights based on computational studies

As discussed above, planners often rely on experience-based heuristics to organize railcar retrieval. Therefore, it is important to evaluate the cost reduction potential of the exact optimization.

In this section, we perform detailed simulations to evaluate the performances of exact optimization and heuristic solution approaches as well as to work out further important managerial insights. We describe the data generation for the simulation in Section 5.1. Section 5.2 evaluates the average performances of the intuitive heuristics. In Section 5.3, we examine the impact of a widespread organizational routine of presorting on the retrieval costs. At several yards, a lot of time and resources are devoted to presorting railcars on the storage tracks according to their types. Section 5.4 evaluates how the yard layout impacts the railcar retrieval cost.

We perform our experiments on a Lenovo notebook with a 2×2.3 GHz Intel i5 CPU, 8 GB RAM and Windows 8.1 operating system. The heuristics were programmed with Java 1.8.0, while the RRT-model (1)-(6) is set in Python 3.5.1 and solved exactly with Gurobi 6.5.0. We refer to the performance of Gurobi 6.5.0 as the *MIP*, meaning that the Mixed-Integer Program was solved to optimality.

5.1 Simulation settings

There are different constellations of storage tracks. There are yards with several dozens of short tracks of about 200 meters in length, and there are yards with fewer, i.e., 20 to 30, long tracks of 800-900 meters in length. In our simulation, we emulate a yard with 25 long tracks. We additionally examine alternative yard layouts in Section 5.4.

The length of a freight railcar, the so-called length over buffers, ranges from 10.6 meter (e.g. for a short Gs model) to 27 meter (e.g. for double-stack railcars transporting automobiles). Most railcars are about 14 meters long; thus, a 850-meter track may host up to 60 railcars. However, railcars rarely fill all the available pitches on the storage tracks, and as a rule, about 50% of pitches are free. Therefore, in our simulation, we assume that each track contain 30 railcars and that $25 \cdot 30 = 750$ railcars are available on the storage tracks.

Typically, five types of railcar account for approximately 90% of railcars, and the most widespread type accounts for approximately one third of railcars. The overwhelming majority of types contain only one railcar each. We closely calibrate the data we observed in practice. We generate 50 types of railcar from the distribution as shown in Figure 7. The prevalence of each of the forty most seldom

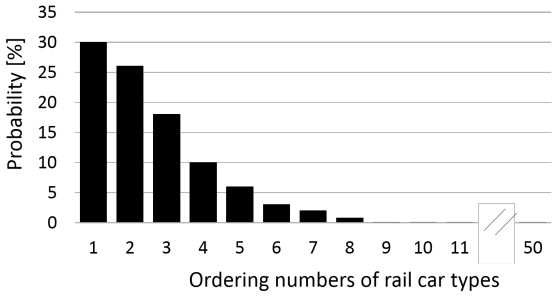


Figure 7: Probability density of types on the storage tracks used in the simulation

types is set to 0.1%. The distribution of the ten most widespread types follows a discretized truncated normal distribution with the highest prevalence set to 30% for type 1 (see Figure 7). The probability of a railcar being one of the frequent types (1, 2, 3, 4 or 5) is 90% in our distribution, as observed in practice. In our simulation, we randomly draw a type for each of the 750 railcars on the storage tracks from the distribution described in Figure 7. There are many more theoretically possible railcar types in practice, but this generation method quite accurately emulates the number of different types that are simultaneously present on storage tracks.

We refer to the scenario where each railcar may take any position on the storage tracks with an equal probability as *random*. When railcars arrive at the storage tracks, their trains typically consist of only a few types of railcar. Therefore, encountering conglomerates of railcars of the same type is very likely. We emulate this situation in our main scenario and designate it *default*. We re-arrange the generated railcars on the storage tracks so that each railcar has the same type as its follower with probability γ provided that railcars of this type are still available. We set $\gamma = 91\%$ so that, if there are enough railcars of a certain type, the average length of a homogeneous-type sequence equals 10 railcars.

In Sections 5.2, 5.3 and 5.4, we introduce eight additional scenarios to mimic different settings that are relevant to practice.

A typical order consists of 20 to 40 railcars. Therefore, we set the number of requested railcars to $N = 30$. The type of each railcar in the order is randomly generated according to the frequency of the types currently available on the storage tracks.

We set $z_0 = 1$ and $z_1 = 2$ and generate 100 RRT-instances for each scenario or $100 \cdot 10 = 1,000$ instances in total.

5.2 Performance of the heuristic solution approaches

Overall, the retrieval costs for the instances in the random scenario are different from those for the instances in the default scenario. For example, random in-

	scenario	Optimal solution Retrieval costs [cost units]	Number of retrieved blocks	Relative optimality gap [%] Naive	LBH	WLBH
average	random	5.03	3.67	372	64	51
	default	6.75	5.02	129	45	30
maximum	random	11.00	7.00	867	200	167
	default	12.00	8.00	400	200	200

Table 1: Comparative performance of the heuristics for default and random scenarios

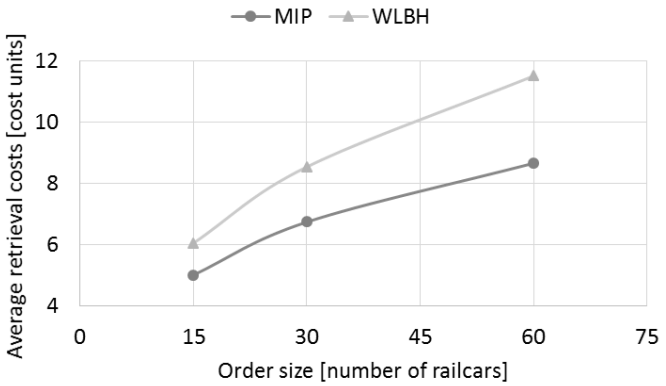


Figure 8: The average retrieval costs for the MIP and the WLBH for different order sizes in the default scenario

stances tend to contain longer blocks of the requested railcars than the default instances. Therefore, an optimal solution in the random scenario consists of 3.67 railcar blocks and that in the default scenario consists of 5 blocks on average (see Table 1).

Table 1 clearly shows that all of the heuristics lead to highly inflated railcar retrieval costs. The average relative optimality gaps in the default scenario are 129%, 45% and 30% for the Naive heuristic, the LBH and the WLBH, respectively. Although the WLBH outperforms the LBH and is significantly better than the Naive heuristic in our simulation, it still misses the optimality by 30-51% depending on the scenario.

Additionally, we solved the default scenario for order sizes of $N = 15$ and $N = 60$. Figure 8 shows the results for these instances provided by the MIP and the WLBH. One can see that the average cost for retrieving the railcars increases more strongly with increasing order size for the WLBH solution than for the optimal solutions.

Because the average computation times of all three heuristics and of the MIP

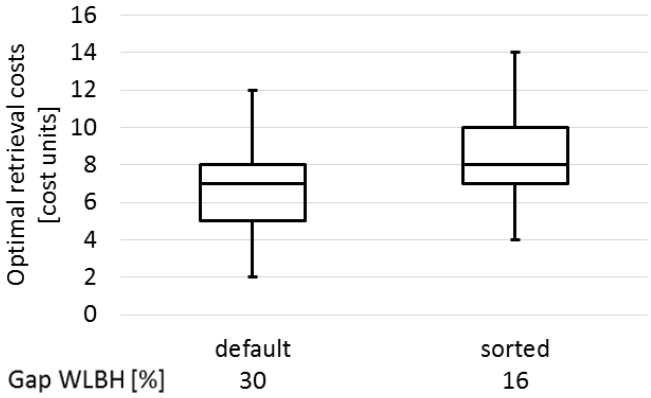


Figure 9: Distribution of the optimal retrieval costs for default and sorted scenarios

are well below one second, we strongly recommend applying the exact optimization in practice. Indeed, the maximum computation time of the MIP in our experiments amounted to 35 seconds for one of the instances. However, this result was a very unusual outlier.

5.3 Presorting railcars by type and the retrieval cost

At several yards, railcars are presorted according to type. This process is extremely expensive and time consuming. To avoid disturbing retrieval operations and the arrival of railcars, presorting is usually performed during night shifts. A major argument in favor of presorting is that it may decrease the variance in the time required to retrieve the ordered railcars. Indeed, the retrieval costs with this organizational policy can never exceed $T \cdot z_1$, and because usually only a few types are requested, i.e. $T < N$, the case of an extremely long retrieval cost of $N \cdot z_1$ becomes less likely. However, the expected retrieval time may increase. As follows from our analysis in Section 3.2, unless *all* the railcars of some type currently available on the storage tracks are requested, the retrieval costs can never fall below $\lceil \frac{T}{2} \rceil \cdot z_0$ in the case of presorting. Therefore, for informed managerial decisions, it is extremely important to visualize risk profiles with and without presorting.

We formulate an additional scenario denoted *sorted* in which all the railcars on the storage tracks are sorted according to type. Note that sorted in this context means that all railcars of one type are placed in sequence (or in two sequences if the remaining slots on the track are not sufficient to host all the railcars of this type).

Figure 9 compares the risk profiles for the generated default and sorted instances as a boxplot. The whiskers indicate the largest and smallest shunting costs that have been observed. The box contains the second and third quartile, and the

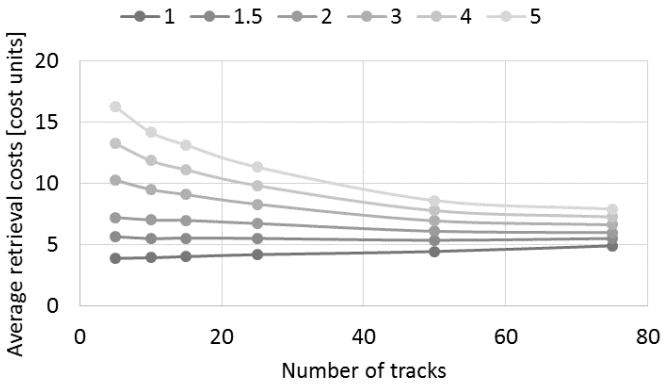


Figure 10: The relation between the average retrieval cost and the number of tracks for different cost ratios

median is indicated as a horizontal bar in the box. Our simulation shows that, in contrast to the common intuition, the sorted scenario does not only lead to a higher average retrieval cost but also does not reduce its variance.

However, the average performance of the WLBH improves in the sorted scenario. The relative optimality gap decreases from 30% to 16%. Indeed, the heuristics have only a small degree of freedom to deviate from the optimal objective function value for the sorted instances, especially with a large number of requested railcars N, because the probability of finding railcar blocks containing different types is low. Therefore, we find that the sorted scenario is more robust towards the implementation of different planning approaches, such as planning heuristics instead of exact optimization.

5.4 Different yard layouts and the retrieval cost

The number and lengths of the storage tracks have important consequences for the fixed (e.g. investment cost) and variable costs of the rail company.

We expect the retrieval cost to vary for different yard layouts even if we keep the number of railcars on the storage tracks and the distribution of the railcar types the same. There are two conflicting forces at play. On the one hand, the larger the number of storage tracks, the larger the average number of blocks necessary for the retrieval of railcars and the higher the retrieval cost. On the other hand, the larger the number of storage tracks, the higher the probability of "paying" z_0 to pull out a block of railcars and the lower the retrieval cost. Thus, for a given ratio $\frac{z_1}{z_0}$ and a given yard size K, there is an optimal number of storage tracks that minimizes the expected retrieval cost. In this section, we determine this number in a large-scale simulation study.

Based on the *default* scenario, we formulate six additional scenarios with dif-

ferent yard layouts. We keep the number of railcars on the storage tracks at $K = 750$ for each scenario. We set the number of tracks to 5, 10, 15, 30, 50 and 75, and the number of railcars per track to 150, 75, 50, 25, 15 and 10, respectively. Afterwards, we compute the average retrieval cost for $\frac{z_1}{z_0} = 1, 1.5, 2, 3, 4, 5$. Figure 10 shows the optimal number of tracks for each ratio $\frac{z_1}{z_0}$.

Unless the cost ratio $\frac{z_1}{z_0}$ is lower than 2, short tracks hosting about 10 railcars each should be preferred. The long tracks scenario is the option of choice at very low $\frac{z_1}{z_0}$. For example, at $\frac{z_1}{z_0} = 1.5$, the lowest average retrieval cost is achieved with rather short tracks hosting about 15 railcars each.

6 Conclusion

In this article, we formulate the problem of railcar retrieval from storage tracks (RRT) as a mixed-integer linear program. We show that the RRT is NP-hard in the strong sense but that some of its practice-relevant special cases are polynomially solvable. We perform thorough worst- and average-case analyses of the intuitive heuristic solution approaches that are either utilized in practice or motivated by the observed planning routines. Our analysis indicates a large cost-reduction potential for rail companies if they adopt exact optimization approaches instead of heuristic planning routines. Thus, even the best performing planning heuristics, the LBH and the WLBH, lead to 30%-65% higher retrieval costs on average and may lead to solutions with up to fourteen times higher costs than the optimal solutions in settings that are common in practice. Based on our simulations, we also conclude that the railcar retrieval from storage tracks results in a practically significant cost reduction potential and therefore should be considered in studies on the operations of maintenance centers.

Our simulations also show that, in contrast to the widespread opinion, railcar presorting results not only in a higher average retrieval cost but also does not reduce the variance of the retrieval cost or time. Moreover, for moderate ratios of $1.5 \geq \frac{z_1}{z_0}$, storage yard layouts consisting of multiple short tracks lead to lower average retrieval costs.

Future research must explicitly integrate the railcar retrieval from storage tracks into a wide range of planning problems on rail maintenance operations.

Acknowledgements. *The authors cooperated with Dr. Christian Otto at DB Schenker Rail and are very grateful to him for his valuable insights into the planning issues of railcar maintenance.*

References

[1] Blasum U, Bussieck MR, Hochstättler W, Moll C, Scheel HH, Winter T (1999) Scheduling trams in the morning. Mathematical Methods of Operations Research 49(1):137–148

[2] Boysen N, Fliedner M, Jaehn F, Pesch E (2012) Shunting yard operations: Theoretical aspects and applications. European Journal of Operational Research 220(1):1–14

[3] Budai G, Huisman D, Dekker R (2006) Scheduling preventive railway maintenance activities. Journal of the Operational Research Society 57:1035–1044

[4] Corsi T, Casavant K, Graciano T (2012) A preliminary investigation of private railcars in north america. Journal of the Transportation Research Forum 51(1):53–70

[5] Di Stefano G, Koči ML (2004) A Graph Theoretical Approach to the Shunting Problem. Electronic Notes in Theoretical Computer Science 92:16–33

[6] Doganay K, Bohlin M (2010) Maintenance plan optimization for a train fleet. In: Ning B, Brebbia CA (eds) Computers in Railways XII, WIT Press, Southampton, pp 349–358

[7] EU (2011) Impact assessment: Roadmap to a single european transport area – towards a competitive and resource efficient transport system. Tech. rep., EU Commission Staff Working Paper, Bruessels

[8] Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, San Francisco

[9] Gatto M, Maue J, Mihalák M, Widmayer P (2009) Shunting for Dummies: An Introductory Algorithmic Survey. In: Ahuja R, Möhring R, Zaroliagis C (eds) Robust and Online Large-Scale Optimization, Lecture Notes in Computer Science, Springer, Berlin, pp 310–331

[10] Hall RW (2000) Scheduling and facility design for transit railcar maintenance. Transportation Research Part A: Policy and Practice 34(2):67–84

[11] Hansmann RS, Zimmermann UT (2008) Optimal sorting of rolling stock at hump yards. In: Krebs HJ, Jäger W (eds) Mathematics - Key Technology for the Future, Springer Berlin / Heidelberg, pp 189–203

[12] Hecht M (2007) Schneller auf schienen: Beschleunigung des schienengüterverkehrs durch zeitgemäßge technik. TU International 59(Januar):24–25

[13] Jacobsen PM, Pisinger D (2011) Train shunting at a workshop area. Flexible Services and Manufacturing Journal 23(2):156–180

[14] Karp RM (1972) Reducibility Among Combinatorial Problems. In: Miller RW, Thatcher JW (eds) Complexity in Computer Computations, Plenum, New York, pp 85–103

[15] Kobbacy KAH, Murthy DN (eds) (2008) Complex System Maintenance. Springer: Berin

[16] Lidén T (2015) Railway infrastructure maintenance - a survey of planning problems and conducted research. Transportation Research Procedia 10:574–583

[17] Lübbecke ME, Zimmermann UT (2005) Shunting Minimal Rail Car Allocation. Computational Optimization and Applications 31(3):295–308

[18] Nicolin J, Nogly L (2013) Gütertransport mit güterwagen im wettbewerb - bei steigenden ansprüchen/beanspruchungen. In: IFS-Seminar at Aachen University, Downloaded from http://www.ifs.rwth-aachen.de/veranstaltungen/Seminar2013.html on the 9th of October 2015

[19] Ramond F, Dauzère-Pérès S, de Almeida D (2006) Scheduling moves within railcar maintenance centers. In: Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing

[20] Ramond F, de Almeida D, Dauzère-Pérès S (2006) Enhanced operation scheduling within railcar maintenance centers. In: Proceedings of the 7th World Congress on Railway Research

[21] SAG-Gruppe (2015) Anlagealternative eisenbahnwaggon: Transparent - rentabel - zukunftsfähig, downloaded from http://www.railinvest.de/ on the 9th of October 2015

[22] Winter T, Zimmermann UT (2000) Real-time dispatch of trams in storage yards. Annals of Operations Research 96(1–4):287–315

Appendix: Detailed proof of Property 5

Proof. As described in Section 4.3, it is sufficient to consider only the cases where an optimal solution consists of two blocks. In the following, we provide explanations only for the set of instances $\mathcal{I}'$ with $Opt(I') = 2 \cdot z_0, I' \in \mathcal{I}'$. The proofs for the cases of $Opt(I) = 2 \cdot z_1$ and $Opt(I) = z_0 + z_1$ are equivalent.

To find the largest possible estimate for $\frac{WLBH(I')}{Opt(I')}$ for instances $I' \in \mathcal{I}'$, we look for instances with the worst possible objective value of the WLBH, i.e. for which the WLBH withdraws railcars in the largest possible number of blocks, each with a cost of z_1. In the first step, we observe that we may restrict our search to a narrower class of instances $\mathcal{I}''$. In the second step, we will make observations about the behavior of the WLBH for this narrower class of instances. Finally, based on these observations, we define constraints and set up an optimization model to find an upper bound on the worst possible objective value of the WLBH.

Step 1. Let us run the WLBH on some instance I''. We define $\mathcal{I}'' \subseteq \mathcal{I}'$ as the set of instances for which the WLBH retrieves single-car blocks in the last iteration after it has withdrawn all the multiple-car blocks (if any). The worst possible objective value of the WLBH for instances $I' \in \mathcal{I}'$ is not larger (worse) than the worst possible objective value of the WLBH for instances $\mathcal{I}'' \subseteq \mathcal{I}'$.

Indeed, let us construct a WLBH-worst-case-instance I', where the WLBH retrieves a single-car block of a critical type t^* at some iteration $m_1 \geq 1$ and a multiple-car block of critical type t^{**} i some later iteration $m_2 > m_1$. Then, we may modify the instance to have this multiple-car block withdrawn at iteration m_1 and this single-car block at iteration m_2, respectively, (otherwise introducing no changes in the solution of the WLBH) by adding a sufficient number of railcars of type t^{**} intermittently with railcars of type 0 on the tracks (and increasing the lengths of the tracks if necessary). The optimal objective value and the objective value found by the WLBH for instance I' and the constructed instance I'' will be the same. Therefore, the relative performance of the WLBH for the constructed instance I'' is not better than for instance I'.

In the following, we will talk about instances $I'' \in \mathcal{I}''$. We denote a set of railcars belonging to all the optimal solutions of some instance I'' as $\mathcal{X}^*$.

Step 2. The main idea of the proof is the following: At some iteration, for instance, $I'' \in \mathcal{I}''$, the WLBH withdraws a railcar of some type t^* that is currently critical. We denote the vector of the remaining demand, i.e. of the still requested railcars, as $(n'_1, \ldots, n'_T)$. There is necessarily a railcar of this type t^* belonging to the not yet withdrawn railcars of $\mathcal{X}^*$. The WLBH always retrieves a largest possible block of railcars containing the currently critical type. Because $\mathcal{X}^*$ may contain all the railcars of the largest block, we will examine the properties of the blocks of not yet withdrawn railcars in $\mathcal{X}^*$. For this purpose, we introduce several definitions. Afterwards, we formulate two observations.

Definition. Let us denote the railcars retrieved by the WLBH up to iteration m inclusively as $\mathcal{X}_m^{WLBH}$. We call a set of successively positioned railcars $\{i, i+1, \dots, i+l\} \subseteq \mathcal{X}^*, i, \dots, i+l \notin \mathcal{X}_m^{WLBH}$ with $|\{j \in \{i, i+1, \dots, i+l\} | u(j) = t\}| \leq n'_t, \forall 1 \leq t \leq T$ as a *remaining block* $B_m^{(l)}$ at iteration m.

In other words, we receive a remaining block if we delete $|\mathcal{X}_m^{WLBH}|$ railcars of the corresponding types from $\mathcal{X}^*$ and take a sequence of successively positioned remaining railcars.

Definition. We say that b remaining optimal blocks at iteration m form a *remaining solution* $\mathcal{B}_m = \{B_m^{(1)}, \dots, B_m^{(b)}\}$, if they do not overlap ($B_m^{(l)} \cap B_m^{(l')} = \varnothing, 1 \leq l, l' \leq b$) and contain only still requested railcars $\left(|\{j \in \bigcup_{l=1}^{b} B_m^{(l)} | u(j) = t\}| = n'_t, \forall 0 \leq t \leq T\right)$. We may alternatively denote an optimal solution of instance I'' as $\mathcal{B}_0$, i.e. the remaining solution at iteration 0.

The example in Figure 6 has only one optimal solution so that $\mathcal{X}^* = \{52, 53, 54, 69, 70, \dots, 79\}$. In the first iteration, the WLBH withdraws railcars $\{2, 3, 4\}$. Thus, we do not need railcars of types 1 and 2 anymore, and we remove railcars 53, 72 and 76 from $\mathcal{X}^*$. In this case, the remaining solution is unique and consists of five remaining blocks $\mathcal{B}_1 = \{\{52\}, \{54\}, \{69, 70, 71\}, \{73, 74, 75\}, \{77, 78, 79\}\}$. However, for some instances I'' in some iterations, there may be several distinct remaining solutions.

We also introduce additional notation. Let for some instance I'' the WLBH retrieve $\mathcal{N}_1$ single-car blocks, $\mathcal{N}_2$ blocks with two railcars each, . . ., and $\mathcal{N}_N$ blocks with N railcars each, where $\mathcal{N}_f$ are nonnegative integers for $f \in \mathbb{N}$. Recall that all $\mathcal{N}_1$ single-car blocks are retrieved in the last iterations of the WLBH.

Observation 1. Let $\mathcal{B}_{m-1}$ be a remaining solution, $m \geq 1$ and $\mathcal{B}_m$ is received from $\mathcal{B}_{m-1}$ as a result of iteration m of the WLBH, i.e. $\mathcal{B}_{m-1}$ contains all but $|\mathcal{X}_m^{WLBH}(I'') \setminus \mathcal{X}_{m-1}^{WLBH}(I'')|$ elements of $\mathcal{B}_m$.

- If the WLBH retrieves a single-car block at iteration m, then the number of remaining blocks decreases by one: $|\mathcal{B}_m| = |\mathcal{B}_{m-1}| - 1$.
- If the WLBH retrieves a two-railcar block at iteration m, then the number of remaining blocks may increase by *maximally* one: $|\mathcal{B}_m| \leq |\mathcal{B}_{m-1}| + 1$.
- If the WLBH pulls out a f-railcar block with $f \geq 3$ at iteration m, then the number of remaining blocks may *maximally* increase by f: $|\mathcal{B}_m| \leq |\mathcal{B}_{m-1}| + f$.

In the following, we will provide justification for Observation 1.

Case of a single-car block. Let the WLBH retrieve a single-car block at iteration $m \geq 1$. Due to the WLBH algorithm, all of the remaining blocks $B_{m-1}^{(l)}$ containing a railcar of the currently critical type t^* are single-car blocks. Therefore,

regardless of whether or not the WLBH pulls out one of the remaining blocks, the number of remaining railcar blocks decreases by one: $|\mathcal{B}_m| = |\mathcal{B}_{m-1}| - 1$.

Case of an f-railcar block, $f \geq 2$. Let the WLBH retrieve an f-block, $f \geq 2$. We obtain $\mathcal{B}_m$ by deleting exactly one railcar from $\bigcup_{l=1}^{b} B_{m-1}^{(l)}$, where $B_{m-1}^{(l)} \in \mathcal{B}_{m-1} \forall 1 \leq l \leq b$, for each railcar retrieved by the WLBH at the iteration m. If a deleted railcar is located somewhere in the middle of a multiple-car remaining block $B_{m-1}^{(l)}$, then the number of the remaining blocks increases by one. Because the WLBH retrieves an f-railcar block, the number of the remaining blocks may increase by up to f: $|\mathcal{B}_m| \leq |\mathcal{B}_{m-1}| + f$.

Case of a two-railcar block. Let the WLBH retrieve a two-railcar block. From the paragraph above, it follows that the number of remaining blocks cannot increase by more than two: $|\mathcal{B}_m| \leq |\mathcal{B}_{m-1}| + 2$. We strengthen this bound and show that the number of remaining blocks cannot actually increase by more than one. Indeed, at least one of the railcars retrieved by the WLBH at iteration m belongs to the currently critical type t^*. Because there are no remaining blocks $B_{m-1}^{(l)}$ containing type t^* with more than two railcars (otherwise the WLBH would retrieve a larger block in this iteration), it is impossible to increase the number of the remaining blocks by picking a railcar of type t^*. Therefore, the number of remaining railcar blocks may maximally increase by one: $|\mathcal{B}_m| \leq |\mathcal{B}_{m-1}| + 1$.

Observation 2. Let the WLBH have already retrieved all the multiple-car blocks in the previous iterations and be about to withdraw the first single-car block in the current iteration $m \geq 1$, then

- Any remaining solution $\mathcal{B}_{m-1}$ contains exactly $\mathcal{N}_1$ railcars,
- Any remaining solution $\mathcal{B}_{m-1}$ consists of exactly $\mathcal{N}_1$ remaining blocks.

The first statement of Observation 2 follows straightforwardly from the definition of the remaining solution because exactly $\mathcal{N}_1$ railcars are still requested. For the second statement, because the WLBH retrieves a single-car block in each of the remaining iterations, it has to perform $\mathcal{N}_1$ such iterations. Let a remaining solution $\mathcal{B}_{m-1}$ consist of $\mathcal{N}' \neq \mathcal{N}_1$ remaining blocks. Then, according to Observation 1, the number of remaining blocks in any remaining solution has to decrease by one after each iteration $m' > m$ of the WLBH, which leads to a contradiction.

Step 3. In the following, we will combine the statements of Observation 1 and Observation 2 and construct an optimization problem to find an upper bound on the worst possible performance of the WLBH for instances I''.

Let the WLBH have already retrieved all the multiple-car blocks in the previous iterations and be about to withdraw the first single-car block in the current iteration $m \geq 1$. From Observation 1, the upper bound on the remaining optimal blocks of railcars in a remaining solution $\mathcal{B}_{m-1}$ is $\left(2 + \sum_{f=3}^{N} f \cdot \mathcal{N}_f + \mathcal{N}_2\right)$

because each f- and 2-railcar block may maximally increase the number of the remaining optimal blocks by f and 1 in the course of one iteration, respectively. From Observation 2, $\mathcal{N}_1 \leq \left(2 + \sum_{f=3}^{N} f \cdot \mathcal{N}_f + \mathcal{N}_2\right)$.

The number of railcars retrieved by the WLBH must be equal to the number of requested railcars: $\sum_{f=1}^{N} f \cdot \mathcal{N}_f = N$.

Let us sum up the formulated constraints into an optimization model where we want to construct an instance I'' for which the WLBH retrieves railcars in as many blocks as possible:

$$\text{maximize} \qquad \overline{\text{WLBH}}(\mathcal{N}_1, \ldots, \mathcal{N}_N) = \sum_{f=1}^{N} \mathcal{N}_f \tag{7}$$

$$\text{s.t.} \qquad \mathcal{N}_1 \leq \left(2 + \sum_{f=3}^{N} f \cdot \mathcal{N}_f + \mathcal{N}_2\right) \tag{8}$$

$$\sum_{f=1}^{N} f \cdot \mathcal{N}_f = N \tag{9}$$

$$\mathcal{N}_f \in \mathbb{Z}_{\geq 0} \qquad \forall f \in \{1, \ldots, N\} \tag{10}$$

In the optimal solution, $\mathcal{N}_f = 0$ for $f > 3$ and $\overline{\text{WLBH}} = \frac{2 \cdot N + 2}{3}$. Therefore, for instances $I'' \in \mathcal{I}''$, $\frac{WLBH(I'')}{Opt(I'')} \geq \frac{N+1}{3} \cdot \frac{z_1}{z_0}$.

In a similar way, we can show that $\frac{WLBH(I)}{Opt(I)} \geq \frac{N+1}{3} \cdot \frac{z_1}{z_0}$ also for instances with $Opt(I) = z_0 + z_1$ or with $Opt(I) = 2 \cdot z_1$. Because the approximation ratio is true for all the other values of $Opt(I)$, as explained above, we conclude that for any RRT-instance $\frac{WLBH(I)}{Opt(I)} \geq \frac{N+1}{3} \cdot \frac{z_1}{z_0}$.

□