Multistage Methods for Freight Train Classification*

Riko Jacob[1], Peter Márton[2], Jens Maue[3], and Marc Nunkesser[3]

[1] Computer Science Department, TU München, Germany
jacob@in.tum.de
[2] Faculty of Management and Computer Science, University of Žilina, Slovakia
Peter.Marton@fri.uniza.sk
[3] Institute of Theoretical Computer Science, ETH Zürich, Switzerland
{jens.maue|mnunkess}@inf.ethz.ch

Abstract. In this paper we establish a consistent encoding of freight train classification methods. This encoding scheme presents a powerful tool for efficient presentation and analysis of classification methods, which we successfully apply to illustrate the most relevant historic results from a more theoretical point of view. We analyze their performance precisely and develop new classification methods making use of the inherent optimality condition of the encoding. We conclude with deriving optimal algorithms and complexity results for restricted real-world settings.

1 Introduction

In real-world railway classification yards, incoming trains are split up into single cars and then reassembled to form outbound trains. It turns out that this process often constitutes the bottleneck in freight transportation, but it would be expensive to extend or redesign classification yards that were designed decades ago to accommodate traffic requirements substantially different from today. An obvious way to improve the performance of existing classification yards is to optimize the classification process itself. To this end we revisit the history of classification methods and develop an efficient representation of these schemes, which allows their consistent presentation and analysis. In the light of this novel encoding, we characterize optimal classification schemes and analyze the underlying algorithmic questions.

A complete classification yard is shown in Fig. 1. It consists of a receiving yard, where incoming trains arrive, a classification bowl, where they are sorted, and a departure yard, where outgoing trains are formed. Many yards feature a *hump*, a rise in the ground, from which cars roll in on the tracks of the classification bowl. These yards are called *hump yards* in contrast to *flat yards*, which

* This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL). This work was partially supported by the University of Zilina, Faculty of Management and Computer Science, under institutional grant no. 1/2007.

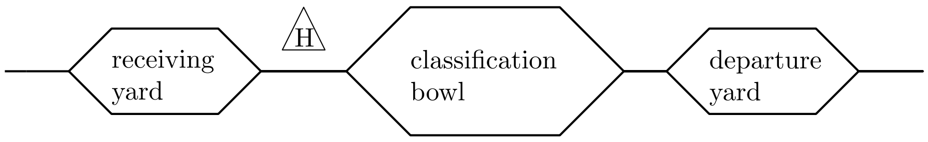

Fig. 1. A typical classification yard with receiving yard, hump (H), classification bowl, and departure yard.

require cars to be hauled by shunting engines. A typical classification bowl is shown in Fig. 2(a). Not all yards have receiving and departure tracks, some have single ended classification bowls as in Fig. 2(b), others have a secondary hump at their opposite end as in Fig. 2(c).

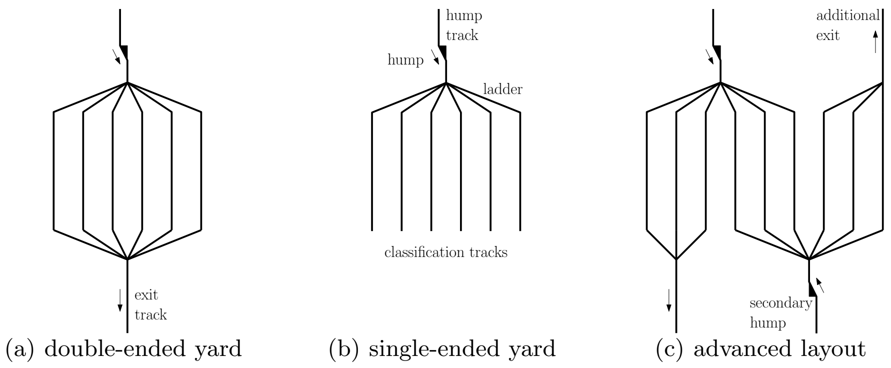

Fig. 2. Some common classification bowl layouts.

General Process of Train Classification The overall classification process looks as follows: inbound trains are collected in the receiving yard on a set of tracks called *receiving tracks*, from where they are moved to the hump track. There, the cars of the train are disconnected and the complete train is pushed over the hump by a yard engine, sending the cars through a series of switches called *ladder*, separately guiding each car on a preassigned *classification track* of the classification bowl. This process is called a *roll-in operation*. Then, the actual sorting process is performed to produce outbound trains, which are picked up by freight locomotives to leave the classification yard.

Regarding the actual classification procedure, there are essentially two modes of operation for shunting yards, which are typically performed in parallel or alternatingly: *single-stage* and *multistage sorting*. In single-stage sorting, each classification track usually corresponds to a common destination, such as a remote classification yard. Departing trains are built by collecting the cars from one or several tracks and coupling them into trains that leave the bowl to the departure yard—if there is any. Single-stage sorting is normally performed for large volume traffic, e.g. traffic between classification yards, and the cars of the created trains are in arbitrary order.

For traffic directly going to its final destination, *multistage* sorting is used. Since the order requirements for this type of outgoing trains are more complex, single-stage sorting is not applicable here. In multistage sorting, after the incoming trains have been pushed over the hump (*primary humping*), a shunting engine repeatedly pulls back the cars from a given track (*pull-out operation*) over the hump on the hump track. These cars are then pushed over the hump again, so that again each car can be independently routed through the ladder to any classification track. This process, called *rehumping*, is iterated until all outgoing trains have been formed. If a classification track is used only for receiving cars of an outgoing train, but the cars on it are never pulled back to the hump track, it is called *train formation* track.

Related Work Multistage methods are presented from an engineering point of view in a number of publications from the 1950s and 1960s [1–4]. Krell [3, 4] compares two basic multistage methods and three improvements of one of them, including an example for dealing with a restricted number of available classification tracks. Some of these methods had been described earlier in a different fashion by Flandorffer [1].

Some of these methods were again considered by Siddiqee [5] in 1972 and in a series of publications in the 1980s by Daganzo, Dowling, and Hall [6–9]. These publications generally deal with multiple outbound trains, but the actual structure of inbound trains is completely ignored.

A classification problem similar to single-stage sorting was studied by Dahlhaus, Horak, Manne, Miller, and Ryan [10, 11] in 2000. For their train classification model, they give a notion of presortedness of the input train which is used to improve the classification process. Several degrees of freedom in the order requirement of the outbound train are regarded in [11], while finding an optimal schedule for one specific such type is shown to be NP-complete in [10].

A systematic framework for classifying single- and multistage classification methods is given by Hansmann and Zimmermann [12]. For the case of a limited number of classification tracks and a different output requirement, which they call 'splitting', they independently obtain a result similar to the one we give in Sect. 6 for the case without splitting. Furthermore, the authors show for a specific multistage method that finding an optimal schedule is NP-hard for the output specification of [10] mentioned above.

Baumann [2] explains the design aspects concerning multistage train formation for the design of the classification yard 'Zürich-Limmattal' in Switzerland. The resulting layout features a secondary hump similar to Fig. 2(c), which is currently not used due to cost and organizational reasons [13].

The historic results mentioned in this section are reconsidered in Sect. 4 from a more theoretical point of view.

Outline In the following section we introduce the above described problem and concepts formally, including the objective of our problem. Then, we present an efficient encoding for representing the classification process in Sect. 3. This allows us to concisely describe and analyze the above methods as done in Sect. 4, followed by analyzes of new problem variants in Sect. 5 and Sect. 6 and some concluding remarks in Sect. 7.

2 Model and Notation

In this section we introduce the terminology and notation used in our model. We assume the common yard layout of a single- or double-ended classification bowl with a single hump as shown in Fig. 2(b) and Fig. 2(a), where the *classification tracks* are denoted by $\theta_1, \ldots, \theta_W$. We denote their number by W, the *width* of the yard, and denote by $C_{\max}$ the *capacity* of the yard, i.e. the maximum number of cars that fit on any of these tracks.

Cars will be represented by positive integer numbers and trains by (ordered) n-tuples of cars; the number of cars n of a train T will be referred to by the *length* of T. In our model, there is a set of ℓ *input trains* and an ordered set of m *output trains*, together called a *classification task*, for which we make the following assumptions: for the ℓ input trains $T^i = (\tau_1^i, \ldots, \tau_{n'_i}^i)$, $i = 1, \ldots, \ell$, with a total number of cars $n := \sum_{i=1}^{\ell} n'_i$, we assume $\tau_j^i \in \{1, \ldots, n\}$ and all cars are distinct. We further assume that concatenating the output trains in their given order yields the sequence $(1, \ldots, n)$, i.e., if n_i denotes the length of the i-th output train, $i = 1, \ldots, m$, the first output train is given by $(1, \ldots, n_1)$, the second by $(n_1 + 1, \ldots, n_1 + n_2)$, and the last by $(n - n_m + 1, \ldots, n)$.

For any train $T = (\tau_1, \ldots, \tau_n)$, car τ_1 is called the *head* of T, and, for any pair of cars τ_i, τ_j of T with $i < j$, we say τ_i is *in front of* τ_j. For a train T located on the hump track, the head of T represents the car that is closest to the hump. For a train T located on some classification track, its head represents the car closest to the dead-end. Thus, the train in Fig. 3(b) is represented by $(6, 1, 4, 2, 3, 5)$ and the train in Fig. 3(f) by $(1, 2, 3, 4, 5, 6)$.

Any multistage sorting method consists of a sequence of alternating roll-in and pull-out steps. In order to specify a single pull-out step, it suffices to specify which is the classification track to pull out cars from. However, to fully specify a roll-in operation, a target track must be given for every car on the hump track. We call such a pair of operations a *hump step*, and an initial roll-in followed by a sequence of h hump steps is called a *classification schedule* of *length* h. A classification schedule is called *valid* for a classification task if applying it transforms the given set of input trains into the set of output trains. Unless otherwise stated, our objective is to find classification schedules of minimum length.

Definition 1 (Optimal Classification Schedule). *Given a* classification task *by ℓ input trains $(\tau_1^i, \ldots, \tau_{n'_i}^i)$, $i = 1, \ldots, \ell$, and the lengths $(n_1, \ldots, n_m)$ of the m outgoing trains, find a valid classification schedule of minimum length.*

Note that, according to the definitions above, the term *length* may refer either to the number of hump steps of a schedule or to the number of cars of a train. In the remainder of this paper, the respective meaning will always be clear from the context. Moreover, we will sometimes abbreviate statements referring to pull-out steps, such as abbreviating 'the cars of a track are pulled out' to 'a track is pulled (out)'.

3 Classification Schedules

In this section we describe an encoding of classification schedules by sets of binary numbers. Conversely, we show how to interpret such sets as schedules, which yields a bijective relation between both. Furthermore, a notion of presortedness is introduced, which allows deducing optimal schedules. As it turns out, the core of a classification scheme can already be given by specifying how a single input train is sorted into a single output train. For this reason we first consider this case and develop the encoding scheme. At the end of this section we show how to extend the results to the general case.

Single train We start by introducing a simplified view on the tracks. After a track has been emptied, cars may be sent to it in subsequent steps, so one physical track might be filled and emptied more than once during a classification procedure. We model this by introducing *logical tracks* that we define such that pull-out i is performed on logical track i. This means that the logical tracks are pulled out in the order $(1, 2, \ldots, h)$. For a classification schedule of length h, the mapping from the h logical to the W physical tracks is given by a sequence $(\theta_{i_1}, \ldots, \theta_{i_h})$ of physical track names, called the *track sequence*.

The course of a single car τ can now be represented by an h-bit binary string $b = b_h \ldots b_1$, $b_i \in \{0, 1\}$, where $b_i = 1$ if and only if τ visits the i-th (logical) track. (In the following these strings are interpreted as little-endian numbers, i.e. b_h is the *most* significant bit of b.) This representation uniquely defines the course of car τ: τ is pulled out in the i-th step if $b_i = 1$ simply because it has been sent there in some earlier step. Then, it is rolled in on the k-th track given by $k := \min_{j>i, b_j=1} j$, i.e. the lowest bit $b_k = 1$ left of i. If $b_j = 0$ for all $j > i$, then τ is guided to the train formation track of its target train. The track for the initial roll-in is given by the least significant bit b_i with $b_i = 1$. The complete schedule for a train of n cars can be simply represented by a *binary encoding* $B = (b^1, \ldots, b^n)$ consisting of a sequence of binary numbers, such that $b^i = b_h \ldots b^i_1$ encodes the course of the i-th car, $i = 1, \ldots, n$.

An example is given in Fig. 3, which shows a classification procedure and the binary representation of its schedule for a single input train of six cars. There are more classification tracks than schedule steps, so the above mentioned mapping from logical to physical tracks is one-to-one. Note that in our model the classification process is not yet finished in situations (d) or (e); a valid output train is obtained only when the situation depicted in (f) has been reached.

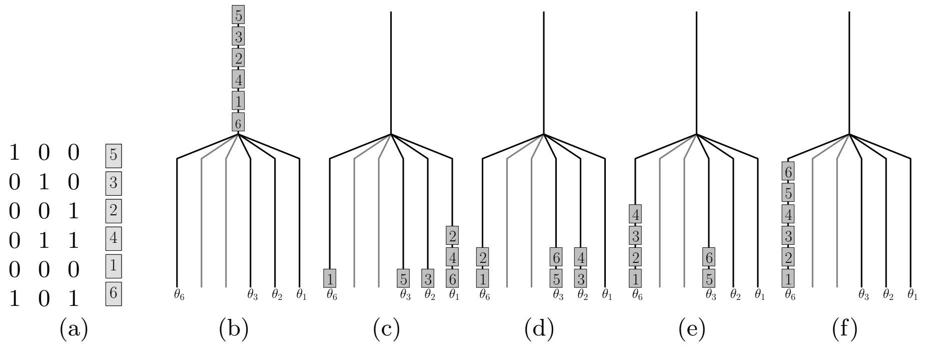

Fig. 3. An exemplary classification procedure of $h = 4$ steps for a train of six cars, using θ_6 for output train formation. The encoding is shown in (a), the input train in (b). Figures (c)–(f) show the consecutive situations after each hump step, always pulling out the cars of the rightmost occupied track.

The following lemma shows how to read the binary representation of schedules: if two cars have different codes, the car with the smaller code will be located in the target train at a position closer to the head of the train. If two cars have the same code, they will not swap their relative order.

Lemma 1. *For a classification schedule for an incoming train $(\tau_1, \dots, \tau_n)$ given by a binary encoding $B = \{b^1, \dots, b^n\}$ two cars τ_i and τ_j for $i < j$ swap their relative position if and only if $b^i > b^j$.*

Proof. There are three possible cases for the order of b^i and b^j. First, if $b^i = b^j$ the two cars will go exactly the same course and end up in the same order as in the input train. Second, if $b^i < b^j$, let k be the most significant index k with $b^i_k = 0$ and $b^j_k = 1$. Car τ_j is sent to some track θ_{next} in hump step k. As b^i and b^j are identical on all bits left of k, car τ_i was sent directly to θ_{next} in a previous step, so τ_i appears at a position in front of τ_j. For the same reason, the two cars will not swap their relative order at any step later than k, so τ_i ends up on the output track at a position in front of τ_j in the output train. Finally, by symmetry if $b^i > b^j$ car τ_j ends up in front of τ_i. The three cases together give the statement of the lemma.

In the following theorem, we show that there is a bijection between valid classification schedules and binary encodings with a special property.

Theorem 1. *A classification schedule for an incoming train $(\tau_1, \dots, \tau_n)$ of h steps is valid if and only if its binary encoding $B = (b^1, \dots, b^n)$, where the b^i are h-bit binary numbers, has the following property:*

$$\textit{For all } \; i, j \in \{1, \dots, n\} \; \textit{ with } \; i < j, \; \textit{if } \; \tau_i > \tau_j \; \textit{ then } \; b^i > b^j \qquad \text{(P)}.$$

Proof. If a classification schedule translates into a binary encoding with property (P), then, by Lemma 1, exactly the cars that need to be swapped are swapped and the classification schedule is thus valid. Conversely, if a binary encoding does not have property (P), then again by Lemma 1 the corresponding classification schedule cannot be valid.

From the above theorem it is clear that an optimal schedule corresponds to a binary encoding B of minimum length that satisfies property (P). For constructing B we need to specify, which cars can get the same code, which leads to a notion of presortedness. We show that to this end it is enough to look at consecutive cars in the output train that are in the wrong order in the input train:

Definition 2. *Given a train $T = (\tau_1, \ldots, \tau_n)$, we say that a pair $(i, i+1), i \in \{1, \ldots, n-1\}$ defines a* break *$\tau_j = i+1, \tau_k = i$ for indices j, k with $j < k$. The set of breaks canonically decomposes each train into* chains *that can be ordered by their first elements.*

For example, train $T = (9, 4, 5, 7, 1, 2, 8, 6, 3)$ decomposes into the disjoint chains $c_1 = (1, 2, 3)$, $c_2 = (4, 5, 6)$, $c_3 = (7, 8)$, and $c_4 = (9)$.

Lemma 2. *Only cars of the same chain can get the same code. For two cars of two different chains the smaller one must get a smaller code.*

Proof. By Definition 2 all cars of a chain are in the correct order in the input train. By Lemma 1 these cars can get the same code. For the other direction note that for each break $(\tau_j = i+1, \tau_k = i)$ in a valid schedule $b^j > b^k$ holds by Lemma 1. Now take any two cars τ_ℓ, τ_m from two neighboring chains separated by break $(i, i+1)$. If $\ell < m$ and $\tau_\ell > \tau_m$ they cannot get the same code directly by Lemma 1. So assume $\ell < m, \tau_\ell < \tau_m$. Car τ_k is the last element of the chain of τ_ℓ, and τ_j is the first element of the chain of τ_m, therefore $b^\ell \leq b^k < b^j \leq b^m$, which implies $b^\ell < b^m$. The claim of the lemma follows by transitivity.

The main result of this section now follows as a corollary of this.

Theorem 2 (optimal schedules). *Let $T = (\tau_1 \ldots \tau_n)$ be a train of length n and c its number of chains. T can be reclassified within $\lceil \log_2 c \rceil$ hump steps in a hump yard of unrestricted width and capacity. This bound is optimal.*

This result can easily be extended to more complicated objective functions. One of the most general such objectives is to charge a cost of α for a pull-out of a train and β for a roll-in of a single car. It still holds that for an encoding B the number of bits equals the number of pull-outs of the corresponding classification schedule. The number of 1's in the encoding equals the number of roll-ins. For an incoming train of c chains and a fixed number h of steps we can construct the optimal classification schedule of length h by choosing greedily the c h-bit binary numbers having the least 1's. By evaluating the objective functions for the admissible range $\lceil \log_2 c \rceil \leq h \leq c$ the optimal classification schedule can be found.

Multiple Trains Any reasonable classification task involves multiple incoming and multiple outgoing trains. However, as we will see in this section, once the order of the incoming trains has been determined, such a shunting task is not more difficult than sorting a single incoming into a single outgoing train.

Observation 1 *Given ℓ incoming trains $I = (\tau_1^1, \dots, \tau_{n_1'}^1), \dots, (\tau_1^\ell, \dots, \tau_{n_\ell'}^\ell)$* in the order in which they are to be rolled into the yard*, and m outgoing trains by their lengths $(n_1, \dots, n_m)$ then the optimal classification schedule for these trains for the case of unrestricted capacity is determined by the union of the optimal classification schedules $\{B_1, \dots, B_m\}$ for the following m classification tasks: Let $I' = (\tau_1, \dots, \tau_n)$ denote the concatenation of the ℓ input trains. Then the i-th classification task, $1 \le i \le m$, is to sort the subsequence of I' that corresponds to the i-th output train. The length of the resulting schedule for the whole classification task is given by $\max_{1 \le i \le m} \text{length}(B_i)$.*

An analogous observation holds for classification with width restriction as discussed in Sect. 6, but not for restricted length as discussed in Sect. 5. It is also important to note that the observation assumes a fixed order of the incoming trains. This assumption is realistic in cases where the input trains arrive scattered over time or have some other natural order. If this is not the case, the problem of choosing an optimal order arises. This problem is closely connected to a special minimum feedback arc problem [14].

Lemma 3. *There is a one-to-one correspondence of finding the permutation of input trains $I = \{T_1, \dots, T_\ell\}$ that leads to the optimal classification schedule (OPT-PERM) and computing minimum feedback arc sets in directed multigraphs, the edges of which form a Eulerian path.*

Proof. We first show how to transform OPT-PERM into an minimum feedback arc set instance $G = (V, E)$. Each incoming train T_i is mapped to a node $n(T_i)$. For each pair of cars $\tau_k = i \in T_\alpha, \tau_j = i + 1 \in T_\beta$ we add a directed edge $(n(T_\alpha), n(T_\beta))$. It follows that in total n (potential self-)edges are added to the graph. These edges correspond to a Eulerian path in G. For any given permutation π of I the number of breaks of $\pi(I)$ equals the number of arcs pointing backwards in the linear arrangement $\pi(V)$. By deleting exactly these arcs the graph becomes acyclic. Thus by Theorem 2 the objective function of OPT-PERM equals the logarithm of the objective function of minimum feedback arc set plus one. For the other direction it is easy to see that the following construction will transform any multigraph with an Eulerian path into an OPT-PERM instance with the same relation of the objective functions. For each node $n \in V$ we introduce an incoming train $T(n)$. Then we walk along the Eulerian path $P = (n_{i_1}, \dots, n_{i_{m+1}})$ and add for each $n_{i_j} \in P$ car j to train $T(n_{i_j})$.

To the best of our knowledge, the complexity status of minimum feedback arc set in such graphs is open. However, by a lemma of Newman, Chen, and Lovász [15, Theorem 4], a polynomial algorithm for OPT-PERM would lead to a $\frac{16}{9}$-approximation algorithm for the general minimum feedback arc set problem, improving over the currently best known $O(\log n \log \log n)$ algorithm [16].

4 Multistage Classification Methods

With the efficient encoding of schedules at hand, we illustrate the most prominent classification methods in this section and analyze their performance in detail.

4.1 Basic Multistage Methods

Multistage methods can be categorized into two general classes: *sorting by train* and *simultaneous marshalling*. In the following we assume that we are given m output trains by their lengths $n_1, \ldots, n_m$ and define $n_{\max} = \max_{1 \leq i \leq m} n_i$ and $n_{\min} = \min_{1 \leq i \leq m} n_i$.

Sorting by Train Sorting by train comprises two stages. First, inbound cars are separated according to their outbound trains by sending all cars of a common output train to the same track. Second, the resulting unordered trains are processed successively: a train is pulled back over the hump and rolled in again, sorting the cars according to their position by sending each car to a different track. Finally, the single cars are moved from the tracks in the required order and coupled to form an outgoing train. In double-ended yards this can be performed by a shunting engine from the opposite end of the yard. As in the rest of the paper the train formation tracks will not appear in the encoding as they are implicitly given. The process continues with the next train.

The length h of the schedule is given by $h = m + \sum_{i=1}^{m} n_i$. For the encoding $b_h \ldots b_1$ of a car τ_ℓ^k, bit $b_i = 1$ if $i = k + \sum_{j=1}^{k-1} n_i$ (corresponding to the initial roll-in) or $i = k + \sum_{j=1}^{k-1} n_i + \ell$ (corresponding to the second stage).

This method occupies exactly m classification tracks after the first stage, so the total number of tracks is at least $m + n_{\min} - 1$, and at most $m + n_{\max} - 1$, while the latter number is tight if a train with $n_{\max}$ cars is processed in the second stage first.

Sorting by train is also called *initial grouping according to outbound trains* [5].[1]

Simultaneous Marshalling Unlike sorting by train, the first stage of the two-stage method *simultaneous marshalling* sorts according to the cars' position in the output train. In terms of codes this step forces $b_i = 1$ for every $i-th$ car τ_i^k of any train $1 \leq k \leq m$. In the second stage, the cars are sorted according to their target trains: the tracks are successively pulled out in the order of the positions, and each set of cars pulled out is directly rolled back in, always sending cars of

[1] The according names used in the German literature are *Ordnungsgruppenverfahren* for sorting by train, *Simultanverfahren* for simultaneous marshalling, and furthermore *Elementarverfahren* to explicitly refer to the basic version of the latter. Triangular sorting is called *Vorwärtssortierung bei höchstens zweimaligem Ablauf*, geometric sorting *maximale Vorwärtssortierung* in [3].

a common output train to the same classification track. This is already implied by the above codes.

This multistage method minimizes the number of cars rolled-in, which must be paid for by a number $n_{\max}$ of hump steps that is maximal for an unrestricted classification yard.

Regarding the track requirement, exactly $n_{\max}$ tracks are used in the first stage. Thus, at most $n_{\max} + m - 1$ tracks are needed since up to $m - 1$ further tracks are needed for train formation, and at least $n_{\min} + m - 1$: pulling the first track of the last $n_{\min}$ tracks to be pulled forces starting the formation of all m output trains (if not yet started), so $n_{\min} + m - 1$ tracks are occupied then.

In contrast to sorting by train the formation of all output trains is performed simultaneously. Simultaneous marshalling is also called *sorting by block*[1], the *simultaneous method*, or *initial grouping according to subscript* [5].

The notion of a *block* corresponds to a set of cars that take a common itinerary over potentially many shunting yards. A block is not broken up at the intermediate classification yards. The associated blocking and makeup problems are out of the scope of this paper, see [17] for references. Blocking is particularly advantageous in large countries like the U.S. and often not applied in most smaller freight systems [18]. If in some freight system blocks are built in multistage sorting, a classification task with cars of unspecified order in some of the target trains arises. Blocks that are broken up have no influence on the classification schedule, blocks that are not broken up at the current classification yard can be treated as a weighted car.

This method never guides cars to a track of the final train formation at the first stage, which is a necessary assumption for a layout as shown on the right of Fig. 2. However, if the tracks for target train formation are accessible from the primary hump, the schedule becomes one step shorter, which also holds for the following variants.

4.2 Variants of Simultaneous Marshalling

In the basic variant of simultaneous marshalling, every car is pulled out once and rolled in twice, once in either stage. In other variants this restriction is dropped. Instead of stages, these variants are specified by sequences of hump steps, and each method is characterized by a class of encodings of common attributes.

Triangular Sorting A variant of simultaneous marshalling called *triangular sorting* is given by allowing at most three roll-in operations (including the final roll-in of a car to its output train) for each car. For the schedule encoding, this yields a restriction of not more than two bits equal to one per car.

For this method Krell gives an upper bound of $\frac{1}{2}h(h+1)$ on the maximum length $n_{\max}$ of an output train that can be sorted in h steps [3]. This result can be reformulated in terms of chains yielding a better bound in general. If $c_1, \ldots, c_m$ denote the respective numbers of chains of the trains, for a sufficiently large classification yard, classifying by triangular sorting can be done within h

hump steps if $c_{\max} \leq \frac{1}{2}h(h+1)$. This follows immediately by our encoding: the number of distinct codes $b_h \dots b_1$ of length h and $b_i = 1$ for at most two different $i \in \{1, \dots, h\}$ is given by $\binom{h}{1} + \binom{h}{2} = \binom{h+1}{2}$, and the required number of distinct codes is not greater than the maximum number of chains by Lemma 2.

The triangular-like occupation of the classification tracks after the initial roll-in explains the name of this variant.[1] The method can be generalized to any restriction on the number of roll-ins for a car.

Geometric Sorting The method of *geometric sorting*[1] is derived from simultaneous marshalling by dropping the number restriction of roll-ins completely, which corresponds to binary codes with no restriction at all. The performance of this method is given in the literature by $n_{\max} \leq 2^h - 1$ for h hump steps [3]. In combination with the notion of chains this yields exactly the classification scheme of Theorem 2 with a bound of $c_{\max} \leq 2^h - 1$, where $c_{\max}$ denotes the maximum number of chains in any output train.

Considering the special case of a single output train of length $2^k - 1$ for some positive integer k, the initial roll-in sends $2k - i$ cars to the i-th track, $i = 1, \dots, k$; the sum of these numbers gives the geometric sum, which explains this method's name. As mentioned before, geometric sorting minimizes the number of hump steps, assuming the number and capacities of tracks are unrestricted. If this cannot be assumed, simultaneous marshalling variants of the following sections should be considered.

5 Restricted Track Capacities

Real world classification yards have classification tracks of bounded capacity for (intermediate) sorting and final train formation. In this section we show that the problem of finding an optimal classification schedule becomes NP-complete with this additional constraint and point out a special case where the problem remains easy.

5.1 General Case

Assuming bounded track capacities for the classification tracks yields an NP-hard problem as shown in Thm. 3 below. The bound on the track capacities is formalized as follows: All tracks have a bounded capacity of $C_{\max}$, i.e., they can accommodate at most $C_{\max}$ cars, with the exception of specific train formation tracks where the outbound trains are formed. We do not allow to pull-out from these tracks.

Theorem 3. *It is NP-hard to find the optimal classification schedule for capacity-bounded tracks.*

Proof. By reduction from "Not ALL Equal 3-SAT" (NAE3SAT) which is known to be NP-complete [19, LO3]. Given an instance of NAE3SAT having n variables and m clauses, we construct an instance of $2n$ input trains that are to be sorted into $2n$ outgoing trains without any interaction between the trains, i.e., the ith input train has cars only for the ith outgoing train. Note that even though there are multiple input trains their order is irrelevant, because there is a one-to-one correspondence of input to output trains (this is in contrast to the general situation discussed in Lemma 3). For ease of exposition we start the proof by making two assumptions, and show later that these can be easily enforced. First, each car can be part of at most one additional roll-in. Second, we can have individual capacity bounds for all logical tracks.

The main idea of the proof is to allow to use a given number $M = 4n + 2m$ of steps and thus logical tracks and to let all input trains have exactly $M - 1$ chains. It follows that at most one of the chains of each train can be split or a single logical track can be left unused (if two chains of the same train end up on the same track they must be in wrong order, which necessitates an additional roll-out in contradiction to the first assumption). The transformation enforces the latter possibility for all trains. Thus, the "local decisions" that we can encode are for each train, which track should be left unused.

We proceed to show how to use this idea in the transformation and give an example in Fig. 4. First, for the input trains it is enough to specify the length of each of their chains, instead of giving the full sequence that leads to these chains. For example we will define a train as $(1, 4, 2)$ by its sequence of chains (*chain sequence*) and ignore whether this comes from an input train $(2, 6, 1, 3, 4, 7, 5)$ or $(6, 2, 3, 1, 4, 5, 7)$. chains and logical tracks are tightly connected. As all chain sequences will have one chain less than there are logical tracks, the chain-to-track assignment can be specified by giving the position of the *gap*, i.e., the logical track left out, e.g., $(1, *, 4, 2)$. In this example the chain of length 1 goes to logical track 1, length 4 goes to 3, and length 2 goes to 4.

$$x_1 \vee \bar{x}_2 \vee x_3 \wedge \bar{x}_1 \vee x_2 \vee \bar{x}_3$$

	x_1	x_1	x_2	x_2	x_3	x_3	C_1^+	C_1^-	C_2^+	C_2^-	x_1	x_1	x_2	x_2	x_3	x_3
	1	0	1	0	1	0					1	0	1	0	1	0
x_1	k	1	1	1	1	1	k	1	1	1	k	1	1	1	1	
$\bar{x}_1$		k	1	1	1	1	1	1	1	k	1	k	1	1	1	1
x_2		1	1	k	1	1	1	1	1	k	1	1	1	k	1	1
$\bar{x}_2$	1	1	k	1	1	1	k	1	1	1	1	1	k	1	1	
x_3		1	1	1	1	k	1	k	1	1	1	1	1	1	1	k
$\bar{x}_3$	1	1	1	1	k	1	1	1	k	1	1	1	1	1	k	

Fig. 4. Sketch of the transformation for an example with two clauses on three variables.

Each chain sequence has $4n + 2m - 1$ chains, which correspond to a *start variable part* of length $2n$, followed by a *clause part* of length $2m$ and an *end variable part* of length $2n - 1$. There are $2n$ chain sequences, one for each literal. All chains have either length k ("ON") or length 1 ("OFF"). The purpose of the start and end part of the chain sequences is to force the gap into these sequences. This is achieved by defining the start and end sequences of both x_i and $\bar{x}_i$ as follows:

$$(\overbrace{\underbrace{1,1}_{\text{1 pair/variable}},\ldots,\underbrace{k,1}_{\text{pair } i},1,1,\ldots,}^{\text{start part}}\overbrace{\ldots}^{\text{clause part}},\overbrace{1,1,\ldots,\underbrace{k,1}_{\text{pair } i},1,1,\ldots,1,1}^{\text{end part}})$$

Both sequences have length $M-1$ together with the clause part that remains to be specified.

The first $2n$ logical tracks and the last $2n$ logical tracks have all capacity $2n+k-1$, except for the first and the last track which have both capacity $n+k-1$. The total capacity of the first $2n$ positions of all chain sequences exceeds the total available capacity for the start part by n, the same holds for the end part. This situation forces at least n gaps in the start part and at least n gaps in the end part, thus exactly n gaps in both parts. Having identical sequences for a variable and its negation enforces together with the capacity bound that for each variable either there is a gap at the beginning of the chain sequence for x_i and the end of the one for $\bar{x}_i$ or vice versa. Thus, we can think of the chain sequences for variable x_i as either being to the left ($x_i = $ TRUE) or to the right ($x_i = $ FALSE).

The clause sequence has $2m$ logical tracks, 2 for each clause. The first track of clause j stands for a literal making this clause true (contributing to set C_j^+), the second for one making it false (contributing to set C_j^-). From above it follows that there can be no gaps in the clause parts. We indicate the occurrences of literals in clauses by turning on the corresponding position in the chain, as exemplified in Fig. 4. The chains for each literal can be either left or right and therefore contribute either to C_j^+ or C_j^- for each clause j. By setting the capacity constraint to $2n+2k-2$ for each logical track in the clause part we enforce the not all equal constraint. This follows because this capacity limit is exceeded if and only if three literals contribute to the same of the sets C_j^+ and C_j^-. Therefore, under the assumptions above there is a yes-instance for NAE3SAT if and only if there is a classification schedule for the transformed instance that respects the given capacity bound.

It remains to specify how to enforce the two properties above. First, we want to replace the individual capacity constraints by a uniform one. To this end, we add one chain sequence of full length M. As every car is only allowed to be pulled once, the classification schedule for this chain sequence is unique. By adjusting the lengths of the chains of this chain sequence, the differences in the capacity constraints can be adjusted.

To enforce that every car is pulled at most once, we add one chain sequence with one big non-trivial chain. The length of this chain is exactly the excess capacity of the logical tracks w.r.t. all chain sequences constructed before. Now, if any car were pulled twice another car could not be pulled at all, which is impossible in a correct classification schedule.

5.2 Other Results

Optimal classification schedules for tracks of bounded capacity $C_{\max}$ translate to binary encodings B with the property that for each bit position the total sum of 1's *weighted by the lengths of the corresponding chains* is bounded by $C_{\max}$. We have recently shown [20] that if all chains have the same length optimal codes can be constructed efficiently (in the size of the resulting codes). On the other hand, for arbitrary chain length the above proof shows NP-completeness.

6 Restricted Number of Classification Tracks

In this section we consider the width constraint of a shunting yard. In particular we are interested in classification tasks for which the optimal classification schedule without width restriction needs a number n of pull-outs and thus logical tracks that is greater than the available number of physical tracks W. This schedule is in general not directly implementable. In this section we show how to construct optimal schedules under restricted width. From Observation 1 we know that it is enough to consider the case of a single input and a single outgoing train. As mentioned in Sect. 1, an example for this setting is given in [3] including the corresponding schedule and maximum number of cars that can be sorted for a number of given tracks. In [12], Hansmann and Zimmermann obtain a corresponding result for a similar setting that differs from our model as follows: After the last rehumping step, the tracks are treated in a stack-like way, such that an outgoing train can be composed by popping its cars from multiple tracks. This assumption is reasonable particularly for self-propelled units.

To simplify the exposition, we slightly deviate from the notation in the other sections and assume that at the start the input train is already in track θ_1. For this initial roll-in we count one step (all codes have $b_1 = 1$). We also count the track of the outgoing train as part of the code (all codes have $b_h = 1$).[4]

A complete specification of the classification schedule now requires in addition to the binary codes of length h the track sequence $(\theta_{i_1}, \ldots, \theta_{i_h})$ for this schedule. By the assumption above the first pulled track is the input track θ_1, and the last pulled track is the output track. The binary codes are restricted by the destination track being available which leads to the following restriction for the codes. More precisely, assume that a code has a 1 at a certain position. Then there are precisely W next choices for tracks, namely the first occurrence of a θ_i in the remaining sequence of pull-outs, $1 \leq i \leq W$, and these are the only possibilities for the next 1.

Observation 2 *The binary encoding $\{b^1, \ldots, b^n\}$ for valid classification schedules on yards of width W and unrestricted width have the property that if any of the codes $b^i, 1 \leq i \leq n$ has $b^i_j = 1$ then the set of indices of follow-up tracks over all codes $\{k | \exists i', k = \min_{j'>j, b^{i'}_{j'}=1} j'\}$ has cardinality at most W.*

[4] These assumptions are not crucial for the correctness of the statements below. However they make the recurrence equations easier to read.

If the tracks are pulled in a round robin fashion these are exactly the next W logical tracks and thus bit positions in the code, i.e., for round robin there must not be W consecutive zeros in any of the codes. We will show that such a round-robin strategy dominates all other strategies.

Let us analyze the number R_h of runs that can be sorted by h pull-outs on W tracks that are used round-robin. We have $R_1 = R_2 = 1$, and $R_h = 2^{h-2}$ for $3 \leq h \leq W$ as there are $h-2$ positions with an unrestricted binary code.

Then, we get the recurrence equation $R_h = \sum_{i=h-W}^{h-1} R_i$ for $h > W$: All valid codes of length h have a 1 at position h, then have the next most significant 1 at a position in the range $h-W$ to $h-1$. Now the number of such codes is the sum of the number of codes starting with a 1 at this particular position, having a trailing 1, and no W consecutive zeros.

For $W = 2$ these numbers are the Fibonacci numbers F_i, for larger values of W a generalization of them. In any case we have $F_h \leq R_h \leq 2^{h-2}$.

Once we know the correct h for a given W and a number of chains n the corresponding codes can also be efficiently constructed, for example by a recursive algorithm that branches in each node into the W choices for the next 1.

Now it remains to be shown that it is optimal to pull the tracks in a round robin fashion. We will do this inductively. Of course for $h \leq W$ this is the case. Assume we already know that the maximal number of codes on h' positions (for the best possible track sequence) is $R_{h'}$ for all $h' < h$. Now take one optimal track sequence and set of codes for h pull-outs. The codes divide into at most W classes by their second 1 (the positions depend on the track sequence). Order the classes according to this position of the second 1. Then, the first class has codes of length at most $h-1$, the second of length at most $h-2$, and so on. Hence, the number of codes in the classes is bounded by R_{h-1}, R_{h-2} and so on (even if the different classes were allowed to have different track sequences), yielding that at most R_h codes are possible. The following theorem sums up these results.

Theorem 4. *A classification schedule for a yard of width W and unrestricted length and an input train of n chains needs h steps in the above model, where $h \in I\!N^+$ is the smallest integer h such that r is greater or equal to the solution of the recurrence equation*

$$R_h = \begin{cases} 1 & \textit{for} \quad h = 1 \\ 2^{h-2} & \textit{for} \quad 2 \leq h \leq W \\ \sum_{i=h-W}^{h-1} R_i & \textit{for} \quad h > W \end{cases}$$

The corresponding track sequence is round-robin, i.e, $(\theta_1, \theta_2, \ldots, \theta_W, \theta_1, \theta_2, \ldots)$. This classification schedule is optimal and can be constructed in linear time (in the size of the schedule). For $h = 2$ we have that $R_h = F_h$ where $F_h = \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}}$ is the h-th Fibonacci number, and φ the golden ratio.

Proof. We have already shown that a round-robin track sequence dominates all other sequences, and that R_h equals the maximum number of chains that can be sorted by "round-robin" codes. The optimality now follows directly from Lemma 2. The construction is via the mentioned recursive algorithm.

7 Concluding Remarks

We have developed an efficient encoding of freight train classification schedules to present, analyze, and develop train classification methods for real-world hump yards. This surprisingly simple though powerful encoding can be used to analyze the efficiency of commonly used multistage methods, of which we proved the optimality of the simultaneous variant geometric sorting in terms of hump steps, considering presorted input.

Future Work It might be interesting to find further optimization criteria for train classification in the literature which are relevant in practice, in order to incorporate these objectives in the encoding scheme. There are further possibilities to specify output requirements, similar to the mentioned concept of blocks, and a straightforward question is how to derive optimal schedules in such settings. Finally, if the presented methods can be simulated to successfully work in practice, their implementation may accelerate the classification process in many real-world hump yards.

Acknowledgments

We would like to thank N. S. Narayanaswamy for discussions on the relation of OPT-PERM to minimum feedback arc set problems.

References

1. Flandorffer, H.: Vereinfachte Güterzugbildung. ETR RT **13** (1953) 114–118
2. Baumann, O.: Die Planung der Simultanformation von Nahgüterzügen für den Rangierbahnhof Zürich-Limmattal. ETR RT **19** (1959) 25–35
3. Krell, K.: Grundgedanken des Simultanverfahrens. ETR RT **22** (1962) 15–23
4. Krell, K.: Ein Beitrag zur gemeinsamen Nutzung von Nahgüterzügen. ETR RT **23** (1963) 16–25
5. Siddiqee, M.W.: Investigation of sorting and train formation schemes for a railroad hump yard. In: Proc. of the 5th Int. Symposium on the Theory of Traffic Flow and Transportation. (1972) 377–387
6. Daganzo, C.F., Dowling, R.G., Hall, R.W.: Railroad classification yard throughput: The case of multistage triangular sorting. Transportation Research, Part A **17**(2) (1983) 95–106
7. Daganzo, C.F.: Static blocking at railyards: Sorting implications and track requirements. Transportation Science **20**(3) (1986) 189–199
8. Daganzo, C.F.: Dynamic blocking for railyards: Part I. homogeneous traffic. Transportation Research **21B**(1) (1987) 1–27
9. Daganzo, C.F.: Dynamic blocking for railyards: Part II. heterogeneous traffic. Transportation Research **21B**(1) (1987) 29–40
10. Dahlhaus, E., Horák, P., Miller, M., Ryan, J.F.: The train marshalling problem. Discrete Applied Mathematics **103**(1-3) (2000) 41–54

11. Dahlhaus, E., Manne, F., Miller, M., Ryan, J.: Algorithms for combinatorial problems related to train marshalling. In: Proc. of the 11th Australasian Workshop on Combinatorial Algorithms (AWOCA-00). (2000) 7–16
12. Hansmann, R.S., Zimmermann, U.T.: Optimal sorting of rolling stock at hump yards. In: Mathematics - Key Technology for the Future: Joint Projects Between Universities and Industry. Springer (2007)
13. Holliger, H.P.: Rangierbahnhof Limmattal. Personal communication (2007)
14. Festa, P., Pardalos, P.M., Resende, M.G.C.: Feedback set problems. In: Handbook of Combinatorial Optimization. Volume 4. Kluwer Academic Publishers (1999)
15. Newman, A.: The maximum acyclic subgraph problem and degree-3 graphs. In: Proceedings of the 4th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX. LNCS (2001) 147–158
16. Even, G., Naor, J., Schieber, B., Sudan, M.: Approximating minimum feedback sets and multi-cuts in directed graphs. In: Proceedings of the 4th International Conference on Integer Programming and Combinatorial Optimization. LNCS (1995) 14–28
17. Cordeau, J.F., Toth, P., Vigo, D.: A survey of optimization models for train routing and scheduling. Transportation Science **32**(4) (1998) 380–404
18. Campetella, M., Lulli, G., Pietropaoli, U., Ricciardi, N.: Freight service design for the italian railways company. In Jacob, R., Müller-Hannemann, M., eds.: ATMOS 2006 - 6th Workshop on Algorithmic Methods and Models for Optimization of Railways, IBFI, Schloss Dagstuhl, Germany (2006) <http://drops.dagstuhl.de/opus/volltexte/2006/685>.
19. Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman (1979)
20. Jacob, R.: On shunting over a hump. Technical Report 576, Institute of Theoretical Computer Science, ETH Zürich (2007)