An Improved Train Classification Procedure for the Hump Yard Lausanne Triage[⋆]

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

Department of Transportation Networks, University of Žilina, Slovakia
marton@frdsa.fri.uniza.sk
Institute of Theoretical Computer Science, ETH Zürich, Switzerland
{jens.maue|marc.nunkesser}@inf.ethz.ch

Abstract. In this paper we combine an integer programming approach and a computer simulation tool to successfully develop and verify an improved classification schedule for a real-world train classification instance. First, we derive an integer program for computing train classification schedules based on an earlier developed bitstring representation of such schedules. We show how to incorporate various practical restrictions in this model. Secondly, we apply the model to one day of traffic data of the Swiss classification yard Lausanne Triage. We incorporate all the operational and infrastructural restrictions of this yard instance in our integer program. Even with this high number of restrictions, we are able to compute a schedule that saves a full sorting step and one track compared to the currently applied procedure. We finally show this improved schedule is applicable in practice by a thorough computer simulation.

Keywords. train classification, shunting of rolling stock, simulation tools for transport operations, infrastructure planning, freight trains

1 Introduction

Classification yards are an important unit of freight train systems, and several technical and methodological innovations have improved their operation since their first construction in the 19th century. Many improvements concerning train classification methods were developed in the 1950s and 1960s, and the resulting methods can be divided in single-stage and multistage sorting. Single-stage sorting is applied to large-volume traffic with only basic sorting requirements, while multistage sorting is used for traffic with lower volume but finer sorting requirements. In this paper we focus on multistage sorting.

Even though there are recent theoretical considerations that guarantee good classification procedures, it is still common practice to apply the traditional multistage methods of the 1950s and 1960s today. In order to support transforming

[⋆] Partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP) under contract no. FP6-021235-2 (project ARRIVAL) and by the Slovak grant foundation under grant no. 1-4057-07 (project "Agent Oriented Models of Service Systems").

J. Clausen, G. Di Stefano (Eds): ATMOS 2009
9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2009/2142

the mentioned theoretical results from the academic environment to the application in practice, we introduce a framework for computing classification schedules for real-world problem instances according to the recent theoretical findings. This approach is mainly based on the knowledge of the input for the classification instance. As soon as the order of incoming cars is known, we are able to compute classification schedules that are superior to the established methods with regard to the number of required sorting steps. This number essentially determines the time required to accomplish a classification task. In contrast to the traditional methods, this method considers ordered subsequences of cars in inbound trains when computing schedules. Since in practice trains show a high degree of presortedness, this approach has a high potential to yield shorter schedules than the established methods in many cases. Conversely, our integer programming approach never yields a longer schedule than the established methods; for instances for which an established method does provide an optimal schedule, our method will find a schedule of the same length.

Outline In Sect. 2 we explain the basics of classification yards and multistage sorting, followed by the related work in this field in Sect. 3. Section 4 revises an encoding of classification schedules from [1], which is used in Sect. 5 to introduce an integer programming model for deriving classification schedules. We then apply our model to effectively derive an improved schedule for the classification yard Lausanne Triage in Sect. 6, which we prove to be applicable in practice by a successful computer simulation. Some final remarks follow in Sect. 7.

2 Hump Yards, Multistage Sorting, and Terminology

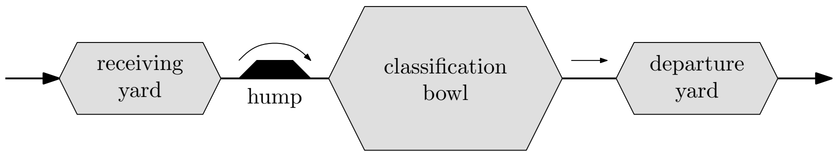

Fig. 1: Typical yard with receiving and departure yard, hump, and classification bowl.

The typical layout of a hump yard, shown in Fig. 1, 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. The yard features a *hump*, a rise in the ground, with a *hump track* from which cars roll in to the tracks of the classification bowl. A typical classification bowl is shown in Fig. 2b. Not all yards have receiving and departure tracks, some have a single end classification bowl as in Fig. 2a, while others have a secondary hump at their opposite end as in Fig. 2c or two parallel hump tracks on one side. Our example of Lausanne Triage is a double-ended hump yard with two parallel hump tracks and no departure yard. Further details are given in Sect. 5 and 6.1. Almost all modern yards built after the 1960s contain the layout of Fig. 2a as a core substructure, in which multistage sorting can be performed as explained in the following paragraph.

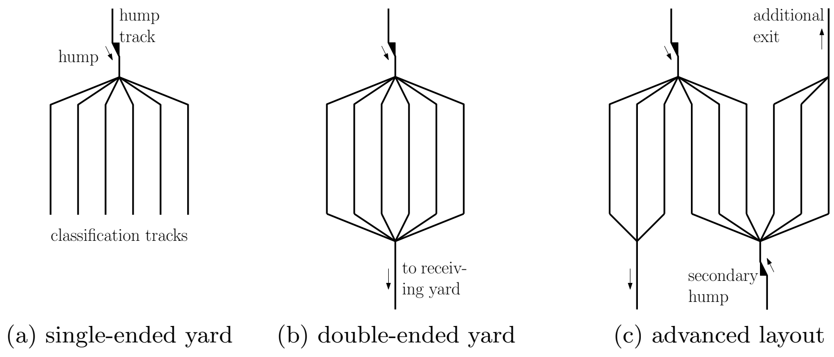

Fig. 2: Common variants of classification bowl layouts.

The following abstract model is a simplification of the actual classification process. Note that this simplification does not impair our results. Every multistage sorting method consists of a sequence of alternating *roll-in* and *pull-out* operations. In a roll-in operation a shunting engine slowly pushes the decoupled cars from the hump track over the hump. The cars roll through a tree of switches, and every car is guided separately to a preassigned classification track. To fully specify a roll-in operation, it suffices to specify the target track for each car. In a pull-out operation an engine drives to some classification track, is coupled to the cars on that track, and pulls back the cars over the hump so that the next roll-in can be performed. A single pull-out can be sufficiently specified by the classification track to pull out cars from. A pull-out followed by a roll-in is called *sorting step* or simply *step*, and an initial roll-in followed by a sequence of h sorting steps is called a *classification schedule of length h*. There is a number of *inbound trains* in the order implied by their arrival times at the yard. This order yields an *inbound train sequence*. Furthermore, there are m order specifications for *outbound trains*. The inbound train sequence has to be sorted on the classification tracks accordingly in order to obtain each of the m outbound trains on a separate track. A classification schedule is called *valid* if applying it accomplishes this sorting task, i.e., if applied to an inbound train sequence, it yields the correctly ordered outbound trains, each on a separate track.

Pulling out a track roughly takes a constant amount of time c_{pull} depending on the distance for the engineer to drive. The time to roll-in the cars in a single hump step is proportional to the number of cars and depends on the time c_{push} required for decoupling and pushing one car, which is roughly constant. Together, a classification process of h steps and a total of r cars rolled in approximately requires a time of $hc_{\text{pull}} + rc_{\text{push}}$. Our main objective is to minimize the number of steps, i.e., the length h of the schedule, which is the approach also taken in [1]. The total number of roll-ins r presents our secondary objective. A more detailed overview of classification yards and their technical implementation is given in [2].

3 Related Work

Multistage classification methods are presented in a number of publications from the 1950s and 1960s in the field of railway engineering [3–10]. Krell [8] compares the two multistage classification methods of *sorting by train* and the often superior *simultaneous method*, as well as two variants: *triangular sorting* and *geometric sorting*. Some of these methods appear in earlier publications of Flandorffer [3] and Pentinga [7]. Boot [4] describes the operational constraints of the simultaneous method in France, Belgium, and The Netherlands. The real-world implementation of the methods with respect to different yard layouts and arrival and departure times of trains is discussed in [9] and [10]. For the Swiss classification yard Zürich Limmattal, Baumann [6] explains the design aspects that make the simultaneous method applicable there. There are more recent descriptions of multistage methods in the papers of Siddiquee [11] and Daganzo et al. [12, 13].

In the 2000s Dahlhaus et al. study a variant of multistage sorting [14] from a more theoretical point of view. They also give a systematic framework for order requirements of outbound trains. These sorting requirements are summarized in [15], which provides a framework for classifying a wide range of single- and multistage methods. There are various shunting problems related to multistage train classification, such as single-stage sorting [12, 14, 16], train matching [16], and blocking and block-to-train assignment [17]. In practice these problems interact with multistage sorting as the practical solution of one problem yields restrictions and simplifications for the other. Further overviews of shunting problems with theoretical focus are given by DiStefano et al. [16] and Gatto et al. [18].

The theoretical concept of *recoverable robustness* [19] is applied by Cicerone et al. [20, 21] to multistage sorting. They regard small deviations in the inbound train and yard infrastructure and three basic recovery strategies, which is an interesting first step towards robustness in train classification.

Computer simulations are a useful tool for evaluating and refining classification methods before applying them in practice. Several such simulations have been performed recently to verify planned modifications of yards or changes in operation for yards in Germany [22], Slovakia [23], and Switzerland [24]. For our computer simulation presented in Sect. 6.3, we used the simulation system "Villon" [25] to verify our schedule.

4 Encoding Classification Schedules

In this section we present the encoding for classification schedules that was derived in [1]. Based on this encoding, we introduce a new integer programming model in Sect. 5, which we apply to a practical classification problem in Sect. 6.

4.1 Model and Notation

We consider the yard layout of a single-ended classification bowl with a single hump as depicted in Fig. 2a. (The same classification procedure can also be applied on double-ended yards such as Lausanne Triage. Moreover, Lausanne Triage has two parallel hump tracks, a setting to which the encoding is adapted in Sect. 5.2.) The number of classification tracks is called the *width* of the yard and denoted by W, the classification tracks are referred to by $\theta_0, \dots, \theta_{W-1}$. The maximum number of cars C that fit on any classification track is called the *capacity* of the tracks.

Every car τ is represented by some positive integer $\tau \in \mathbb{N}$, and a train T is defined as an ordered sequence $T = (\tau_1, \dots, \tau_k)$ of cars $\tau_i \in \mathbb{N}$, $i = 1, \dots, k$. The number k of cars of T is referred to by the *length* of T. There is an ordered sequence of inbound trains, the concatenation of which (according to their arrival at the yard) yields an ordered sequence of cars, called the *inbound sequence of cars.* The order of cars in the inbound sequence is a permutation $T = (\tau_1, \dots, \tau_n)$ of $(1, \dots, n)$, where n is the total volume of cars. Moreover, there are m order specifications for the m *outbound trains.* If n_i denotes the length of the ith outbound train, $i = 1, \dots, m$, then $\sum_{i=1}^{m} n_i = n$. We further assume, w.l.o.g., that the specification of the first outbound train is given by $(1, \dots, n_1)$, the second by $(n_1 + 1, \dots, n_1 + n_2)$, etc., and the last by $(n - n_m + 1, \dots, n)$. During the classification process the cars of different outbound trains are sorted simultaneously on the same set of tracks, called *sorting tracks*, whereas each outbound train is finally formed on an individual track. Those tracks are called *destination tracks.* Our optimization problem can now be defined as follows: Given an inbound sequence of cars $T = (\tau_1, \dots, \tau_n)$ and m outbound trains defined by their lengths $(n_1, \dots, n_m)$, find a valid classification schedule of minimum length.

4.2 Bitstring Representation of Classification Schedules

A track may be filled several times during a classification procedure by sending cars to it after it has been pulled out. We call the track pulled out in the ith step the ith *logical track.* For a classification schedule of length h, we map the h logical tracks to the W physical tracks, obtaining a sequence $(\theta_{i_0}, \dots, \theta_{i_{h-1}})$ of h tracks, where θ_{i_k}, $k = 0, \dots, h-1$, is the physical track pulled out in the kth sorting step. As shown in [1], for tracks of unbounded capacity, there always is an optimal schedule whose track sequence $(\theta_{i_0}, \dots, \theta_{i_{h-1}})$ satisfies $k \equiv i_k \pmod W$ for every $k = 0, \dots, h-1$; in other words, there is an optimal schedule in which the tracks are pulled out in a round robin order. The proof given in [1] still holds for tracks of limited but uniform capacity C, which we consider in this paper.

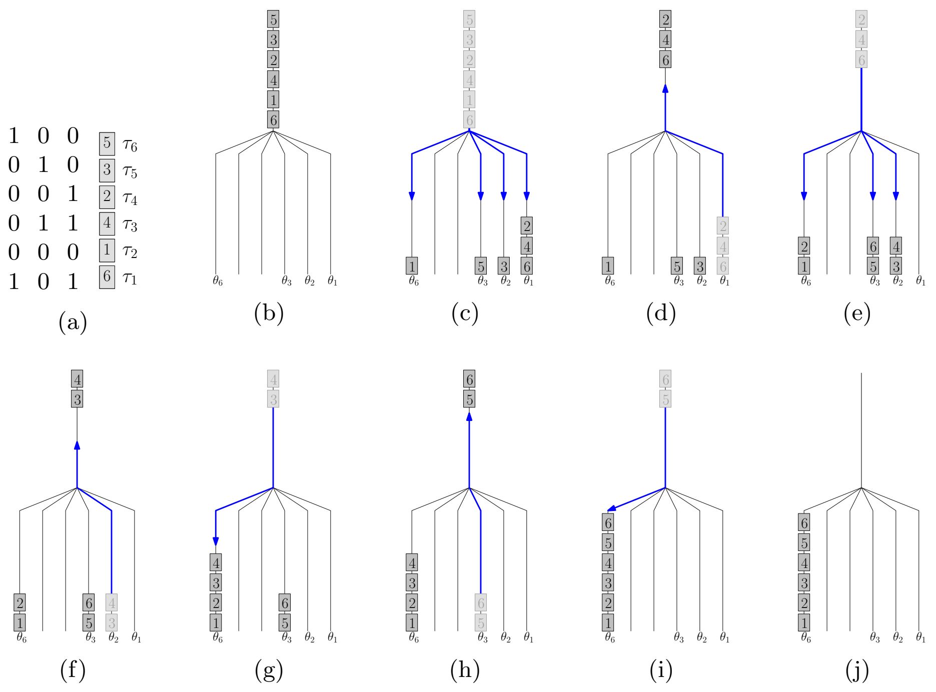

Fig. 3: A classification procedure for $h = 4$ and $n = 6$, using track θ_6 for the only outbound train. The encoding is shown in (a), the inbound sequence of cars in (b). (c)–(j) show the consecutive situations during the procedure, always pulling out the cars of the rightmost occupied track.

For any classification schedule of h steps, the course of any car j can be represented by a binary string $b^j = b^j_{h-1} \dots b^j_0$ with $b^j_k = 1$ iff the jth car visits track θ_{i_k} pulled out in the kth sorting step, $k = 0, \dots, h-1$. After the kth pull-out operation, this car is rolled in to track θ_{i_ℓ} with $\ell = \min\{k < i \leq h-1 \mid b^j_i = 1\}$. If there is no bit $b^j_i = 1$, $k < i \leq h-1$, the car is rolled in to the destination track of its outbound train. In this way, every classification schedule of length h can be represented by an assignment of cars to bitstrings of length h. Figure 3 illustrates this representation in an example with a single outbound train.

Conversely, the bitstring encoding can be applied in order to derive a feasible schedule. First, if two cars with consecutive indices j and $j+1$ of the same outbound train appear correctly ordered already in the inbound train sequence, they may be assigned the same bitstring; then, both cars take exactly the same journey over the tracks during the classification, so they never change their relative order and end up in their correct relative order in the outbound train. Second, assume two consecutive cars j and $j+1$ of the same outbound train occur in reversed relative order in the inbound sequence. Then, the bitstring b^{j+1} assigned to $j{+}1$, regarded as the binary representation of the integer $\sum_{i=0}^{h-1} 2^i b^{j+1}_i$, must be strictly greater than the bitstring b^j assigned to j. To see this, let $b^{j+1} > b^j$ and k be the most significant (i.e. largest) index with $b^j_k = 0$ and $b^{j+1}_k = 1$. After being pulled out from track θ_{i_k}, car $j+1$ is sent to some track θ_{i_ℓ}, $\ell > k$, which car j has been sent to in some earlier step. (Note that θ_{i_ℓ} might be the destination track.) Thus, the two cars appear correctly ordered on this track. Since they never swap their relative order at any later stage of the classification, they arrive correctly ordered on the destination track of their outbound train. By the same argument, if two consecutive cars j and $j+1$ occurs in correct relative order in the inbound sequence, assigning b^{j+1} to $j+1$ and b^j to j is fine if $b^j < b^{j+1}$.

This insight yields a necessary ordering condition for a feasible assignment of cars to bitstrings, which is independent of the number or capacity of classification tracks. This condition presents the most basic constraint of our integer programming model introduced in the following section.

5 Deriving Schedules by Integer Programming

In this section, we present the integer programming model we apply in Sect. 6 to successfully derive an improved schedule for a day of traffic in Lausanne Triage. (Part of this model can be found in the ARRIVAL technical report [26].) We start with the most basic version of this model in Sect. 5.1 and refine the model successively from Sect. 5.2 to Sect. 5.4, incorporating all the required practical constraints. Some constraints are specific for Lausanne Triage only, some apply to other classification yards too.

5.1 Basic IP Model

The integer programming model applies the binary encoding of classification schedules introduced in [1] and explained in Sect. 4. In the basic model below, we enforce an assignment that yields the correctly ordered outbound trains. Note this is the only constraint for completely unrestricted schedules, particularly without any restriction on the number and capacity of tracks. Secondly, the basic model implements limited track capacities.

We introduce binary variables b^j_i, $j = 1, \dots, n$, $i = 0, \dots, h-1$, corresponding to the jth car in the ith sorting step. (We repeatedly introduce binary variables in the following sections without repeating the binary constraint in the actual formulation for space requirements.) The set of indices of cars that are the first of their respective outgoing trains is denoted by $F \subseteq \{1, \dots, n\}$. Let further $\mathrm{rev}(i, j)$ be an indicator function with $\mathrm{rev}(i, j) = 1$ iff the ith and jth car appear in reversed order in the incoming train sequence. Recall that C denotes the maximum number of cars fitting on a track.

$$\textbf{base:}\ \min \sum_{\substack{1 \le j \le n \\ 0 \le i < h}} b^j_i$$

$$\text{s.t.} \quad \sum_{0\leq i<h} 2^i b_i^j \geq \text{rev}(j,j-1) + \sum_{0\leq i<h} 2^i b_i^{j-1} \quad \forall j \in \{1,\dots,n\} \setminus F \qquad (1)$$

$$\sum_{1\leq j\leq n} b_i^j \leq C \qquad \forall i \in \{0,\dots,h-1\} \qquad (2)$$

The objective function in this model minimizes the total number of cars rolled in during the classification process, which presents our secondary objective as mentioned in Sect. 2. In order to minimize our primary objective, i.e. the number of sorting steps, we solve a short sequence of integer programs with increasing length values h. Constraints (1) enforce a valid schedule w.r.t. the ordering of cars in the outbound trains: If two consecutive cars $j-1$ and j of an outbound train are in correct order, they may be assigned the same bitstring; otherwise, $\text{rev}(j-1,j) = 1$, so j will get a strictly larger bitstring than $j-1$ as required according to Sect. 4.2. Constraints 2 implements the restricted capacity of the classification tracks.

5.2 Parallel Classification Procedures

As mentioned before, the classification yard Lausanne Triage features two parallel hump tracks. For the simultaneous method, this means that we can apply two classification procedures in parallel. The two procedures work as two independent systems: there is one shunting engine in either system, and each available classification track is used by only one procedure; furthermore, every outbound train is assigned to exactly one of the systems and remains in that system from its first roll-in until its outbound train is formed. We refer to the two systems of Lausanne Triage by *north partition* and *south partition*.

The assignment of trains to partitions is part of the optimization process. We add binary variables s_i, $i = 1,\dots,m$, with $s_i = 1$ iff the ith outbound train is a member of the north partition. For the sake of comparability, however, we fixed eight out of 24 variables s_i in our test instance as further explained in Sect. 6.2. We further double the binary variables b_i^j into two sets: $\hat{b}_i^j$ for the schedule corresponding to the north and $\check{b}_i^j$ for that of the south partition. In the resulting model, we perform h sorting steps in each partition. Let $t(j)$, $j \in \{1,\dots,n\}$, denote the outbound train of the jth car.

$$\min \sum_{\substack{1\leq j\leq n\\ 0\leq i<h}} \left(\hat{b}_i^j + \check{b}_i^j\right)$$

$$\text{s.t.} \quad \sum_{0\leq i<h} 2^i \hat{b}_i^j \geq \text{rev}\,(j,j-1) - \left(1-s_{t(j)}\right) + \sum_{0\leq i<h} 2^i \hat{b}_i^{j-1} \quad \forall j \in \{1,\dots,n\} \setminus F \qquad (3)$$

$$\sum_{0\leq i<h} 2^i \check{b}_i^j \geq \text{rev}\,(j,j-1) - s_{t(j)} + \sum_{0\leq i<h} 2^i \check{b}_i^{j-1} \quad \forall j \in \{1,\dots,n\} \setminus F \qquad (4)$$

$$\sum_{1\leq j\leq n} \hat{b}_i^j \leq C, \ \sum_{1\leq j\leq n} \check{b}_i^j \leq C \qquad \forall i \in \{0,\dots,h-1\} \qquad (5)$$

Note that with this approach the jth car has *two* bitstrings $\hat{b}_j$ and $\check{b}_j$, one for each partition. Consider two consecutive cars j and $j-1$ of the same outbound train x that appear in reversed order in the inbound sequence of cars. If x is assigned to the north partition, i.e. $s(x) = 1$, then $1 - s_{t(j)} = 0$ and Constraints (3) corresponds to Constraints (1). In this case, the values of $\check{b}_j$ and $\check{b}_{j-1}$ have no meaning. Note that Constraints (4) are satisfied if both $\check{b}_j = 0$ and $\check{b}_{j-1} = 0$ independently of the value of $\mathrm{rev}(j, j-1)$. By the objective function, an optimal solution will satisfy $\check{b}_j = 0$ and $\check{b}_{j-1} = 0$ and its objective value actually equals the total number of cars rolled in. A similar argument applies for $s(x) = 0$.

5.3 Available Classification Tracks

In the classification yard Lausanne Triage, the multistage method for classifying multidestination freight trains is carried out in two stages. First, the trains are collected on a number W of reserved classification tracks, while all other tracks are used for other shunting activities such as single-stage sorting. This first stage corresponds to the initial roll-in of every car (see Sect. 4.1). This constraint is modeled as follows, where $W = \hat{W} + \check{W}$ with $\hat{W}$ and $\check{W}$ being the numbers of tracks corresponding to the north and south system, respectively:

$$\textbf{initial roll-in:} \quad \sum_{0 \le i < \hat{W}} \hat{b}_i^j \ge s_{t(j)} \qquad \forall j \in \{1, \ldots, n\} \tag{6}$$

$$\sum_{0 \le i < \check{W}} \check{b}_i^j \ge 1 - s_{t(j)} \quad \forall j \in \{1, \ldots, n\} \tag{7}$$

Note that for the special case of $h = \hat{W} = \check{W}$, which holds for our solution for the sample instance of Sect. 6, this simply means that the all-zero bitstring is disallowed for every car; in other words, cars may not be sent to destination tracks initially. Note that Constraints (6) and (7) do not implement the limited number of tracks mentioned in Sect. 4.1 in full generality. In the improved schedule of Sect. 6, we do not pull out any track twice, so Constraints (6) and (7) suffice here.

In the second stage, these tracks are pulled out to build outgoing trains, which is usually performed during the night when more than the W reserved tracks are available for multistage sorting. There might be more and more tracks available after every sorting step, so forming more and more outgoing trains can be started. In the integer program, we introduce binary variables $\hat{u}_{x,t}$ and $\check{u}_{x,t}$, $x = 1, \ldots, m$, $t = 0, \ldots, h$, that indicate whether forming the xth outgoing train has started yet at time step t in the north or south partition, respectively.

$$\textbf{train formation:} \quad \sum_{j \in F} \hat{u}_{t(j),t} \le \hat{N}_t \qquad \forall t \in \{0, \ldots, h-1\} \tag{8}$$

$$\sum_{j \in F} \check{u}_{t(j),t} \le \check{N}_t \qquad \forall t \in \{0, \ldots, h-1\} \tag{9}$$

$$\hat{u}_{j,t} \geq s_{t(j)} - \sum_{t \leq i < h} \hat{b}_i^j \qquad \forall j \in F, t \in \{0,\dots,h\} \qquad (10)$$

$$\check{u}_{j,t} \geq 1 - s_{t(j)} - \sum_{t \leq i < h} \check{b}_i^j \qquad \forall j \in F, t \in \{0,\dots,h\} \qquad (11)$$

After every step t, the number of outgoing trains that have started to be formed must not exceed the available number $\hat{N}_t$ or $\check{N}_t$ of tracks, respectively, at this time. This is implemented by Constraints (8) and (9). Constraints (10) and (11) make sure each variable $u_{j,t}$ is actually set if forming the train of the jth car has been started at the tth step.

5.4 Train Departure Times

If an outbound train is finished, it will not wait until the whole classification process is finished but leaves the yard if the traffic on the railway line allows. Some outbound trains even have to depart early to meet the point of time they are expected to arrive at their destinations, and we have to consider these latest-possible departure times in the classification process. We introduce an upper bound on the time it takes to perform one sorting step, which we chose to be 30 minutes for our example of Lausanne Triage. In this way, we obtain the latest sorting step acc_x in which a train x can still receive cars.

$$\textbf{accumulation finish:} \quad \sum_{\mathrm{acc}_{t(j)} \leq i < h} \left(\hat{b}_i^j + \check{b}_i^j\right) = 0 \qquad \forall j \in \{1,\dots,n\} \qquad (12)$$

In the following section, we use this model to derive a schedule for a real-world classification task, to which we have to apply all the Constraints (3) to (12).

6 Case Study: Lausanne Triage

We apply the model of the previous section to real-world traffic data in this section. The problem instance is illustrated in Sect. 6.1, the schedule computation is described in Sect. 6.2, and its successful simulation in Sect. 6.3.

6.1 Classification Yard Lausanne Triage

The train classification yard of Lausanne features a receiving yard, a classification bowl (see Fig. 4) of 38 tracks with two parallel hump tracks, and no departure yard. Regarding the operation, there are ten tracks reserved for forming multidestination freight trains, on which all cars for the multistage method are initially collected. As mentioned in Sect. 5.3, the remaining tracks are needed for other shunting activities. These activities are stopped at some point in the early morning, from which time the humps are exclusively used for multistage sorting. Still, not all multidestination freight trains can start to be formed right after the first pull-out since there are still not enough tracks, but more and more tracks are available after each step as mentioned in Sect. 5.3.

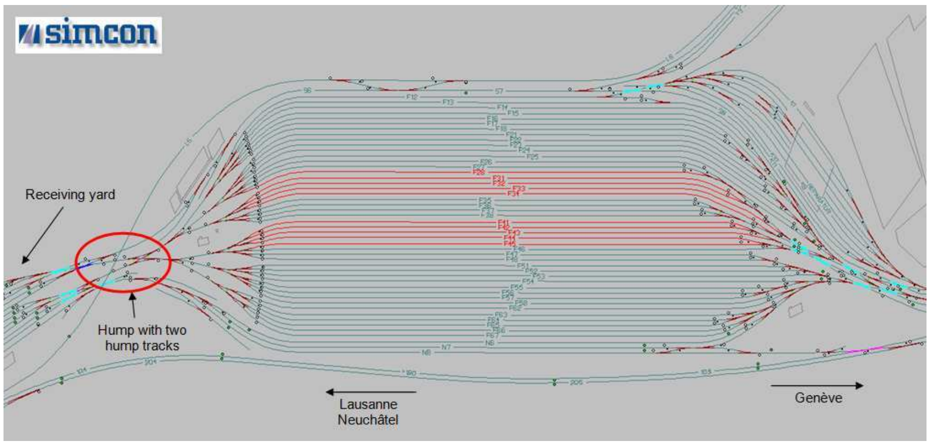

Fig. 4: The classification bowl of Lausanne Triage with ten tracks for multistage sorting.

Our problem instance comprises all the cars of a complete day in 2005, which amount to 1'346. For the multistage method there are 452 cars for 22 outbound trains with between two and seven destinations and two outbound trains with one destination. We extracted 331 cars for which we computed the schedule. The remaining 121 cars of the multistage method were not included in the schedule computation since they receive a special treatment as explained in Sect. 6.2 below.

6.2 Schedule Computation

All IP computations were done with ILOG OPL Studio 3.7 featuring CPLEX 9.0 on an Intel Xeon CPU with 2.80 GHz and 2 GB main memory running Linux.

The schedule originally applied to the above described classification instance in 2005 comprised five steps in each partition, which corresponds to $h = 5$ in the model of Sect. 5.2. Setting the values for C, $\hat{N}_t$, $\check{N}_t$, and acc_x according to the practical requirements, the problem turns out to be infeasible for putting $h = 4$. However, with five steps in the north and only four steps in the south partition, we obtain a feasible schedule. This is implemented by putting $h = 5$ and additionally requiring $\check{b}_i^j = 0$ for $i = 4$ and all cars $j \in \{1, \ldots, n\}$. Computing this schedule took 5.75 hours including the proof of optimality.

As mentioned above, there are 121 cars which we did not consider in the schedule computation. These cars belong to destinations for which there is a very big number of cars. In the original schedule, these cars were not rolled in to the ten classification tracks for multistage sorting but directly sent to their respective destination tracks. Except for one case, for which some extra shunting must be done, these destinations are at the very front of their respective outbound trains, so the classification process is not impaired by this practice. In this way, the cars of the huge destinations did not have to be sent over the hump a second time. For the sake of an easier comparison, we took the same approach: in order not to interfere with the operation of shunting activities other than multistage sorting, we chose the same tracks for the large destinations; this includes a fixed assignment to the north or south partition for the affected outbound trains by forcing $s_i = 0$ or $s_i = 1$, respectively. Our improvement was achieved with this additional constraint.

We also tried to compute a schedule with $h = 5$ steps in each partition and $\hat{W} = \check{W} = 4$, i.e. a schedule in which the first track of either partition is pulled twice. This would save even two classification tracks by revoking the saved sorting step from above, but there is no feasible solution for this combination.

6.3 Simulation and Results

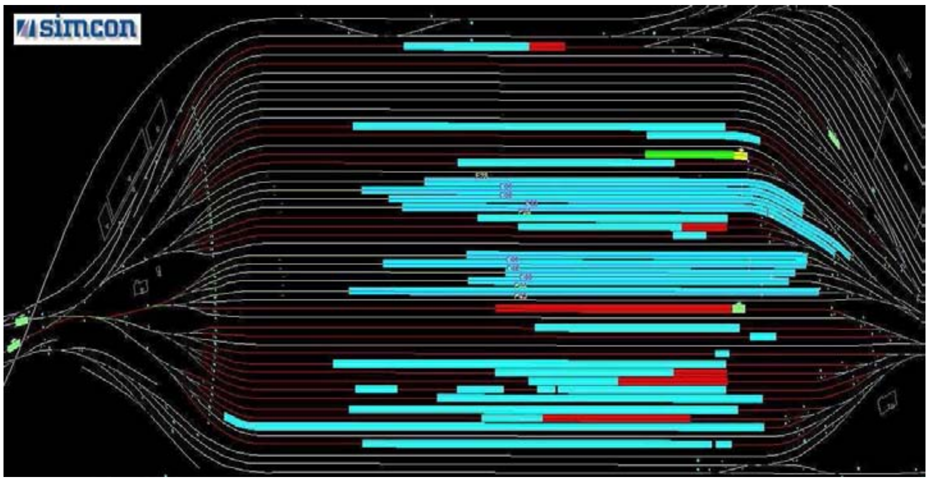

Fig. 5: Situation of the cars on the classification tracks after the initial roll-in for the improved schedule. North is at the bottom of the picture.

We simulated the above described schedule using the simulation system "Villon" [25]. First of all, the above described schedule did not produce any conflicts when our computer simulation was run on it, which basically means, with regard to the technical implementation, that the schedule works in practice.

The total number of cars rolled in during the complete improved classification procedure amounts to 1'700, compared to 1'706 in the original schedule, which is only a marginal saving. Nevertheless, the theoretical considerations on multistage sorting in [1] shows that increasing the number h of steps in the multistage method over the optimum value generally allows decreasing the total number r of cars rolled in and vice versa. Even though the experiments of [26] suggest only a mild rise of r for decrementing h, our schedule does not yield any increase at all. Therefore, the marginal reduction of r by six is a great success since we do not have to pay for the reduced number of sorting steps with more roll-ins compared to the original schedule. This finding also underlines the suboptimality of the schedule originally applied.

The number of settings of switches for our schedule amounts to 789 compared to 914 for the old schedule, which is a considerable saving of 125 settings or 13.7 %. This significantly reduces the wear of the switches and saves maintenance, which is further contributed to by only 1'481 movements of cuts over switches. (A *cut* is a small set of coupled cars—if consecutive cars on the hump track are about to be rolled in to the same track, they will not be decoupled.) Compared to 1'691 for the original schedule, this is a saving of 210 cuts or 12.4 %.

The main improvement, however, consists in saving one full sorting step: in the original procedure the track labeled "F28" in Fig. 5 contained the cars that were pulled out in the fifth sorting step of the south partition. In the improved procedure this track is empty after the initial roll-in, and is now available to be used for various purposes. the original procedure comprised five sorting steps in the south partition, whereas our improved procedure only performs four steps. The track made available by saving the fifth step can be used, for example, for multistage sorting in order to increase the upper limit of traffic with a higher attractiveness for this method through an increased potential traffic volume. The track may also be used for other shunting activities, such as building very long trains with no order restriction by collecting their cars on several classification tracks before coupling them into one train.

7 Conclusion and Future Work

The results of this paper demonstrate the power of the classification schedule encoding established in [1]. We have effectively applied this encoding to obtain a highly flexible integer programming model for train classification that allows incorporating various practical restrictions, which underlines the applicability in practice. As the main result, we are able to derive a schedule for real-world traffic data of the example classification yard Lausanne Triage that outperforms the current schedule by one sorting step. Implementing this schedule in practice would yield a more efficient sorting process with less engine movement and a significantly reduced wear of switches. Most importantly, the improved schedule makes an additional classification track available. This raises a potential for more traffic for the multistage method itself or other shunting methods applied in parallel, such as single-stage sorting.

For Lausanne Triage dropping the fixed assignment of some trains to partitions mentioned in Sect. 6.2 may yield an even better schedule with higher savings. Beyond that, it would also be interesting to derive and simulate more schedules for further real-world data. In particular, there are larger classification yards than Lausanne Triage with higher volumes of traffic for multistage sorting. For such yards an even higher improvement can be expected, so an application to yards with a higher traffic volume and more sorting steps and tracks appears promising.

The commonly applied classification methods triangular and geometric sorting yield correctly ordered outbound trains regardless of the order of inbound trains [18]. Such methods are called strictly robust. However, only a fraction of trains is actually delayed in practice, so providing strict robustness wastes a lot of potential as the results of this paper show. As mentioned before, our improvement is based on complete knowledge of the order of inbound cars. Since trains may be delayed, the actual order may differ from the scheduled order, and the optimal classification schedule for the expected order cannot be applied anymore. This dilemma can be tackled by regarding realistic scenarios of delay and providing optimal robust solutions w.r.t. a limited amount of recovery in case of disturbance [19, 20]. This approach balances between strictly robust and optimal non-robust solutions and may thus yield robust classification methods that still improve on the current practice.

Acknowledgments We would like to thank Michael Gmür and Sigmund Rützler (Rangierbahnhof Limmattal) for their interesting information on yard operation. Thank you very much also to Michael Gatto and Matus Mihalak for the helpful discussions about mathematical programming. Last but not least, many thanks to Stephan Leber and SBB Infrastruktur for making available their valuable traffic data to us.

References

1. Jacob, R., Márton, P., Maue, J., Nunkesser, M.: Multistage methods for freight train classification. NETWORKS—Special Issue: Optimization in Scheduled Transportation Networks (2009)
2. Kumar, S. In: Improvement of Railroad Yard Operations. McGraw-Hill (2004) 25.1–25.28
3. Flandorffer, H.: Vereinfachte Güterzugbildung. ETR RT **13** (1953) 114–118
4. Boot, B.C.M.: Zugbildung in Holland. ETR RT **17** (1957) 28–32
5. Keckeisen, W.: Bau und Betrieb der Stuttgarter Hafenbahn. ETR **7**(10) (1958) 408–420
6. Baumann, O.: Die Planung der Simultanformation von Nahgüterzügen für den Rangierbahnhof Zürich-Limmattal. ETR RT **19** (1959) 25–35
7. Pentinga, K.J.: Teaching simultaneous marshalling. The Railway Gazette (1959)
8. Krell, K.: Grundgedanken des Simultanverfahrens. ETR RT **22** (1962) 15–23
9. Krell, K.: Ein Beitrag zur gemeinsamen Nutzung von Nahgüterzügen. ETR RT **23** (1963) 16–25
10. Endmann, K.: Untersuchungen über die Simultanzugbildung bei der Deutschen Bundesbahn. Bundesbahn **37** (1963) 593–600
11. 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
12. Daganzo, C.F., Dowling, R.G., Hall, R.W.: Railroad classification yard throughput: The case of multistage triangular sorting. Transp. Res., Part A **17**(2) (1983) 95–106

13. Daganzo, C.F.: Static blocking at railyards: Sorting implications and track requirements. Transp. Science **20**(3) (1986) 189–199
14. Dahlhaus, E., Horák, P., Miller, M., Ryan, J.F.: The train marshalling problem. Discrete Applied Mathematics **103**(1–3) (2000) 41–54
15. 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)
16. Di Stefano, G., Maue, J., Modelski, M., Navarra, A., Nunkesser, M., van den Broek, J.: Models for rearranging train cars. Technical Report TR-0089, ARRIVAL (2007)
17. Jha, K.C., Ahuja, R.K., Şahin, G.: New approaches for solving the block-to-train assignment problem. NETWORKS **51** (2008) 48–62
18. Gatto, M., Maue, J., Mihalak, M., Widmayer, P.: Shunting for dummies: An introductory algorithmic survey. In: Robust and Online Large-Scale Optimization. LNCS State-of-the-Art. Springer (2009)
19. Liebchen, C., Lübbecke, M., Möhring, R.H., Stiller, S.: Recoverable robustness. Technical Report TR-0066, ARRIVAL (2007)
20. Cicerone, S., D'Angelo, G., Stefano, G.D., Frigioni, D., Navarra, A.: Robust algorithms and price of robustness in shunting problems. In: ATMOS-07, Wadern, Germany, IBFI Schloss Dagstuhl (2007) 175–190
21. Cicerone, S., D'Angelo, G., Di Stefano, G., Frigioni, D., Navarra, A., Schachtebeck, M., Schöbel, A.: Recoverable robustness in shunting and timetabling. Technical Report TR-0190, ARRIVAL (2009)
22. Edinger, M., König, R., Márton, P., Zat'ko, M.: Die rechnergestützte Simulation des Betriebs in Werkbahn BASF Ludwigshafen. In: Railways on the Edge of the 3rd Millennium (ZEL-04). (2004) 161–165
23. Márton, P.: Experimental evaluation of selected methods for multigroup trains formation. Communications **2** (2005) 5–8
24. Zat'ko, M., Leber, S.: Simulation komplexer Betriebsprozesse in einem Rangierbahnhof am Beispiel von Lausanne Triage. Schweizer Eisenbahn-Revue **11** (2006) 9500–9503
25. Adamko, N., Kavička, A., Klima, V.: Villon - Agent based generic simulation model of transportation logistic terminals. In: Proc. of the 2007 European Simulation and Modelling Conference (ESM-07). (2007) 364–368
26. Maue, J., Nunkesser, M.: Evaluation of computational methods for freight train classification schedules. Technical Report TR-0184, ARRIVAL (2009)
27. Jacob, R., Marton, P., Maue, J., Nunkesser, M.: Multistage methods for freight train classification. In: ATMOS-07, IBFI Schloss Dagstuhl (2007) 158–174