Optimisation of simultaneous train formation and car sorting at marshalling yards*

Sara Gestrelius[1], Florian Dahms[2] and Markus Bohlin[1]

[1]SICS Swedish ICT AB
Box 1263, SE-164 29 Kista, Sweden,
{firstname.lastname}@sics.se

[2]RWTH Aachen, Chair of Operations Research
Kackertstrasse 7, 52072 Aachen, Germany
dahms@or.rwth-aachen.de

Abstract
Efficient and correct freight train marshalling is vital for high quality carload freight transportations. During marshalling, it is desirable that cars are sorted according to their individual drop-off locations in the outbound freight trains. Furthermore, practical limitations such as non-uniform and limited track lengths and the arrival and departure times of trains need to be considered. This paper presents a novel optimisation method for freight marshalling scheduling under these circumstances. The method is based on an integer programming formulation that is solved using column generation and branch and price. The approach minimises the number of extra shunting operations that have to be performed, and is evaluated on real-world data from the Hallsberg marshalling yard in Sweden.

Keywords
Shunting, Marshalling, Classification, Optimisation, Blocking, Column Generation

1 Introduction

To maximise the capacity of a carload transportation system a hub and spoke network is often operated. The hubs are marshalling yards, where incoming cars are sorted into new outbound trains. Outbound trains have predetermined service routes, and cars are assigned to trains that pass through their destination. This implies that cars have to be decoupled at intermediate stops, and to facilitate drop-off it is desirable that all the cars that are to be decoupled at a certain location are right at the end of the train when the train reaches this location. For this reason, cars should be sorted according to their drop-off location, into so called *blocks*, during marshalling. Furthermore, practical limitations such as non-uniform and limited track lengths, and arrival and departure times of the inbound and outbound trains, need to be respected. This paper presents a novel method for freight classification planning under these circumstances. It is an extension of the work in Bohlin et al. [5], where column generation is used to find the classification schedule that minimises the number of extra car pull-backs, but where any car order within the outbound trains is assumed to be acceptable.

*This work was funded in part by the Swedish Transport Administration (Trafikverket) under grant TRV 210/29758.

The yard used as a case study is the Hallsberg marshalling yard in Sweden. The Hallsberg yard is the largest marshalling yard in Scandinavia, and its advantageous location makes it an important hub in the Swedish freight transportation network. Like many other marshalling yards, the Hallsberg yard consists of three sub-yards: an arrival yard, a classification yard (also called classification bowl) and a departure yard. Inbound trains arrive to the arrival yard where their cars are decoupled from the line engine and undergo various inspections. The uncoupled cars of the inbound train are then pushed over a hump and roll to various tracks in the classification yard by means of gravity and a switching system. On the classification yard, the cars are sorted into new outbound trains. When all cars of an outbound train have been rolled to a classification track in the correct order the outbound train can be coupled and then either depart from the classification track, or be pulled to the departure yard and depart from there. In Hallsberg an operational demand is that each track in the classification yard may only contain cars of a single outbound train. That is, there must be a bijection between trains and tracks at any point in time. However, due to capacity limitations and sorting requirements it is generally impossible to roll all cars straight to the tracks that have been reserved for their outbound trains. Therefore, some of the tracks in the classification yard are used as a buffer area where cars of different trains may be temporarily stored. These tracks are called *mixing tracks*. The tracks used for building trains are called *train formation tracks*. At given points in time, a *pull-back* operation is performed which allows any subset of cars to be moved from the mixing tracks to the train formation tracks. During a pull-back the cars of a mixing track are coupled, pulled back over the hump by an engine, and then immediately pushed over the hump to once again be distributed on the classification tracks. The later is called a *roll-in* operation. For a more detailed description of the operations in Hallsberg we refer to [6].

Although sorting according to blocks is omitted in Bohlin et al.[5], it is generally included in previous literature on classification planning. Research has been performed on classification yard operations since the 1950s, and reviews of the area are given by Gatto et al. [13] and more recently Boysen et al. [8]. Four basic methods for classification planning are *Sorting by trains*, *Sorting by block*, *Triangular Sorting* and *Geometric Sorting*, all of which include blocking. However, these methods determine the number of tracks and classification steps by the number of outbound trains and their blocks, and do not take arriving car order into consideration. This makes them robust against disruptions in the incoming car order, but also leads to an excessive use of capacity or pull-backs. Gatto et al. [13] provides a good overview of the trade-offs between the different algorithms. Dahlhaus et al. [10, 11] introduce the train marshalling problem and presents a radix sort scheme that exploit the car order in the arriving trains (called *pre-sortedness*), and show that this reduces the number of sorting steps from $\log_W(n)$ to $\log_W(k)$, where W is the number of classification tracks, n the number of arriving cars and k the number of *batches*. A batch is the maximal sequence of cars that are in the correct relative order in the inbound trains. Jacob et al. [14] also develop a methodology that takes pre-sortedness into consideration by representing the classification schedule of each car by a binary encoding, and then using the intrinsic properties of this representation to generate an optimal schedule. Further, they use their representation to encode and analyse the previously mentioned basic sorting algorithms. Beygang et al. [4] provide a lower bound on the objective, and upper and lower bounds on competitiveness for the online version of the problem as well as give an optimal deterministic online algorithm.

The encoding in Jacob et al. [14] is also used to develop an integer programming (IP) model for deriving optimal classification schedules in Maue and Nunkesser [15]. The IP

model incorporates real world constraints such as multiple humps and track capacity, and it can be extended to take departure times into consideration. The authors model the Lausanne Triage Shunting yard, and report that an optimal schedule was found within 3 minutes. Further, this schedule requires one less pull-back and one less track than the planning method currently used in Lausanne. The Hallsberg marshalling yard in Sweden has also been modelled using first mixed integer programming (MIP) and later pure IP by Bohlin et al. [6, 7, 5]. The three articles present more and more efficient MIP and IP formulations, and in the latest paper [5], an optimal classification schedule for five days is found within 13 minutes for all 30 real world test cases. All models presented by Bohlin et al. [6, 7, 5] take the additional constraints imposed by a car booking system into consideration.

The main contribution of this paper is a new model for optimised marshalling planning that respects operational constraints and allows for cars to be sorted into blocks in the outbound trains. Cars are simultaneously sorted by block and train, and car bookings on specific trains are respected, i.e. cars need not depart with the next train going to their destination, but the freight transportation company can define exactly which train they want a certain car to depart with. As the generated schedule minimises the number of car pull-backs while allowing for blocked outbound trains, it contributes to both efficient shunting and efficient freight transportations.

The paper is structured as follows. In Section 2, the mixing problem with car sorting according to blocks is formally defined, and in Section 3 the branch-and-price based column generation approach from Bohlin et al. [5] is presented. We also show how to extend this formulation to allow for sorting by blocks. Section 4 describes the experimental setup and results, and includes an analysis of how sorting by blocks affects the execution time and solution quality. Finally, Section 5 concludes the paper and outlines future research.

2 Problem Definition

We are given a set of classification tracks O, a set of periods P, a set of car units Q, and a set of outbound trains R. We denote by $B(r) \subseteq B$ the blocks belonging to train r, and $Q(b) \subseteq Q$ the set of car units that belong to block b. A *car unit* consists of cars that belong to the same block and arrive with the same inbound train. For each car unit $q \in Q$, we are given its arrival time $t(q)$, i.e., the time when its inbound train is rolled to the classification yard from the arrival yard, its length $s(q)$, and its corresponding outbound train $r(q) \in R$. Further, each car unit $q \in Q$ belongs to a block $b(q) \in B$. The blocks have a natural order in which they must appear in the outbound train based on their geographical location. However, within a block no special ordering of cars is necessary. We denote by $Q(r) \subseteq Q$ the set of car units that belong to train r. Further, each train $r \in R$ has a departure time $t(r)$, i.e., the time when it leaves the classification bowl. The length $s(r)$ of a train r is the sum of the lengths of its cars, $s(r) := \sum_{q \in Q(r)} s(q)$.

For each classification track $o \in O$ we are given its length $s(o)$. Thus, a train r can be formed on track o if and only if $s(r) \leq s(o)$. Let $R(o)$ denote the set of trains that can be formed on track o. At any point in time, a classification track may only contain cars of one outbound train, and each train is formed on exactly one classification track. We say that a train r is *active* for the time interval during which its corresponding classification track is used exclusively for the formation of r. Further, we call the block currently being built on the classification track the *active block*.

We define the strict partial order $\prec$ on the set of outbound trains R such that $r \prec r'$ if and

only if train $r \in R$ can be scheduled directly before train $r' \in R$ on the same track. Whether $r \prec r'$ holds or not depends on the departure times of trains r and r' as well as on technical setup times (e.g., brake inspection) and other car movements in the marshalling yard. This is further explained in Section 2.1. Note that antisymmetry is ensured as no two trains may be formed on one track at the same time.

As stated previously, capacity limitations and sorting requirements normally make it impossible for all cars to be rolled straight to their train formation tracks. Therefore, we are also given a set of mixing tracks where cars of different outbound trains can be temporarily stored. To simplify our model, we treat these tracks as one concatenated track, called *the mixing track*. This simplification is valid as long as enough time is added to pull back all cars on all physical mixing tracks each time the cars on the concatenated mixing track is pulled back.

The mixing track has a given length s^{mix}. A car that is stored on the mixing track is said to be *mixed*. All mixed cars are pulled back to the arrival yard at certain predetermined times, and are then immediately pushed over the hump again so that the cars can be re-distributed on the classification tracks. All cars belonging to currently active blocks will be rolled to their train formation tracks, while the remaining cars are rolled back to the mixing track. We call this operation a pull-back followed by a roll-in. A pull-back operation defines the beginning of a period $p \in P$, and all cars that are sent to the mixing track after the start time $t(p)$ of period p must remain mixed at least until the start of the next period at time $t(p+1)$.

As an objective function, we choose to minimise the number of car pull-backs. There are several reasons for our choice of objective function. For each period during which a car is mixed, it will be subjected to a pull-back and roll-in operation, which takes effort and time, and wears down switches and tracks. Note that since no car can leave the mixing track until the next pull-back is performed, the total length of the mixed cars within a period is at its maximum at the end of the period.

How many cars that need to be mixed in a period depends on which blocks are active, and how long these blocks have been active for. The start time of a block $b \in B(r')$, denoted $i(b, r)$, can be deduced if the blocks of train r', and the train r that precedes train r' on the classification track, are known (see Section 2.1).

Although the car-ordering within an inbound train is assumed to be unknown, the roll-in order of trains, and therefore of car units from different trains, is know. This means that the order of car units on the mixing track is known to some extent, and in some cases we can deduce that car units belonging to different blocks are in the correct order on the mixing track, and should therefore be rolled to the train formation track in the same pull-back to avoid mixing cars for longer than necessary.

Let q_e^b be block b's final car unit to be mixed. This implies that q_e^b is the final car unit to be rolled in before block b's active period starts. Further, assume that block b is immediately followed by block b' in a train. Then all cars of b' that arrives after q_e^b but before $i(b', r)$ can be moved to the train formation track in the same pull-back as q_e^b if this pull-back ends the active period of block b. If the pull-back does not end block b's active period more cars are going to be rolled to block b after the pull-back, and all cars belonging to block b' must be mixed for at least another period. Let's call the set of car units that fulfill these criteria Q_m, i.e. Q_m is the car units that can be rolled to their train formation track in the same pull-back as cars of the previous block. Note that the pull-back that moves a car $q \in Q(b') \cap Q_m$ to the train formation track will be the pull-back that starts off the active period of block b', but that there may be other mixed cars $q \in Q(b') \backslash Q_m$ that have to be mixed for yet another

period as they were rolled in before $t(q_e^b)$. An example of this is shown in Figure 1, where car $q_2^1 = q_e^1$, and cars $q_2^2, q_3^2 \in Q_m$ and can be rolled to the train formation track in pull P_1 while car $q_1^2 \notin Q_m$ and has to wait until pull P_2.

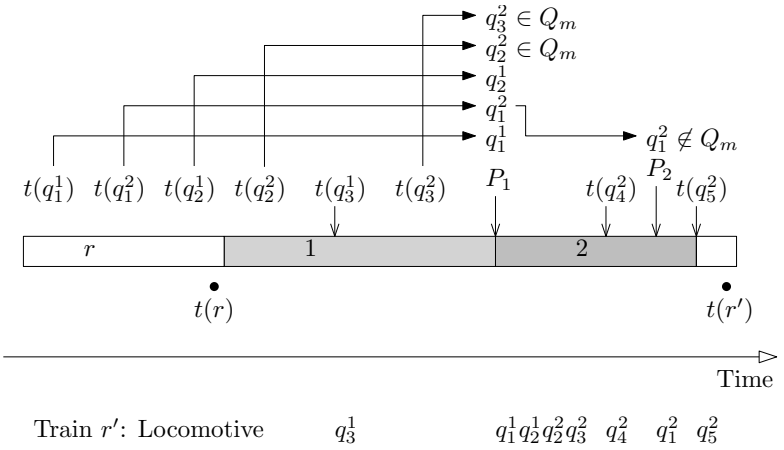

Figure 1: An example of cars of different blocks being in the correct order on the mixing track. Train r' consists of two blocks and train r of one. The boxes represent the active periods of the trains and blocks, and the arrival times of the cars $q_i^b \in r'$ are plotted above the active period boxes. Car units are denoted q_i^b, where b is the block, and i an index to separate the car units from each other. The departure times of the trains are shown by circles underneath the boxes. Pull-backs are represented by P, and the cars that are mixed when a pull-back is executed are printed above the pull-back in the order that they were rolled to the mixing track (so e.g. car q_1^1 will be the first car to be rolled into the classification yard after pull-back P_1). Also, the final car ordering of train r' is shown in the bottom of the picture where the cars have been plotted right underneath the times when they're rolled to the train formation track.

The number of extra car roll-ins needed if train r is followed by train r' on a classification track can now be calculated as follows:

$$c(r, r') = \sum_{q \in Q(r')} c(q, r)$$

where

$$c(q, r) = |q| \cdot |P(q, r)|$$

is the number of extra roll-ins needed for car unit q if its train is shunted after r, and

$$P(q,r) = \begin{cases} \emptyset & \text{if } i(b(q), r) \leq t(q) \\ \{p \in P : t(p) > t(q) \wedge t(p-1) \leq i(b(q), r)\} & \text{elseif } q \notin Q_m \\ \{p \in P : t(p) > t(q) \wedge t(p-1) < i(b(q), r)\} & \text{else} \end{cases}$$

is the set of all periods in which q will be mixed, when its train is shunted after r. $|q|$ is

defined as the number of cars in car unit q. Note that if p is the first pull-back in P, $t(p-1)$ is defined as the very first point in time.

Likewise, the length of cars that need mixing in each period, denoted $s_p(r, r')$, is given by,

$$s_p(r, r') = \sum_{b \in b(r')} s_p(b)$$

where,

$$s_p(b) = \begin{cases} \sum_{q \in Q(b): t(q) < \min(i(b,r), t(p+1))} s(q), & \text{if } t(p) < i(b,r) \\ \sum_{q \in Q(b) \setminus Q_m: t(q) < \min(i(b,r), t(p+1))} s(q), & \text{if } t(p) = i(b,r) \\ 0, & \text{otherwise.} \end{cases}$$

The formulae above calculate the cost of train r' if it's scheduled straight after train r on a classification track. However, in a schedule there will also be a first train on each formation track. In Bohlin et al. [5] all cars belonging to the first train on a track will always be rolled straight to this formation track. However, when sorting cars into blocks some of the cars of this first train may need to be mixed. To calculate the mixing cost of the first train an imaginary train, u, with a departure time at 0.0, is introduced. The number of pull-backs incurred by the first train, $c(u, r)$, can then be calculated using the formula above. Likewise, the mixing length, $s_p(u, r)$ can be calculated using the formula above and the same imaginary predecessor train u.

Deducing active periods of blocks

The start time of a block $b \in B(r')$ where r' is scheduled to follow r on a formation track o is denoted $i(b, r)$, and the end time $t(b, r)$. Also, let a block b be denoted by its natural order in the outbound train, i.e. call the block that needs to be built first on the track 1, the next one 2 etc. But from the first block, the start time of a block will always be the end time of the previous block, i.e. $i(b+1, r) = t(b, r)$. Note that all cars belonging to a block must be rolled to the track *after* the start time of the active period of the block. The only exception to this rule are cars in Q_m, which may be rolled to the formation track in the pull-back that starts the active period of their block. The start time of the first block of a train r will be the departure time of the train preceding r on the formation track. Further, the end time of a block's active period is the time when its final car is rolled to the train formation track. This may be as a result of an initial roll-in (arrival) or a mixing track pull-back and its following roll-in. If a block b requires mixing, its end time is $\max(p_b, l_b)$ where $l_b = t(q_f)$ and $q_f \in Q(b)$ is the car with the latest roll-in time, and p_b is the time of the earliest pull-back after $i(b, r)$, or, if all the cars of the two blocks are in the correct order on the mixing track (i.e. every mixed car belonging to block b exists in the set Q_m), p_b is the time of the pull-back that ends the active period of block $b-1$. In the latter case the active period of block b may be very short, and only consist of the time it takes for its cars to be rolled-in after the pull-back. Blocks that do not require mixing always end with a roll-in, namely $t(q_f)$. Given these simple rules the start and end times of all the active periods of all blocks of train r' following a train r on a formation track can be calculated as described in Algorithm 1. Note that in the final iteration of the algorithm the end time of the final block will be calculated and saved as the start time of a dummy block $b = |B(r')| + 1$.

Algorithm 1: Blocking(r, r'), calculates the start time for all blocks in a train r' given that its immediate predecessor is train r.

input : Two trains r, r'
output : Start times $i(b, r)$ for all blocks $b \in b(r')$
$i(1, r) \leftarrow t(r)$
foreach $b \in b(r')$ in natural block order **do**
  $e_b \leftarrow \min_{q_i \in Q(b)} t(q_i)$
  $l_b \leftarrow \max_{q_i \in Q(b)} t(q_i)$
  $M_b \leftarrow \{q_i | q_i \in Q(b) : t(q_i) \leq i(b, r)\}$
  if $e_b > i(b, r)$ **then**
    $p_b \leftarrow 0$
  else
    if $|M_b| = |M_b \cap Q_m|$ and the active period of b starts with a pull-back **then**
      $p_b \leftarrow i(b, r)$
    else
      $p_b \leftarrow \min_{p_i \in P : i(b,r) < t(p_i)} t(p_i)$
    end
  end
  $i(b + 1, r) \leftarrow \max(p_b, l_b)$
end

2.1 Sequences and Feasible Solutions

We define feasible solutions to our problem in terms of *sequences* of trains, which can be allocated to individual tracks. A sequence g is a totally ordered subset of trains, including the dummy train u which defines the beginning of each sequence. Let us denote the fact that two trains $r, r' \in R$ appear consecutively in a sequence g by $(r, r') \in g$. We call $(r, r') \in g$ a pairing. Note that in a pair (r, r') train r occupies the formation track before train r'. For example, given a sequence $g = \langle u, r_1, r_2, r_3 \rangle$, it holds that $(r_1, r_2) \in g$ and $(r_2, r_3) \in g$, but note that $(r_1, r_3) \notin g$ and $(r_2, r_1) \notin g$. A sequence which is ordered by $\prec$ is *feasible*. Let G denote the set of feasible sequences. For each sequence $g \in G$ and period $p \in P$, let $s_p(g)$ be the total length of the mixed cars from g in p, i.e.,

$$s_p(g) = \sum_{(r,r') \in g} s_p(r, r') \quad .$$

Further, let $c(g)$ be the sum of all extra roll-ins for g:

$$c(g) = \sum_{(r,r') \in g} c(r, r') \quad .$$

A *schedule* $f : O \rightarrow G$ is an injective mapping from tracks to feasible sequences. A feasible sequence g can be scheduled on a track o if and only if all trains of the sequence fit on the track, i.e., $g \subseteq R(o)$. Let us denote by $G(o)$ the set of all feasible sequences that can be scheduled on track o. A *feasible solution* to our problem can now be defined as a schedule f that

1. assigns a feasible sequence to each track,

$$\forall o \in O : f(o) \in G(o) \quad ,$$

2. such that each train occurs exactly once in a sequence,

$$\forall r \in R : \exists o \in O : r \in f(o) \wedge \forall o' \in O : o \neq o' \rightarrow r \notin f(o') \quad ,$$

3. and such that in each period, the capacity of the mixing track is respected,

$$\forall p \in P : \sum_{o \in O} s_p(f(o)) \leq s^{\text{mix}} \quad .$$

Feasible Sequences

For two trains (r, r'), where r departs before r', to be comparable by $\prec$ the active periods of all blocks $b \in B(r')$ must fit within the active period of train r'. The departure time of train r determines when the active period of r' may start. If the departure time of r is earlier than the arrival time of the earliest car of the first block of train r' the two trains are trivially comparable, else they are only comparable if there are enough pull-backs for all blocks of train r' to be built before its departure time. The end-time of the final block of train r' can be calculated using Algorithm 1, and if the end-time of the final active block is earlier than, or at the same time as, the departure time of train r' minus a given time for technical set-up, the two trains are comparable by $\prec$ and may be used in the same sequence. An example of two feasible and one infeasible train pairings is shown in Figure 2.

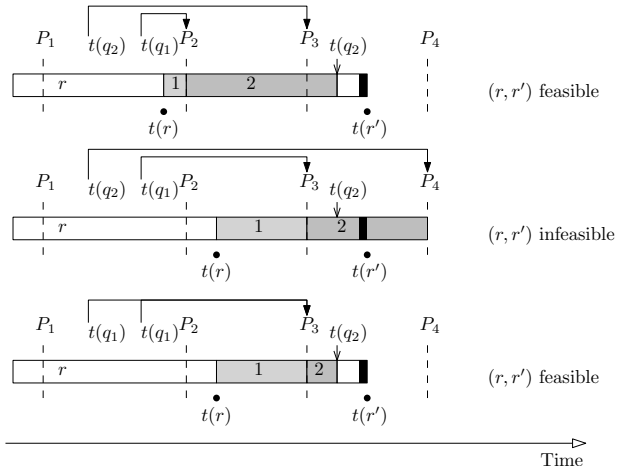

Figure 2: An example of two feasible pairings and one infeasible. Train r is not blocked while train r' has two blocks. The active periods of the trains are marked with white boxes, and the active periods of block 1 and 2 of train r' are marked with grey boxes of different darkness. The departure times are shown as circles underneath the active period boxes, and the technical set-up time as a black thick line. The roll-in times of cars belonging to train r' are shown above the boxes, and the arrows show when the cars are rolled to the train formation track. The block of a car is indicated as a subscript. For the pairing to be feasible all cars of train r' must be rolled to the train formation track before the technical set-up preceding its departure time $t(r')$.

3 The column generation model

In this section we introduce the column generation model from Bohlin et al. [5] and describe the necessary adaptations to allow for sorting the cars into blocks in the outbound trains. The essence of column generation is to reduce the problem size of a linear programming problem (LP) by starting out with a few variables and solving this reduced problem to optimality. If the solution satisfies all the constraints of the full dual problem the solution will be optimal also for the full LP, but if some constraints are broken more variables need to be introduced. In the model we use, the variables to be included are generated using a method called pricing, where the variable that corresponds to the most violated constraint in the full dual is found. As the shunting problem is an integer programming problem (IP) rather than an LP, we work with the LP relaxation of the IP, and then branching is used to find an IP solution. For a more thorough explanation of column generation, see Derosiers and Lübbecke [12].

In this paper we briefly outline the integer programming model and its dual. For further details on the model and the branching algorithm, we refer to Bohlin et al. [5].

3.1 The IP model and its dual

We use variables x_{go} to encode whether a sequence g is assigned to a track o or not. Further, variables y_{ro} are included to encode that train r is assigned to track o, as this variable is useful during branching. This gives the IP,

$$\min \quad \sum_{\substack{o\in O\\ g\in G(o)}} c(g)\cdot x_{go} \tag{1}$$

$$\text{s.t.} \quad \sum_{o\in O} y_{ro} \geq 1 \qquad r\in R \tag{2}$$

$$\sum_{\substack{g\in G(o)\\ r\in g}} x_{go} \geq y_{ro} \qquad r\in R, o\in O \tag{3}$$

$$\sum_{g\in G(o)} x_{go} \leq 1 \qquad o\in O \tag{4}$$

$$\sum_{\substack{o\in O\\ g\in G(o)}} s_p(g)\cdot x_{go} \leq s^{\text{mix}} \qquad p\in P \tag{5}$$

$$x, y \in \{0,1\} \tag{6}$$

(1) is the objective function, i.e. the number of car pull-backs, and to adapt the IP formulation for sorting by blocks the formula for $c(g)$ from Section 2 should be used. Inequalities (2) and (3) ensure that every train is present in at least one sequence. Should a train be present in more than one sequence in the final solution all but one instance of the train can be removed. Inequality (4) ensures that at most one sequence is assigned to each track. In inequality (5) the formula for $s_p(g)$ from Section 2 should be used to make sure that the mixing track capacity is never exceeded. The dual of the problem is then the following,

$$\max \quad \sum_{r\in R}\alpha_r + \sum_{o\in O}\gamma_o + s^{\text{mix}}\sum_{p\in P}\delta_p \tag{7}$$

$$\text{s.t.} \quad \alpha_r \leq \beta_{ro} \qquad r\in R, o\in O \tag{8}$$

$$\sum_{r\in g}\beta_{ro} + \gamma_o + \sum_{p\in P} s_p(g)\cdot\delta_p \leq c(g) \qquad o\in O, g\in G(o) \tag{9}$$

$$\alpha,\beta \geq 0 \qquad \gamma,\delta \leq 0 \tag{10}$$

3.2 Pricing

In the pricing step we want to find the variable that violates

$$\sum_{r\in g}\beta_{ro} + \sum_{p\in P} s_p(g)\cdot\delta_p - c(g) \leq -\gamma_o$$

the most for each track o. That is the variables that maximise,

$$\max_{g\in G(o)} \sum_{r\in g}\beta_{ro} + \sum_{p\in P} s_p(g)\cdot\delta_p - c(g) \quad . \tag{11}$$

To this aim we make use of a directed graph $G = (V, E)$, where the nodes represent trains as well as the start and end of a sequence (special nodes u and v), $V = R(o) \cup \{u, v\}$. The edge set E includes an edge (u, r) and (r, v) for every train $r \in R(o)$, and an edge (r_1, r_2) if $r_1 \prec r_2$. Any path from u to v in G corresponds to a feasible sequence $g \in G(o)$.

Next, weights should be added to the edges. The weights represent how much cost a certain train pairing adds to Expression (11). As pointed out previously the first train in the sequence may incur a cost when sorting according to blocks. This potential cost is added to the edge (u, r) for every train $r \in R(o)$, where u is once again treated as a dummy train with departure time 0.0:

- $w_{(u,r)} = \beta_{ro} + \sum_{p\in P} s_p(u, r)\cdot\delta_p - c(u, r)$
- $w_{(r_1,r_2)} = \beta_{r_2o} + \sum_{p\in P} s_p(r_1, r_2)\cdot\delta_p - c(r_1, r_2)$
- $w_{(r,v)} = 0$

Now, for every sequence $g \in G(o)$ there is one equivalent path in G which has a total weight equal to

$$\sum_{r\in g}\beta_{ro} + \sum_{p\in P} s_p(g)\cdot\delta_p - c(g) \quad ,$$

where $s_p(g)$ and $c(g)$ are defined as in Section 2 and allow for sorting according to blocks. This is the quantity that should be maximised, which is done by finding the longest path in G from u to v. Bohlin et al. [5] points out that the partial ordering of trains makes G cycle-free, and therefore the longest path can be calculated in $\mathcal{O}(|V| + |E|)$ time (see [9]). As the train graph could be close to complete, the complexity in our case would be $\mathcal{O}(|R|^2)$.

4 Experiments

Case study
The Hallsberg classification yard, which is the main hub for car-load rail freight in Sweden, was used as a case study. To evaluate the approach, a historic data set obtained from the Swedish Transport Administration (Trafikverket) was used. The set included trains that arrived to or departed from the Hallsberg yard between December 2010 and May 2011. Further, the physical constraints of Hallsberg marshalling yard were taken from [3].

The geographical layout of the yard in Hallsberg can be seen in Figure 3. Hallsberg's arrival yard consists of 8 tracks, while the classification yard has 32 tracks and the departure yard 23 tracks. The length of the arrival yard tracks range from 595 to 693 meters, and the classification track lengths range from 374 to 760 meters. The departure yard tracks have lengths between 562 and 886 meters. Two classification tracks, of length 704 m and 724 m, were chosen as mixing tracks, giving a total mixing capacity of 1428 m.

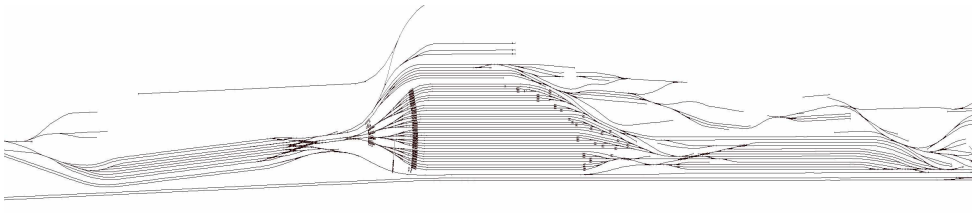

Figure 3: Layout of the Hallsberg Marshalling yard, taken from [3]. The three sub-yards can easily be identified, with the arrival yard to the left. But from the three sub-yards a number of other tracks used for e.g. repair work are visible. These tracks were not included in the problem set-up as they are normally not used for shunting. The image is not to scale.

Arrival and departure track scheduling was done in a pre-processing step as described in Bohlin et al. [7]. Likewise, a hump schedule for roll-ins and pull-backs was generated using the same pre-processing code as in Bohlin et al. [7]. However, as sorting by blocks requires more shunting steps extra pull-backs were scheduled between two consecutive inbound train roll-ins whenever possible. The time duration estimates used in the pre-processing, and when adding extra pull-backs, were taken from [2].

The resulting data set consisted of 18 365 car groups. All cars in a car group arrive with the same inbound train, and depart with the same outbound train. Note that this is different from the car units used for generating a classification schedule that allows for sorting by blocks, as these car units consist of cars that arrive with the same inbound train, and belong to the same block in an outbound train. However, due to lack of better data car groups were used as car units. Since operational planning normally is done for a few days at a time, the resulting data set was split into separate planning problems of three days each, resulting in a total of 50 test cases. Every three day planning problem included all car groups that arrived to or departed from the yard during the period.

Generating block data
Unfortunately real-life block data was not available for the sampling period. However, the Hallsberg yard staff provided us with example data for one week to use as a basis for our

experimental setup. We used this example data to generate the block composition of trains randomly, based on the estimated frequency and composition of blocked trains in the example data set. Trains that require sorting by blocks normally depart at approximately the same time each day, and a daily departure time pattern consisting of seven times was therefore chosen. This time pattern corresponded to times when blocked trains typically departed from Hallsberg. A sweep algorithm was then used to identify the first trains that departed after the times specified by the pattern and consisted of cars from more than one inbound train. The latter requirement is a consequence of the direct translation of car groups to car units, and the need for at least two different car units to generate blocks. Trains that had been identified by the sweep algorithm and that consisted of cars from only two inbound trains were assumed to have two blocks, all other trains that were identified were assumed to require two or three blocks with equal probability. We restricted the number of blocks to at most three as the example data set never contained more than three blocks for any train. The number of blocks that a train required was sampled using the random module in Python 2.7 (for more details on the random module see [1]).

After the blocked trains had been chosen, their cars had to be assigned to blocks. We call a blocked train that can be correctly built on one train formation track within its unrestricted active period, i.e. when its active period is determined by the roll-in times of the cars and the departure time of the train only, a *feasible train*. Whether a train is feasible or not depends on the blocks of the train, and the pull-back and roll-in schedule. Blocked trains that are not feasible can never be built before their departure time and thereby render the entire problem infeasible.

Let $n_b^r = |B(r)|$ be the number of blocks of a train r, and let a block b be defined by its natural order, starting with 1. Then the following lemma states a sufficient condition for feasible trains.

Lemma 1. *A train r is feasible if for all cars q there exist at least $n_b^r - b(q)$ pull-backs before $t(r)$ but after $t(q)$.*

Although this condition is sufficient, it is not necessary. A trivial example of a feasible blocked train that requires no pull-backs is a train whose cars are rolled in from the arrival yard in the correct order. However, a feasible two block train that does not fulfill Lemma 1 can be treated as a non-blocked train (as all cars of the train must already be in the correct order during roll-in). Likewise, a feasible three block train not fulfilling Lemma 1 can be treated either as a train with no blocks (if all cars are rolled in in the correct order), or as a two block train.

To sample a block for a car all blocks that were feasible for the car with respect to Lemma 1 were identified. Then the random number generator of Python 2.7 was used to sample one of the feasible blocks and the car was assigned to this block. All feasible blocks had an equal probability of being chosen. To ensure that all blocks were represented at least once, the set of feasible cars was identified for each block, and then one car was sampled from each set and assigned to the respective block. This was done in the beginning of the block-sampling, and the cars were sampled in the natural order of the blocks. Cars were removed from the sampling as soon as they had been assigned to a block. If a train did not have any feasible cars for a certain block it was either assumed to not require sorting by blocks, or it was assumed to be a train of fewer blocks than originally sampled. More precise, two block trains were always assumed to not require sorting by blocks, while three block trains were re- marked as a two block trains and the block sampling was restarted.

The initial sweep algorithm identified 1054 trains to be blocked in the historic data. Out of these, 581 two block trains and 473 three block trains were sampled. However, due to lack of pull-backs 29 two block trains and 38 three block trains were rejected by the algorithm at later stages. Further, 15 three block trains were re-marked as two block trains. Therefore the final experimental data included 567 two block trains and 420 three block trains. Out of the 567 two block trains 260 were trivial, i.e. the blocks happened to be sampled such that all cars were rolled in from the arrival yard in the correct order. Likewise, 109 of the three block trains were trivial.

Introducing blocked outbound trains will decrease the number of feasible pairings as the number of pull-backs needed in the active period of a train is increased (see Section 2.1). In our case, the number of feasible pairings was reduced from 135 514 to 131 755 for the entire sample period.

Technical details

SCIP 3.1.0 was used as the branch-and-price framework, with CPLEX 12.5.00 as the LP solver. The pre-processing code was run using Python 2.7.2. Experiments were performed on Linux workstations running openSUSE 12.1 with eight Intel Core i7-2600 quad-core CPUs running at 3.4 GHz and equipped with 16 GB of RAM.

4.1 Results

Two different test cases, one with blocked outbound trains (called B-IP) and one without (N-IP), were executed and the results are presented in Table 1. Two different measurements of car pull-backs are presented: P-EP is the number of extra car pull-backs that the problem set-up includes, while O-EP is the number of extra car pull-backs that would be needed to properly sort all cars according to the generated classification schedule in an operational setting. The difference between the two measurements is largely caused by the pull-backs that were added between consecutive train roll-ins. These pull-backs were scheduled without consideration of how useful they would be, and therefore they are likely to not move any cars from the mixing track to the train formation tracks, but rather all mixed cars will just be pulled out to the arrival yard and then immediately rolled back to the mixing track. This is obviously an unnecessary operation, and in a real world setting the pull-back would not be executed. O-EP include only the extra pull-backs which are strictly necessary, while P-EP includes all extra pull-backs present in the schedule. Further, while all pull-backs of N-IP may be considered as "extra" pull-backs enforced by capacity limitations, a certain amount of pull-backs will be necessary for sorting the cars into blocks in B-IP. Therefore the total number of pullbacks, as calculated by the formula in Section 3, is presented in brackets for B-IP. The extra pull-backs have been calculated as the total number of pull-backs minus the number of pull-backs necessary for sorting the cars into blocks given that all the trains' active periods are unrestricted.

As can be seen in Table 1, the optimal solution is found for all test instances. Further, the average execution times for the two test cases are comparable, indicating that generating a schedule that allows for blocked outbound trains will not negatively affect the execution time. As expected the total number of pull-backs is greatly increased when blocking is introduced. However, the number of "extra" pull-backs enforced solely by capacity constraints remains low. In fact, the average of P-EP enforced solely by capacity constraints is reduced when sorting cars according to blocks. This is explained by the fact that some of the pull-backs that

are necessary for sorting the cars into blocks will also alleviate capacity problems. However, the average O-EP is slightly higher for B-IP than for N-IP. This may be explained by the fact that more pull-backs will be useful when sorting cars into blocks. A car that spends the same amount of time on a mixing track in B-IP and N-IP is therefore likely to be subjected to more pull-backs in an operational setting if the mixing track is used not only for capacity alleviation but also for sorting cars according to blocks. The effect of this is not seen in P-EP as all pull-backs are included whether they are useful or not.

Table 1: The experimental results for two different test cases, one with blocked outbound trains (B-IP) and one without (N-IP). P-EP are the number of extra pull-backs as defined by the problem set-up, while O-EP are the extra pull-backs that need to be carried out in an operational setting to sort the cars according to the generated classification schedule. The execution times and optimality gaps are also included. $\bar{x}$ is the arithmetic mean. The execution time is reported as clock-seconds, and includes problem set-up and post-processing.

	Instance		*N-IP*				*B-IP*			
	Trains (#)	Groups (#)	P-EP (#)	O-EP (#)	Time (s)	Gap (%)	P-EP (Tot) (#)	O-EP (Tot) (#)	Time (#)	Gap (#)
1	65	257	0	0	10.2	0	0 (82)	0 (44)	8.5	0
2	107	581	68	33	124.5	0	48 (474)	34 (306)	99.9	0
3	35	148	0	0	1	0	0 (162)	0 (76)	1.2	0
4	88	340	0	0	29.9	0	0 (248)	0 (106)	44.8	0
5	14	46	0	0	0.1	0	7 (9)	7 (8)	0.1	0
6	55	170	0	0	3.3	0	0 (267)	0 (133)	3.3	0
7	44	170	0	0	2.6	0	0 (109)	0 (31)	2.8	0
8	51	155	0	0	3.8	0	0 (49)	0 (39)	2.8	0
9	70	268	0	0	12	0	0 (89)	0 (31)	11.3	0
10	44	155	0	0	2.2	0	0 (23)	0 (12)	2.6	0
11	113	575	89	38	191.2	0	61 (500)	53 (335)	154.8	0
12	57	282	0	0	4.4	0	0 (262)	0 (194)	4.7	0
13	91	485	30	28	43.4	0	17 (278)	17 (224)	38.7	0
14	80	440	4	2	25.4	0	0 (276)	0 (216)	24.8	0
15	63	259	0	0	7.6	0	0 (118)	0 (52)	6.5	0
16	73	308	0	0	9	0	0 (162)	0 (84)	8.1	0
17	53	229	0	0	4.6	0	0 (64)	0 (33)	3.6	0
18	97	485	5	3	51.8	0	3 (144)	2 (69)	61.7	0
19	66	296	0	0	5.5	0	0 (449)	0 (258)	5.1	0
20	78	440	18	16	15.7	0	9 (177)	8 (104)	14.6	0
21	75	371	0	0	12.3	0	0 (600)	0 (370)	13.9	0
22	59	251	0	0	8.8	0	0 (413)	0 (260)	8.3	0
23	106	594	64	54	70.3	0	51 (651)	51 (420)	52.4	0
24	43	200	0	0	1.5	0	0 (561)	0 (352)	1.7	0
25	108	538	3	3	93.5	0	0 (455)	0 (277)	66.8	0
26	74	327	0	0	10.9	0	0 (269)	0 (132)	9.2	0
27	82	467	4	2	14.2	0	2 (208)	2 (158)	14.1	0
28	72	413	130	24	12.1	0	130 (222)	68 (140)	5.6	0
29	62	259	0	0	8.1	0	0 (286)	0 (156)	7.2	0

Table 1: The experimental results (continuation from previous page).

	Instance		*N-IP*				*B-IP*			
	Trains (#)	Groups (#)	P-EP (#)	O-EP (#)	Time (s)	Gap (%)	P-EP (Tot) (#)	O-EP (Tot) (#)	Time (#)	Gap (#)
30	103	572	29	20	173.5	0	21 (352)	21 (246)	169.6	0
31	50	234	0	0	3.1	0	0 (267)	0 (189)	3.1	0
32	118	715	100	81	130.4	0	64 (599)	59 (481)	156	0
33	78	420	1	1	13.1	0	0 (392)	0 (212)	16.1	0
34	88	492	49	39	36.3	0	20 (236)	16 (187)	43.7	0
35	84	503	14	9	36	0	3 (471)	3 (310)	28.6	0
36	65	288	0	0	6.8	0	0 (204)	0 (71)	7	0
37	94	546	118	78	49.4	0	85 (682)	63 (456)	41.8	0
38	51	268	0	0	4.5	0	0 (345)	0 (253)	5.6	0
39	111	640	86	76	151.4	0	67 (483)	67 (355)	148	0
40	59	269	0	0	5.1	0	0 (411)	0 (269)	4.8	0
41	95	513	158	62	47	0	145 (261)	83 (150)	54.7	0
42	86	514	29	22	21.3	0	28 (364)	25 (310)	29.4	0
43	70	319	6	4	9.9	0	6 (245)	5 (106)	14.6	0
44	84	503	14	12	37.8	0	7 (464)	6 (298)	32.4	0
45	5	9	0	0	0	0	0 (0)	0 (0)	0	0
46	91	470	19	12	24	0	4 (503)	4 (444)	33.2	0
47	71	391	0	0	6.8	0	0 (273)	0 (177)	5.3	0
48	94	533	247	118	45.2	0	235 (373)	187 (294)	62.5	0
49	74	413	4	4	13.3	0	3 (231)	3 (149)	20.2	0
50	56	244	0	0	4.4	0	0 (148)	0 (107)	4.6	0
$\bar{x}$	73.0	367.3	25.8	14.8	32.0	0	20.3 (298.2)	15.7 (193.7)	31.2	0

5 Conclusions

In this paper we introduced a novel approach for optimisation of simultaneous car sorting and train formation for freight shunting yards. The approach builds on the integer programming approach of Bohlin et al. [5] to enable the generation of classification schedules that allow for cars to be sorted into blocks within outbound trains. This is accomplished by constructing new formulae for calculating the mixing and pull-back costs, as well as adapting the pricing algorithm. Experiments were performed on real traffic data from Hallsberg marshalling yard, generated block data, and a planning horizon of three days. The new model found an optimal solution within three minutes for all 50 test instances, and the results show that although the solutions require more car pull-backs in total, the number of extra pull-backs enforced by capacity constraints is comparable with the number of extra pull-backs that is required when no outbound trains have blocks. In fact, if all pull-backs that were included in the problem set-up are executed, the number of extra pull-backs enforced solely by capacity constraints is reduced slightly when sorting the cars into blocks. This is because some of the pull-backs that are necessary for sorting (and hence not extra) also alleviate the capacity problem. However, if only the useful pull-backs are executed, the number of extra pull-backs

is slightly increased when including sorting by blocks. This is explained by the fact that more pull-backs are useful in the blocked problem, and therefore a car that has been mixed for capacity reasons is likely to be subjected to more pull-backs while waiting for track capacity to become available.

5.1 Future Work

There is a number of ways in which this method could be further improved. First of all, the heuristic used to decide the roll-in and pull-back times does not take blocking into consideration. Changing the heuristic such that blocking is included would most likely improve our results. Further, in this paper the mixing track modelling is simplified, and we view the different mixing tracks as one long track. In reality there are many mixing tracks, and they could be pulled-out at different times providing an opportunity to sort cars by blocks on the mixing tracks, and thereby make better use of the yard capacity. For this reason, more detailed mixing-track modelling is highly desirable. For the methods described in this paper to be applicable in real operations, robustness and efficient ways to update the schedule and extend the planning horizon are also important topics to address.

Acknowledgments

We are grateful to Stefan Huss and Pelle Andersson at Green Cargo AB for providing information and data on the shunting process in Hallsberg, and to the staff at the Hallsberg shunting yard for providing example data and for helpful and interesting discussions. Finally, we would like to thank Hans Dahlberg at the Swedish Transport Administration for his support.

References

[1] Python v2.7.3 documentation chapter 9.6. random –generate pseudo-random numbers. http://docs.python.org/2/library/random.html, Jan. 2013.

[2] C. Alzén. *Handbok BRÖH 313.00700: Trafikeringsplan Hallsbergs rangerbangård*. Banverket, May 2006.

[3] K.-Å. Averstad. *Handbok BRÖH 313.00001: Anläggningsbeskrivning Hallsbergs rangerbangård*. Banverket, February 2006.

[4] K. Beygang, S. O. Krumke, and F. Dahms. Train marshalling problem - algorithms and bounds -. Technical Report 132, TU Kaiserslautern, Fachbereich Mathematik, 2010.

[5] M. Bohlin, F. Dahms, H. Flier, and S. Gestrelius. Optimal Freight Train Classification using Column Generation. In D. Delling and L. Liberti, editors, *12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 25 of *OpenAccess Series in Informatics (OASIcs)*, pages 10–22, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[6] M. Bohlin, H. Flier, J. Maue, and M. Mihalák. Hump Yard Track Allocation with Temporary Car Storage. In *The 4th International Seminar on Railway Operations Modelling*

and Analysis (RailRome), 2011. Available on `http://soda.swedish-ict.se/5089/`.

[7] M. Bohlin, H. Flier, J. Maue, and M. Mihalák. Track Allocation in Freight-Train Classification with Mixed Tracks. In *11th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 20 of *OpenAccess Series in Informatics (OASIcs)*, pages 38–51, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

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

[9] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction To Algorithms*. MIT Press, 3rd edition, 2009.

[10] E. Dahlhaus, P. Horák, M. Miller, and J. F. Ryan. The train marshalling problem. *Discrete Applied Mathematics*, 103(1–3):41–54, 2000.

[11] E. Dahlhaus, F. Manne, M. Miller, and J. Ryan. Algorithms for combinatorial problems related to train marshalling. In *Proceedings of the Eleventh Australasian Workshop on Combinatorial Algorithms (AWOCA)*, pages 7–16, 2000.

[12] J. Desrosiers and M. Lübbecke. A primer in column generation. In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, pages 1–32. Springer, Berlin, 2005.

[13] M. Gatto, J. Maue, M. Mihalák, and P. Widmayer. Shunting for dummies: An introductory algorithmic survey. In *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 310–337. Springer, 2009.

[14] R. Jacob, P. Márton, J. Maue, and M. Nunkesser. Multistage methods for freight train classification. *Networks*, 57(1):87–105, 2011.

[15] J. Maue and M. Nunkesser. Evaluation of computational methods for freight train classification schedules. Technical report, ARRIVAL Project, 2009.