Optimal Freight Train Classification using Column Generation*

Markus Bohlin[1], Florian Dahms[2], Holger Flier[3], and Sara Gestrelius[1]

1 **Swedish Institute of Computer Science**
{firstname.lastname}@sics.se
2 **RWTH Aachen, Chair of Operations Research, Germany**
dahms@or.rwth-aachen.de
3 **ETH Zürich, Institute of Theoretical Computer Science, Switzerland**
holger.flier@inf.ethz.ch

Abstract

We consider planning of freight train classification at hump yards using integer programming. The problem involves the formation of departing freight trains from arriving trains subject to scheduling and capacity constraints. To increase yard capacity, we allow the temporary storage of early freight cars on specific mixed-usage tracks. The problem has previously been modeled using a direct integer programming model, but this approach did not yield lower bounds of sufficient quality to prove optimality. In this paper, we formulate a new extended integer programming model and design a column generation approach based on branch-and-price to solve problem instances of industrial size. We evaluate the method on historical data from the Hallsberg hump yard in Sweden, and compare the results with previous approaches. The new method managed to find optimal solutions in all of the 192 problem instances tried. Furthermore, no instance took more than 13 minutes to solve to optimality using fairly standard computer hardware.

1998 ACM Subject Classification G.1.6 Integer Programming, G.1.10 Applications, F.2.2 Sequencing and scheduling

Keywords and phrases Column generation, integer programming, scheduling, shunting, classification, marshalling, transportation, railways

Digital Object Identifier 10.4230/OASIcs.ATMOS.2012.10

1 Introduction

In this paper, we solve a planning problem from the largest Swedish hump yard, Hallsberg, to optimality using a column generation approach. In previous papers [4, 5], we have modeled the problem as a special kind of list coloring problem on interval graphs, proved the NP-completeness of several variants of the problem, and developed both heuristics and mixed integer programming formulations for the problem. In this paper, we are for the first time able to find provably optimal solutions for the problem instances within practical time limits. We evaluate our results on historical data taken from a five month period of traffic at Hallsberg.

There are two basic modes of operation in railway freight transportation, namely single wagon load and full train transportation. In *full train* transportation, all cars of a train belong

* This work was supported by the Swedish Transport Administration under grant TRV 2010/29758 and by the Swiss National Science Foundation (SNF) under grant 200021-125033/1.

12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'12).
Editors: Daniel Delling, Leo Liberti; pp. 10–22

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

to the same customer and share a common destination. In *single wagon load* transportation, shipments of several customers are transported together in a hub and spoke network, where trains are typically composed of cars with different destinations. In order to route each car to its final destination, trains are decoupled into single cars at *classification yards* (also: marshalling or shunting yards). New outbound trains are then formed from cars which share a common intermediate destination.

Classification yards constitute a bottleneck in rail freight transportation. If a car misses its next train along the route, the incurred delay can be up to several days. In order to make single wagon load transportation more competitive, it is highly desirable to route cars through the network as quickly as possible while maintaining all connections.

Since resources at classification yards are usually scarce, production planning is a necessity. Of particular interest are so called *hump yards*, the largest class of classification yards where cars are pushed over a hump in order to roll onto their respective classification track by means of gravity. The problem we study in this paper is restricted to the scheduling of the *classification bowl* of the hump yard, i.e., a set of parallel *classification tracks* that are used for the *formation* of outbound trains from single cars. In particular, we consider the common case where at any point in time, there is a bijection between outbound trains and classification tracks, i.e., each track may only contain cars of a single outbound train. Therefore, we need to decide for each train on which track it will be formed.

Due to the high amount of traffic it is impossible, however, to reserve a whole track for each outbound train from the time of arrival of its first car to the time of its departure. Therefore, some of the tracks are used as a buffer area where cars of different trains may be temporarily stored. We refer to these tracks as *mixing tracks*. At given points in time, a *pull-out* operation is performed which allows to move any subset of cars from the mixing tracks to the classification tracks, provided that the formation of each such car's respective outbound train has started. In all brevity, a pull-out comprises that the cars of each mixing track are coupled, pulled back over the hump by an engine, to be immediately pushed over the hump to once more be distributed on the classification tracks. The latter is called a *roll-in* operation. A more detailed description of the operations as well as further planning problems at Hallsberg is given in [5].

Early literature on freight classification considers sorting schemes that essentially perform the same sorting steps for any input sequence of a given length [13, 14, 10]. More recently, it has been studied how to utilize the "pre-sortedness" of the input in order to minimize the number of pull-out operations [8, 12], as well as variants thereof [7]. A recent survey by Gatto et al. [11] gives an overview of this topic. The problem we study in this paper however does not require the cars of outbound trains to be sorted in any particular order.

The rest of the paper is structured as follows. We first define the mixing problem formally in Section 2 and give a direct integer programming model used for comparison in Section 3. In Section 4 an extended formulation is presented together with a new solution approach using branch-and-price column generation, a corresponding polynomial pricing problem, and the branching rules employed. Section 5 describes the experimental setup and results, including a comparison with previous approaches. Finally, Section 6 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 cars Q, and a set of outbound trains R. Groups of cars with the same destination arriving at the same time are

handled as single units. For each car $q \in Q$, we are given its arrival time $t(q)$, i.e., the time of its first roll-in into the classification bowl, its length $s(q)$, and its corresponding outbound train $r(q) \in R$. Each train $r \in R$ has a departure time $t(r)$, i.e., the time when it leaves the classification bowl. We denote by $Q(r) \subseteq Q$ the set of cars that belong to train r. 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 the cars of one outbound train, and each train is formed on exactly one classification track. We say that a train r is *active* during the time interval in which its corresponding classification track is used exclusively for the formation of r.

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, amongst others, on the departure times of trains r and r' as well as on technical setup times (e.g., brake inspection), the details of which we omit for the sake of clarity. Note that antisymmetry is ensured as no two trains may be formed on one track at the same time. We denote with $r \parallel r'$ that r and r' cannot be formed on the same track (i.e. they are incomparable by $\prec$).

In general, there are not enough classification tracks such that each train is active from the arrival of its earliest car until its departure. Therefore, we are also given a set of mixing tracks on which one can temporarily store cars of different outbound trains. To simplify our discussion, we treat these tracks as one concatenated track, called *the mixing track*. The mixing track has a given length s^{mix}. A car that is stored on the mixing track is said to be *mixed*. Whether a car needs to be mixed or not depends solely on the departure time of the preceding train on the same track. Once a train becomes active, mixed cars of that train can be retrieved from the mixing track by a pull-out operation on the mixing track, which is performed once at the beginning of each period. For each period $p \in P$, we are given its starting time $t(p)$. During the pull-out starting at time $t(p)$ each mixed car of a currently active train is moved to the allocated classification track. The remaining mixed cars return to the mixing track and remain there at least until the next period begins.

We seek to avoid the mixing of cars for several reasons. First, for each period during which a car is mixed, it will be subject to a roll-in operation, which takes effort and time, and wears down switches and tracks. As an objective function, we therefore choose to minimize the number of extra roll-in operations performed due to mixing. Second, the limited capacity s^{mix} of the mixing track must be respected in each period. Since no car can leave the mixing track until the next pull-out is performed, the total length of the mixed cars within a period is at its maximum at the end of the period. For two trains $r \prec r'$ and a period p, let $s_p(r, r')$ denote the total lengths of all cars of r' which have to be mixed in p:

$$s_p(r, r') = \begin{cases} \sum_{q \in Q(r') : t(q) < \min(t(r), t(p+1))} s(q), & \text{if } t(p) < t(r)\,, \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, let $c(r, r')$ denote the total number of extra roll-ins for two trains $r \prec r'$:

$$c(r, r') = \sum_{q \in r' : t(q) < t(r)} |q| \cdot k_{qr}\,,$$

where $|q|$ is the number of actual cars represented by the car group q and k_{qr} is the number

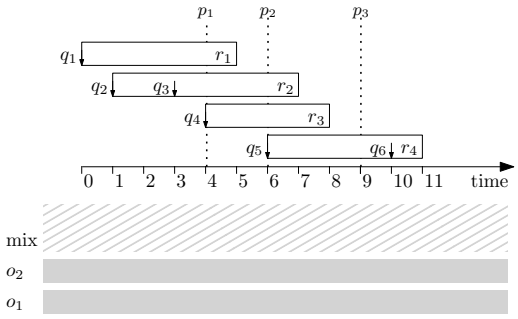
of periods between the arrival of car group q and the departure of train r:

$$k_{qr} = \#\{p \mid p \in P \wedge t(q) < t(p+1) \wedge t(p) < t(r)\}.$$

There are several subtleties in how mixing is performed that are noteworthy. Consider two consecutive trains r, r' for which $r \prec r'$. Any car of the second train r' that arrives after $t(r)$, the time of departure of the first train, may enter the classification track directly, since r' can become active immediately after r has departed. However, any car q of r' arriving before the departure of r, i.e., $t(q) < t(r)$, has to be send to the mixing track, since the classification track o may only hold cars of one outbound train at the same time. This also means that the partial order $\prec$ actually depends on the given times of periods, trains, and cars. If $r \prec r'$ and the second train r' has a car q that must be mixed, then in order to pull back and roll-in of q to o in time before r' departs, there must also be a period p with $t(r) < t(p) < t(r')$. Note also that a train whose earliest car arrival and departure time lie both within the same period cannot have cars that are mixed, for otherwise such a car could not be moved in time to the classification track.

				$c(r_i, r_j)$		
r_i	Car arrivals	s	t	r_2	r_3	r_4
r_1	$t(q_1) = 0$	1	5	4	1	0
r_2	$t(q_2) = 1$	1	7		1	1
	$t(q_3) = 3$					
r_3	$t(q_4) = 4$	1	8			1
r_4	$t(q_5) = 6$	2	11			
	$t(q_6) = 10$					

(a) Problem instance data.



(b) Illustration of problem instance. Downward-pointing arrows indicate car arrivals.

Figure 1 Example instance with two classification tracks o_1, o_2, departing trains $r_1 - r_4$, car arrivals $q_1 - q_6$, and three periods $p_1 - p_3$. The total number of mixed cars if two trains are allocated on the same track are also shown. Of the two tracks, only o_2 can accommodate the longest train r_4; all other trains fit on any track.

An example problem instance is illustrated in Fig. 1. Here, four trains with car arrivals and departure times as below are to be allocated to two classification tracks o_1, o_2. All trains fit on the longest track o_2, but only the first three trains fit on the shorter track o_1. Mixing capacity is assumed to be infinite, and pull-outs are performed at time 4, 6 and 9. Traversing the trains in order, all trains can precede all later trains (as defined by $\prec$).

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. Let us denote the fact that two trains $r, r' \in R$ appear consecutively in this order in a sequence g by $(r, r') \in g$. For example, given a sequence $g = \langle r_1, r_2, r_3 \rangle$, it holds that $(r_1, r_2) \in g$ and $(r_2, r_3) \in g$, but note that $(r_1, r_3) \notin g$. A sequence which is ordered by $\prec$ is *feasible*. Let G denote the set of feasible sequences. For each $g \in G$ and period $p \in P$, let $s_p(g)$ be the total length of the mixed cars for g in p, i.e.,

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

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

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

A *schedule* $f : O \to G$ 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 each track a feasible sequence,

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

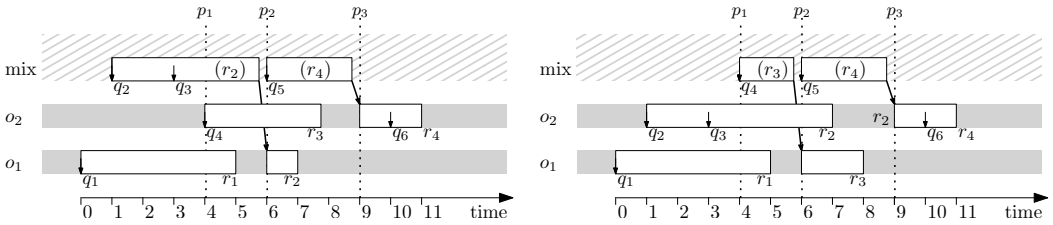
2. such that each train occurs exactly once in a sequence,

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

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

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

A feasible solution f is optimal if it minimizes the total number of roll-ins $\sum_{o\in O} c(f(o))$. Figure 2 illustrate one suboptimal solution as well as one optimal solution to the example problem instance from Figure 1.



(a) Feasible solution, with car q_2 and q_3 mixed in two periods each, and q_5 in one.

(b) Optimal solution, with car q_4 and q_5 mixed in one period each.

Figure 2 Two feasible solutions for the problem instance in Fig. 1. Cars are mixed for in total five periods in 2a and two periods in 2b.

3 A Binary Integer Programming Formulation

In order to evaluate the main contribution of this paper, namely the column generation approach described in Section 4, we give a brief outline of the binary integer programming formulation for the problem which we developed in [5].

The model is based on the observation that both the number of extra roll-ins and the amount of used mixing capacity in each period depends solely on times when trains become active. Further, only a few points in time turn out to be relevant for a train to become active, namely each time one of its cars arrives or is pulled-out of the mixing track. Let $\mathcal{T}(r)$ denote the set of all such relevant points in time for train r. We introduce binary variables

x_{rt} that indicate at which time t the reservation of a classification track for outbound train r starts, i.e., at which time r becomes active. Further, binary variables y_{ro} indicate whether the outbound train r is allocated to track o.

$$\text{minimize} \quad \sum_{r \in R} \sum_{t \in \mathcal{T}(r)} c(r,t) \cdot x_{rt} \tag{1}$$

$$\text{subject to} \quad \sum_{o \in O(r)} y_{ro} = 1, \qquad r \in R \tag{2}$$

$$\sum_{t \in \mathcal{T}(r)} x_{rt} = 1, \qquad r \in R \tag{3}$$

$$\sum_{t \in \mathcal{T}(r'): t < t(r)} x_{r't} + y_{ro} + y_{r'o} \leq 2, \qquad r \prec r', o \in O(r) \cap O(r') \tag{4}$$

$$\sum_{r \in R} \sum_{t \in \mathcal{T}(r)} s_p(r,t) \cdot x_{rt} \leq s^{\text{mix}}, \qquad p \in P \tag{5}$$

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

Objective (1) gives the objective in terms of the number $c(r,t)$ of extra roll-ins due to mixing, which results from using start time t for train r. Equalities (2) ensures that each train $r \in R$ is allocated to a track on which it fits. Equalities (3) ensures that each train $r \in R$ becomes active at a relevant time point $t \in \mathcal{T}(r)$. Inequalities (4) states that for each pair of trains $r \prec r'$ that can be scheduled consecutively on the same track, either r and r' are scheduled on different tracks, or r' becomes active only after r has departed. Finally, Inequalities (5) ensures that the mixing capacity is respected in each period, where $s_p(r,t)$ denotes the length of all cars of r that are mixed in period p if r becomes active at time t.

Scheduling problems in general often yield weak LP relaxations, and not surprisingly, this is true also for the compact problem formulation in the previous section, which have scheduling properties such as those encoded in Intequalities (4). This is confirmed by the results in [5] and [4].

4 Extended Formulation Solution

In this section, we introduce an extended formulation, where the variables represent pairings of entire sequences and tracks. We will see that this formulation can be solved efficiently by branch-and-price and leads to a strong LP relaxation for our problem instances (see Section 5.1).

For each track o and every possible sequence $g \in G(o)$ we use a variable x_{go} to encode whether we use g on o or not. As we saw in Section 2, it is sufficient to know the sequence for each track to calculate the mixing track usage and the number of extra roll ins of a schedule. Furthermore, each sequence-track pair included in the final solution will add to these quantities independently of all other pairs, which allows us to use a linear objective function and linear constraints only. To aid in branching, we also use the variables y_{ro} from the previous section, encoding that train r is assigned to track o. The full integer program for our extended formulation (EF) looks as follows:

$$\min \quad \sum_{\substack{o\in O\\ g\in G(o)}} c(g)\cdot x_{go} \tag{7}$$

$$\text{s.t.} \quad \sum_{o\in O} y_{ro} \geq 1 \qquad r\in R \tag{8}$$

$$\sum_{\substack{g\in G(o)\\ r\in g}} x_{go} \geq y_{ro} \qquad r\in R, o\in O \tag{9}$$

$$\sum_{g\in G(o)} x_{go} \leq 1 \qquad o\in O \tag{10}$$

$$\sum_{\substack{o\in O\\ g\in G(o)}} s_p(g)\cdot x_{go} \leq s^{\text{mix}} \qquad p\in P \tag{11}$$

$$x, y \in \{0,1\} \tag{12}$$

Objective (7) counts the total number of extra roll-ins as the sum of the roll-ins for the sequences selected for each track. Inequalities (8) and (9) ensure that every train appears in one sequence. We do not have to ensure equality as using a single train several times can never improve the objective. If a single train occurs several times in an optimal solution, then it can be removed from all but one sequence. Inequalities (10) state that at most one sequence per track can be used, and inequalities (11) ensure that we do not use more than the available mixing capacity in any period. Inequalities (8–11) are equivalent to the conditions for a feasible schedule in Section 2.

In the model above, there is one x variable for each combination of sequence $g \in G(o)$ and track $o \in O$. As the size of $G(o)$ is of order $\mathcal{O}(|R|!)$ only a subset of the x variables are initially included, and column generation is used to generate new variables as needed. For a detailed introduction to column generation see [9].

To use column generation, we first look at the dual of the LP relaxation of (EF):

$$\max \quad \sum_{r\in R} \alpha_r + \sum_{o\in O} \gamma_o + s^{\text{mix}} \sum_{p\in P} \delta_p \tag{13}$$

$$\text{s.t.} \quad \alpha_r \leq \beta_{ro} \qquad r\in R, o\in O \tag{14}$$

$$\sum_{r\in g} \beta_{ro} + \gamma_o + \sum_{p\in P} s_p(g)\cdot \delta_p \leq c(g) \qquad o\in O, g\in G(o) \tag{15}$$

$$\alpha, \beta \geq 0 \qquad \gamma, \delta \leq 0 \tag{16}$$

The dual variables are chosen as follows: the α variables correspond to primal inequalities (8), β to (9), γ to (10) and δ to (11). For every primal variable x we get an inequality of the form (15) and for every y an inequality of the form (14).

As stated above we start off with a reduced set of x variables. This reduced set contains sequences from a heuristically generated solution. Following the literature we will refer to this smaller variant of (EF) as the restricted master problem (RMP). We can now solve the LP relaxation of (RMP) to optimality with regard to the chosen subset of variables using any LP solver. Note that this solution may or may not be optimal for the relaxed (EF). From duality theory we know that the optimal solution for the relaxed (EF) will have a corresponding dual solution that satisfy all inequalities in (15). Further, the inequalities in (15) corresponding to the x variables in the (RMP) are guaranteed to be satisfied by our solution.

If our solution to the relaxed (RMP) satisfies all inequalities (15), it is also optimal for the relaxed (EF). If not, we need to add variables corresponding to violated inequalities to the

(RMP) and start over again until no inequalities are violated anymore. The new sub-problem is now to identify violated constraints (15) without checking every single one (which would be inefficient due to their number). This step, called pricing, will be explained in Section 4.1.

When we have reached an optimal LP solution for (EF), the solution might not be integral. Normal branch-and-bound on (RMP) does not guarantee optimality of the resulting integral solution, as the LP relaxation of (RMP) is not necessarily a lower bound given the branching decisions made to reach integrality. Therefore we may have to price in new variables in each node of the search tree. Furthermore, we must also take care not to include variables which, given the current branching decisions, represent infeasible train sequences. In Section 4.2 we will show how we implement a branching rule and how the branching decisions are taken into account in the pricing step. Algorithms of this type are referred to as branch-and-price algorithms [3].

4.1 Pricing

We will now discuss how we can efficiently determine variables that can improve (RMP) by finding violated inequalities from (15). First, as the set O is not too large, we can easily look at the inequalities for each track separately. Thus for each track $o \in O$ we need to find a sequence $g \in G(o)$ for which

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

is violated. There might be many such inequalities, and we want to search for the inequality that is violated the most, i.e., we want to maximize the left hand side:

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

To do so we use the fact that $s_p(g)$ and $c(g)$ are calculated as sums over pairs of trains appearing consecutively in the sequence, i.e. the quantities only depend on the immediate predecessor for each train (see Section 2).

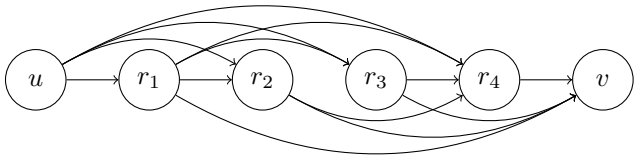


Figure 3 Longest path graph for the root pricing problem for track o_2 from the example in Figure 1.

First, a directed graph $G = (V, E)$ is constructed with the node set $V = R(o) \cup \{u, v\}$, i.e., one node for each train fitting on o plus two additional nodes. 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 $r_1 \prec r_2$. Note that any path from u to v in G corresponds to a feasible sequence $g \in G(o)$. Figure 3 shows what this graph looks like for the example presented in Figure 1, if we search for a new sequence for track o_2.

Next, we add edge weights to G. We choose the following weights:

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

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

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

As this is the quantity we want to maximize, we can search for a longest path in G from u to v. Due to the partial ordering of the trains, G is cycle-free, and calculating a longest path can be done in $\mathcal{O}(|V| + |E|)$ time (see[6]). In our case, this would be $\mathcal{O}(|R|^2)$, as the graph could be close to complete (i.e., complete except for edge (u, v), if $\prec$ is a total order).

4.2 Branching

When the LP relaxation of (EF) is solved in one of the branch-and-bound trees nodes and the solution turns out to be fractional, the solution space of the relaxed problem needs to be further restricted. This is known as branching. However, branching on x is problematic, as in the $x_{go} = 0$ branch just one sequence g on track o will be excluded. The pricing problem would therefore have to exclude individual paths, which in general is much more difficult than only computing the longest path. This is a common issue in column generation, see for example [3].

Fortunately, we can circumvent this problem by branching on allocation of trains r to tracks o instead, corresponding to the original y_{ro} variables in the direct model. In every fractional node, we will choose a fractional variable y_{ro} (such a variable must exist as otherwise the solution would not be fractional). The problem is then divided into two cases, one where y_{ro} equals 0 and one where it equals 1. Next we need to consider how to make sure branching decisions are respected in subsequent pricing steps. This can be accomplished by modifying the graph used in the pricing (see Section 4.1). We look at the two possible cases:

Case $y_{ro} = 0$: Remove node r from node set V and all edges connected to it. This way the generation of a sequence which contains train r is prohibited.

Case $y_{ro} = 1$: For all nodes r' and r'' where $t(r') \prec t(r) \prec t(r'')$, remove the edges (r', r''), together with (u, r'') and (r', v), from the edge set E. Also remove all nodes r' for which it holds that $r' \parallel r$ along with all their edges. Now all paths from a node before r to one after r will include r. Therefore no path from u to v can skip node r, forcing r to be contained in every generated sequence.

To illustrate this, consider the example graph in Figure 3. Figure 4 shows how the transformed longest path graphs would look like in both branches for variable $y_{r_2o_2}$.

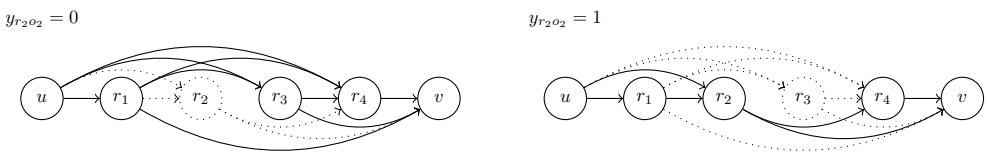


Figure 4 Longest path graphs for the pricing problem for track o_2 in the example from Figure 1 after branching on $y_{r_2o_2}$. Dotted nodes and edges are removed from the pricing problem.

5 Experiments

We evaluate the new approach on a historic data set, provided to us by the Swedish Transport Administration (Trafikverket). The data set is for arriving and departing trains and cars at the Hallsberg Rangerbangård hump yard in central Sweden, and covers a period of five months between December 2010 and May 2011. Hallsberg has 8 tracks of length 595 to 693 meters on the arrival yard, two parallel humps, of which only one is in use, 32 available classification tracks of length between 374 and 760 meters, and 12 tracks with length 562 to 886 meters on the departure yard. The layout of Hallsberg is shown in Figure 5.

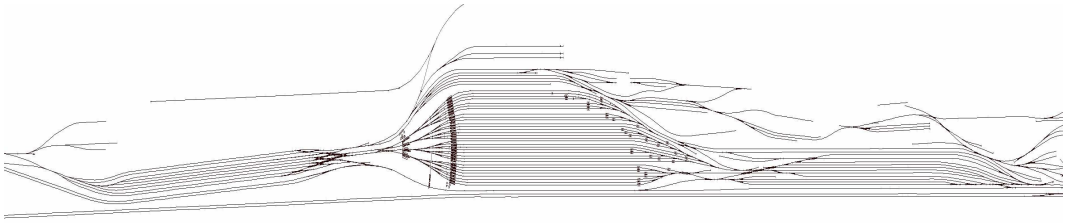


Figure 5 Layout of the Hallsberg classification yard in Sweden. The arrival yard is on the left, followed by the hump, the switching system, classification tracks, and finally the departure yard on the right. The image is taken from [2], and is scaled to emphasize details.

There are also several other tracks on the yard, for example tracks going to light and heavy repair facilities. These additional tracks are not considered since they are not normally used for shunting. Furthermore, arrival and departure tracks as well as hump scheduling is done in a preprocessing stage to obtain a suitable data set (see [5] for more details). Duration estimates were taken from [1].

The resulting data set consists of arrival times for 3653 outbound trains and 18366 car groups. Since operational planning is in practice done for a few days at a time, we split the resulting data set into separate planning problems, which each contain all car groups and the corresponding trains which are handled on the yard during the chosen interval. We chose to evaluate plans of length between two and five days, and assume that the yard is empty at the beginning of each planning interval. In a deployed implementation, cars which were already on the yard at the beginning of the planning interval would also have to be handled, but this is not considered in this paper. In total, 192 problem instances were generated for evaluation.

To evaluate our approach, we optimized the resulting problems using the improved heuristic from [5] (Heuristic++), the direct model from Section 3 (D-IP) and the new column generation approach from Section 4 (CG-IP). When found, heuristic solutions were used as starting points for both D-IP and CG-IP. CPLEX 12.4.0.0 in deterministic parallel mode with up to 8 threads was used to solve D-IP. CG-IP uses SCIP 2.1.0 as a branch-and-price framework with the same CPLEX version as LP solver. Experiments were performed on Linux workstations running openSUSE 12.1 with two Intel Core i7-2600 quad-core CPUs running at 3.4 GHz and equipped with 16 GB of RAM. All times are reported as wall-clock seconds and includes problem setup and post processing. A time limit of 20 minutes was set for each problem instance, after which the best integer solution found was returned.

Table 1 Experimental results for different planning horizons and solution methods. For each planning horizon, the number of instances in the sample data and the average instance problem size are included. For the different solution methods the table shows the average number of extra car-roll-ins due to mixing, the average execution time and the number of instances for which optimal, feasible and no feasible solutions were found. The average optimality gap is also reported for instances where feasible solutions with a non-zero lower bound were found. Finally, the number of times CG-IP generated a schedule with less extra roll-ins is reported along with the average improvement. Only the schedules that improved the number of extra roll-ins are included in this average. Only feasible instances are included in the average extra roll-ins and improvement values.

		Planning horizon (days)			
		2	3	4	5
	Number of instances	75	50	37	30
	Avg. number of trains	48.7	73.0	97.7	121.7
	Avg. number of groups	244.9	367.3	492.0	612.2
Heuristic++	Avg. extra roll-ins	8.3	16.1	26.0	31.6
	Avg. time (s)	0.0	0.1	0.1	0.2
Inst. classes	Feasible solution	73	47	34	27
	No feasible solution found	2	3	3	3
D-IP	Avg. extra roll-ins	10.1	18.8	29.8	25.2
	Avg. time (s)	360.6	530.9	684.1	722.4
Inst. classes	Optimality proven	50	27	14	12
	Feasible solution, LB>0 (avg. gap)	3(6.0)	2(16.2)	3(18.9)	1(7.4)
	Feasible solution, LB=0	22	21	20	14
	No feasible solution found	0	0	0	3
CG-IP	Avg. extra roll-ins	10.1	18.2	27.6	39.0
	Avg. time (s)	2.0	17.5	76.4	168.2
Inst. classes	Optimality proven	75	50	37	30
	Feasible solution, LB>0 (avg. gap)	0(0.0)	0(0.0)	0(0.0)	0(0.0)
	Feasible solution, LB=0	0	0	0	0
	No feasible solution found	0	0	0	0
Improvement	Heur++ No. (Avg. improvement)	19(7.7)	16(15.0)	15(11.7)	13(14.4)
	D-IP No. (Avg. improvement)	1(1.0)	8(3.6)	9(9.1)	7(3.4)

5.1 Results

The experimental results are shown in Table 1. The relative MIP gap reported is calculated as $|p - d|/|p|$, where p is the primal bound (the objective of the best found integral solution) and d is the dual bound. Note also that when a method fails to find a feasible solution for an instance, we exclude that instance from the extra roll-in average. As these instances often have a lot of traffic this reduces the average number of extra roll-ins for this method. This is why the heuristic has a lower extra roll-in average than the optimizing methods, and likewise why D-IP has a lower extra roll-in average than CG-IP for the five day planning horizon.

As we can see, CG-IP manages to prove optimality for all problem instances, compared to only 54 % of the instances for D-IP. This indicates a quite strong LP relaxation for CG-IP. In contrast, for D-IP, 90 % of the feasible instances not proven optimal terminates with a trivial lower bound of zero. For CG-IP, no instance took longer than 13 minutes to solve to optimality, while D-IP reaches the time limit for approximately 45 % of the instances.

Further, CG-IP always improves the shunting schedules with respect to the number of extra roll-ins compared to Heuristic++ and D-IP, and the percentage number of improvements increases as the planning horizon is extended. The reason the comparison between CG-IP and D-IP gives lower values than expected for the five day planning horizon is that D-IP fails to return a solution for 3 instances, and these 3 instances are subsequently omitted from the calculations.

6 Conclusions

We presented a compact IP model alongside an extended IP formulation to solve the train classification problem arising at (among other) the Hallsberg hump yard. For the extended formulation we provided all steps necessary to make it solvable in an efficient manner. The pricing problem was shown to be a longest path problem on a directed acyclic graph. We provided a branching rule that could easily be incorporated into the pricing problem by means of modifying this graph.

In the experiments performed on the data from Hallsberg the extended formulation turns out to provide a strong dual bound. Using the new approach we were able to find provably optimal solutions to all problem instances in a reasonable amount of time. These solutions often turn out to be a lot better than the solutions generated by the improvement heuristic from our previous paper [5, 4]. Therefore the extended IP model seems to be a good starting point for further research. In particular, the new results open up for a real-world implementation of the developed models.

6.1 Future Work

Though the results are promising, there are still open questions to be considered in future research.

The NP-completeness of the problem has been shown by reduction from μ-coloring [5]. The proof inherently needs the existence of differently sized classification tracks. It is still an open question whether the problem stays NP-complete if all tracks are of equal length.

As can be seen in Section 5.1, all our problem instances could be solved to optimality within a reasonable amount of time. Still, finding optimal solutions takes too much time for larger instances. We believe that this is mainly due to symmetry arising in the problem formulation, since train sequences on equivalent tracks can be interchanged. A possible way to reduce symmetry could be to aggregate the extended formulation variables and use a more sophisticated branching rule.

In real-world applications, the exact car lists and times of all incoming and outgoing trains are normally not known far in advance. Therefore the planning needs to be flexible, and the shunting schedule should be updated on a regular basis. This suggests possible further research directions:

- Changing the shunting schedule might be complicated as the formation of trains may already have begun. Therefore it would be preferable if the original schedule was constructed such that it could easily be recovered. Research from the field of recoverable robustness might apply here.
- Creating the shunting schedule while not possessing all information about the future adds an online component to the problem. It would be interesting to see if the problem can be handled as an online problem and if competitive analysis could lead to applicable results.

It also appears natural that less shunting would reduce the total load on the yard, implying a possibility to classify more freight with the same resources. To be consistent with current practices in Sweden, we however had to assume a fixed timetable and a fixed allocation of cars to trains. Planning timetables and freight car allocations with optimal yard operation in mind may yield such positive effects, and should therefore be investigated further.

Acknowledgements

We are grateful to Hans Dahlberg at the Swedish Transport Administration and Stefan Huss at Green Cargo AB for providing information on the working processes for shunt yard operation.

References

1 C. Alzén. *Handbok BRÖH 313.00700: Trafikeringsplan Hallsbergs rangerbangård*. Banverket, May 2006.
2 K.-Å. Averstad. *Handbok BRÖH 313.00001: Anläggningsbeskrivning Hallsbergs rangerbangård*. Banverket, February 2006.
3 C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, pages 316–329, 1998.
4 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/`.
5 M. Bohlin, H. Flier, J. Maue, and M. Mihalák. Track Allocation in Freight-Train Classification with Mixed Tracks. In *11$^{\text{th}}$ 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.
6 T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction To Algorithms*. MIT Press, 3rd edition, 2009.
7 E. Dahlhaus, P. Horák, M. Miller, and J. F. Ryan. The train marshalling problem. *Discrete Applied Mathematics*, 103(1–3):41–54, 2000.
8 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.
9 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.
10 H. Flandorffer. Vereinfachte güterzugbildung. *ETR RT*, 13:114–118, 1953.
11 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.
12 R. Jacob, P. Márton, J. Maue, and M. Nunkesser. Multistage methods for freight train classification. *Networks*, 57(1):87–105, 2011.
13 K. Krell. Grundgedanken des simultanverfahrens. *ETR RT*, 22:15–23, 1962.
14 M. W. Siddiqee. Investigation of sorting and train formation schemes for a railroad hump yard. In *Proceedings of the 5th International Symposium on the Theory of Traffic Flow and Transportation*, pages 377–387, 1972.