SWEDISH ICT SICS

OPTIMIZED SHUNTING

WITH MIXED-USAGE TRACKS

SICS Technical Report T2013:XX

Markus Bohlin, Sara Gestrelius SICS Swedish ICT
markus.bohlin@sics.se, sara.gestrelius@sics.se

Florian Dahms RWTH Aachen University
dahms@or.rwth-aachen.de

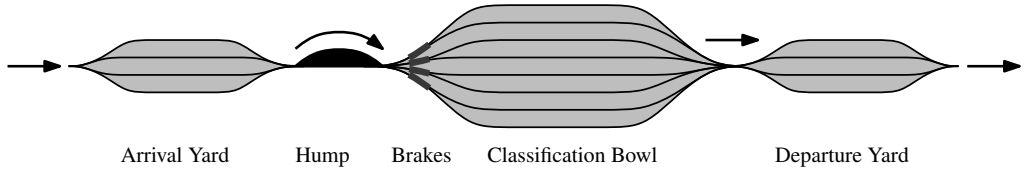
Matúš Mihalák, Holger Flier ETH Zürich
mmihalak@inf.ethz.ch, hflier@inf.ethz.ch

2013-12-19

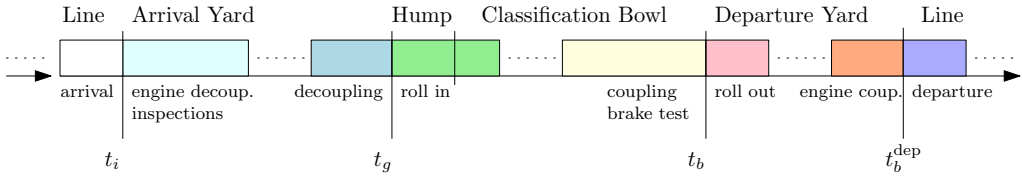
Executive Summary

We consider the planning of railway freight classification at hump yards, where the problem involves the formation of departing freight train blocks from arriving trains subject to scheduling and capacity constraints. The hump yard layout considered consists of arrival tracks of sufficient length at an arrival yard, a hump, classification tracks of non-uniform and possibly non-sufficient length at a classification yard, and departure tracks of sufficient length. To increase yard capacity, freight cars arriving early can be stored temporarily on specific mixed-usage tracks. The entire hump yard planning process is covered in this paper, and heuristics for arrival and departure track assignment, as well as hump scheduling, have been included to provide the neccessary input data. However, the central problem considered is the classification track allocation problem. This problem has previously been modeled using direct mixed integer programming models, but this approach did not yield lower bounds of sufficient quality to prove optimality. Later attempts focused on a column generation approach based on branch-and-price that could solve problem instances of industrial size. Building upon the column generation approach we introduce a direct arc-based integer programming model, where the arcs are precedence relations between blocks on the same classification track. Further, the most promising models are adapted for rolling-horizon planning. We evaluate the methods on historical data from the Hallsberg shunting yard in Sweden. The results show that the new arc-based model performs as well as the column generation approach. It returns an optimal schedule within the execution time limit for all instances but from one, and executes as fast as the column generation approach. Further, the short execution times of the column generation approach and the arc-indexed model make them suitable for rolling-horizon planning, while the direct mixed integer program proved to be too slow for this.

Extended analysis of the results shows that mixing was only required if the maximum number of concurrent trains on the classification yard exceeds 29 (there are 32 available tracks), and that after this point the number of extra car roll-ins increases heavily.



(a) A schematic layout of a hump yard.



(b) Activities performed on a single car g, from its arrival in train i at the yard, via roll-in to the classification bowl, roll-out, and finally departure as part of the new train b.

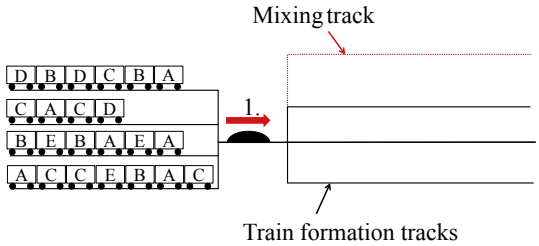
Figure 1: Typical layout of a hump yard, and activities performed on each car passing through the yard.

1 Introduction

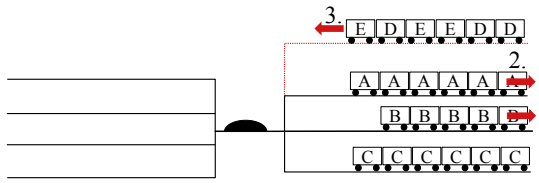
In railway freight transportation, there are two distinct types of services offered, namely full train load and car load transportation. In full train load transportation, all cars of a train share the same origin and destination. Typical examples are trains transporting coal or ore. In car load traffic, a freight train consists of cars with various origins and destinations, often from different customers. Car load shipments are transported through a network of railway lines and classification yards (also known as marshalling or shunting yards) where the cars from inbound trains are arranged into new outbound trains. This paper considers the optimization of the operational tasks at the largest class of classification yards, so called *hump yards*, where cars are pushed over a hump in order to roll onto their respective classification track by means of gravity.

Fig. 1 shows the layout of a typical hump yard (Fig. 1a) and the basic shunting activities performed there (Fig. 1b). A hump yard typically consists of an *arrival yard*, a *hump*, a *classification bowl*, and a *departure yard*. Trains arriving to the shunting yard are parked on an available track at the arrival yard. The arrival time of a train i is denoted t_i. After the train has been parked on its designated arrival track, the cars of the train are inspected and decoupled before they are ready to be rolled over the hump into the classification bowl. The time at which a car g is rolled into the classification bowl is called the *roll-in time*, denoted t_g.

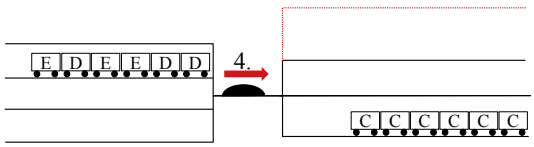
Once a car has been rolled into the classification bowl, it is typically parked on the classification track designated for its outbound train. Tracks that are used for compounding outbound trains are called *formation tracks*. An outbound train is ready to be rolled out of the classification bowl once all cars of the train have been coupled on its allocated



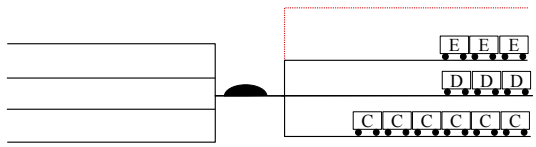
(a) Trains are rolled over the hump in arbitrary sequence. The train formation tracks have been allocated to trains A, B and C, while cars for trains D and E have to be mixed.



(b) A and B depart, freeing up tracks for trains D and E. The cars on the mixing track are pulled back to the arrival yard (a pull-back).



(c) The mixed cars are rolled in a second time (a roll-in).



(d) Final train formation of (C–E).

Figure 2: Illustration of multi-stage train formation of five trains (A–E) on four tracks through shunting movements (1–4). Movements 3 and 4 together constitute a pull-back operation. In real instances, freight trains arrive at and occupy arrival tracks during the train formation process, and are rolled in as needed, typically in arrival order.

classification track, and a compulsory brake test has been performed. The time at which a train b is rolled out from the classification bowl is called the *roll-out time*, and is denoted t_b. The train will then be parked on the departure yard until it is coupled to a line engine and departs. The departure time of an outbound train b is denoted t_b^{dep}.

High traffic volumes can make it impossible to reserve a classification track for the full time interval between the first roll-in and the roll-out time of an outbound train. Therefore, some of the tracks, called *mixing tracks*, are used as a buffer area where cars of different outbound trains may be temporarily stored. By temporarily storing cars, it is possible to build more trains simultaneously at a given yard, which helps in forming the overall transportation plan. Fig. 2 illustrates the mixing process. At given points in time, the cars on the mixing tracks are coupled and pulled back to the arrival yard (a *pull-back*). The cars are then once again rolled over the hump (a *roll-in*). Cars for which a train formation track has been reserved are directed there, while the remaining cars are again directed to the mixing track. The time from the start of a pull-back to the next pull-back is called a *stage*. As the pull-back and roll-in operations of mixed cars require manual effort (coupling, driving shunting engines, decoupling, etc.) and wear down switches and tracks, a natural objective is to minimize the total number of car pull-backs for all stages.

1.1 Related Work

Railway freight transportation is a classical application area of operations research; a recent overview is given by Nemani and Ahuja [36]. Much of the focus has been on railway freight in the U.S., where the infrastructure is typically owned by the operator and freight trains are longer and heavier and the market share for rail freight is higher than in Europe. For freight operations in particular, the *railroad blocking problem*, which considers the formation of a cost-minimal freight transportation plan, has been studied. In this problem, freight yards are considered on a macroscopic level, and as operations within the yard are not considered, solutions to this problem are not applicable for the classification problem considered here. An early model for the blocking problem was introduced by Bodin et al. [9], and since then various approaches have utilized dynamic programming [4], a successive shortest path method [40], a linear MIP model and Lagrangean relaxation [32], simulated annealing [29], genetic and tabu search [26], column generation [37], Lagrangean relaxation with column generation to solve the sub-problem [7] and VLSN (very large-scale neighborhood) search [2].

Operations research approaches for freight yard planning on a micro- or mesoscopic level, where the actual shunting operations are considered, have in comparison received less attention. Literature reviews are given by Boysen et al. [14] and Gatto et al. [23]. Single-stage methods, where cars are pushed over the hump exactly once, and all trains are formed directly on classification tracks without pull-back operations, are discussed by Krell [33]. As mixing requires cars to be pushed over the hump multiple times, single-stage methods are not applicable for the shunting problem considered in this paper.

Comparatively more research efforts have focused on multi-stage methods for arranging the cars in freight trains in a particular order. Given a sequence of n cars labeled from 1 to n, the general goal of the procedure is to form a sorted sequence of cars by roll-in and pull-back operations, similarly to what is considered in this paper. Algorithms that produce a desired sequence of cars without considering the inbound car order are called *sorting schemes* (39; see also 17). All sorting schemes construct a classification schedule and can be carried out without the aid of a computer, but they have different requirements regarding the number of re-classification steps and tracks needed. More recently, it has been studied how the order of the incoming cars could be utilized to minimize the number of pull-back operations [19, 31], as well as variants thereof [18]. Jacob et al. [30] introduced a binary encoding of sorting schedules and their requirements. This encoding was further used by Maue and Nunkesser [35] and Jacob et al. [31] to derive optimal sorting schedules, with regard to the number of pull-backs, on yards with a restricted number of tracks and classification tracks of sufficient length. However, the approach by Jacob et al. [31] is not applicable to the MSTF problem, as it violates the assumptions on sufficient track capacity, allow temporary storage and thereby more simultaneous trains than available classification tracks, and finally has additional constraints on the timely departure of trains.

Multi-stage train formation with mixing has been modeled, in preliminary versions of this paper, using a MIP model [11], a pure IP model [12], and utilizing column generation with branch-and-price [10]. This article summarizes the previous work and extends it

with a new arc-based model, rolling-horizon planning and analysis of yard capacity.

Finally, methods that incorporate robustness, and thus allow solutions to be valid even under disturbances such as delays and variances in car order, have also been considered. In particular, recoverable robustness [34], in which well-defined recovery actions can be taken to counter the effects of a disturbance, has been considered for freight shunting problems. In general, approaches for recoverable robustness have to be simplified substantially in order to be tractable, and are therefore mostly of theoretical value. The concept was first considered for shunting by Cicerone et al. [16] for two disruption types (a car being in the wrong place, and a new car being introduced) and three recovery strategies (not changing anything, changing the re-classification schedule for one car group, and changing the re-classification schedule for any number of car groups). Büsing and Maue [15] use the same concepts and present a general algorithm for finding a classification schedule that allows for a recovery strategy where k additional sorting steps can be inserted after the p first steps while minimizing the total number of re-classification steps in the original schedule.

1.2 Contributions

The contributions of the paper are summarized below.

1. A formal definition of the multi-stage train formation problem with non-uniform track lengths, which arises in a core part of the shunting yard planning process (Section 2). The problem models a real-world practice where tracks are reserved in advance for either train formation or for temporary storage of cars. Activity durations, non-uniform track lengths, timetabled arrivals and departures, as well as procedures for temporary storage of cars and their re-classification, are taken into account. By allowing temporary storage, the time cars spend on formation tracks can be shortened, thus allowing the simultaneous handling of more outbound trains than there are formation tracks on the yard. Schedules for the arrival yard, hump, and departure yard are assumed to be determined in advance.

2. An overview of the computational complexity of several variants of the multi-stage train formation problem with both non-uniform and uniform but sufficient track lengths. The results build on a proof of NP-completeness (Theorem 1) for the case where mixing capacity is either zero or unrestricted.

3. A column-based optimization model giving tight lower bounds (Section 3.1), and a branch-and-price approach capable of solving the problem to optimality in reasonable time (Sections 3.1.2 and 3.1.3).

4. A conceptually simpler arc-based optimization model (Section 3.2), with the same LP relaxation as the column-based model (Theorem 2).

5. A thorough evaluation of the models, based on a five-month data set from the largest hump yard in Scandinavia. The evaluation covers both the case when the

yard is initially empty, and a rolling horizon planning approach. Using the results, the computational performance of the methods and the capacity of the yard are analyzed. (Section 4).

1.3 Paper Outline

The rest of the paper is structured as follows. In Section 2, the core problem considered — the multi-stage train formation (MSTF) problem — is defined formally. In Section 3, two optimization models for the MSTF problem are presented. Section 4 describes the experimental setup, including the preprocessing and computation of the hump schedule, and provides results, including a comparison between the new and previous approaches, and an analysis of the relationship between the results and features of the input data. Finally, Section 5 concludes the paper with possible future research.

2 Problem Definition

In this section, the multi-stage train formation (MSTF) problem is formally defined in terms of the pairwise train schedulability. Finally, the complexity of the problem is discussed.

2.1 Problem Data

Assume sets of integers for the formation tracks $\mathcal{A} = \{1, 2, \ldots, |\mathcal{A}|\}$, stages $\mathcal{P} = \{1, 2, \ldots, |\mathcal{P}|\}$, car groups $\mathcal{G} = \{1, 2, \ldots, |\mathcal{G}|\}$, and outbound trains $\mathcal{B} = \{1, 2, \ldots, |\mathcal{B}|\}$. Formation tracks $a \in \mathcal{A}$ have length $l(a)$. There is in addition a single *mixing track*, not included in $\mathcal{A}$, which can temporarily store cars. The mixing track has length l^{mix}, and all cars on the mixing track are pulled back at the end of each stage. Stage 1 starts at $-\infty$ and ends at time t_1, all other stages p start at time t_{p-1} and end at time t_p. Further, the stages are totally ordered, so that for $p, p' \in \mathcal{P}$, if $p < p'$ then $t_p < t_{p'}$.

Car groups $g \in \mathcal{G}$ contain $\#g$ cars that are handled as a single unit with physical length $l(g)$, arriving to the classification bowl from the hump at time t_g to be formed into (outbound) train $b(g) \in \mathcal{B}$. The set $\mathcal{G}$ contain all car groups in the classification bowl during the planning period. Further, car groups are referred to simply as cars. A train b is rolled out from the classification bowl at time t_b. The set $\mathcal{B}$ contain all trains which have at least one car in $\mathcal{G}$ in the classification bowl during the planning period, and no two trains are rolled out from the bowl at the same time. Each train consists of a disjoint set of cars $\mathcal{G}(b) \subseteq \mathcal{G}$, i.e., $\mathcal{G}(b) := \{g \mid g \in \mathcal{G} \wedge b = b(g)\}$. The length $l(b)$ of a train b is the sum of the lengths of its cars: $l(b) := \sum_{g \in \mathcal{G}(b)} l(g)$.

2.2 Pairwise Train Scheduling and Sequences

To define the problem, a strict partial order $\prec$ on the set of trains $\mathcal{B}$ is used, denoting which pairs of trains $b \prec b'$ that can be scheduled in immediate succession on any single track, taking necessary technical setup times and previous assignments into account. More

formally, the relation is defined as follows.

$$b \prec b' \equiv t_b \leq t_{b'} - S_1 \wedge \forall g \in \mathcal{G}(b') : t_g < t_b \rightarrow \exists p \in \mathcal{P} : t_b < t_p < t_{b'} - S_2. \quad (1)$$

Eq. (1) states that for two trains b, b to be consecutively scheduled on a track, there must be a minimum roll-out setup time S_1 between the train roll-out times, and for all cars in b' arriving before the roll-out of b, there must exist a pull-back between the roll-out of b and the beginning of roll-out setup activities for b'. In S_2, both the pull-back duration and the roll-out setup time S_1 is included.

The virtual train u is used as predecessor to trains which are first on any track, and assume $u \prec b$ for any $b \in \mathcal{B}$. Further, the virtual train v is used as successor for trains which are last on any track, and assume $b \prec v$ for any $b \in \mathcal{B}$. Let $t_u = 0, t_v = \infty$ and $\mathcal{G}(u) = \mathcal{G}(v) = \emptyset$. Finally, assume $u \prec v$ in order to ensure transitivity. The relation is asymmetric as the unique roll-out times t_b define a fixed roll-out order which determines the order trains may be scheduled on a single track, and u never follows a train, while v never precedes one. Further, transitivity follows as the setup times are assumed to be constant and train-independent, $u \prec v$, and $u \prec b \prec v$ for any $b \in \mathcal{B}$.

Let a *sequence* s be a subset of trains, totally ordered by roll-out time, but also including the virtual train u as start and v as end. Two trains $b, b' \in \mathcal{B}$ appearing consecutively in this order in a sequence s is denoted by $(b, b') \in s$. For example, given a sequence $s = \langle u, b_1, b_2, b_3, v \rangle$, it holds that $(b_1, b_2) \in s$ and $(b_2, b_3) \in s$, but $(b_1, b_3) \notin s$. A sequence for which the relation $\prec$ holds for any two trains in immediate succesion is said to be *feasible*.

2.3 Pairwise Mixing

In this section, the mixing capacity usage and mixing cost of a train pair scheduled in immediate succession, are defined. For simplicity, it is assumed in this section that $t_0 = -\infty$. For two trains $b \prec b'$ and a period $p \in \mathcal{P}$, let $\mathcal{G}_p(b, b')$ denote the set of cars in b' which have to be mixed in p if both trains are scheduled in immediate succession on the same track:

$$\mathcal{G}_p(b, b') = \begin{cases} \{g \mid g \in \mathcal{G}(b') : t_g < \min(t_b, t_p)\} & \text{if } t_{p-1} < t_b, \\ \emptyset, & \text{otherwise.} \end{cases} \quad (2)$$

$l_p(b, b')$ then denote the total length of all such cars:

$$l_p(b, b') = \sum_{g \in \mathcal{G}_p(b,b')} l(g). \quad (3)$$

For the number of extra roll-ins, only the cost of decisions made in the current planning period are included. In other words, if a train b already has cars on a formation track when the planning period begins, the track allocation for b is *committed* and cannot be undone, and the fixed extra roll-ins incurred by this decision should not be included in the cost of the current planning period. That a train is committed implies that it is also

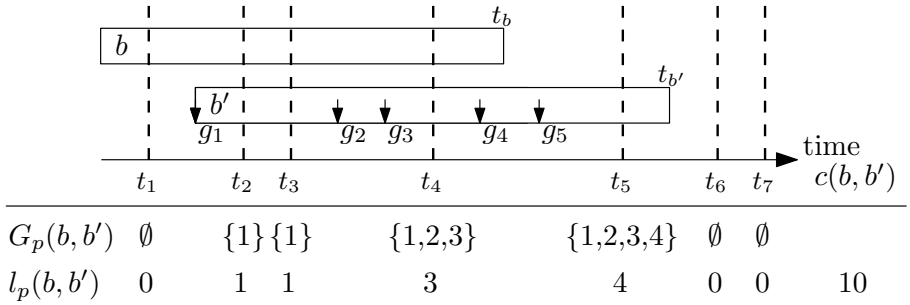


Figure 3: Calculation example for $\mathcal{G}_p(b,b')$, $l_p(b,b')$ and $c(b,b')$, assuming one car in each group, and a car length of one.

the first train on its track. So, let $c(b,b')$ denote the total number of extra roll-ins for two trains $b \prec b'$ scheduled in immediate succession on the same track:

$$c(b,b') = \begin{cases} 0, & \text{if } b' \text{ is committed,} \\ \sum_{p \in \mathcal{P}} \sum_{g \in \mathcal{G}_p(b,b')} \#g & \text{otherwise.} \end{cases} \tag{4}$$

An example illustrating the calculations above is shown in Fig. 3.

2.4 Formal Problem Definition

The MSTF problem can now be stated as follows. Partition the trains $\mathcal{B} \cup \{u, v\}$ into $|\mathcal{A}|$ sequences $\{\mathcal{B}(a) \mid a \in \mathcal{A}\}$ ordered by roll-out time, corresponding to trains allocated to tracks, such that

$$l(b) \leq l(a), \qquad \forall b \in \mathcal{B}(a) \setminus \{u, v\}, \tag{5}$$

$$\mathcal{B}(a) \text{ is a feasible sequence w.r.t. } \prec, \qquad \forall a \in \mathcal{A}, \tag{6}$$

$$\sum_{a \in \mathcal{A}} \sum_{(b,b') \in \mathcal{B}(a)} l_p(b,b') \leq l^{\text{mix}}, \qquad \forall p \in \mathcal{P}, \tag{7}$$

and such that

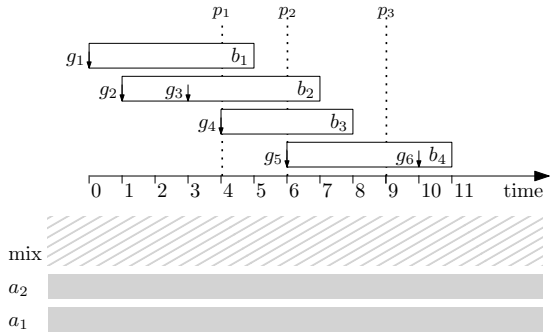
$$\sum_{a \in \mathcal{A}} \sum_{(b,b') \in \mathcal{B}(a)} c(b,b') \tag{8}$$

is minimized. Inequality (5) states that each train must fit on its allocated track, Condition (6) states that each ordered train pair allocated to the same track must be schedulable, and Inequality (7) states that for each stage, the length of the mixed cars must be shorter than the mixing track length. Finally, the objective is to minimize the total number of car roll-ins. As all cars are rolled in at least once, it is equivalent to instead minimize the number of extra car-roll-ins due to mixing only.

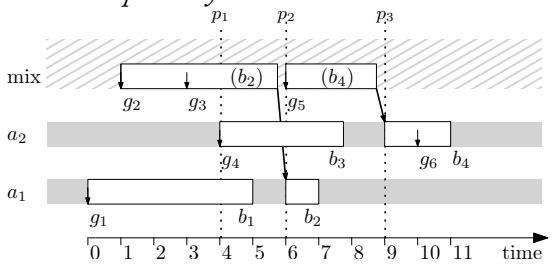
An example problem instance is illustrated in Figs. 4a and 4b. In the example, four trains are to be allocated to two formation tracks a_1 and a_2. All trains fit on the longest track a_2, but only the first three trains fit on the shorter track a_1. The mixing capacity

Train data			$c(b,\cdot)$			Cars	
b	$l(b)$	t_b	b_2	b_3	b_4	g	t_g
b_1	1	5	4	1	0	g_1	0
b_2	1	7			1	g_2	1
						g_3	3
b_3	1	8			1	g_4	4
b_4	2	11				g_5	6
						g_6	10

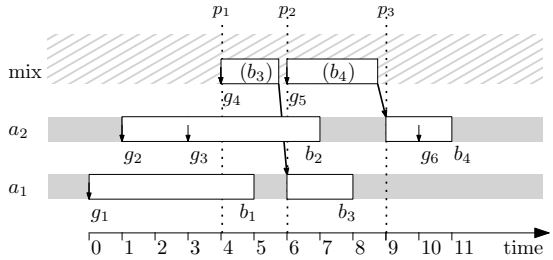
(a) Problem instance data for trains (length, roll-out time and mixing) and cars (arrival time). There are two tracks a_1, a_2 with length 1 and 2 respectively.



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



(c) Feasible solution, with car g_2 and g_3 mixed in two stages each, and g_5 in one, for a total cost of five extra car roll-ins.



(d) Optimal solution, with car g_4 and g_5 mixed in one stage each, for a total cost of two extra car roll-ins.

Figure 4: Example problem illustration (4a and 4b) and two solutions (4c and 4d) for an instance with two formation tracks a_1, a_2, departing trains $b_1 - b_4$, car arrivals $g_1 - g_6$, and three stages $p_1 - p_3$. The mixing cost for consecutive train pairs is also shown.

is assumed to be infinite and pull-backs are performed at times 4, 6 and 9. Traversing the trains in order of roll-out time, all trains can succeed all earlier trains, as defined by $\prec$, except train b_3 which cannot follow b_2 due to the lack of a stage start in between their roll-outs. All car groups consist of one car, albeit of different lengths. Figs. 4c and 4d show two feasible solutions in which the resulting number of extra car roll-ins are five and two respectively.

2.5 Complexity Analysis

This section shows that the feasibility version of the MSTF problem is NP-complete and that MSTF variants can be seen as coloring problems of intervals, where an interval models the presence of the cars of a particular train on a classification track. The following equivalent reformulation of the MSTF problem is used. The MSTF problem is to determine, for each train b, a track and an active time interval J_b, during which the track is reserved exclusively for the formation of that train.

The roll-in time of the earliest car g_1 in b and its bowl roll-out time form an interval $I_b = (t_{g_1}, t_b)$ in which all cars of b arrive at the classification bowl, which is also the longest interval that b ispossibly active. It is required w.l.o.g. that J_b is a suffix of I_b, so that $J_b = (x, t_b)$ where $t_{g_1} \leq x < t_b - S$ and S is the total duration of technical setup activities and an allowance for necessary pull-backs. For any train b, the minimal suffix J_b^- is the shortest suffix during which b is active in any feasible solution. Further, two intervals $J_b, J_{b'}$ must not intersect whenever b and b' are on the same formation track.

The MSTF problem thus translates to assigning a track of sufficient length and a suffix J_b to every outbound train b, where $J_b^- \subseteq J_b \subseteq I_b$ and no two suffixes of trains allocated to the same track intersect. If suffixes have been decided, then the problem of assigning tracks of sufficient length to the outbound trains can be seen as a list-coloring problem of the intervals. In this, each train b has a list L of classification tracks that the train fits on, each track represents a color, and every interval J should be colored with a color from the list L such that any two overlapping intervals J, J' are assigned different colors.

In general, the list-coloring problem of intervals is NP-complete. The lists in the MSTF problem do not have an arbitrary structure, as the trains and tracks can be ordered in increasing length. The resulting list-coloring problem is called a μ-coloring problem, which for interval graphs unfortunately also is NP-complete [see 13]. As a consequence, the following theorem, first appearing in Bohlin et al. [11], is obtained:

Theorem 1 *Feasibility in the MSTF problem is NP-complete when either 1) there is no mixing track, or 2) the mixing track has unlimited capacity.*

Proof of Theorem 1 If no mixing track is present, no car can be mixed, and thus $I_b = J_b^- = J_b$. If mixing capacity is unlimited, then the problem is feasible if and only if there exists a feasible track assignment where $J_b = J_b^-$. In both cases, any instance of the μ-coloring problem can be trivially translated to a corresponding MSTF instance. □

From the proof for these two special cases, it follows trivially that the general MSTF problem is also NP complete. If however each train fits on each track and there is no mixing track, then the problem reduces to the problem of interval graph coloring, which is well-known to be polynomially solvable by a simple greedy algorithm [see 25]. The optimization variant where the objective is to minimize the total number of car pull-backs can also be solved in polynomial time, by solving an assignment problem [see 11].

The complexity results for other variants of the MSTF problem are shown in Table 1. The variants where track length are all of the same length and can accommodate all trains is equivalent to the case where track lengths are infinite. Worth noting is that the question of whether the problem variant with finite mixing capacity but infinite formation track lengths is in P or not is currently open.

3 Optimization Models

In this section, we develop three different integer programming models for the mixing problem. The first model, D-IP, is a direct model based on the problem formulation in

Mixing capacity	Formation track lengths	Class	Comment
0	non-uniform	NP-Complete	See Theorem 1.
∞	non-uniform	NP-Complete	See Theorem 1.
finite	non-uniform	NP-Complete	Trivial from 0/non-uniform variant.
0	∞	P	Solved as an interval graph coloring problem.
∞	∞	P	Solved as an assignment problem in [11].
finite	∞	Unknown	Currently open.

Table 1: Complexity results for variants of the MSTF problem.

Section 2. The second model, CG-IP, instead uses a sequence-based representation, and is solved using column generation. The third model, AI-IP, is a reformulation of the second model as an arc-based integer program.

3.1 Extended Formulation Solution

In this section an extended formulation first introduced in [10] is described and modified for rescheduling in a rolling horizon setting.

3.1.1 Model.

The extended formulation is based on binary variables representing feasible sequences of trains on a single track. A feasible solution is a train sequence for each formation track, where all trains fit on the track and are compatible according to $\prec$. Further, each train should be present on exactly one track. Formally, sequence $s \in \mathcal{S}$ is a set of totally ordered trains, starting with the virtual train u and ending with the virual train v. Let $\mathcal{S}(a)$ denote the set of feasible sequences of trains which all fit on track a. The number of mixed cars in a sequence s can then be calculated as

$$c(s) = \sum_{(b,b') \in s} c(b, b').$$

Similarly, the length of mixed cars in a certain stage p can be calculated as

$$l_p(s) = \sum_{(b,b') \in s} l_p(b, b').$$

The following integer program models the MSTF problem problem using variables x_{sa}, which encode whether sequence s is chosen for track a or not:

$$\min \quad \sum_{a\in\mathcal{A}} \sum_{s\in\mathcal{S}(a)} c(s)\cdot x_{sa}, \tag{9}$$

$$\text{s.t.} \quad \sum_{a\in\mathcal{A}} y_{ba} \geq 1, \qquad b\in\mathcal{B}, \tag{10}$$

$$\sum_{\substack{s\in\mathcal{S}(a):\\ b\in s}} x_{sa} \geq y_{ba}, \qquad b\in\mathcal{B}, a\in\mathcal{A}, \tag{11}$$

$$\sum_{s\in\mathcal{S}(a)} x_{sa} \leq 1, \qquad a\in\mathcal{A}, \tag{12}$$

$$\sum_{a\in\mathcal{A}} \sum_{s\in\mathcal{S}(a)} l_p(s)\cdot x_{sa} \leq l^{\text{mix}}, \qquad p\in\mathcal{P}, \tag{13}$$

$$x_{sa}, y_{ba} \in \{0,1\}, \qquad s\in\mathcal{S}, a\in\mathcal{A}, b\in\mathcal{B}. \tag{14}$$

The objective function (9) counts the total number of extra roll-ins as the sum of the roll-ins for the sequences selected for each track. The variable y_{ba} encodes that train b shall be placed on track a, and is included for use in the IP solver during branching. Inequalities (10) and (11) ensure that each train appears in at least one sequence. If a train appears several times in an optimal solution, then, since the pairwise train cost $c(b,b')$ is always non-negative, the train can be removed from all but one sequence without affecting the cost. Inequalities (12) state that at most one sequence per track can be used, and inequalities (13) ensure that at most the available mixing capacity is used in any stage. Inequalities (10)–(13) are equivalent to the conditions (5)–(7), in Section 2, for feasibility of schedules.

In the model above, there is one x variable for each combination of sequence $s\in\mathcal{S}(a)$ and track $a\in\mathcal{A}$. As the size of $\mathcal{S}(a)$ is of order $\mathcal{O}(|\mathcal{B}|!)$, it is only possible to work with a subset of the x variables. Column generation [see for example 24, 22, 20] is used to generate new variables as needed in every node of the branch-and-bound algorithm. Such algorithms are referred to as branch-and-price algorithms; see Barnhart et al. [8] for a general description and Desrosiers et al. [21] for a survey of specialized branch-and-price algorithms for routing and scheduling.

3.1.2 Pricing.

The identification of necessary new variables is called pricing and forms a combinatorial sub-problem of its own. To identify a missing variable x_{sa}, a corresponding dual constraint, violated by the current solution, needs to be found. The dual of the LP relaxation of (10)–(14) is

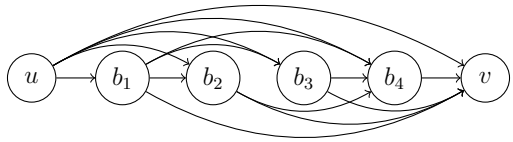


Figure 5: Pricing graph for track a_2 in the example in Fig. 4.

$$\max \quad \sum_{b\in\mathcal{B}} \alpha_b + \sum_{a\in\mathcal{A}} \gamma_a + l^{\text{mix}} \sum_{p\in\mathcal{P}} \delta_p, \tag{15}$$

$$\text{s.t.} \quad \sum_{b\in s} \beta_{ba} + \gamma_a + \sum_{p\in\mathcal{P}} l_p(s)\cdot\delta_p \le c(s), \qquad a\in\mathcal{A}, s\in\mathcal{S}(a), \tag{16}$$

$$\alpha_b \le \beta_{ba}, \qquad b\in\mathcal{B}, a\in\mathcal{A}, \tag{17}$$

$$\alpha_b, \beta_{ba} \ge 0, \tag{18}$$

$$\gamma_a, \delta_p \le 0. \tag{19}$$

The dual α variables correspond to primal inequalities (10), β to (11), γ to (12) and δ to (13). For any x_{sa} the corresponding dual constraint for the problem in (9)–(14) is

$$\sum_{b\in s} \beta_{ba} + \gamma_a + \sum_{p\in\mathcal{P}} l_p(s)\cdot\delta_p \le c(s), \qquad a\in\mathcal{A}, s\in\mathcal{S}(a).$$

The problem can be solved separately for each track a by identifying a sequence s where

$$\sum_{b\in s} \beta_{ba} + \sum_{p\in\mathcal{P}} l_p(s)\cdot\delta_p - c(s) > -\gamma_a,$$

which can be done by solving the following maximization problem:

$$\max_{s\in\mathcal{S}(a)} \sum_{b\in s} \beta_{ba} + \sum_{p\in\mathcal{P}} l_p(s)\cdot\delta_p - c(s). \tag{20}$$

As shown in Bohlin et al. [10], this is equivalent to finding a longest path in a directed acyclic graph. This could for example be done using the reaching algorithm [see 3, Section 4.4] in $\mathcal{O}(|\mathcal{B}|^2)$ time.

The directed acyclic graph $G = (V, E)$ is constructed as follows. There is a node for every train which fits on a plus a source u and a sink v, so that $V = \{b \in \mathcal{B} \mid l(b) \le l(a)\} \cup \{u, v\}$. For every train b there are edges (u, b), (b, v) and (u, v) as well as (b, b') for all other trains where $b \prec b'$. Fig. 5 shows the resulting graph for track a_2 for the example in Fig. 4. Edge weights for the directed acyclic graph are chosen as follows:

$$w_{b,b'} = \begin{cases} 0, & \text{if } b' = v, \\ \beta_{b'a} + \sum_{p\in\mathcal{P}} l_p(b', b)\cdot\delta_p - c(b', b), & \text{otherwise.} \end{cases} \tag{21}$$

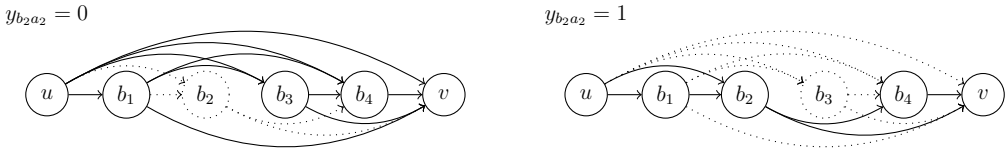


Figure 6: Modified pricing graphs from Fig. 5 after branching on $y_{b_2a_2} = 0$ (left) and $y_{b_2a_2} = 1$ (right). Dotted nodes and edges are removed from the pricing problem.

Any path from u to v corresponds to a feasible sequence of cars corresponding to the visited nodes. The cost on the edges used is equal to the cost in Equation (20), so finding a maximum (u, v)-path solves our pricing problem.

3.1.3 Branching and Rolling Horizon.

As the branching in the branch-and-price search tree is performed on the y_{ba} variables, the branching decisions made need to be reflected in the pricing problem such that only feasible variables are generated. Furthermore, when planning using a rolling horizon approach (see Section 4) it is necessary to fix the track allocations for cars already on a formation track, and hence the corresponding y_{ba} variables. Both of these constraints can be handled via the removal of nodes and edges from the pricing graph as follows.

Case $y_{ba} = 0$**:** Remove node b from V and all edges connected to it from E, so that sequences containing train b cannot be generated.

Case $y_{ba} = 1$**:** For all nodes b' and b'' where $t_{b'} \prec t_b \prec t_{b''}$, remove (b', b''), (u, b''), (b', v) and (u, v) from E. In addition, remove all nodes b' for which neither $b' \prec b$ nor $b \prec b'$ along with all edges connected to them. Thereby, all paths from a node before b to one after b will include b, forcing b to be contained in every generated sequence.

Fig. 6 illustrates the two possible modifications of the graph from Fig. 5 when branching on $y_{b_2a_2}$.

3.2 An Arc-Indexed Integer Programming Formulation

Although the branch-and-price algorithm in Section 3.1 can solve the problem efficiently, it may still be possible to achieve similar performance using a compact IP model. In this section, such a compact IP model based on an arc formulation of the extended formulation in Section 3.1 is presented. The model was first introduced in Güçlü [27] and is easier to implement and maintain than CG-IP. Furthermore, the LP relaxation of the new model yields the same lower bound as the extended formulation from Section 3.1.

The model is built around threefold indexed variables $x_{bb'a}$, encoding that train b' is scheduled immediately after train b on track a. Only variables $x_{bb'a}$ defining feasible train sequences and train-to-track allocations are included in the problem, which corresponds

directly to the edge set in the directed acyclic graph for the pricing subproblem, as defined in Section 16. Furthermore, the source u and sink node v from the pricing graph are included as virtual starting and final train on a track. Formally, the variable assignment $x_{ub'a} = 1$ represents that b' is the first train in the schedule on track a, and $x_{bva} = 1$ represents that b is the last train on track a. Let $\mathcal{B}_a$ be the set of trains which fit on track a, including virtual trains u and v. The MSTF problem can now be formulated as the following binary program:

$$\min \quad \sum_{a \in \mathcal{A}} \sum_{\substack{b,b' \in \mathcal{B}_a: \\ b \prec b'}} c(b,b') \cdot x_{bb'a} \tag{22}$$

$$\text{s.t.} \quad \sum_{\substack{a \in \mathcal{A}: \\ b' \in \mathcal{B}_a}} \sum_{\substack{b \in \mathcal{B}_a: \\ b \prec b'}} x_{bb'a} \geq 1, \qquad b' \in \mathcal{B}, \tag{23}$$

$$\sum_{b' \in \mathcal{B}} x_{ub'a} \leq 1, \qquad a \in \mathcal{A}, \tag{24}$$

$$\sum_{a \in \mathcal{A}} \sum_{\substack{b,b' \in \mathcal{B}_a: \\ b \prec b'}} l_p(b,b') \cdot x_{bb'a} \leq l^{\text{mix}}, \qquad p \in \mathcal{P}, \tag{25}$$

$$\sum_{\substack{b \in \mathcal{B}_a: \\ b \prec b'}} x_{bb'a} = \sum_{\substack{b \in \mathcal{B}_a: \\ b' \prec b}} x_{b'ba}, \quad a \in \mathcal{A}, b' \in \mathcal{B}_a \setminus \{u,v\}, \tag{26}$$

$$x_{bb'a} \in \{0,1\}, \qquad a \in \mathcal{A}, b, b' \in \mathcal{B}_a. \tag{27}$$

The objective (22) is again counting the total number of extra roll-ins (cf. Section 3.1). Inequality (23) requires every train to be scheduled in at least one sequence, i.e., it must follow either the virtual train u or some other train on any track. Inequality (24) ensures that at most one sequence of trains starts on any track, while inequality (25) ensures that no more mixing length than available is used in any given stage. Finally, inequality (26) is the flow conservation constraint that ensures that every train has the same number of predecessors and successors on every track. For rolling horizon planning, a train b that is already being built on a formation track a is restricted to only fit on that track, and can only follow the virtual start train u. That is, the only allocation variable for train b is x_{uba} and $b \in \mathcal{B}_a$ only.

The compact model should ideally have a strong LP relaxation. The following Theorem states that the compact model has the same LP relaxation as the model in Section 3.1.

Theorem 2 *AI-IP and CG-IP yield the same LP relaxation.*

The proof is given in Appendix A.

4 Experiments

The two optimization models were experimentally compared with two previously developed methods: a heuristic approach based on interval graph coloring, and a straightfor-

ward integer programming model [see 12]. The experiments were run on a data set from the Hallsberg shunting yard in Sweden, which is the largest hump yard in Scandinavia. The yard has 8 arrival tracks, two parallel humps (of which only one is used), 32 classification tracks, and 12 departure tracks. The lengths of the arrival tracks range from 595 m to 693 m, the classification tracks range from 374 m to 760 m, and the departure tracks from 562 m to 886 m. The layout of the yard is shown in Fig. 7. Apart from the tracks mentioned above there are also some additional tracks that are normally not used for shunting, e.g., tracks going to repair facilities. Timing estimates for all tasks, e.g., setup times, roll-ins, pull-backs, etc., were taken from Averstad [5]. Two of the 32 available classification tracks were pre-allocated for mixing only, leaving 30 formation tracks.

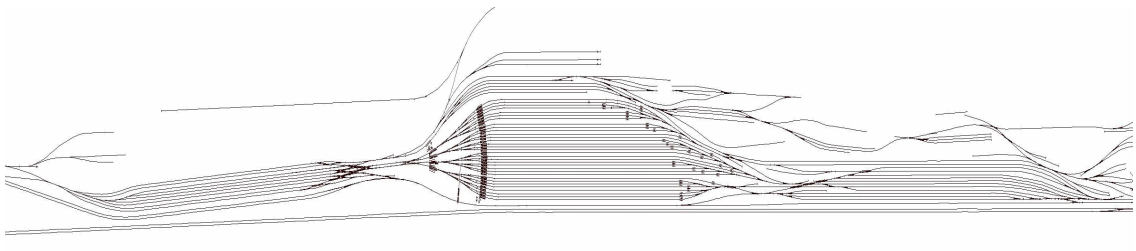


Figure 7: Layout of Hallsberg shunting yard in Sweden. The arrival yard is on the left, followed by the two humps, classification tracks, and finally the departure yard on the right. The image is taken from Averstad [6] and has been scaled to aid visibility (in reality the yard is approximately 3.5 km long and 0.25 km wide).

4.1 Preprocessing

Historical traffic data was provided by the Swedish Transport Administration (Trafikverket in Swedish) and spanned a 5-month period from December 2010 to May 2011. The data contained the arrival times of all inbound trains and departure times of all outbound trains, as well as the cars on each train and their length.

The data exhibited ambiguity and inconsistencies in the form of missing and duplicate train and car entries. Yard staff stated that cars often were rebooked on new freight trains, and both the old and new entries could be present simultaneously in the data, which contained 1.37 times as many car departures as arrivals. Furthermore, some cars would have failed the arrival inspection and been sent to the repair facilities rather than the classification bowl, but no indication of this existed. The arrival and departure yard track allocations as well as the hump schedule were also not provided and had to be constructed.

The following sections contains a brief explanation of how problem instances were generated from the available data.

4.1.1 Filtering and Matching Cars to Trains.

Each arriving car was matched to the earliest associated train departing between 140 and 2880 minutes (48 hours) after the arrival time. The lower limit was due to regulations

of the yard, and the upper limit was chosen in cooperation with the yard staff to exclude cars that were sent to repair facilities and therefore not present in the classification bowl. Furthermore, trains were trimmed to the length of the longest formation track, and surplus cars were removed from the problem. All in all, 32% of the car arrivals and 51% of the car departures were removed. Green Cargo considered the resulting data set to be of realistic size.

4.1.2 Creating a Hump Schedule.

The problem definition assumes that the hump-schedule (i.e., roll-ins and pull-backs) and roll-out times are given as input data, as deciding roll-ins, pull-backs and roll-outs are already part of the daily planning of the yard. However, these data were not included in the example traffic data, and therefore sensible allocations and schedules had to be generated. This section briefly discusses the algorithms used for deciding roll-in, pull-back and roll-out times. For a more detailed description we refer to Appendix B.

The goal of the algorithms was to construct a hump schedule and roll-out times that were likely to define a feasible optimization problem. This was accomplished by making rough arrival and formation track allocations, and then constructing a pull-back schedule that ensured that every mixed car could be moved to its allocated formation track in time. The optimization problem defined was feasible given infinite mixing track capacity.

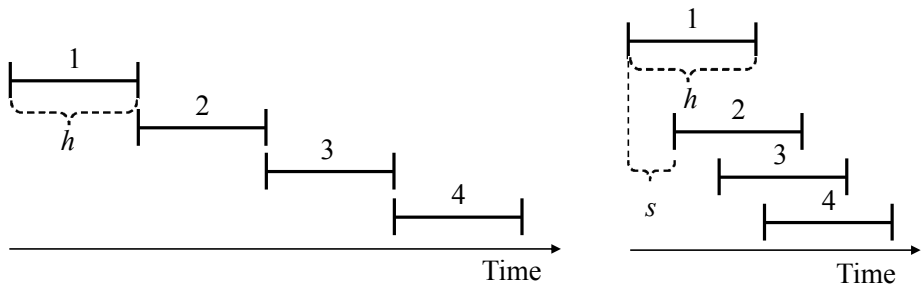
As both roll-ins and pull-backs occupy the hump they had to be scheduled together such that no two hump-actions overlapped in time. Therefore the algorithms used time intervals rather than exact time points up until the last stage, when a hump schedule with exact roll-in and pull-back times was constructed based on the constraints imposed by these intervals. If this final consolidation of hump actions failed, the trains served by a pull-back that could not be scheduled were delayed, and another iteration of the algorithms was required. Likewise, sometimes the arrival of a train had to be delayed as the arrival yard was overloaded.

Appendix B.1 presents the algorithm used for scheduling the arrival yard and finding feasible roll-in intervals for all arriving trains. Appendix B.2 describes the classification bowl scheduling which returns pull-back intervals and roll-out times. Finally, Appendix B.3 outlines how roll-ins and pull-backs were combined into a hump-schedule.

After pre-processing, the data set included 3606 arrivals, 3653 departures, and 18 366 car groups (making up 61 593 individual cars). The length of the inbound trains varied in the range 12.8 m to 929 m and outbound trains in the range 12 m to 759.1 m. When extracting associations between inbound and outbound trains, cars had to be discarded from eight outbound trains in order to stay below the maximum track length of 760 meters. When computing the hump-schedule and roll-out times, 0.9 % of the inbound trains had to be delayed for in total 84 minutes and 0.1 % of the outbound trains for in total 62 minutes.

4.1.3 Problem Instances.

At classification yards, planning is normally done on a daily basis for a few days ahead. To evaluate the performance of the implemented methods, the problem data set was divided into smaller instances of h days each. Two instance types where generated: *independent instances* where it was assumed that no cars were present at the yard in the beginning of the planning period, and *rolling-horizon instances* where cars from previous days were already present in the yard. In this set-up, a full plan was generated by incrementally fixing a prefix of s days out of the h days available, using the state after the prefix as input for the next instance. The goal of the experiments on the independent instances was to evaluate performance before implementing rolling-horizon planning for the most promising methods. Fig. 8 is a graphical representation of how the problem instances of the two different set-ups were distributed in time.



(a) Independent instances *(b) Rolling-horizon instances*

Figure 8: A graphical representation of independent and rolling-horizon problem instances.

The planning horizon h was varied between two and five days in one-day increments. A step size s of one day was used in the rolling horizon approach.

Table 2: Instance data for the different planning set-ups.

Instance type	*Independent*				*Rolling horizon*			
Horizon h (days)	2	3	4	5	2	3	4	5
No. instances	75	50	37	30	150	149	148	147
Avg. no. trains	48.7	73.0	97.7	121.7	69.4	93.9	118.4	143.0
Avg. no. groups	244.9	367.3	492.0	612.2	366.6	490.5	614.6	738.6

Data on the resulting problem instances is shown in Table 2, where the following attributes are presented:

No. instances The total number of instances in each data set.

Avg. no. trains The average number of outbound trains in each data set.

Avg. no. groups The average number of car groups in each data set (note that each car group represent several physical cars).

4.2 Optimization Set-up

The column generation integer program (CG-IP) and the arc-based integer program (AI-IP), as well as two previously developed methods, D-IP [12] and Heuristic [11], were included in the experiments. D-IP is a binary integer program based on choosing an allocation start time and a train formation track for each outbound train, and Heuristic is a clique-based heuristic followed by an improvement reassignment heuristic. As these methods did not perform satisfactory in previous experiments [see 10] they where not adapted for rolling horizon planning.

The pre-processing, heuristics and experimental framework were implemented using Python 2.7.2 and CPLEX 12.5.0.0 was used as the LP solver. CPLEX was configured to use a single thread of execution. A multiple thread setting was tested for AI-IP but this did not improve the results. Further, SCIP 3.0.1 [see 1] was used as the branch-and-price framework for CG-IP, with the pricing sub-problem in Section 3.1.2 implemented in C++ using GCC 4.6.2. All experiments were run on identical Linux workstations with eight Intel Core i7-2600 quad-core CPUs running at 3.4 GHz and equipped with 16 GB of RAM.

For the independent problem set-up a limit of 20 minutes, the time of an extended coffee break, was set on the optimization in CPLEX and SCIP, after which the best integer solution found was returned. Further, the solution from Heuristic was used to hot-start the optimizations. For the rolling horizon set-up, no time limit was set, and no hot-start provided.

4.3 Computational Results

This section presents an execution time analysis of the various solution methods on the two sets of problem instances.

Table 3 provides results for all planning set-ups. Only the results relevant for a certain solution method have been included. The following results are reported:

Optimal The number of instances for which an optimal solution was returned (all methods).

Subopt avg. Δ The average distance to optimality for suboptimal solutions (only D-IP and Heur).

Feasible The number of feasible solutions found (only Heur).

Feas. $LB > 0$ The number of feasible solutions where the maximum lower bound was positive (only D-IP).

Table 3: Computational results for the different solution methods and planning set-ups.

Instance type		*Independent*				*Rolling horizon*			
	Horizon h (days)	2	3	4	5	2	3	4	5
Heuristic	Optimal (#)	54	31	18	13	–	–	–	–
	Subopt avg. Δ	7.6	21.0	17.9	13.7	–	–	–	–
	Feasible (#)	73	47	34	27	–	–	–	–
	No sol. found (#)	2	3	3	3	–	–	–	–
	Avg. time (s)	0.1	0.1	0.1	0.2	–	–	–	–
	Max. time (s)	0.4	0.3	0.4	0.5	–	–	–	–
D-IP	Optimal (#)	75	42	29	18	–	–	–	–
	Subopt avg. Δ	–	4.8	7.6	3.8	–	–	–	–
	Opt. proven (#)	48	27	14	12	–	–	–	–
	Feas. LB > 0 (#)	6	3	5	1	–	–	–	–
	Feas. LB $= 0$ (#)	21	20	17	14	–	–	–	–
	No sol. found (#)	0	0	1	3	–	–	–	–
	Avg. time (s)	380.5	531.7	671.7	672.1	–	–	–	–
	Max. time (s)	1365.3	1266.9	1273.9	1259.9	–	–	–	–
CG-IP	Optimal (#)	75	50	37	30	150	149	148	147
	Opt. proven (#)	75	50	37	30	150	149	148	147
	Avg. time (s)	2.0	18.4	76.4	202.9	7.6	31.9	117.7	317.3
	Max. time (s)	16.1	134.8	686.1	1065.1	45.8	355.6	1033.6	2472.2
AI-IP	Optimal (#)	75	50	37	30	150	149	148	147
	Opt. proven (#)	75	50	37	30	150	149	148	147
	Avg. time (s)	3.1	11.5	33.2	94.2	5.4	16.1	41.7	85.9
	Max. time (s)	12.1	58.4	151.2	466	21.7	93.3	284.5	338.9

Feas. LB = 0 The number of feasible solutions where the maximum lower bound was zero (only D-IP).

Opt. proven The number of instances where optimality was proven (all methods but from Heur).

No sol. found The number of instances where no feasible solution was found (only Heur and D-IP).

Avg. time The average run time in wall-clock[1] seconds (all methods).

[1]The sum of CPU time, I/O time, and delays due to communication.

4.3.1 Independent Instance Results.

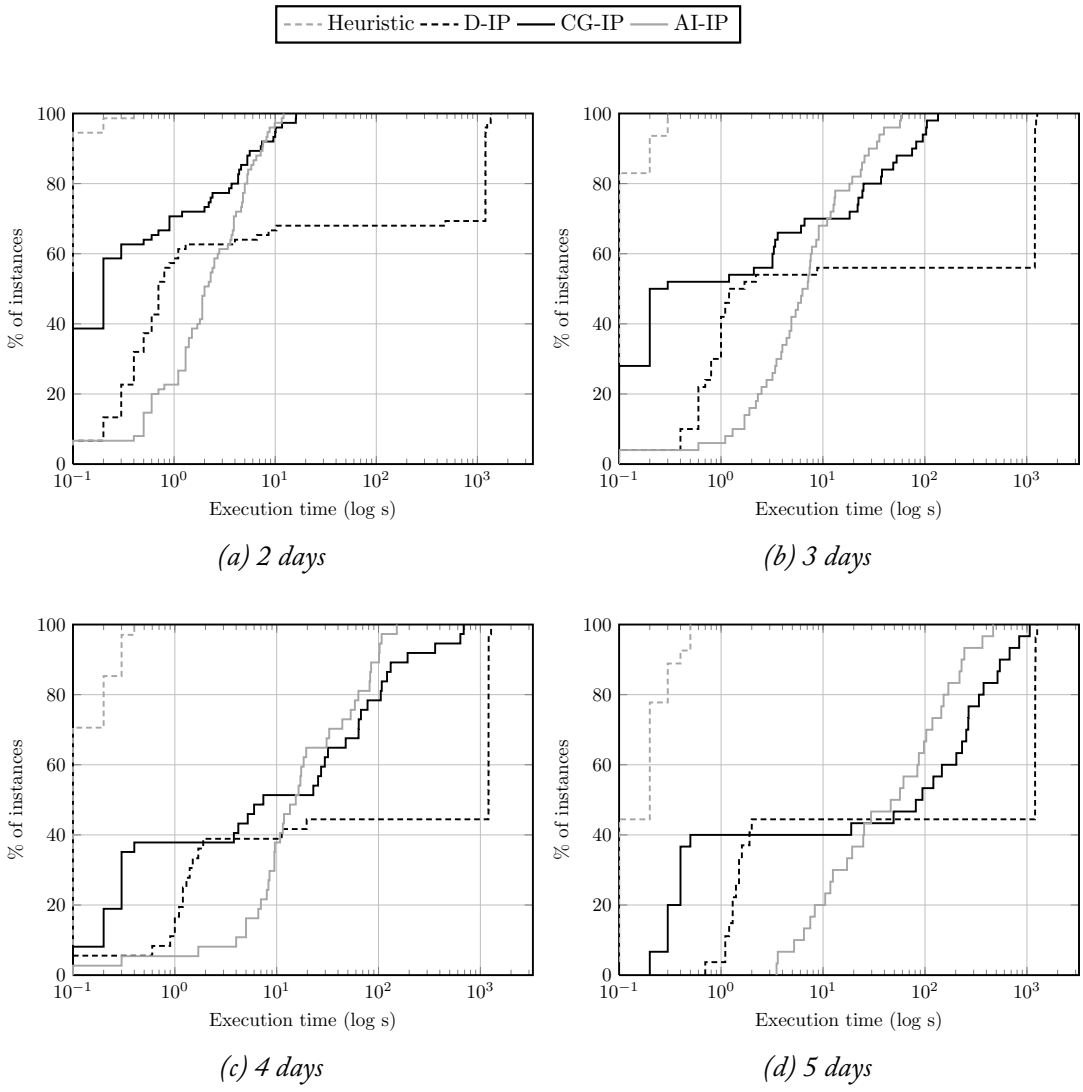


Figure 9: Execution time graphs for the independent problem instance set-up, in logarithmic time scale.

For the independent instances, all methods were evaluated on the data set, with the intention to determine which methods were suitable for rolling horizon implementation. As can be seen in Table 3, for the independent instances, Heuristic returned an optimal solution for 116 out of 192 instances in total, with an execution time below 0.5 s. However, the method failed to return even a feasible schedule for 11 instances. The average difference in number of extra roll-ins (ER) between the heuristic suboptimal solutions and the optimal solutions, varied between 7.6 and 21 roll-ins. As the method failed to provide even feasible solutions, it was not further considered for rolling-horizon planning.

D-IP returned optimal solutions in 164 out of 192 instances in total. For the suboptimal solutions, the average difference in ER from the optimal solutions were lower than for Heuristic for all planning horizons. For D-IP, optimality was proven in 101 of the instances. For the remaining 91 instances, the trivial lower bound of zero was reported in 72 instances. Further, for four instances, D-IP failed to return a feasible solution within the given time limit.

For both CG-IP and AI-IP, proven optimal schedules were always returned within the execution time limit, and for the three problem sets with the longest planning horizons, AI-IP did so in less time than CG-IP, on average. Altogether, this indicates that D-IP yields a weak LP relaxation in comparison to the two newer models (CG-IP and AI-IP), and was therefore not further considered for rolling horizon planning.

Fig. 9 plots the percentage of finished computations versus execution time. A logarithmic scale is used for the execution time. The optimization methods (D-IP, CG-IP and AI-IP) terminated either at proven optimality or at the execution time limit, while the heuristic terminated when a schedule had been generated.

As can be seen in Fig. 9, Heuristic always executed in less than 1 s. The execution times of D-IP are clustered around 1 s to 2 s or around 1200 s. For D-IP, either an optimal schedule was found quickly (typically when mixing was not required and the hot-start schedule was optimal) or else, the time limit was reached and optimality was not proven. D-IP finished executing within 10 s in 104 instances, and 100 of these did not require mixing.

The general trend for CG-IP and AI-IP was that the majority of the instances were solved fairly quickly, while in particular the larger problem instances took more time to solve. In particular, for CG-IP and the five-day instance set shown in Fig. 9d, approximately 80 % of the instances are solved in less than 400 s each, while the remaining 20 % of the instances take 400 s to 1065 s. The situation is similar for the smaller problem instances, although the execution time is shorter.

There is also a difference between CG-IP and AI-IP for short execution times. For example, CG-IP solves more than 38 % of instances within 1 s regardless of size, while AI-IP have not solved a single five-day instance in the same time. By examining the log files it became clear that the disperse execution times were caused by CPLEX taking time to solve the relaxed root problem for AI-IP. In all instance sets, AI-IP solves at least as many instances as CG-IP given a time budget of 30 s or more.

4.3.2 Rolling Horizon Results.

The most promising methods, CG-IP and AI-IP, were chosen for implementation in a rolling-horizon setting. The goal of the rolling horizon experiments were to compare the computational performance to optimality for the two methods.

As no time limit was set, an optimal schedule was always returned by both methods and for all instances, as can be seen in Table 3. The time to optimality was always less than 42 minutes for CG-IP and less than 6 minutes for AI-IP. Further, the mean time to optimality for AI-IP was 31.05 s compared to 117.8 s for CG-IP. Execution time graphs for the rolling-horizon set-up are shown in Fig. 10. The graphs from the independent

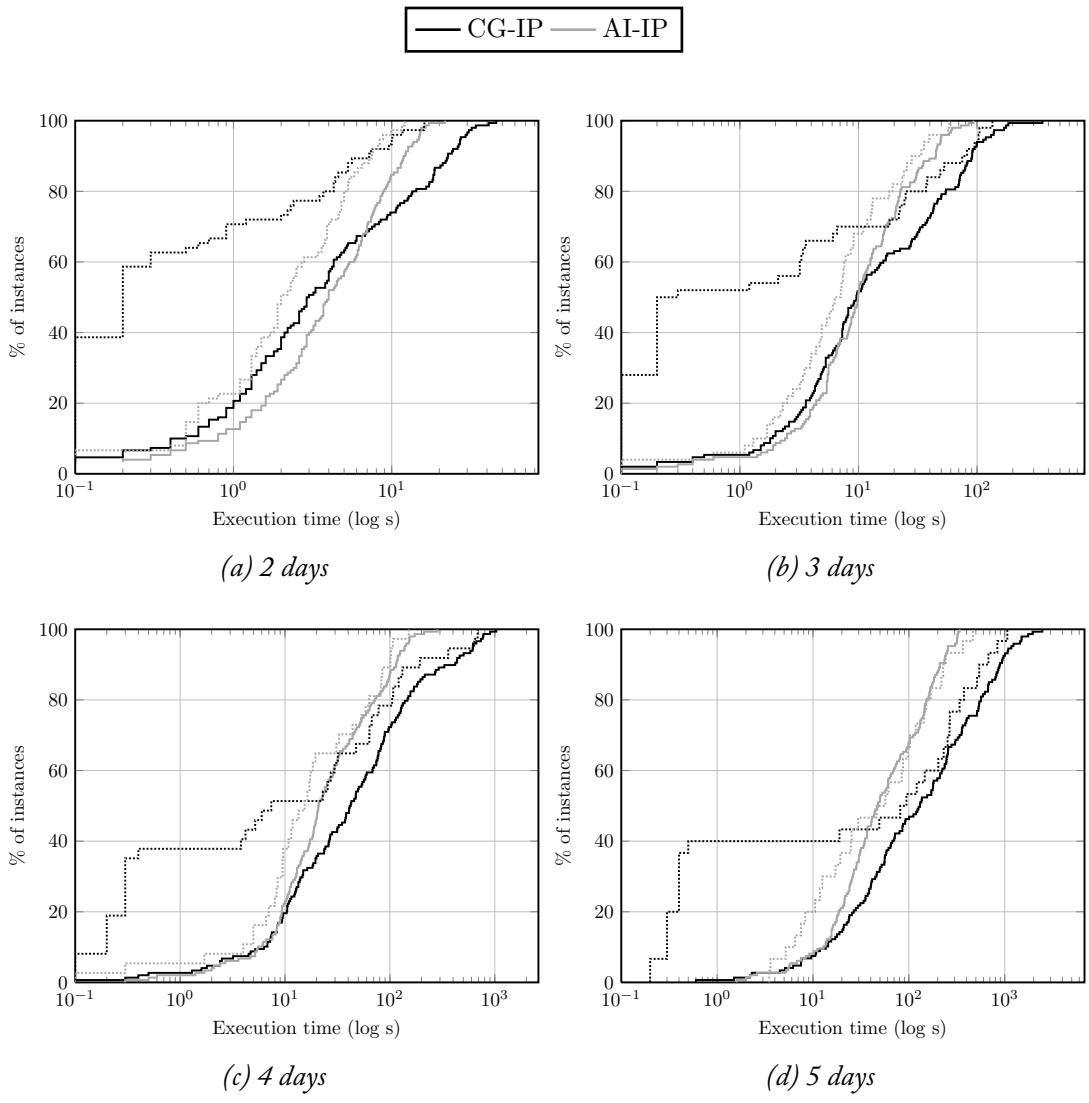


Figure 10: Execution time graphs for the rolling-horizon set up (solid lines). Data from the independent problem instance set-up has also been included for comparison (dotted lines). A logarithmic time scale has been used.

Table 4: Mixing results for CG-IP and AI-IP for the all instances combined.

Instance type		*Independent*				*Rolling horizon*			
Planning horizon h (days)		2	3	4	5	2	3	4	5
CG-IP	Total instance ER (#)	757	913	1019	1167	–	–	–	–
	Total Roll. H. ER (#)	–	–	–	–	1372	1374	1380	1384
	Days w. mixing (#)	47	55	55	51	61	61	61	61
AI-IP	Total instance ER (#)	757	913	1019	1167	–	–	–	–
	Total Roll. H. ER (#)	–	–	–	–	1387	1388	1379	1360
	Days w. mixing (#)	48	55	55	51	61	61	61	61

problem instance set-up have been included as dotted lines for comparison. As can be seen, AI-IP no longer performed worse than CG-IP for short execution times, which is likely a result of the lack of hot-start solution for the rolling horizon experiments. The long tail for the CG-IP results, corresponding to long execution times for a few instances, is still clear in comparison to the AI-IP results.

Finally, in both problem set-ups the LP relaxation in the root node of the branch and bound tree was tight for CG-IP and AI-IP for all real world instances. Therefore it stands to reason that practical instances of the problem exhibited a lot of structure which could be exploited by the two IP models and made them efficiently solvable even for larger problems (despite the NP-hardness). This observation means that a good way to further improve the two methods is to find new, problem specific heuristics, or enhance the existing ones.

4.3.3 Results for the Full Data Set.

In a realistic scenario, planning for shunting yards would be done for a few days at a time. This section, in contrast, analyzes the overall performance of CG-IP and AI-IP on the full five-month data set. Table 4 contains results pertaining to mixing, i.e. the number of extra-roll ins (ER), for the full data set. The following attributes are reported:

Total instance ER The total number of extra roll-ins for all independent instances in the 5-month data set.

Total Roll. H. ER The total number of extra roll-ins for the entire 5-month schedule.

Days w. mixing The number of days that required mixing in either all independent instances, or in the rolled-out 5-month schedule.

From Table 4, it is clear that for the independent instance set-up the sum of ER from all instances increased with the planning horizon. This can be explained by the empty

formation tracks at the start of each planning period allowing more trains to be scheduled without mixing, and as the planning horizon was increased the number of planning restarts (and emptying of the yard) decreased, and more mixing was required. In the rolling horizon planning scenario, the yard was assumed to be empty only once, which is why the total number of extra roll-ins was then higher and did not increase with the planning horizon.

For the independent problem instances, AI-IP and CG-IP return the same number of extra roll-ins (ER). However, the ER in the complete schedules differed for the rolling horizon scenario. It was expected that the two methods might return schedules of different ER for the same horizon length, as there were often more than one schedule with the optimal ER for each problem instance, and the two methods would not necessarily return the same one. Depending on which schedule that was returned, different trains would be fixed on tracks resulting in different problem set-ups in the next instance, and subsequently possibly resulting in two different optimal costs as well.

For the independent instance planning, the number of days that required mixing varied depending on the horizon length, but was the same for both solution methods for all horizons but from the 2-day horizon when AI-IP mixed cars for one extra day. For rolling-horizon planning, the number of days that required mixing was the same for all horizons.

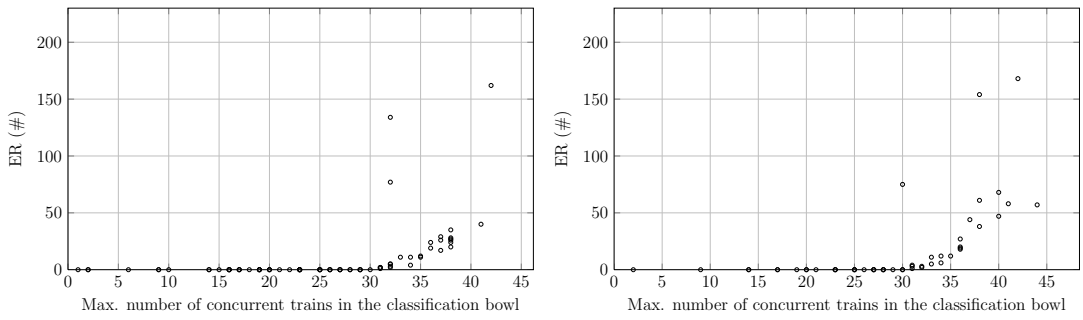
As there was no clear relationship between the ER in the complete schedules returned by the rolling-horizon planning and the horizon length, there is no clear indication as to which planning horizon was the most suitable. However, as cars should, in practice, stay at most one day at the yard, and in our experiments stayed at most two days, it is reasonable to conclude that any planning period longer than a few days should be sufficiently long. Further experimentation is however required to determine if this conclusion is valid in practice.

4.4 Capacity Analysis

Railway operators generally want as much track capacity as possible for shunting as this provides flexibility and robustness. On the other hand, the infrastructure providers have to maintain and build the tracks, and should therefore prefer as few tracks as possible being used for shunting. Being able to analyze how much capacity is required to shunt a certain traffic pattern, and understand when the capacity is running out, is therefore highly interesting.

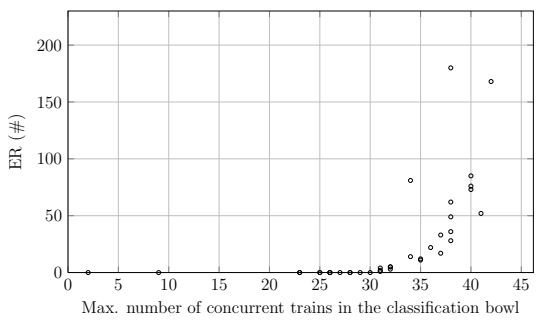
The results of the experiments indicated that no mixing was required for most days (see Table 4). In other words, a track allocation existed that allowed all cars to be rolled straight to their allocated formation tracks, indicating that on most days, there was enough capacity at Hallsberg. To investigate this further, an analysis of the classification bowl capacity was performed using the optimal allocations. Results from the independent set-up planning were chosen as the planning periods were independent in this set-up.

Fig. 11 shows the number of extra roll-ins versus the maximum number of simultaneous trains in the classification bowl, for all optimal solutions to the independent instances. The maximum number of concurrent trains on the 32 classification tracks varied between 42 and 44. As can be seen, no problem instance required mixing when the maximum

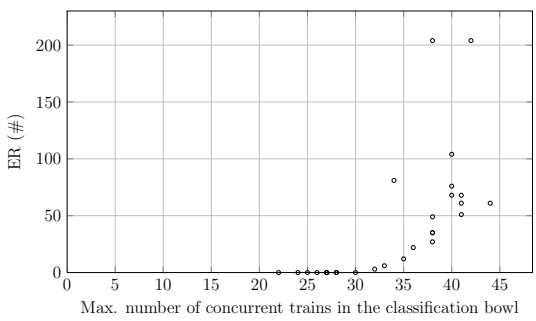


(a) 2 days

(b) 3 days



(c) 4 days



(d) 5 days

Figure 11: Scatter graph of the number of extra roll-ins versus the maximum number of trains simultaneously present in the classification bowl for AI-IP instances, independent problem set-up.

number of concurrent trains in the bowl was fewer than 30. For more than 30 concurrent trains, the number of extra roll-ins, and thereby also the work effort, increased quickly. In other words, when the number of concurrent trains was lower than the number of available formation tracks, work effort was low. However, when the number of concurrent trains was higher than the number of available formation tracks, the work effort required to form all freight trains increased rapidly with the number of concurrent trains. For the instance set considered, the same is true for all other horizon lengths, and for all solution methods (including the sometimes sub-optimal schedules obtained from Heuristic and D-IP). The results show that infrastructure providers may, at the cost of extra work effort, be able to offset limitations in the number of available tracks by implementing multi-stage train formation procedures.

5 Conclusions

In this paper, two models were developed and used to solve the multi-stage train formation problem, which is a core problem in the classification process carried out at shunting yards. A straightforward formulation turned out to yield weak LP relaxations on a real-world historic data set spanning five months. The new methods, CG-IP and AI-IP, were capable of calculating optimal solutions for real world instances of sizes considered large enough for deployment. The problem definition and the two algorithms were modified for rolling horizon planning, for which they also produced optimal solutions within a time span that would allow practical application. Further analysis of the results showed that it was possible to increase the maximum number of concurrently formed trains by 31 % to 37 %, by using a multi-stage train formation process. However, in the experiments, the increase in number of simultaneously formed trains came at a steep cost: the work effort, measured in number of extra roll-ins required, increased rapidly when simultaneously forming more trains than the number of available tracks.

As both CG-IP and AI-IP had acceptable computation times for the instances investigated, both appear suitable for solving the presented problem. However, CG-IP is harder to implement, and AI-IP may therefore be more suitable for a real-world implementation.

Despite the work presented in this paper there are still some open ends that should be addressed by future research.

5.1 Symmetry Reduction

As many classification tracks in reality are of the same or similar length, the problem exhibits a large amount of symmetry: one solution can easily be turned into another one by swapping the train schedule for two similar tracks. The performance of both CG-IP and AI-IP might suffer from this problem. Even though this symmetry will be reduced in the rolling horizon case, it might still be worth to address it.

We are currently working on eliminating symmetries using Halls Theorem [28]. For a bipartite graph $G = (V \uplus V', E)$, a complete matching M is a set of $|V|$ independent edges from G. For any subset $S \subseteq V$, let $\Gamma(S)$ be the vertices in V' connected to a vertex

in S. More formally, $\Gamma(S) = \{j \in V' \mid \exists i \in S : (i,j) \in E\}$. Hall's theorem can now be stated as follows.

Theorem 3 (Hall's Theorem) *For any bipartite graph $G = (V \uplus V', E)$, there is a complete matching M from V to V' if and only if*

$$|S| \leq |\Gamma(S)| \text{ for each } S \subseteq V.$$

For CG-IP, V is the set of sequences, V' is the set of tracks, and $(s,a) \in E$ if and only if $l(s) \leq l(a)$, where $l(s)$ denotes the length of the longest train that occurs in s. It is now possible to remove the indices a for track allocation in the sequence selection variables x_{sa} as well as the track selection variable y_{ba} and the constraints (10), (11) and (12) from the model in (9)–(14), if the following two sets of inequalities are included:

$$\sum_{s \in \mathcal{S}: b \in s} x_s \geq 1, \qquad b \in \mathcal{B}, \tag{28}$$

$$\sum_{s \in \mathcal{S}(a)} x_s \leq |\mathcal{A}_a|, \qquad a \in \mathcal{A}, \tag{29}$$

where $\mathcal{A}_a$ is the subset of tracks not longer than a. The constraint (28) expresses that each train should be present in at least one sequence (set cover constraints) and replaces the constraints (10) and (11). Further, the constraint (29) expresses the conditions in Hall's Theorem. Branching in the resulting model can be (non-trivially) achieved by e.g. Ryan-Foster Branching [see 38]. After solving the model, the track allocation can be recovered by solving an assignment problem on the final sequences and the tracks.

5.2 Hump Scheduling and Roll-out Order

In the experiments, a heuristic hump scheduling algorithm was used to create the missing hump schedule, including the roll-in order, stages and pull-back times. The roll-out times and order is also assumed to be given. These input parameters indirectly determine the outcome of classification bowl planning, and although there is little hope to find optimal strategies for determining them, it would nonetheless be interesting to investigate how more elaborate methods affect the outcome of classification yard planning.

5.3 Mixing Track Scheduling

In this paper, a single mixed track with constant capacity is assumed. In a real world setting, it is however possible to use a variety of mixing tracks at different time points over the day. It is easy to extend the models in this paper so that mixing capacity varies between stages, and the use of multiple tracks can be modeled as long as these are pulled out at the same time points. However, situations where the mixing track designations, and consequently the available set of formation track, varies over time requires a substantial extension of the presented models. It would be of great interest to see how such an extension, together with a method for determining a suitable mixing track designation, would perform.

References

[1] Achterberg, T. 2009. SCIP: Solving constraint integer programs. *Mathematical Programming Computation* **1**(1) 1–41.

[2] Ahuja, R. K., K. C. Jha, J. Liu. 2007. Solving real-life railroad blocking problems. *Interfaces* **37**(5) 404–419.

[3] Ahuja, R.K., T.L. Magnanti, J.B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.

[4] Assad, A. A. 1983. Analysis of rail classification policies. *Infor* **21**(4) 293–314.

[5] Averstad, K.-Å. 2006. *Handbok BRÖ 05-35/BA50: Trafikeringsplan Hallsberg rangerbangård*. Banverket. In Swedish.

[6] Averstad, K.-Å. 2006. *Handbok BRÖH 313.00001: Anläggningsbeskrivning Hallsbergs rangerbangård*. Banverket. In Swedish.

[7] Barnhart, C., H. Jin, P. H. Vance. 2000. Railroad blocking: A network design application. *Oper. Res.* **48**(4) 603–614.

[8] Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, P. H. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* 316–329.

[9] Bodin, L. D., B. L. Golden, A. D. Schuster, W. Romig. 1980. A model for the blocking of trains. *Transportation Res. B* **14**(1) 115–120.

[10] Bohlin, M., F. Dahms, H. Flier, S. Gestrelius. 2012. Optimal Freight Train Classification using Column Generation. *Proc. 12th Workshop on Algorithmic Approaches for Transportation Modelling, Optim., and Systems (ATMOS 2012)*, vol. 25. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 10–22.

[11] Bohlin, M., H. Flier, J. Maue, M. Mihalák. 2010. Hump yard track allocation with temporary car storage. Tech. Rep. T2010:09, Swedish Institute of Computer Science.

[12] Bohlin, M., H. Flier, J. Maue, M. Mihalák. 2011. Track Allocation in Freight-Train Classification with Mixed Tracks. *Proc. 11th Workshop on Algorithmic Approaches for Transportation Modelling, Optim., and Systems (ATMOS 2011)*, vol. 20. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 38–51.

[13] Bonomo, F., G. Durán, J. Marenco. 2009. Exploring the complexity boundary between coloring and list-coloring. *Ann. Oper. Res.* **169**(1) 3–16.

[14] Boysen, N., M. Fliedner, F. Jaehn, E. Pesch. 2012. Shunting yard operations: Theoretical aspects and applications. *Eur. J. Oper. Res.* **220**(1) 1–14.

[15] Büsing, C., J. Maue. 2010. Robust algorithms for sorting railway cars. *Proc. 18th Annual Eur. Conf. on Algorithms: Part I (ESA'10)*. Springer-Verlag, Berlin, Heidelberg, 350–361.

[16] Cicerone, S., G. D'Angelo, G. Di Stefano, D. Frigioni, A. Navarra. 2007. Robust algorithms and price of robustness in shunting problems. *Proc. 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optim., and Systems (ATMOS 2007)*. 175–190.

[17] Daganzo, C. F., R. G. Dowling, R. W. Hall. 1983. Railroad classification yard throughput: The case of multistage triangular sorting. *Transportation Res. A* **17**(2) 95–106.

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

[19] Dahlhaus, E., F. Manne, M. Miller, J. F. Ryan. 2000. Algorithms for combinatorial problems related to train marshalling. *Proc. 11th Australasian Workshop on Combinatorial Algorithms (AWOCA 2000)*. 7–16.

[20] Desaulniers, G., J. Desrosiers, M.M. Solomon. 2005. *Column Generation*. Cahiers du GERAD, Springer.

[21] Desrosiers, J., Y. Dumas, M. M. Solomon, F. Soumis. 1995. Time constrained routing and scheduling. *Handbooks in operations research and management science* **8** 35–139.

[22] Desrosiers, J., M. E. Lübbecke. 2005. A primer in column generation. G. Desaulniers, J. Desrosiers, M.M. Solomon, eds., *Column Generation*. Springer, Berlin, 1–32.

[23] Gatto, M., J. Maue, M. Mihalák, P. Widmayer. 2009. Shunting for dummies: An introductory algorithmic survey. *Robust and Online Large-Scale Optimization*, *LNCS*, vol. 5868. Springer, Berlin Heidelberg, 310–337.

[24] Gilmore, P. C., R. E. Gomory. 1965. Multistage cutting stock problems of two and more dimensions. *Operations Research* **13**(1) pp. 94–120.

[25] Golumbic, M.C. 2004. *Algorithmic Graph Theory and Perfect Graphs: Second Edition*. Annals of Discrete Mathematics, Elsevier Science.

[26] Gorman, M. F. 1998. An application of genetic and tabu searches to the freight railroad operating plan problem. *Ann. Oper. Res.* **78**(0) 51–69.

[27] Güçlü, T. 2012. Ein Spaltengenerierungsansatz für die Zuordnung von Güterzügen. Master's thesis, Chair of Operations Research, RWTH Aachen University, Germany.

[28] Hall, P. 1935. On representatives of subsets. *J. London Math. Soc* **10**(1) 26–30.

[29] Huntley, C. L., D. E. Brown, D. E. Sappington, B. P. Markowicz. 1995. Freight routing and scheduling at CSX transportation. *Interfaces* **25**(3) 58–71.

[30] Jacob, R., P. Márton, J. Maue, M. Nunkesser. 2007. Multistage methods for freight train classification. *Proc. 7th Workshop on Algorithmic Approaches for Transportation Modelling, Optim., and Systems (ATMOS 2007)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 158–174.

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

[32] Keaton, M. H. 1992. Designing railroad operating plans: A dual adjustment method for implementing lagrangian relaxation. *Transportation Sci.* **26**(4) 263–279.

[33] Krell, K. 1962. Grundgedanken des Simultanverfahrens. *Rangiertechnik* **22** 15–23.

[34] Liebchen, Christian, Marco Lübbecke, Rolf Möhring, Sebastian Stiller. 2009. The concept of recoverable robustness, linear programming recovery, and railway applications. Ravindra K. Ahuja, Rolf H. Möhring, Christos D. Zaroliagis, eds., *Robust and Online Large-Scale Optimization*, *Lecture Notes in Computer Science*, vol. 5868. Springer Berlin Heidelberg, 1–27.

[35] Maue, J., M. Nunkesser. 2009. Evaluation of computational methods for freight train classification schedules. Tech. Rep. TR-0184, ARRIVAL Project.

[36] Nemani, A. K., R. K. Ahuja. 2011. OR models in freight railroad industry. James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, J. Cole Smith, eds., *Wiley Encyclopedia of Oper. Res. and Management Sci.*. John Wiley & Sons, Inc.

[37] Newton, H. N., C. Barnhart, P. H. Vance. 1998. Constructing railroad blocking plans to minimize handling costs. *Transportation Sci.* **32**(4) 330–345.

[38] Ryan, D.M., B.A. Foster. 1981. An integer programming approach to scheduling. *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling* 269–280.

[39] Siddiqee, M. W. 1972. Investigation of sorting and train formation schemes for a railroad hump yard. G. F. Newell, ed., *Proc. 5th Internat. Symp. on the Theory of Traffic Flow and Transportation*. Elsevier, New York, 377–387.

[40] Van Dyke, C. D. 1986. The automated blocking model: A practical approach to freight railroad blocking plan development. *Transportation Research Forum*, vol. 27. 116–121.

A Proof of Theorem 2

The theorem states that AI-IP and CG-IP yield the same LP relaxation.

Proof of Theorem 2 To establish the result sought, it is enough to show that any feasible solution to the LP relaxation of one problem can be translated into a feasible solution of the other one, with the same objective value. First, the y variables in the CG-IP model are solely for the purpose of facilitating branching decisions and do not carry additional information about the solution. They could be removed by joining constraints (10) and (11) to

$$\sum_{a\in\mathcal{A}}\sum_{\substack{s\in\mathcal{S}(a):\\ b\in s}} x_{sa} \geq 1, \qquad b\in\mathcal{B}. \tag{30}$$

They are therefore w.l.o.g. ignored in this proof and constraint (30) considered instead.

First let x^* be a feasible solution to the LP relaxation of CG-IP. A feasible solution $\tilde{x}$ to AI-IP can then be constructed as:

$$\tilde{x}_{bb'a} = \sum_{\substack{s\in\mathcal{S}(a):\\ (b,b')\in s}} x^*_{sa}, \qquad \forall b\in\mathcal{B}\cup\{u\},\ b'\in\mathcal{B}\cup\{v\},\ a\in\mathcal{A}. \tag{31}$$

Replacing $x_{bb'a}$ in the objective (22) with the right-hand side of (31) yields

$$\begin{aligned}
\sum_{a\in\mathcal{A}}\sum_{\substack{b,b'\in\mathcal{B}_a:\\ b\prec b'}} c(b,b')\cdot\tilde{x}_{bb'a} &= \sum_{a\in\mathcal{A}}\sum_{\substack{b,b'\in\mathcal{B}_a:\\ b\prec b'}} c(b,b')\cdot\sum_{\substack{s\in\mathcal{S}(a):\\ (b,b')\in s}} x^*_{sa} \\
&= \sum_{a\in\mathcal{A}}\sum_{s\in\mathcal{S}(a)}\left(\sum_{(b,b')\in s} c(b,b')\right)\cdot x^*_{sa} \\
&= \sum_{a\in\mathcal{A}}\sum_{s\in\mathcal{S}(a)} c(s)\cdot x^*_{sa},
\end{aligned}$$

which is the same as Objective (9), showing that $\tilde{x}$ and x^* have the same objective values. Similarly, from the left-hand side of Inequality (25) and Equation (31) it follows that

$$\begin{aligned}
\sum_{a\in\mathcal{A}}\sum_{\substack{b,b'\in\mathcal{B}_a:\\ b\prec b'}} l_p(b,b')\cdot\tilde{x}_{bb'a} &= \sum_{a\in\mathcal{A}}\sum_{\substack{b,b'\in\mathcal{B}_a:\\ b\prec b'}} l_p(b,b')\cdot\sum_{\substack{s\in\mathcal{S}(a):\\ (b,b')\in s}} x^*_{sa} \\
&= \sum_{a\in\mathcal{A}}\sum_{s\in\mathcal{S}(a)}\left(\sum_{(b,b')\in s} l_p(b,b')\right)\cdot x^*_{sa} \\
&= \sum_{a\in\mathcal{A}}\sum_{s\in\mathcal{S}(a)} l_p(s)\cdot x^*_{sa},
\end{aligned}$$

which is the same as the left-hand side of Inequality (13).

The left hand side of Constraint (23) in AI-IP can be rewritten as follows:

$$\begin{aligned}\sum_{\substack{a\in\mathcal{A}:\\ b'\in\mathcal{B}_a}}\sum_{\substack{b\in\mathcal{B}_a:\\ b\prec b'}} x_{bb'a} &= \sum_{\substack{a\in\mathcal{A}:\\ b'\in\mathcal{B}_a}}\sum_{\substack{b\in\mathcal{B}_a:\\ b\prec b'}}\sum_{\substack{s\in\mathcal{S}(a):\\ (b,b')\in s}} x_{sa}\\ &= \sum_{\substack{a\in\mathcal{A}:\\ b'\in\mathcal{B}_a}}\sum_{\substack{s\in\mathcal{S}(a):\\ (b,b')\in s}} x_{sa}\\ &= \sum_{\substack{a\in\mathcal{A}:\\ b'\in\mathcal{B}_a}}\sum_{\substack{s\in\mathcal{S}a):\\ b'\in s}} x_{sa}\\ &= \sum_{a\in\mathcal{A}}\sum_{\substack{s\in\mathcal{S}a:\\ b'\in s}} x_{sa}.\end{aligned}$$

As this is equivalent to the left-hand side in the fulfilled Constraint (30) in CG-IP, (23) must also hold. Similarly, constraint (12) ensures that (24) is fulfilled. The flow conservation constraints (26) are fulfilled as every column in CG-IP represents exactly one path through the arc indexed variables and the combination of those will be a valid flow.

For the other direction we assume that $\tilde{x}$ is a feasible solution to the LP relaxation of AI-IP. Constraints (26) are flow conservation constraints, therefore $\tilde{x}$ can be interpreted as a flow on a network for every track $a \in \mathcal{A}$. The nodes in this network are the trains $b \in \mathcal{B}_a$. The virtual trains u and v are the source and the sink node.

For each track let $s_1, \ldots, s_n \in \mathcal{S}$ be a decomposition of this flow into distinct simple paths and let $x^*_{s_i a}$ be the flow sent over path s_i in this decomposition. Such a decomposition exists as there are no cycles in the flow (due to the partial order $\prec$). See e.g. [3, Theorem 3.5, pp. 89] for a proof of this. Also note that the set of possible sequences $\mathcal{S}$ is sufficient as it contains every possible path from u to v by its definition.

As the flow does not change in the decomposition, we know that the total flow entering or leaving a node must stay the same i.e. for $a \in \mathcal{A}$ and $b \in \mathcal{B}_a \setminus \{u, v\}$ it holds that

$$\sum_{\substack{s\in\mathcal{S}a:\\ b\in s}} x^*_{sa} = \sum_{\substack{b'\in\mathcal{B}_a:\\ b\prec b'}} \tilde{x}_{bb'a} = \sum_{\substack{b'\in\mathcal{B}_a:\\ b'\prec b}} \tilde{x}_{b'ba}$$

From here it easily follows that the aforementioned constraint (30) has to hold as for any train $b \in \mathcal{B}$ we have

$$\sum_{\substack{a\in\mathcal{A}:\\ b\in\mathcal{B}a}}\sum_{\substack{s\in\mathcal{S}(a):\\ b\in s}} x^*_{sa} = \sum_{\substack{a\in\mathcal{A}:\\ b\in\mathcal{B}a}}\sum_{\substack{b'\in\mathcal{B}_a:\\ b'\prec b}} \tilde{x}_{b'ba} \geq 1$$

Furthermore as all flow has to leave the sink and enter the source node it holds that

$$\sum_{s\in\mathcal{S}a} x^*_{s,a} = \sum_{b\in\mathcal{B}_a} \tilde{x}_{uba} = \sum_{b\in\mathcal{B}_a} \tilde{x}_{bva}$$

This implies that constraint (12) holds for x^* as constraint (24) was satisfied for $\tilde{x}$.

Equality of the two objective functions is satisfied by the same argument as before. Also constraint (13) is implied by (25), as seen in the first half of the proof. This means that x^* is a valid solution of CG-IP with the same objective value. Therefore CG-IP and AI-IP have the same LP bound. □

B Hump Scheduling and Roll-out Time Algorithms

B.1 Calculating Roll-in Intervals

This algorithm calculates roll-in intervals (A_i, D_i) defining between which two times the roll-in of a train i may be scheduled, and arrival yard track allocations $k(i)$, denoted by the track number from 1 to $|L|$ where L is the set of arrival tracks, for all inbound trains i.

1. Initiate the roll-in intervals with trivial early and late roll-in times:
   (a) $(A_i, D_i) \leftarrow \left(t_i + S\ ,\ \min\{t^{\text{dep}}_{b(g)} - T \mid \text{Car } g \text{ in } i\}\right)$, where S is inspection and decoupling time, and T is the minimal yard time needed (140 minutes)
2. Let t_h be the time when the hump is available next:
   (a) $t_h \leftarrow 0$
3. Let $\mathcal{I}$ be the set of all non-processed trivial intervals, and $\mathcal{I}_y$ the set of intervals of inbound trains currently on yard:
   (a) $\mathcal{I} \leftarrow \{(A_i, D_i) \mid \text{all inbound trains } i\}$
   (b) $\mathcal{I}_y \leftarrow \emptyset$
4. Find a round-robin allocation for the first $|L|$ trains to arrive.
   (a) For each $n = 1...|L|$:
      i. $i \leftarrow \text{argmin}_i\{A_i \mid (A_i, D_i) \in \mathcal{I}\}$
      ii. $k(i) \leftarrow n$
      iii. $\mathcal{I} \leftarrow \mathcal{I} \setminus \{(A_i, D_i)\}$
      iv. $\mathcal{I}_y \leftarrow \mathcal{I}_y \cup \{(A_i, D_i)\}$
5. For each non-processed arriving train in time order, find an allocation if possible, else delay train:
   (a) While $\mathcal{I} \neq \emptyset$:
      i. $i \leftarrow \text{argmin}_i\{A_i \mid (A_i, D_i) \in \mathcal{I}\}$
      ii. $P = \{(A_k, D_k) \mid A_k \leq t_i, (A_k, D_k) \in \mathcal{I}_y\}$, the set of trains on the arrival yard ready to be rolled in when i arrives
      iii. $P_e = \{(A_k, D_k) \mid D_k \leq D_l \forall (A_l, D_l) \in P, (A_k, D_k) \in P\}$, trains with the earliest trivial late roll-in time
      iv. $j \leftarrow \text{argmin}_j\{A_j \mid (A_j, D_j) \in P_e\}$, train with the earliest trivial early roll-in time (also earliest arrival time)
      v. if $j \neq \emptyset$:

A. $k(i) = k(j)$
B. $A_j = \max\{t_h, A_j\}$
C. $D_j = \min\{t_i, D_j\}$
D. $t_h = A_j + R$, where R is the time needed for a roll-in.
E. $\mathcal{I} \leftarrow \mathcal{I} \setminus \{(A_i, D_i)\}$
F. $\mathcal{I}_y \leftarrow \mathcal{I}_y \cup \{(A_i, D_i)\} \setminus \{(A_j, D_j)\}$

else:

A. $l \leftarrow \text{argmin}_l\{A_l \mid (A_l, D_l) \in \mathcal{I}_y\}$
B. $t_i = A_l$, arrival of train i delayed as no space on arrival yard.
C. $A_i \leftarrow t_i + S$
D. Go to 5(a)ii.

B.2 Calculating Pull-back Intervals

Time intervals (A_p, D_p) for pull-backs are determined as follows.

1. Set the roll-out time of all trains $b \in \mathcal{B}$ to the latest possible time.
   (a) $t_b = t_b^{dep} - S^{dep}$, where S^{dep} is the time needed between roll-out and departure.
2. Assign to each train b a formation track $f(b)$ such that the total time between consecutive same-track roll-outs is maximized. This is done by solving an assignment problem.
3. For each track, form an interval graph of trains requiring mixing. For the earliest maximal clique, let $D_p \leftarrow$ the earliest departure time and $A_p \leftarrow$ the latest allocation start time. Remove trains covered by this pull-back interval (i.e. trains in the clique) from the interval graph and repeat until the graph is empty.

B.3 Consolidating Pull-backs and Roll-ins Into a Hump Schedule

The hump schedule is fixed so that pull-backs and roll-ins do not overlap as follows.

1. For all roll-in intervals (A_i, D_i) in order of increasing A_i:
   (a) Schedule the roll-in as early as possible in the interval. Note that this is a feasible roll-in schedule as the hump availability was considered when generating the roll-in intervals.
2. For all pull-back intervals (A_p, D_p) in order of increasing A_p:
   (a) Schedule the pull-back as early as possible in the interval. If possible, reschedule any roll-ins until after the pull-back.

(b) If a pull-back cannot be scheduled in its time interval, delay the affected outbound trains alongside the pull-back until the hump is free. Do not schedule the pull-back.

3. If any pull-back was delayed, restart the process from the beginning of the arrival yard planning with the new delayed departure times. Else, return the roll-in, pull-back and roll-out times.