Technical University of Denmark

DTU

Integrating rolling stock scheduling with train unit shunting

Haahr, Jørgen Thorlund; Lusby, Richard Martin

DTU Library
Technical Information Center of Denmark

Accepted Manuscript

Integrating Rolling Stock Scheduling with Train Unit Shunting

Jørgen Haahr, Richard M. Lusby

Highlights

- We develop two integrated frameworks for scheduling rolling stock and train unit shunting.

- We provide a comparison of several methods for solving the Track Allocation Problem.

- We give a counter example demonstrating the suboptimality of a previously proposed optimal method.

Integrating Rolling Stock Scheduling with Train Unit Shunting

Jørgen Haahr[a], Richard M. Lusby[b,*]

[a]*Optivation, Rued Langgaards Vej 7, 2300 Copenhagen, Denmark*
[b]*Department of Management Engineering, Technical University of Denmark*

Abstract

In this paper we consider integrating two important railway optimization problems, in particular the Rolling Stock Scheduling Problem and the Train Unit Shunting Problem. We present two similar branch-and-cut based approaches to solve this integrated problem and, in addition, provide a comparison of different approaches to solve the so-called Track Assignment problem, a subcomponent of the Train Unit Shunting problem. In this analysis we demonstrate, by way of a counter example, the heuristic nature of a previously argued optimal approach. For the integrated problem we analyze the performance of the proposed approaches on several real-life case studies provided by DSB S-tog, a suburban train operator in the greater Copenhagen area. Computational results confirm the necessity of the integrated approach; high quality solutions to the integrated problem are obtained on instances where a conventional, sequential approach ends in infeasibility. Furthermore, for the considered instances, solutions are typically found within a few minutes, indicating the applicability of the methodology to short-term planning.

Keywords: Transportation, Passenger Railway Optimization, Integer programming, Integration, Rolling Stock, Shunting

1. Introduction

A passenger railway operator must exercise careful planning in order to make detailed schedules for the day-to-day operations. A railway line plan stipulates the underlying structure of the timetable, which must be operated by the fleet of rolling stock units and any required crew members. Identifying a high quality solution for the railway operation has proven to be a difficult and complex optimization problem. The current practice is to decompose the problem into smaller subproblems and solve them sequentially. Two apparent drawbacks to this approach are the degradation of overall solution quality and the infeasibility occurring due to poor decisions made in previous stages; e.g. a timetable with a valid rolling stock schedule may be infeasible with respect the required shunting movements. Finding feasible solutions, as opposed to optimal ones, is even more important in a short-term (or real-time) setting, where the time to react is limited. Integrating the planning problems remedies the need for iterating through infeasibility and the generation of sub-optimal solutions. Due to the size and complexity of each of these problems, not to mention the short reaction time available to find a solution, it is not surprising that there has recently been an increased interest in computer aided decision support, or disruption management tools that assist planners in recovery operations within the railway industry, see e.g. Kroon and Huisman (2011). A study in the airline industry shows that integrating aircraft routing and crew pairing not only results in cost reductions but also produces more robust solutions, compared to those used in practice, see Weide et al. (2010).

In this work we consider integrating two important scheduling problems, in particular the Rolling Stock Scheduling Problem (RSP) and the Train Unit Shunting Problem (TUSP). The RSP involves allocating a fleet of rolling stock units to *trips* in such a way that the overall cost of

---

*Corresponding author, ph: +45 4525 3084

Email address:* `rmlu@dtu.dk` (Richard M. Lusby)

operating the timetable is minimized. A trip is a timetabled train service connecting two stations. Units, on the other hand, are individual vehicles that come in different *types* and which can be arranged in different ways, giving so-called *compositions*, which can then be assigned to trains. Units of the same type have the same physical characteristics. The allocation of units to trains should ensure a sufficient number of seats for the passengers, but also respect several constraints from an operational perspective. In order to cover the expected passenger demand as closely as possible, the composition of trains can be modified throughout the planning horizon by coupling and uncoupling units. Finding optimal or high quality rolling stock schedules is the focus of the RSP. Often, the RSP focuses on the allocation of unit types only, and the assignment of specific units is left until a later stage.

The TUSP, on the other hand, deals with the planning of *shunting movements* within railway *depots*. A depot (or shunting yard) is an area where units not in service can be parked. It usually consists of a number of parallel tracks and is often located close to a railway station. A shunting movement refers to the process of driving a unit to (or from) a depot track from (or to) a platform in the station. Given an initial inventory of unit types parked in the depot, this problem typically involves first *matching* specific units in the depot to arriving and departing trains (for which only the types required are usually specified). Simply put, an arriving train must be matched to a specific (later) departure or be assigned to depot track permanently, and a departing train must similarly be matched to an (earlier) arrival or an initially parked train. Once a matching has been found, the units must be assigned to depot tracks in a such a way that the resulting shunting movements can be performed in an efficient way. On which track and in which order to park idle units, given a matching, is commonly referred to as the Track Assignment Problem (TAP). A depot track has a finite capacity, and this limits the number of units that it can simultaneously hold. Furthermore, a common characteristic of depot tracks is that they are only accessible to and from one direction. In other words, they essentially function as last-in first-out (LIFO) stacks.

The RSP and TUSP are clearly interdependent; any rolling stock schedule induces activity at the depots of a railway network. The conventional approach is to solve these problems sequentially, with the RSP solved first. Such an approach can potentially lead to an infeasible TUSP. This is more likely to occur in networks with scarce depot capacity. Infeasibilities in the TUSP typically arise due to unavoidable ordering violations, i.e., two units cannot enter and exit the depot in the order specified by the rolling stock schedule. This research is in collaboration with Copenhagen Suburban Railway Operator (DSB S-tog), a suburban train operator in the greater Copenhagen area. DSB S-tog operates a fleet of approximately 130 physical units (of two different types) over a rail network, totalling 170km in length, that connects 85 stations. Approximately, 350,000 passengers use the network daily. Talks with DSB S-tog revealed that an optimized rolling stock schedule is usually infeasible with respect to the induced shunting movements. Consequently, manual repairs are needed in order to obtain a feasible solution to both the RSP and the TUSP. This provides the necessary motivation for our research. We focus on solving the integrated problem, which we term the Integrated Rolling Stock and Unit Shunting Problem (IRSUSP). To the best of our knowledge, such an approach has not been previously attempted. We test the performance of the developed algorithms on several real-life case studies provided by DSB S-tog.

In what follows we propose two similar branch-and-cut based approaches to solve the IRSUSP. Two possibilities arise depending on what information is included in the RSP. The first approach extends the Branch-and-Price (BAP) framework of Haahr et al. (2014) for the RSP. In particular, we introduce a new routine to assess the feasibility of rolling stock schedules from a shunting perspective at all depots. The model proposed in Haahr et al. (2014) only implicitly considers depot capacity in the form of an aggregated constraint on the total length of track available. It does *not* consider any ordering conflicts that could arise, nor if it is possible to park the units when individual track capacities are considered. Haahr et al. (2014) allocate specific units to trips and generate sequences of trips, or *itineraries*, for each unit in the fleet via a column generation approach. From these individual unit itineraries it is possible to determine at which times each unit enters, or exits, a given depot. Utilizing this information we extend the BAP method of Haahr et al. (2014) to a Branch-and-Price-and-Cut (BAPC) approach. A feasible solution to the RSP is only accepted if it is feasible with respect to all depots from a shunting perspective. Otherwise, we

3

add a constraint to cut this solution away and generate a new rolling stock schedule. As specific unit assignments are provided by the model for RSP, to determine the shunting feasibility of a depot only a TAP needs to be solved. We refer to this variant as the Route First Method (RFM) since the itineraries for all individual units are available before solving the shunting problem.

The second proposed approach extends a flow based Mixed Integer Program (MIP) inspired by Fioole et al. (2006) for the RSP. As with the model Haahr et al. (2014), very little information regarding depots is included in this formulation of the RSP. Furthermore, being a flow based model, it only assigns unit types to timetabled service, and not specific units. As is the case with the first approach, we introduce an additional routine to test the shunting feasibility of rolling stock schedules. As the RSP formulation is anonymous from a unit perspective, this feasibility check requires solving the full TUSP (i.e. both the matching and the TAP must be solved). If a rolling stock schedule provides a feasible solution to the TUSP, it is retained, otherwise it is a cut away through the addition of a constraint. We refer to this method as the Route Last Method (RLM) since individual unit itineraries are implicitly found as part of the shunting problem.

In developing the two frameworks for the IRSUSP this paper makes several contributions to the literature on the RSP and TUSP. First and foremost, the proposed solution methods themselves are the biggest contribution. This has not been addressed previously, and this paper highlights the necessity of such approaches in short term planning. Secondly, as the TAP is a core component of the RFM, we present a comparison of different methodologies for solving this problem. In particular, we compare two Branch-and-Cut (BAC) procedures for this, an extension of the heuristic described in Freling et al. (2005), and an exact BAP approach. In doing so, we highlight, using a counter example, the heuristic nature of the previously argued optimal column generation procedure in Freling et al. (2005) for generating a lower bound. Finally, we also develop a heuristic unit swapping routine that can be used to repair an infeasible solution to the TAP.

This paper is structured as follows. In Section 2 we provide a brief summary of the relevant literature, where any differences to existing approaches are described. Section 3 introduces formal descriptions of both the RSP and TUSP and provides mathematical formulations of each. In Section 4 we introduce the BAC frameworks for solving the IRSUSP. This section also describes existing approaches to solve the TAP. We extend some of these and correct an error in the proposed column generation approach of of Freling et al. (2005). It is in this section where we also as propose a heuristic to recover feasibility of an infeasible TAP instance. Computational results are provided in Section 5. This includes a comparison of the methods to solve the TAP on artificial instances as well as a performance assessment of the integrated frameworks and real-life case studies provided by DSB S-tog. Finally, conclusions and future research directions are outlined in Section 6.

2. Literature

In this section we review relevant literature that addresses either the RSP or the TUSP. For the RSP we include models and methods used for solving the tactical and short-term variants. However, for the TUSP, to our knowledge there are no studies that address short-term planning. The ordering restrictions inherent in the TUSP are not unique to passenger rail, nor trains for that matter; we draw analogies with bus depot planning and freight rail car classification.

Over the past few years several different models have been put forward for solving variants of the RSP, both from a planning and a rescheduling perspective. An approach for determining the minimum circulation of units required to operate a timetable is described by Schrijver (1993). A more elaborate MIP model that can handle the combining and splitting of trains is proposed by Fioole et al. (2006). This model forms the basis of Nielsen et al. (2012), where a rolling horizon framework is developed for solving a rolling stock rescheduling problem. These three approaches can all be classified as anonymous unit flow models. As such, one is unable to track individual unit movements. As an alternative, Haahr et al. (2016) present a path based formulation for the same problem and solve this using a BAP framework. Each path specifies an itinerary for a particular unit type. Train compositions are handled by the master problem, making sure that composition changes are feasible. Note that the model does not assign individual itineraries to specific units, but sets of itineraries that are consistent with the number of unit types available at their specified

4

initial locations. This methodology does therefore not deal with anonymous unit flows as in Fioole et al. (2006) and is equally applicable at the tactical planning level. For a more extensive overview of models and methods to reschedule rolling stock in real-time, the reader is referred to Cacchiani et al. (2014).

Several studies concerning variants of the TUSP have been conducted. The problem of dispatching trams from a storage yard is addressed in Winter and Zimmermann (2000). To achieve a departure order satisfying the scheduled demand, several shunting operations may be necessary. The authors describe combinatorial optimization models as well as both exact and heuristic approaches for solving the real time dispatch problem. Scheduling trams in the morning is also the topic of Blasum et al. (1999). The authors prove the NP completeness for the problem of finding an assignment of parked trams (of different types in stacks) to departure times without any additional shunting movements. Di Stefano and Koči (2004) consider the shunting of units in a railway depot. The authors primarily focus on depot track ordering restrictions and address the complexity of several subproblems. The objective of all problems is to minimize the number of tracks needed to park the units. Topics from graph theory including hypergraphs, permutation graphs, and graph colouring are applied. Three heuristic approaches to train shunting in a workshop area are described in Jacobsen and Pisinger (2011), while Føns (2006) considers different mathematical models and approaches for the TUSP. Both Føns (2006) and Jacobsen and Pisinger (2011) consider cases arising at the Copenhagen suburban railway operator, DSB S-tog.

The problem of dispatching buses from a bus terminal is the focus of Gallo and Miele (2001). Initially the problem is formulated using the Quadratic Assignment model approach of Winter and Zimmermann (2000). In addition, the authors also describe a new model that takes into account the fact that the buses can have different lengths. The resulting model is shown to be well suited to decomposition, and hence the authors present a Lagrangian Decomposition based approach. Real-life instances from the Florence Public Transportation Company are studied. It is concluded that the algorithm can find good quality solutions to practical problems in reasonable time (within two minutes). Other bus parking related research includes Hamdouni et al. (2006, 2007). The study by Hamdouni et al. (2006) considers identifying robust parking solutions and argues that one should generate solutions in which the bus lanes have at most two different types. Having fewer types on a lane reduces the potential for ordering (or crossing) conflicts, possibly at the expense of wasting parking capacity. The work of Hamdouni et al. (2007) introduces a Benders Decomposition approach for minimizing bus type mismatches between arrival and departure pairs. That is, in the problem considered it is possible to supply a bus of a different type to the one requested, but at a cost. Having few distinct bus types on a given parking lane is also a priority.

In Freling et al. (2005) a model for the TUSP is defined; however, the described solution method decomposes the problem into a matching problem and a TAP. The matching is solved first and then used as input to the TAP. The authors propose a MIP model for the matching problem, while the TAP is solved using a column generation approach. Computational experiments focus on two case studies at station Zwolle in the Netherlands. It is not completely clear if a column generation approach actually outperforms a MIP approach for the instances considered. Furthermore, we argue, with counter example that the dominance criteria stated are not exact; i.e. they can potentially remove the optimal solution.

A similar decomposition is also suggested in Lentink et al. (2006). However, two additional steps focusing on the routing of the units from their arrival platforms to their designated depot tracks are also included. Between the matching of arrivals with departures and the parking of the units, an estimate on the cost of routing the units is calculated. These costs are then used when assigning units to tracks. Upon parking the units, actual routes from the platform to the depot tracks are obtained. A heuristic that sequentially routes the units is devised, and the complete routing solution is improved using a 2-opt heuristic.

Haijema et al. (2006) propose a dynamic programming based heuristic for solving the TUSP. A realistic test case from the railway station Zwolle in the Netherlands is used to test the developed methodology. The test case considers a 24 hour period during which 45 units arrive and 55 units depart. The depot has 19 tracks with a total length of 4000m. The simple heuristic is fast and flexible and produces promising results; however, only a single instance from real-life is considered.

The work of Kroon et al. (2008) extends the work in Freling et al. (2005) and presents a fully integrated model for solving both the matching problem as well as the TAP. A large MIP model is proposed that attempts to minimize the number of split compositions and the number of different unit types simultaneously parked on the same depot track. The authors indicate how the model can be strengthened through the addition of clique inequalities and also how to model more practical restrictions. For example, a discussion on how to deal with trains composed of several units as well as how to model depot tracks that can be approached from both sides is included. The concept of a virtual shunting track is introduced in order to help identify which tracks should be heterogeneous and which tracks should be homogeneous from a unit type perspective. Computational experiments focus on two Dutch stations and consider up to 125 units of 12 different types that need to be parked.

An advanced planning tool for shunting operations is described by van Wezel and Riezebos (2011). Like almost all previous work on the TUSP, this approach first solves a matching problem that determines how arriving units are matched to departing units. This is done via a network flow algorithm. A k-shortest path algorithm is used to assign a track to each unit, while a modified version of an undirected k-shortest path is used to route the units. Finally the approach also assigns drivers to shunt operations.

Aside from passenger railway operators, ordering problems on storage tracks are highly prevalent in the freight rail industry. In order to reach their final destination freight rail cars are sorted at so-called *classification* yards where incoming trains are sorted into blocks of rail cars that share the same destination. The blocks are subsequently combined to form outbound trains. The order in which to process inbound trains, which blocks to build, which track to assign a block, and how to build outbound trains, are all decisions that need to be made. For a survey of shunting in the freight rail industry, the reader is referred to e.g. Gatto et al. (2009) and Boysen et al. (2012).

To summarize, when solving the TUSP one first typically solves a matching problem before solving a parking (and possibly a routing) problem. This stems from the fact that rolling stock allocation is typically done using MIP models that generate anonymous unit routes. In what follows, we present and compare two approaches that integrate rolling stock scheduling and train unit shunting. The RFM generates individual unit itineraries in the RSP and tests feasible solutions to this problem for depot feasibility. The RLM, on the other hand, only assigns unit types in the RSP. Feasible solutions to this problem must also be checked for depot feasibility. Unlike the RFM, this check must be extended to include the assignment of individual units to compositions.

3. Problem Description

A sound understanding of both the RSP and the TUSP is required in order to grasp the details of the integrated frameworks. In this section we therefore provide more formal descriptions of each of these problems. Section 3.1 focuses on the RSP, while Section 3.2 introduces the TUSP. The necessary notation is defined, and mathematical formulations of each problem are provided.

3.1. Rolling Stock Scheduling (RSP)

The RSP involves allocating a fleet of units, possibly of different types, to a set of timetabled trips, $\mathcal{T}$. Associated with any trip $t \in \mathcal{T}$ is a departure station, a departure time, an arrival station, an arrival time, and the forecasted passenger demand. Furthermore, in our modelling we assume that the two stations, which a trip connects, each have a depot. As such, a trip may not necessarily connect adjacent stations. It is always possible to obtain a set of such trips from an input timetable by merging trips that do not run between stations with depots until this is achieved. Train compositions can only be changed at stations with a depot. The set of all depots on the network is denoted by $\mathcal{D}$, while the set of unit types in the fleet is given by $\mathcal{U}$. Naturally, the trips to which each unit can be assigned to, depend on the unit's initial location, and potentially also its type. The constant inv_d^u states how many units of type $u \in \mathcal{U}$ are initially available at depot $d \in \mathcal{D}$. The assignment of units to trips is constrained by the set of possible train compositions, $\mathcal{C}$. In other words, only certain combinations of unit types can be coupled together

to form a train. In addition to the unit types it is comprised of, a composition also states where in the corresponding sequence each of the types is placed. Changing the composition of a train on consecutive trips is only possible if the layout of the station where the connection occurs permits this. For example, at some stations it may only be possible to (un)couple units from the front of the train, while at other stations it may only be possible to (un)couple units from the rear of the train. Also, composition changes can occur even in the absence of (un)coupling units. A train that turns on a platform is one such example; the train leaves in the direction it came from, and the composition of units is reversed. A set of so-called *connections*, $\mathcal{R}$, is usually introduced to model this. Every element $r \in \mathcal{R}$ is a pair of trips $(t, t') \in (\mathcal{T} \times \mathcal{T})$, where t' is the successor trip of trip t. A composition change can occur at the arrival station of trip t (the departing station of trip t'). The constants $coup^u_{c,c'}$ and $uncoup^u_{c,c'}$ are used to denote the number of units (of type $u \in \mathcal{U}$) requiring shunting to and from the depot as a result of a change in composition from $c \in \mathcal{C}$ to $c' \in \mathcal{C}$.
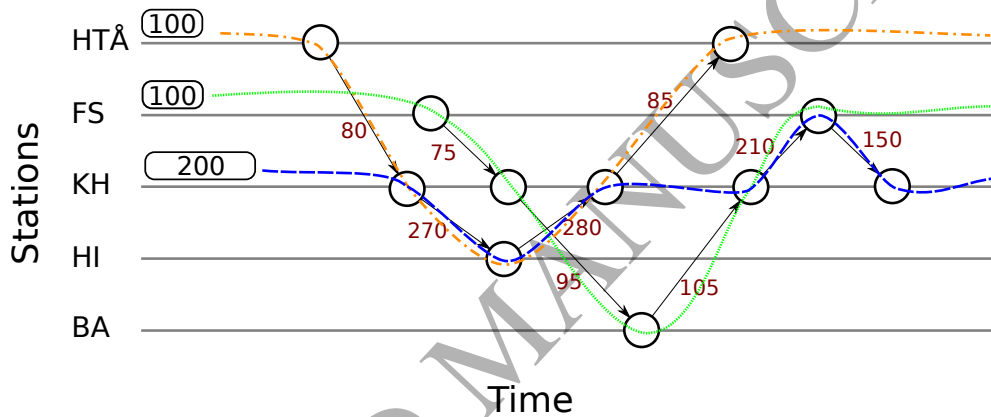

Figure 1: An example of a RSP instance using a time and space diagram. Stations (with depots) are listed vertically. some of these contain initial units with stated capacities. Arcs represent trips that run between two stations. The number associated with each arc specifies the passenger demand. The colored paths illustrate the itineraries taken by the available units

An example of an RSP is shown in Figure 1. All units end at the depots where they were initially positioned. Furthermore, by (un)coupling units at key stations, the compositions of the trains can be modified such that the passenger demand is satisfied. For example, a small and large unit are coupled together at KH in order to satisfy the demand of 270. After performing two trips the large unit is uncoupled at KH, where it is then coupled to the other small unit to better match the demand on the KH-FS trip. Note that whenever a unit is not operating a trip it must be parked on a depot track.

The first MIP that we present is based on the one proposed by Fioole et al. (2006). This forms the basis of the RLM and uses two families of binary variables. One set, x^t_c, is needed to control the composition assignment to trips, while the other, $z^r_{c,c'}$ monitors the composition changes. Formally, x^t_c has a value of one if composition $c \in \mathcal{C}$ is selected for trip $t \in \mathcal{T}$, while $z^r_{c,c'}$ is set to one if, at connection $r \in \mathcal{R}$, the train changes from composition $c \in \mathcal{C}$ to $c' \in \mathcal{C}$. The formulation is shown below. To ease its readabiltity we introduce a number of additional sets and some shorthand notation. We denote the two trips associated with any connection $r \in \mathcal{R}$ as $\texttt{source}(r) \in \mathcal{T}$ and $\texttt{target}(r) \in \mathcal{T}$, respectively, where $\texttt{target}(r)$ is the successor trip of $\texttt{source}(r)$. Furthermore, we use $\texttt{depot}(t) \in \mathcal{D}$ to indicate the depot associated with the arrival station of trip $t \in \mathcal{T}$. Finally, we introduce the set $\mathcal{R}^t \subset \mathcal{R}$ which contains all connections at the departure station of trip $t \in \mathcal{T}$ that occur prior to or at the departure time of trip t. A general MIP formulation of the RSP can hence be stated as:

$$\text{Minimize:} \quad F(x, z) \tag{1}$$

7

$$\text{s.t.} \quad \sum_{c \in \mathcal{C}} x_c^t = 1 \qquad\qquad \forall t \in \mathcal{T} \qquad (2)$$

$$x_c^{\texttt{source}(r)} = \sum_{c' \in \mathcal{C}} z_{c,c'}^r \qquad\qquad \forall r \in \mathcal{R}, c \in \mathcal{C} \qquad (3)$$

$$x_{c'}^{\texttt{target}(r)} = \sum_{c \in \mathcal{C}} z_{c,c'}^r \qquad\qquad \forall r \in \mathcal{R}, c' \in \mathcal{C} \qquad (4)$$

$$0 \le inv_{\texttt{depot}(t)}^u \qquad\qquad\qquad (5)$$

$$- \sum_{c \in \mathcal{C}} \sum_{c' \in \mathcal{C}} \sum_{r \in \mathcal{R}^t} coup_{c,c'}^u \cdot z_{c,c'}^r$$

$$+ \sum_{c \in \mathcal{C}} \sum_{c' \in \mathcal{C}} \sum_{r \in \mathcal{R}^t} uncoup_{c,c'}^u \cdot z_{c,c'}^r \qquad \forall u \in \mathcal{U}, t \in \mathcal{T} \qquad (6)$$

$$x_c^t \in \{0,1\} \qquad\qquad \forall t \in \mathcal{T}, c \in \mathcal{C} \qquad (7)$$

$$z_{c,c'}^r \in \{0,1\} \qquad\qquad \forall r \in \mathcal{R}, c \in \mathcal{C}, c' \in \mathcal{C} \qquad (8)$$

A general objective function is given in (1). This depends on the compositions assigned to the trips as well as any composition changes that must be performed. It is normally a weighted sum of multiple objectives. These can include such factors as the number of trip cancellations if it is impossible to service all trips, the operating cost, the shunting costs, the unmet passenger demand, and the end-of-day *balance* deviations. Unmet passenger demand arises whenever the composition assigned to a trip has fewer seats than the forecasted passenger demand on the trip. In some cases it is desirable to end with a specified number of each unit type at each depot. Any deviation from this specified balance is penalised. Constraints (2) ensure that every trip is assigned one composition. Note, that the empty composition is included in $\mathcal{C}$ and represents a cancellation. Any composition can be removed from consideration by fixing the x variables to zero. For example, a short platform at a particular station may prevent large compositions from being assigned. Constraints (3) and (4) link the set of binary variables. In order to enforce certain transition rules, a number of the z variables can be removed from consideration by fixing them to zero. For example, a station where units can only be (un)coupled from the rear of the train has fewer possibilities than stations where it is also possible to (un)couple from both ends. Constraints (6) ensure that inventory is always non-negative. Finally, the domains of the variables are defined by (7)-(8). We note, that instead of fixing variables to zero, the corresponding variables should be left out of the mathematical model altogether in the implementation.

When assigning rolling stock it is usually assumed that all depots have sufficient capacity for the total number of shunting movements that must be performed in the resulting schedule. For depots with scarce capacity, this may not necessarily be true. In such cases it is possible to append constraints on aggregated depot capacity to the RSP formulation above. These constraints ensure, on an aggregated level, that the total length of units parked in any depot at the same time is never more than the total available track length of the depot. In reality depots consist of multiple parallel tracks of different lengths. Solely respecting the aggregated length does not guarantee that units under consideration can in fact be parked on the discrete set of tracks. Furthermore, this does not rule out the possibility for ordering violations on the depot tracks when considering all tracks as well as the exact timings of shunting movements into and out of depots.

Model (1)-(8) is essentially a flow based model; a solution to it states the how the different unit types move through the network in operating the timetable. It does not, however, provide individual itineraries for the specific units. An alternative formulation by Haahr et al. (2016) models the RSP from this perspective. A path based formulation, which generates individual unit itineraries, is developed and solved using column generation in which promising unit itineraries are dynamically generated. The Haahr et al. (2014) model has been extended to include compositions by Haahr et al. (2016) and is used in the RFM here. For each of the depots on the network it assigns a certain number of itineraries corresponding to the units parked there. Recall that an itinerary is a sequence of trips a given unit will perform and can be visualized as a "path" in a time-space network of the form given in Figure 1. Given an initial inventory of inv_u^d units in depot

$d \in \mathcal{D}$ of type $u \in \mathcal{U}$, the model must allocate exactly inv_u^d itineraries (including the option of not moving) for units of type u. The set of all possible unit type $u \in \mathcal{U}$ itineraries that originate depot $d \in \mathcal{D}$ is denoted by $\mathcal{P}_d^u$, while the full set of itineraries is hence $\mathcal{P} := \bigcup_{u \in \mathcal{U}, d \in \mathcal{D}} \mathcal{P}_d^u$. Binary variables λ_p are introduced and govern the inclusion of path $p \in \mathcal{P}$ in the solution. Additional parameters are needed to tie the itineraries to the composition variables of model (1)-(8). The binary parameter α_p^t indicates whether itinerary $p \in \mathcal{P}$ operates trip $t \in \mathcal{T}$, μ_c^u indicates the number of units of type $u \in \mathcal{U}$ required in train composition $c \in \mathcal{C}$. The path based formulation for the RSP can hence be stated as follows:

$$\text{Minimize:} \quad F(\lambda, x, z) \tag{9}$$
$$\text{s.t.} \quad (2) - (4)$$
$$\sum_{d \in \mathcal{D}} \sum_{p \in \mathcal{P}_d^u} \alpha_p^t \lambda_p = \sum_{c \in \mathcal{C}} \mu_c^u x_c^t \qquad \forall u \in \mathcal{U}, t \in \mathcal{T} \tag{10}$$
$$\sum_{p \in \mathcal{P}_d^u} \lambda_p = inv_d^u \qquad \forall u \in \mathcal{U}, d \in \mathcal{D} \tag{11}$$
$$(7) - (8)$$
$$\lambda_p \in \{0, 1\} \qquad \forall p \in \mathcal{P} \tag{12}$$

This formulation is noticeably similar to that of model (1)-(8). The objective function, given in (9), has been modified to include the path variables, and two new types of constraints are required. Constraint (10) ensures that the number of units assigned to each specific trip is consistent with the number of units of each type in the chosen composition for a trip. Constraint (11) restricts the number of paths for a specific unit type originating a depot to be exactly equal to the inventory of that unit type available. These perform the role of Constraints (6) in model (1)-(8). The binary nature of the λ_p variables is given by (12).

Formulations (1)-(8) and (9)-(12) both model the RSP. The latter, being a path based formulation, allows unit specific requirements to be modelled, e.g. itinerary mileage. Depending on which is used, the subsequent TUSP takes on a different form. In the path based formulation units can be followed through the network. The variation stems from the fact that an itinerary contains all information on when units enter and exit depots and what trips they perform. The flow based formulation, on the hand, does not have such information.

3.2. The Train Unit Shunting Problem (TUSP)

The TUSP focuses on the movements in one depot and tries to coordinate them in a conflict free manner. At the same time, it must ensure that depot track capacity is never violated when parking units. An instance of the TUSP is thus characterized by a set of tracks, T, each of which has a known length, and a set of required shunting movements $\mathcal{E}$. The set $\mathcal{E} := \mathcal{E}_{arr} \cup \mathcal{E}_{dep}$ is partitioned into two subsets, where $\mathcal{E}_{arr}$ contains all arriving movements and $\mathcal{E}_{dep}$ contains all departing movements. An arrival movement arises whenever a unit is uncoupled from a train and driven to the depot, while a departure movement arises when a unit requires coupling to a train and must leave the depot. Associated with any movement $e \in \mathcal{E}$ is information concerning the time the movement occurs, denoted by $\texttt{time}(e)$, and its unit type, denoted by $\texttt{type}(e)$. Typically the aim of the TUSP is to determine whether or not a given set of shunting movements induced by a rolling stock schedule is feasible; however other objective functions are also possible. We argue that in a short term planning, it is sufficient to detect feasibility of the depots as the majority of the cost is incurred by the rolling stock schedule. Note, if feasible, depots can always be resolved independently to achieve a "better" solution.

If a flow based model has been used to obtain the rolling stock schedule, all that is known at a particular depot are the times when units of certain types arrive and depart. For a given arrival it is not specified what trip the incoming unit type will perform next. In this case the TUSP includes a matching problem; arriving movements must be paired with departing movements and assigned
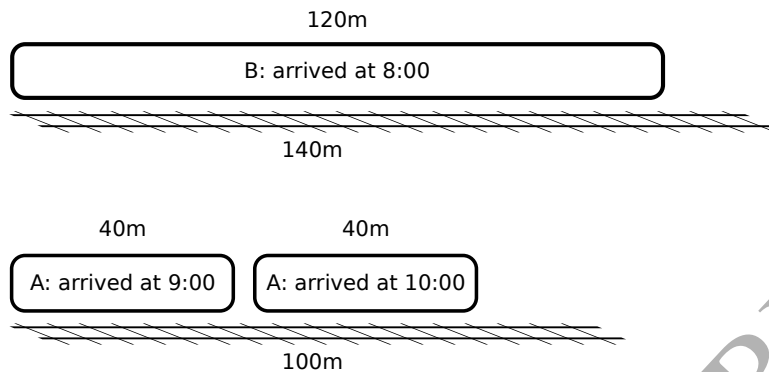
Figure 2: An example of the TUSP with three units of two distinct types and two tracks.

a physical unit. Once such a matching has been obtained, the tracks on which (and the order in which) the corresponding units are to be parked can be determined. An instance of the TUSP is feasible if there exists a feasible matching in which the corresponding units can be ordered feasibly on the tracks.

As an example, consider a depot with two tracks. Each track can only be accessed from one end. The first has length 100m, and the second has length 140m. Three arrivals are scheduled. A unit of type A (of length 40m) arrives at 9:00 and at 10:00. One unit of type B (of length 120m) arrives at 8:00. Three departures are scheduled, a unit of type A must leave at times 12:00 and 16:00, while a unit of type B must leave at 20:00. Figure 2 depicts the depot at time 11:00. Due to the track lengths, this is the only feasible ordering of the units on the tracks. The unit that arrives at 10:00 must be matched to the departure at 12:00, otherwise the movements of the two A units would be in conflict; the unit at the open end of the track would block the other.

In general, every arrival movement can be matched to one of possibly many departures, as long as the unit type associated with the two movements is the same. We define the set of all possible matchings at a depot to be:

$$
\begin{aligned}
\mathcal{M} = \{(e_a, e_d) | e_a &\in \mathcal{E}_{arr} \wedge e_d \in \mathcal{E}_{dep} \\
&\wedge \texttt{type}(e_a) = \texttt{type}(e_d) \\
&\wedge \texttt{time}(e_a) \leq \texttt{time}(e_d)\}
\end{aligned}
$$

An arrival and departure movement can be feasibly matched if and only if the unit type is identical and the departure occurs after the arrival in time. Note that the set of matchings should in practice be restricted further, e.g. by requiring a minimum separation time between the arrival and departure. Initially parked units can be easily considered in the form of artificial arrival movements that have occurred prior to the planning horizon under consideration. Additional information concerning which tracks any initial units are parked on (and their) respective order is assumed to be given. Similarly, artificial departure movements that correspond to a unit remaining at the depot beyond the end of the planning horizon can also be used. As such we can assume without loss of generality that $|\mathcal{E}_{arr}| = |\mathcal{E}_{dep}|$, each set possibly augmented with artificial movements. The set $\mathcal{M}_e \subset \mathcal{M}$ is assumed to contain all matchings that include movement e, and c_{mt} indicates the cost or unattractiveness of assigning matching $m \in \mathcal{M}$ to track $t \in T$. For convenience $\texttt{len}(m)$, a_m, and d_m give the length of the unit associated with matching $m \in \mathcal{M}$, the time of its arrival movement, and the time of its departure movement.

The TUSP can be formulated as a binary integer program. Decision variables x_{mt} can be used to indicate whether or not matching $m \in \mathcal{M}$ is assigned to track $t \in T$. A value of one indicates that the corresponding arrival and departure movements are matched, and, furthermore, that the arriving unit is parked on track t at the depot until its departure time. Any two matchings for which the corresponding movements cannot be assigned the same depot track due to an ordering

10

violation (i.e. LIFO restrictions) are said to be in conflict. All such pairs are contained in the set $\mathcal{C}$. The length of track $t \in \mathcal{T}$ is denoted by $L_t \in \mathbb{R}$. Finally, we define the set $\mathcal{N}_e \subseteq \mathcal{M}$. This set contains all matchings that overlap (in time) with the arrival e. The full formulation can now be stated as follows and is similar in structure to that which appears in Lentink et al. (2006) and Kroon et al. (2008).

$$\text{Minimize:} \quad \sum_{m \in \mathcal{M}} \sum_{t \in T} c_{mt} x_{mt} \tag{13}$$

$$\sum_{t \in T} \sum_{m \in \mathcal{M}_e} x_{mt} = 1 \qquad \forall e \in \mathcal{E}_{arr}, \tag{14}$$

$$\sum_{t \in T} \sum_{m \in \mathcal{M}_e} x_{mt} = 1 \qquad \forall e \in \mathcal{E}_{dep}, \tag{15}$$

$$\sum_{m \in \mathcal{N}_e} \texttt{len}(m) \cdot x_{mt} \leq L_t \qquad e \in \mathcal{E}_{arr}, t \in T, \tag{16}$$

$$x_{m_1 t} + x_{m_2 t} \leq 1 \qquad \forall (m_1, m_2) \in \mathcal{C}, t \in T, \tag{17}$$

$$x_{mt} \in \{0, 1\} \qquad \forall m \in \mathcal{M}, t \in T, \tag{18}$$

The objective (13) is to minimize the overall cost of the found solution. For illustrative purposes we have defined the problem to have an objective function here, despite the fact we regard it as a feasibility problem in this paper. Constraints (14) and (15) require that every arrival and departure movement are selected in exactly one matching. Every movement must be covered exactly once. Constraint set (16) guarantees that the track capacity is never violated by ensuring that the capacity is respected on each track whenever an arriving movement occurs. Constraints (17) stipulate ordering restrictions; a pairwise packing constraint is included for any two conflicting movements. Variable domains are given by (18). Note that constraints (17) can be strengthened by identifying stronger clique inequalities. Typically this set of constraints can be problematic as the formulation can become prohibitively large if there are many conflicts.

When the rolling stock schedule also states individual unit itineraries, as is the case with model (9)-(12), the TUSP reduces to the TAP. This is because all matchings are specified in the schedule and all that remains is to order the units in a non-conflicting way on the depot tracks while respecting track capacity. Thus constraints (14) or (15) can be omitted and replaced by a set of constraints that ensure all of the prespecified matchings are assigned to tracks. We denote this set of matchings as $\mathcal{M}^* \subset \mathcal{M}$. The sets $\mathcal{N}_e$ and $\mathcal{C}$ are adjusted accordingly, where $\mathcal{N}_e^* \subseteq \mathcal{M}^*$ and $\mathcal{C}^*$ contains only conflicts relevant to the chosen matching. In addition, we introduce decision variables $y_m \geq 0$ to account for infeasibility. The variable states the portion of matching $m \in \mathcal{M}^*$ that cannot be assigned to any track. It is penalized with a large cost, M, in the objective function. For completeness we state the full model here.

$$\text{Minimize:} \quad \sum_{m \in \mathcal{M}^*} \sum_{t \in T} c_{mt} x_{mt} + M \sum_{m \in \mathcal{M}^*} y_m \tag{19}$$

$$\sum_{t \in T} x_{mt} + y_m = 1 \qquad \forall m \in \mathcal{M}^*, \tag{20}$$

$$\sum_{m \in \mathcal{N}_e^*} \texttt{len}(m) \cdot x_{mt} \leq L_t \qquad e \in \mathcal{E}_{arr}, t \in T, \tag{21}$$

$$x_{m_1 t} + x_{m_2 t} \leq 1 \qquad \forall (m_1, m_2) \in \mathcal{C}, t \in T, \tag{22}$$

$$x_{mt} \in \{0, 1\} \qquad \forall m \in \mathcal{M}^*, t \in T, \tag{23}$$

$$y_m \geq 0 \qquad \forall m \in \mathcal{M}^*. \tag{24}$$

The objective, (19) minimizes the cost of the track assignment and penalties incurred for not covering a matching (effectively not parking a unit). Constraints (20) ensure each matching is

11

assigned a track or left uncovered, while constraints (21) and (22) are equivalent to their respective counterparts in model (13)-(18). Variable domains are specified by (23) and (24).

We have described two possible formulations for the RSP and shown that each yields a slightly different version of the TUSP. In the coming section we discuss various aspects of the integrated frameworks. In the RFM we effectively combine formulations (9)-(12) and (19)-(24), while in the RLM we combine formulations (1)-(8) and (13)–(18). The TAP and TUSP essentially function as subproblems in the respective approach, generating constraints to cut away rolling stock schedules that are infeasible from a shunting perspective. In this development we also try to identify the most effective way to solve the TAP problem and report the results of this study.

4. Solution Approach

This section focuses on different aspects concerning the proposed solution approaches. We begin in Section 4.1 by providing an overview of the two BAC based approaches for solving the IRSUSP. Furthermore, given its importance to one of the two methods, and as various approaches have been proposed in the literature to solve it, we specifically consider solution methods for the TAP in more detail in Section 4.2. Being a sub-component of a larger framework, it is essential that we identify the most efficient way to solve this. Finally, in Section 4.3 we introduce a heuristic which attempts to obtain a feasible solution to an infeasible instance of the TAP through swapping the departure movements of matchings.

4.1. Integrated Approaches for the IRSUSP

In developing the integrated frameworks for the IRSUSP we make use of known methods for solving the two different formulations of the RSP outlined in Section 3. The flow based formulation, which is central to the RLM, is solved using a Branch-and-Bound (BAB) approach, while the path-based formulation, part of the RFM, is solved using the BAP procedure outlined in Haahr et al. (2016). To ensure rolling stock schedules which are feasible from a shunting perspective are obtained, we augment both formulations with additional constraints that cut away solutions that would otherwise give rise to infeasible shunting movements at depots. As a rolling stock schedule indicates where train composition changes take place, any given rolling stock schedule can be prohibited by explicitly adding a constraint that prevents some composition changes appearing. The constraint can be improved by constraining only the composition changes giving rise to the infeasible depot. It can be further improved by limiting the composition changes to the minimal set that are infeasible. Using the same notation as Section 3, all rolling stock schedules infeasible with respect to shunting movements could be removed by adding the following set of constraints:

$$\sum_{(t,c)\in\phi} x_c^t \le |\phi| - 1 \quad \forall \phi \in \Psi, \tag{25}$$

where Ψ is a full list of all feasible solutions to the RSP that are infeasible from a shunting perspective. Each element in $\phi \in \Psi$ is list of (trip, composition) pairs. As a depot infeasible rolling stock schedule is associated with a specific depot, a stronger cut is obtained by only specifying trip compositions for all trips that either enter or leave the infeasible depot. We note that the cut is still not strong as similar rolling stock solutions are not necessarily covered by the same constraint; e.g., the cut for one depot infeasible solution does not prohibit a very similar solution where one of the arriving unit types is shifted to arrive a little earlier or later.

There can be a potentially large number of depot infeasible solutions to the RSP. Consequently, the number of constraints that comprise (25) can be very large. Enumerating all such constraints is impractical. Furthermore, identifying all of them in advance is even more cumbersome. We therefore propose extending the two solution approaches for the RSP with a dynamic cutting routine, where constraints (25) are initially removed from the formulation. On finding a feasible rolling stock schedule in BAB tree we test the feasibility of the induced shunting movements. If any depot is infeasible, we add a cut of the form (25) and continue the optimization. On the other

12

hand, if the rolling stock schedule is shunting feasible, a solution to the IRSUSP has been found. Recall that we treat the TUSP as a feasibility problem. That is, the cost of the IRSUSP comes from the rolling stock schedule and no cost stems from the TUSP

BAC is a well known technique for efficiently solving large scale MIPs, see e.g. Desrosiers and Lübbecke (2010). This approach combines the addition of cutting planes within a BAB framework. More specifically, whenever the relaxation of a node in the branch-and-bound tree is solved to optimality, so-called *separation* routines are performed. Each separation routine attempts to identify valid inequalities, i.e. constraints that are not part of the original formulation, but which are violated by the node's fractional solution. By adding such valid inequalities one hopes to improve the bound by removing infeasible fractional solutions that would otherwise be branched on. The valid inequalities produce a tighter relaxation and often a less fractional solution.

In addition to generating valid inequalities from the problem's full constraint set it is also possible to remove a set of problem constraints, thus obtaining a relaxation, and run a separation routine that identifies any violated, relaxed constraints. This can be particularly useful if there is a large set of constraints, many of which are unlikely to be binding in an optimal solution. See e.g. the application of BAC algorithms to the traveling salesman problem and its variants Padberg et al. (1987), where the exponential number of sub tour elimination constraints can be more efficiently handled via a separation routine. This is the approach we adopt here. We initially remove all constraints (25) and then check for any violated ones whenever an integral solution to the RSP is found. In this way we circumvent the problem of bloating the rolling stock formulations with redundant constraints as any necessary ones are dynamically generated.

The purpose of the separation routine is the same for the RFM and the RLM; however, the form of the separation routine is slightly different due to the information available in the rolling stock schedules each approach provides. In both cases, however, all depots must be assessed (independently) when testing the feasibility of any induced shunting movements. That is, on finding a feasible rolling stock schedule, the separation routine sequentially iterates through all depots and solves a TUSP. If an infeasible depot is encountered, the separation routine terminates and returns a violated constraint (25). The current node of the BAB tree for the RSP is then reoptimized. Due to the absence of unit itineraries in Model (1)-(8), for the RLM the full TUSP (Model (13)-(18)) must be solved. For this a commercial MIP solver is used. For the RFM, however, only the TAP (Model (19)-(24)) is needed in the separation routine. In order to strengthen the cuts obtained, if an infeasible depot is encountered in the separation routine, we attempt to find the smallest set of shunting movements that create the infeasibility, thereby making the cut more general and not specific to one or a few solutions.

To determine this minimal set of movements, we propose an iterative approach. On detecting infeasibility, the TUSP is reduced in size so that it only contains the first i movements. Naturally, if the first i movements are infeasible from a shunting perspective, then no extended set of movements can remedy this. If this reduced set is feasible, its size is incremented by one (to include the next movement) and resolved. This process terminates when infeasibility is detected.

Note that for the RFM there may exist many different combinations of unit itineraries that yield the same train composition assignment for the train trips. Hence, if a particular TAP is infeasible with a given set of unit itineraries, it does not necessarily mean that the given composition assignment is infeasible. Since enumerating the entire set of unit itineraries for a given composition assignment is intractable, we also include the possibility of *swapping* units when infeasibility occurs. Swapping units effectively changes unit itineraries without modifying the train compositions. This heuristic approach is a tool for achieving feasibility and is described in detail in Section 4.3. Consequently, in the absence of an optimal method for swapping units, Constraint (25) for the RFM may actually cut away a feasible composition assignment for the integrated problem. With a heuristic swapping routine, the RFM is therefore heuristic. Although not considered here, if unit specific constraints must be respected, the swapping method can only swap units if the unit-specific constraints are satisfied.

On the other hand, as the routing of units is performed last, in the absence of unit specific constraints, the RLM is an exact procedure. An infeasible TUSP instance for one of the depots indicates the composition assignment could never be feasible for the integrated problem. The RLM

can also be extended to include unit specific requirements. The routing step of the algorithm would have to be modified such that itineraries that a feasible with respect to the unit specific constraints are satisfied. This may not be possible. As such, in this case, before cutting away a composition assignment, all feasible solutions to the TUSP instances would need to be considered. Iterating over these solutions is a trivial, but perhaps a time consuming task.

Both variants of the approach therefore have strengths and weaknesses. Without modeling unit-specific constraints, the RLM has an advantage. However, when modeling unit-specific constraints, it is less clear which variant is better for solving the TUSP.

4.2. Solving The Track Allocation Problem

In this section we introduce three different approaches for modelling and solving the TAP. A comparison of these approaches is provided in Section 6. Note that we do not claim to be the first to propose these; all have been suggested before. We do, however, make some new extensions as well as correct a flaw in one of them. The first two approaches, discussed in Section 4.2.1, are BAC procedures applied to Model (19)–(24). The third approach, discussed in Section 4.2.2, applies column generation to a reformulation of Model (19)–(24). The approaches have been suggested by, among others, Freling et al. (2005).

4.2.1. Branch-and-Cut

The first BAC procedure is the simplest and involves directly solving Model (19)–(24) with a commercial solver, e.g. Cplex. The solver is permitted to use its default settings, which includes the generation of any valid cuts it can identify. We refer to this approach as BAC_1. Explicitly including the large number of pairwise incompatibility constraints, (22), can result in a prohibitively large formulation as there are potentially many of these for large problem instances. Furthermore, the vast majority of these are likely to be redundant. An alternative approach therefore is to initially remove constraints (22) and dynamically add only violated ones as necessary. We refer to this approach as BAC_2. It is essentially the same as BAC_1 with the exception that constraints (22) are removed from the formulation and dynamically separated during the Cplex BAB solve. That is, whenever we have obtained the solution to a given node of the BAB tree, we analyse its solution for any LIFO violations to confirm/disprove the feasibility of the current solution. If a violated constraint (22) is found, it is added, and the problem is resolved. Note that if a LIFO violation is found for a pair matchings, we add a constraint (22) for each track in the depot to ensure that the conflict doesn't simply move to one of the other compatible tracks. In this approach we cut on both fractional and integer solutions. As we are considering a relaxation of Model (19)–(24), integer solutions to this problem may not necessarily satisfy constraints (22). Such solutions are cut away when introducing the violated constraints.

4.2.2. A Column Generation Model

Column generation is an efficient method for solving large scale linear programs and can, via a BAP algorithm be adapted to solve large scale MIPs, see e.g. Desrosiers et al. (2005); Barnhart et al. (1998). The method is typically used when it is computationally intractable to enumerate all possible variables a priori. To apply column generation, the problem is decomposed into a *master* problem and one or more independent *subproblems*. The master problem usually only contains a subset of its possible variables (often being referred to as the *restricted* master problem), while the subproblem is an optimization problem that implicitly considers all non-basic variables omitted from the master problem and attempts to identify promising candidates (i.e. non-basic master variables with favourable reduced cost) using the optimal dual solution to the master problem. Column generation is thus an iterative procedure between the master and subproblem(s) and only terminates when the subproblems cannot identify any improving variable for the master problem.

Freling et al. (2005) were the first to propose a column generation procedure for the TAP. An exact procedure is used to solve the models Linear Programming (LP) relaxation, and this is combined with a heuristic branching routine to provide an integer solution. The proposed model identifies so-called *track assignments* for the depot tracks. A track assignment for a given

14

track $t \in \mathcal{T}$ simply refers to a set of matchings that can be feasibly assigned to track t. That is, the movements associate with the matchings are conflict-free and respect the track's capacity. This formulation is obtained by applying Dantzig-Wolfe Decomposition to Model (19)–(24), where constraints (21), (22), and (23) are placed in the subproblem and thereby implicitly satisfied in the construction of the variables. We denote the set of all track assignments for $t \in \mathcal{T}$ as $\mathcal{P}_t$ and introduce a new set of binary variables x_{tp} that govern the selection of a particular track assignment $p \in \mathcal{P}_t$ for track $t \in \mathcal{T}$. A cost c_{tp} is associated with each such variable and indicates the unattractiveness of the assignment; this cost is just the sum of the costs of assigning each of the matchings contained in the track assignment. Finally, the binary parameter a_{mp} indicates whether matching $m \in \mathcal{M}^*$ is included in track assignment $p \in \mathcal{P}_t$ or not. The full formulation is given below.

$$\text{Minimize:} \quad \sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}_t} c_{tp} x_{tp} + M \sum_{m \in \mathcal{M}^*} y_m \tag{26}$$

$$\sum_{p \in \mathcal{P}_t} x_{tp} = 1 \qquad\qquad \forall t \in \mathcal{T}, \tag{27}$$

$$\sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}_t} a_{mp} x_{tp} + y_m = 1 \qquad\qquad \forall m \in \mathcal{M}^*, \tag{28}$$

$$x_{tp} \in \{0, 1\} \qquad\qquad \forall t \in \mathcal{T}, p \in \mathcal{P}_t, \tag{29}$$

$$y_m \geq 0 \qquad\qquad \forall m \in \mathcal{M}^*. \tag{30}$$

The objective function (26) is identical in structure to (13); however, the aim is to find a minimum cost selection of track assignments. Constraints (27) ensure that each track is assigned exactly one of its possible track assignments (including the null assignment, i.e. a track assignment containing no matchings), while Constraints (28) enforce the restriction that each matching must be present in exactly one of the track assignments, or left uncovered. Variable domains are stated by (29) and (30). This formulation has a constant row dimension (it is independent of the number of conflicting matchings), but can have an exponential number of variables. For large problems there can be many possible track assignments. Thus, column generation can be used to dynamically generate only those variables that have the potential to improve the objective function.

To apply column generation to Model (26)–(30) the integer requirement on the x variables is replaced with $x \geq 0$, and the set of possible track assignments is reduced to a smaller set $\mathcal{P}^* \subset \bigcup_{t \in \mathcal{T}} \mathcal{P}_t$. All that is assumed from this initial set is that it permits a feasible solution to the resulting formulation. A subproblem can then be identified for each track (of distinct length), and its role is to identify promising track assignments for the corresponding track using an optimal dual solution to the master problem. For completeness, we denote the dual values on constraints (27) and (28) in this relaxation as π_t and μ_m, respectively. In this way the many track assignment variables that would otherwise assume a non-basic status in the optimal linear programming are purposely omitted from explicit consideration. As described in Freling et al. (2005), the problem of generating the track assignments for a given track can be modelled as a Resource Constrainted Shortest Path (RCSP) problem. We provide a brief overview of this in the following section.

4.2.3. Resource Constrained Shortest Path

RCSP problems typically arise in the application of column generation to vehicle routing and crew rostering problems. They involve finding a shortest path between two nodes of a graph while adhering to a number of *resource constraints*, which limit the set of feasible paths. A feasible path in the graph corresponds to a solution of the subproblem. Freling et al. (2005) model the track assignment generation using an acyclic graph $G = (V, A)$. The authors show that the methodology is applicable to free tracks (i.e. tracks that are open from both ends); however, we describe it in the context of LIFO tracks as this is the only type of depot track in DSB S-tog's network. In addition to a source node, $\mathcal{O}$ and sink node $\mathcal{D}$, the node set contains a pair of nodes for each matching $m \in \mathcal{M}^*$. The first indicates that the matching is assigned to the track, while the second indicates

it is not. Each such pair of nodes comprises a layer, and these layers are sorted in increasing time of the arrival movements. Associated with each layer is hence a time-stamp corresponding to the time of the relevant matching's arrival movement. Arcs are used to connect the nodes of one layer with nodes of the subsequent layer and indicate assignment choices. The source node, $\mathcal{O}$, is connected to nodes in the first layer, while the nodes of the last layer are connected to the sink node, $\mathcal{D}$. Figure 3 gives an example of such a graph, obtained from a TAP instance with six matchings. Nodes that correspond to unassigned matchings are marked with ($'$). There is a cost an all arcs entering a node that corresponds to assigning a matching that indicates reduced cost contribution of the assignment; i.e. it is the matching's assignment cost c_{mt} adjusted by the dual value for the matching constraint it covers, μ_m. The dual contribution of π_t must be applied to any track assignment for track t. For convenience, it can be included on all arcs which connect the source node with nodes of the first matching. The rectangles in the figure illustrate the layers.
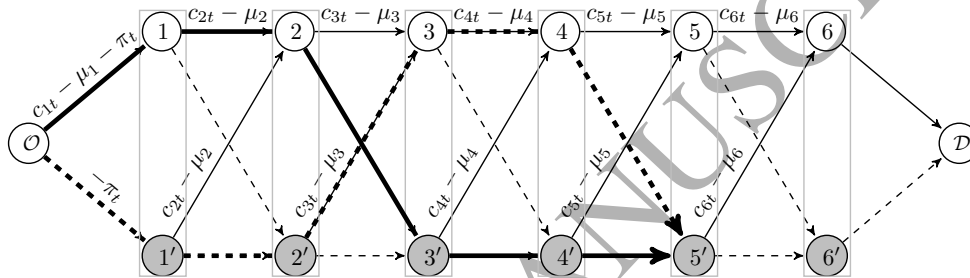

Figure 3: An example subproblem network with six matchings. There are two nodes for every matching, indicating whether the matching is assigned to the track or not.

Enumerating all paths between $\mathcal{O}$ and $\mathcal{D}$ provides the set of all subsets of matchings. Due to the resource constraints, however, not all subsets are feasible. Some subsets might violate track capacity or contain conflicting movements. The aim of the subproblem is to find the best set of feasible matchings given dual vectors $\boldsymbol{\pi}$ and $\boldsymbol{\mu}$. RCSP problems are typically solved using a *label setting algorithm*, in which the nodes of the graph are processed in topological order. A label is associated with a given node v and contains information, in terms of resource levels, about a unique partial path to v from $\mathcal{O}$. When processing node v new labels are generated by *extending* the set of labels associated with node v to each of node v's successor nodes. Each new label reflects the label from which it is generated, updated with any changes to resource levels that have occurred in traversing the arc. Dominance strategies are used to restrict the number the partial paths extended from a given node by removing labels that would ultimately lead to inferior paths. Figure 3 also shows two examples of partial paths, given by the dashed and bold lines. The solid line indicates a partial track assignment that includes matchings 1 and 2, while the dashed line corresponds to a partial track assignment which includes matchings 3 and 4. We refer the reader to Irnich and Desaulniers (2005) and Irnich (2008) for more details on solving RCSPs.

Freling et al. (2005) claim that three resources must be monitored when generating track assignments. These are: the `cost` (or total reduced cost contribution) of the partial path, the `remaining length` of the track, and the `earliest departure time` included in the partial track assignment that has not yet occurred. Track length must be monitored to avoid generating track assignments that violate the track's capacity, while the earliest departure time must be updated to avoid extending the partial track assignment with conflicting matchings. Updating these resource levels is relatively straightforward. Below we give an example of this between nodes two layers. Given in the extension step is a partial path p_u, containing a set of matchings $\mathcal{M}_{p_u}$, and an arc $(u, v) \in A$ with cost ρ_{uv}. The cost, c_{p_v}, remaining length, l_{p_v}, and earliest departure time, δ_{p_v}, of the resulting partial path p_v are calculated as follows. Here m_v states the matching associated with node $v \in V$, and the set $\mathcal{D} \subseteq \mathcal{M}_{p_u}$ which all matchings whose departure movement occurs on the interval. $[a_{m_u}, a_{m_v}]$. How to extend the partial path depends on whether node $v \in V$
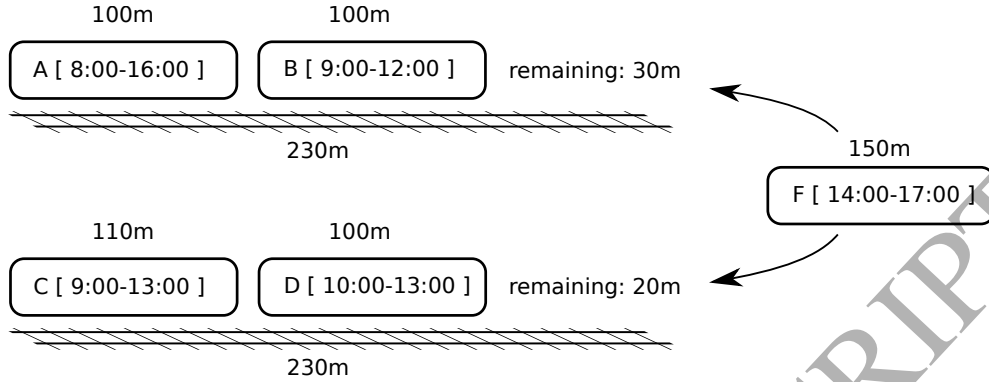
16

Figure 4: Example of two partial track assignments. The boxes represent units of individual lengths and parking durations. Both track assignments do not include matching E, leaving them in the same state (node in the graph), thus making it possible to check for dominance. The lower track assignment should not be dominated because the upper track assignment cannot include matching F in contrast to the lower one. If matching F has a negative cost, then a feasible and better track assignment is wrongly pruned.

corresponds to assigning the associate matching or not. Set $\mathcal{V}$ contains all nodes that indicate the corresponding matching is assigned, while $\mathcal{V}'$ contains the nodes that indicate the matching is not assigned.

$$
c_{p_v} = c_{p_u} + \rho_{uv}
$$

$$
\delta_{p_v} = \begin{cases} d_{m_v} & v \in \mathcal{V} \\ \min_{m \in \mathcal{M}_{p_u} \setminus \mathcal{D}} d_m & v \in \mathcal{V}' \end{cases}
$$

$$
l_{p_v} = \begin{cases} l_{p_u} + \mathtt{len}(m_v) - \sum_{m \in \mathcal{D}} \mathtt{len}(m), & v \in \mathcal{V} \\ l_{p_u} - \sum_{m \in \mathcal{D}} \mathtt{len}(m) & v \in \mathcal{V}' \end{cases}
$$

an extension is assumed to be feasible if $l_{p_v} \geq 0$ and, in the case $v \in \mathcal{V}'$, $\delta_{p_v} \leq \min_{m \in \mathcal{M}_{p_u} \setminus \mathcal{D}} d_m$. From a dominance perspective, Freling et al. (2005) argue that given two partial paths at a given node, one dominates the other if it is "better" in all resources. In this case better means having a lower cost, greater remaining track length, and earlier departure time. We show that this is not necessarily true in the following section.

4.2.4. Dominance Rule Counter Example

By way of a counter example we demonstrate that the dominance procedure stated in Freling et al. (2005) is inherently heuristic. While the authors do acknowledge that their full solution algorithm is heuristic, the reason is claimed to be due to their heuristic branching strategy – as columns are only generated at the root node – and *not* their variable generation procedure. In fact, the paper argues that the column generation procedure applied to the LP relaxation of Model (26)–(30) produces a *lower bound* for Model (26)–(30). This is incorrect. Using the prescribed dominance procedure, we show the algorithm may fail to optimally solve a subproblem by incorrectly removing non-dominated labels. Our counter example is given in Figures 4 and 5. Figure 4 gives a schematic view of the situation, while Figure 5 shows the graphical structure of the subproblem in this case

In the example a 230 meter track and a set of six matchings {A, B, C, D, E, F } are considered. The arrival movement of each matching is assumed to appear in the listed order. Let us assume that the costs of assigning the matchings (track assignment cost - associated dual value) are {-2, -2, -1, -2, 10, -10 }. Two partial paths to node E′ are depicted in Figure 5. The solid line

corresponds to a partial path (track assignment), p_1, in which matchings A and B are included, while the dashed line corresponds to a partial path, p_2, in which matchings C and D are included. If we assume that the arrival movement of matching E occurs at 10:30 and its departure movement occurs at 18:00, the label for the first path would contain resource levels (-4, 30, 12:00), while the label for the second path would have resource levels (-3, 20, 13:00). In each case the cost of the path is the accumulated cost of assigning the respective matchings to the track, the remaining length is the total track length adjusted by the combined length of the physical units in the matchings, and the earliest departure time is the departure time of the outmost unit on the track. If we apply the dominance procedure of Freling et al. (2005), then the label associated with path p_2 would be dominated by the label for path p_1 as it has a higher cost, less remaining track length, and a later earliest departure time. However, doing so actually removes the optimal solution {C,D,F}, which has cost -13.
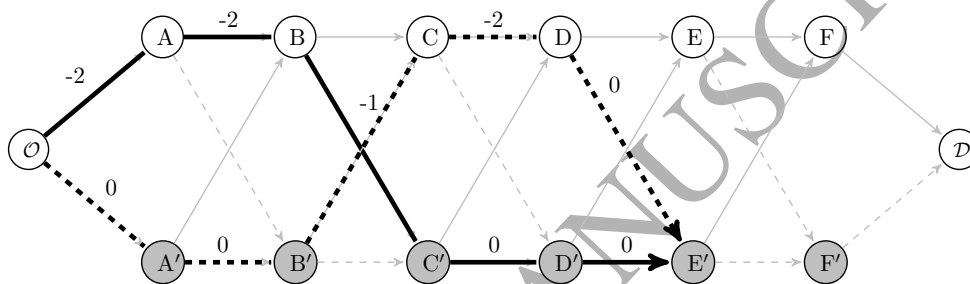

Figure 5: Dominance counter example: The dashed partial path corresponds to assigning movements C and D to the track, while the solid partial path corresponds to assigning movements A and B.

The reason why this occurs is because parking on tracks is only a temporary resource consumption; a unit will claim the track capacity for a specified interval of time only. Therefore, it is incorrect to assume that one partial track assignment has more available future capacity than another by simply comparing the times of the next departures in each partial assignment. We illustrate this with another figure, Figure 6. To assess the capacity of a depot track over time, we can partition the time horizon (from the time of the first arrival movement to the time of the last departure movement) into a set of smaller intervals of time, T. The end points of such intervals correspond to points in time when the track utilization can change, namely when arrival or departure movements occur. Between such events the track utilization remains constant. Figure 6 considers the track utilization for partial paths p_1 and p_2 and indicates the remaining track capacity by time interval, where $|T| = 9$. If we consider interval [10:30-12:00] (the time between the arrival movement of matching E and the departure movement of matching B) we see that p_1 has more available track capacity; however, if we look at future intervals, this is not the case. In particular, in interval [14:00-16:00], when the arrival movement of matching F occurs, more capacity is available on p_2. Future capacity thus depends on the time of all departure movements for the matchings in a partial track assignment, not just the next to leave. In our results in Section 5 we report cases where the method of Freling et al. (2005) obtains suboptimal solutions. As expected, the simpler non-optimal dominance does result in faster solution times.

To produce an optimal column generation procedure we thefore revise the dominance criteria specified Freling et al. (2005). In addition to the cost, remaining length, and earliest departure time (the last two assist in extension feasibility), we introduce a remaining length for each time interval $t \in T$. Such resources are updated similarly to the remaining length; when assigning a matching it consumes track capacity for all intervals the corresponding unit is in the depot. When the unit leaves it releases capacity for future time intervals. Given two labels at a node, one dominates another if it is better in cost and has at least as much capacity in all future time intervals. Clearly, p_2 would not be dominated by this approach.
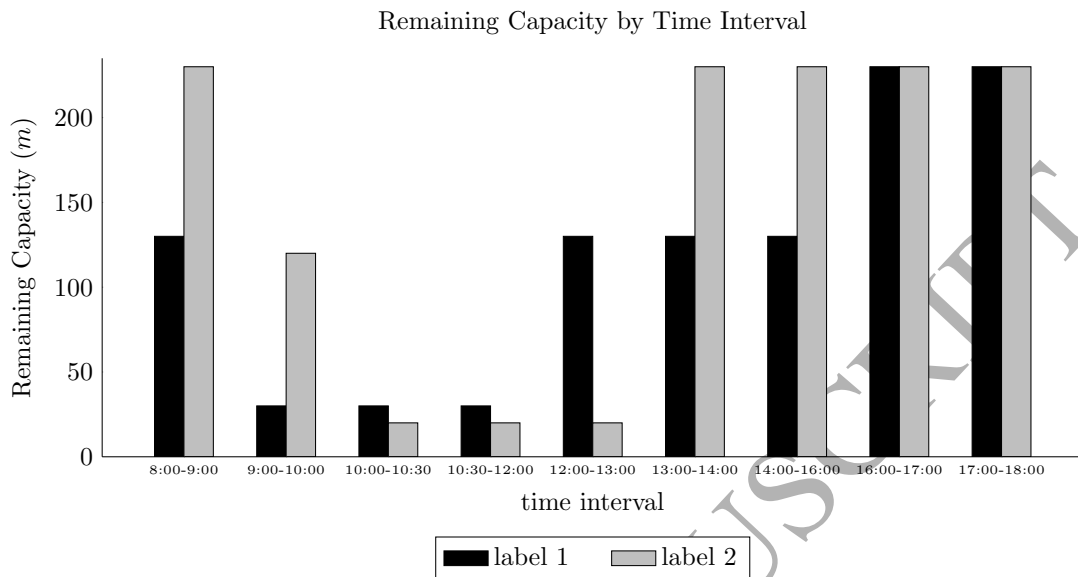
18

Remaining Capacity by Time Interval

Figure 6: A comparison of track utilization by time period for the example given in Figure 4

4.2.5. Constraint Branching

The column generation procedure by Freling et al. (2005) is combined with a heuristic branching strategy to produce an integer solution. In particular, a MIP problem is solved using the columns obtained during the convergence of the root node solve of the BAB tree. In this section we propose an application of constraint branching to ensure optimality of the column generation procedure described in Section 4.2.2 (with the revised dominance). The constraint branching technique, developed by Ryan and Foster (1981), is a well known method for solving set partitioning problems. The authors observe that in an optimal, but fractional solution there must exist two constraints (say c_1 and c_2) such that $\sum_{j \in J(c_1,c_2)} x_j < 1$, where $J(c_1, c_2)$ defines the set of variables that cover (or contribute to) both constraints c_1 and c_2. Two branches are then created by ensuring that either $\sum_{j \in J(c_1,c_2)} x_j \geq 1$ (the *one* branch), or $\sum_{j \in J(c_1,c_2)} x_j \leq 1$ (the *zero* branch). On the *one* branch both constraints must be covered by the same variable, while in the *zero* branch, they must be covered by different variables. For the TAP, we branch on track and matching pairs. In other words, on the *one* branch, a matching is assigned to a specific track, while on the *zero* branch it is prohibited from being assigned to a given track. In an optimal, fractional solution the track and matching pair with the closest fractional coverage to 0.5 is selected to branch on. When branching, all variables that are in the master problem and which do not satisfy the imposed branches are removed, and the subproblems are modified to ensure they return track assignments consistent with the imposed branches. One advantage of this constraint branching approach is that minimal changes to the subproblems are required. Except for removing columns the master problem remains unchanged. When branching, we adopt a depth first strategy in which we successively enforce *one* branches. For set partitioning problems we do not expect the optimality gap to be large; the approach should therefore quickly find an incumbent of high quality.

4.3. Unit Swapping

Feasibility of the TAP is unlikely to be a problem if the number of depot tracks is sufficiently large; however, for DSB S-tog, depot capacity is a scarce commodity. If an instance of the TAP is infeasible, instead of cutting the solution to the RSP away, feasibility can possible be restored by swapping the departure movements of two conflicting matchings. This approach would, however, relax the fixed matching assumption in the TAP. Swapping can only be done if the units associated

with the conflicting matchings are considered interchangeable, i.e., they must be of the same type and the must also be compatible with respect to any unit specific constraints. With the absence of unit specific constraints, such as maintenance, all units of the same type are interchangeable. At railway operators, such as DSB S-tog, maintenance checks on units are issued at certain times and itineraries are allocated to units whilst adhering to this. Thus, swapping units is not considered a general remedy to correct for feasibility. Note that in this work, we do not take unit maintenance requirements into account; we assume this is done in a post-processing phase.

To swap departure movements of two matchings, we propose a heuristic that iterates over the set of unassigned matchings, $\mathcal{U}$ and, for each $u \in \mathcal{U}$, identifies a set of swap possibilities $\mathcal{S}_u$ by considering the track assignments of each of the assigned matchings. In particular we look at the matchings present in the depot when the arrival movement of u occurs as well as the future arrival movements of other matchings that will happen. An attempt is then made to swap the departure movement of u with either a unit at the front of a stack, or with one of the next ones to arrive.

An example of a swap is given in Figure 7. This example contains three matchings: A, B, and C. The respective units are all of the same type and the arrival and departure time of each unit is given on the matching. Given these times, and assuming the depot only has one track (with its open end to the right), it is not possible to assign all matchings; unit associated with matching C would block that associated with matching A. On this example the heuristic would work as follows. The solution with the two matchings would be retained and matching C would be placed in set $\mathcal{U}$. The heuristic then considers each track of the depot and attempts to swap the time of the departure movement of C with the matchings assigned to the depot track under consideration. The arrival movement of C occurs at 8:00. The only unit on the track at this time is that associated with matching A. A check is then made to see if swapping the departure times of the two matchings reduces the infeasibility (i.e. effectively results in an extra unit being parked). This involves checking to see that the remaining matchings assigned to the track are still feasible. If they are, this results in a swapping possibility which, if performed in isolation, would increase the number of parked units by one. In this simple example, there is only one assigned matching and the swap results in feasibility. Updated matchings A$'$ and C$'$ are indicated in the figure. For more complicated situations, there might be more than one unassigned matching. In such cases several iterations might be needed, where at any given iteration the heuristic considers making at most one swap per track.

| 7:00 A 16:00 | 12:00 B 15:00 | | 8:00 C 17:00 |

Infeasible

| 7:00 A' 17:00 | 8:00 C' 16:00 | 12:00 B 15:00 |
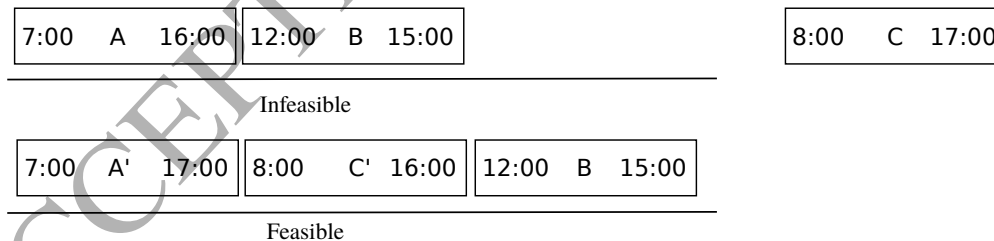
Feasible

Figure 7: An example of a swap

Essentially the heuristic is given an infeasible solution to the TAP, containing one or more unassigned matchings. It then attempts to obtain feasibility by re-matching some of the departure movements; the aim being to make the minimal number of changes to achieve feasibility. The heuristic method is outlined in Algorithm 1. The algorithm terminates when no unassigned matchings exist, no swapping possibilities remain, or an upper limit on iterations is reached. Swap possibilities are ranked such that a matching with fewest swap possibilities is considered first. The algorithm's sequential nature makes it inherently heuristic.

Algorithm 1 Unit Swapping Heuristic

1: **procedure** UNITSWAP(*sol*)
2: *iterations* ← 0
3: *stop* ← false
4: **while** unassignedMatchings(*sol*) > 0 and stop=false and *iterations* < *limit* **do**
5:    $\mathcal{U}$ ← getUnassignedMatchings(*sol*);
6:    $\mathcal{S}$ ← ∅
7:    **for** $u \in \mathcal{U}$ **do**
8:      **for** $t \in \mathcal{T}$ **do**
9:        $\mathcal{S}$ ← $\mathcal{S} \bigcup$ createSwapPossibilities(*sol*, *t*, *u*))
10:    **if** $|\mathcal{S}| > 0$ **then**
11:      rankSwaps($\mathcal{S}$)
12:      *sol* ← performSwaps($\mathcal{S}$)
13:    **else**
14:      *stop* ← true
15:    *iterations* = *iterations* + 1

---

5. Computational Results

Two computational studies are presented in this section. The first, described in Section 5.1, is dedicated to the TAP, while the second, described in Section 5.2 looks at the performance of the integrated approaches on real-life instances from DSB S-tog. All tests have been performed on a dedicated `Intel(R) Xeon(R)` CPU X5550 @ 2.67GHz with 24 gigabytes of main memory running `Ubuntu` Linux 14.04. The commercial solvers `Gurobi` 6.0 and `Cplex` 12.6 have been used to experiment with solving the MIP and LP relaxations.

5.1. TAP Benchmark

In order to determine the best way of solving the TAP, we benchmark the approaches, described in Section 4, on a test set of 11 artificial problem instances. We compare four different solution methods described. These include the two BAC approaches (BAC_1 and BAC_2) and two versions of the column generation procedure. The first, BAP, is the proposed exact procedure (using our revised dominance strategy and constraint branching). The second, FEL, is almost the same as Freling et al. (2005); however, we extend it with the constraint branching procedure outlined in Section 4.2.5. A preliminary study of the available problem instances for DSB S-tog shows that the realistic instances are quickly solved by all methods. They have not been included for this reason. The artificial problem instances constitute a more difficult set of instances that vary in, among other things, the level of infeasibility, the number of matchings, the number of available depot tracks, and the number of unit types. Table 1 summarizes the problem instances. Note that the instances have been purposely created with infeasibilities and that these are in line with the size of the problems DSB S-tog faces. In general these instances have few unit types and consider the maximum number of matchings that could be encountered in the worst case for DSB S-tog.

The results are given in Table 2. An upper bound of 10 minutes is enforced on the solution time. We note, that the objective value here is equal to the number of unassigned matchings in the problem instance. The results show that the BAC_1 outperforms the other methods in general. No added value is observed in dynamically separating Constraints (22). Infeasibility is expected for these instances as they have been constructed with this in mind. As such, a subset of Constraints (22) must be active in any optimal solution. The generation of these cuts depends on the order in which the integer solutions are found. Given the hardness of the instances, it is understandable that BAC_1 performs better; it has full knowledge of the problem structure, while BAC_2 iteratively cuts away infeasible integer solutions in a potentially very large BAB tree. Omitting Constraints (22) initially in the BAC_2 procedure means we are effectively solving a relaxation and the separation routine is needed to ensure we get a feasible solution to Model (19)– Model (24). The FEL approach is slightly faster than the exact BAP approach, however, it also provides suboptimal solutions. As discussed in Section 4.2.4, this is due to the heuristic nature

Instance	$|\mathcal{M}|^*$	$|\mathcal{T}|$	L_{max}	Horizon (s)	Types	Unit Lengths
data0	66	6	300.0	17113	2	[35,70]
data1	69	6	500.0	17785	2	[35,70]
data2	62	5	850.0	21103	2	[42,84]
data3	75	5	850.0	24837	3	[30,60,90]
data4	72	5	700.0	21602	2	[42,84]
data5	59	5	740.0	21602	3	[30,60,90]
data6	79	5	800.0	25202	2	[35,70]
data7	79	6	790.0	25202	3	[35,50,75]
data8	78	7	900.0	24897	1	[42]
data9	101	8	1000.0	24964	2	[42,84]
data10	109	8	400.0	28529	2	[42,84]

Table 1: The artificial problem instances. The columns show the instance name, the number of matchings, the number of depot tracks, the length of the longest track (denoted by L_{max}), the planning horizon, the number of unit types, and the unit type lengths.

of the dominance rule presented by Freling et al. (2005). We conclude that, for the instances considered, the BAC_1 approach is the best solution method. This may not necessarily be true for larger problem instances as this MIP formulation, which explicitly contains all conflict constraints, is unlikely to scale well. From a scalability perspective the BAP and BAC_2 methods, which dynamically generate interesting components of their respective formulations, would be preferable.

			BAP Framework				BAC_1	BAC_2		FEL		
Instance	Z^*_{root}	Z^*	cols	n	l	t (s)	t (s)	t (s)	%	Z_{root}	Z	t (s)
data0	8.00	8	1030	37	18	**1.33**	3.21	*	12.50	13.25	11	1.30
data1	6.00	6	1317	41	20	1.74	**0.36**	*	16.67	11.68	7	1.43
data2	7.00	7	727	13	6	0.75	**0.12**	5.05	0.00	10.12	8	0.41
data3	8.00	8	1181	23	11	1.71	**0.11**	7.63	0.00	13.03	10	2.38
data4	11.00	11	834	25	12	1.05	**0.14**	*	9.09	14.79	14	1.17
data5	4.00	4	754	29	14	0.98	**0.05**	1.09	0.00	7.81	8	0.43
data6	11.00	11	841	17	8	**1.18**	1.99	*	36.36	17.53	14	0.58
data7	8.00	8	1740	41	20	2.18	**1.76**	*	12.50	9.66	9	1.83
data8	1.00	1	3236	99	49	4.86	**0.05**	0.36	0.00	4.00	1	3.78
data9	3.00	3	5125	87	43	17.33	**0.42**	*	66.67	5.65	3	6.78
data10	0.00	0	2413	147	73	6.14	**0.23**	1.29	0.00	0.00	0	5.17

Table 2: The TAP benchmark results. The first column identifies the instance, while columns two to seven report statistics on the proposed optimal column generation procedure (i.e. the revised dominance strategy is combined with a constraint branching approach). Z^*_{root} reports the optimal LP objective value, while Z^* gives the objective value of the optimal integer solution. In addition, we report the number of columns generated (*cols*), the number of nodes in the BAP tree (n), the deepest level of same tree (l) and the time needed to solve the instance (t) in seconds. For the two BAC approaches the columns show the time taken to solve the problem and, for BAC_2, the gap from optimality at termination. For the *FEL* approach, we give the root node objective value (not necessarily optimal), the objective value of the integer solution obtained, and the time taken.

Finally, we benchmark the swapping heuristic. The results are summarized in Table 3. We compare the BAP approach with BAC_1. In almost all cases feasibility through swapping can be achieved, and the heuristic nature is seen by the fact that the two approaches can yield different results. The heuristic does not optimize the number of swaps; such a "re-matching" problem could also be performed via a MIP, if running times are fast enough. The integration of the matching and parking in the general TUSP is thus motivated by the considered problem instances.

Figure 8: DSB S-tog network

23

			BAP Framework			BAC$_1$	
Instance	Z^*	Z	Swaps	t (s)	Z	Swaps	t (s)
data0	8	0	12	7	0	19	3
data1	6	0	10	20	0	11	4
data2	7	0	11	6	0	16	1
data3	8	0	12	15	3	15	3
data4	11	1	21	126	1	16	5
data5	4	0	4	3	0	5	0.2
data6	11	0	22	19	1	18	16
data7	8	1	16	145	0	18	44
data8	1	0	1	9	0	1	0.1
data9	3	0	4	42	0	4	1

Table 3: The unit swapping results. For each approach the columns report the new objective (Z), the number of swaps, and the time taken. For any instance, up to 30 swapping iterations are allowed.

Name	Stops	Trips	Trips*	Weekday	Lines
Mon	28 017	4 468	868	Monday	A,B,Bx,C,E,F&H
Fri	28 719	4 558	886	Friday	A,B,Bx,C,E,F&H
Sat	20 474	1 916	590	Saturday	A,B,C&F
Sun	19 919	1 871	574	Sunday	A,B,C&F

Table 4: Four timetables operated by DSB S-tog. The columns respectively show the instance names, total number of stops, total number of trips, total number of non-reducible trips (Trips*), weekday, and finally the lines that are running.

5.2. The IRSUSP Benchmark

In this Section we perform tests on both BAC integrated frameworks, RFM and RLM, described in Section 4.1. Recall that they are identical in structure but differ in how the RSP is solved, and this difference results in a slightly different separation for determining cuts. The real-life problem instances are provided by DSB S-tog. An overview of the network is given in Figure 8. DSB S-tog operates a weekly schedule that consists of four daily distinct timetables. The main difference between the timetables is in the number of train trips operated; on the weekends there is typically a reduced number of total services. It is also normal to have additional night trip services on Fridays and Saturdays. The instances are summarized in Table 4. The table reports among other things the number of trips and the number of *irreducible* trips. Some stations do not have depots and thus do not permit (un)coupling activities. The irreducible set is obtained from the full set of trips by aggregating trips between stations where the train's composition cannot change. The "Lines" column simply states which of DSB S-tog's lines are operating.

One key aspect of the problem is whether an initial position (on the depot tracks) is given for the units in inventory, or whether this can be determined by the solver. This essentially highlights the difference between tactical level planning and operational planning. For the former it is usually possible to determine the initial positions of the units in the network (i.e. on which depot track and in which order); however, this is not usually the case on an operational level when a schedule is being executed and the initial positions of the units is fixed. We therefore perform two sets of experiments, one where initial positions for the units can be decided by the solver and one where they are fixed. We expect the need for integration to be more pronounced when the initial positions are fixed. When the initial positions can be decided there is significantly more flexibility. If depot capacity is satisfied on an aggregated level in the RSP, it is unlikely that TUSP will be infeasible. This is confirmed by the results in Table 5; no infeasible solutions to the RSP are encountered when checking for depot feasibility when the initial positions of the units are not fixed. Note that for both formulations of the RSP the objective balances penalties for seat-shortages, driven mileage, end-of-day balance deviations, and (un)couplings. In particular,

24

the penalty for a lack of seats is 0.5 per seat; the cost for driving one kilometer is set to 0.1; a depot imbalance incurs a penalty of 10000 per missing unit; and the cost of (un)coupling is set to 1000. These settings were determined in collaboration with DSB S-tog.

The results demonstrate that the proposed methodologies are acceptable for tactical planning purposes.

| | | RFM | | RLM | |
Instance	Objective	Time (s)	#Cuts	Time (s)	#Cuts
Fri	731,589.5	53.46	0	14.27	0
Mon	717,371.1	363.33	0	13.87	0
Sat	418,791.8	5.72	0	6.73	0
Sun	411,940.2	5.92	0	6.51	0

Table 5: Summary of benchmarks performed on all instances without enforcing any particular initial positions. The columns respectively show the instance name, runtime, number of depot cuts generated and the objective value.

Table 5 indicates that on a tactical level all optimal rolling stock schedules are feasible. The integrated loop therefore seems unnecessary as no cuts are generated in either BAC framework. A possible reason for this could be as follows. At the start of the planning horizon, most units exit the depots and at the end of the planning horizon all units enter the depots. In between only a few units enter and exit the depots as a result of the need for rush-hour patterns. With the absence of an initial ordering, the model is always able to reorder the units such that they can initially leave the depot (aggregated depot capacity is enforced in the RSPs). Handling the rush-hour patterns is not difficult since the depots are far from being at capacity during these periods. Further experiments were performed with varying penalties for (un)coupling units, in effect encouraging more shunting movements; however, these, more artificial problem instances, were unable to generate depot activity that resulted in infeasibility. Weekend days appear to be easier to solve than weekdays. This is expected due to the reduced number of units in service.

In an operational planning context, the train dispatchers do not have the luxury of deciding the initial positions of the units. All unit positions must be adhered to; units are fixed to certain initial locations. In the following we benchmark different, randomly generated initial positions for the units. At DSB S-tog only two unit types exist, but most depots contain at least three tracks. In effect a random order is not expected to be very limiting. A total of ten random initial parkings are generated for every instance and solved using both solution methods. The averaged results of this are given in Tables 6 and 7. Table 6 summarizes the results when an optimality tolerance of 1% is acceptable, while Table 7 indicates what happens when optimality is required. We note here that for one Saturday instance no feasible solution was found with the RFM and that we enforce a time limit of two hours when trying to solve the instances to optimality.

Several important observations can be made from the results in Tables 6 and 7. The first is that both approaches are able to, in most cases quickly find solutions of good quality. A 1% gap is usually acceptable in a practical setting. Furthermore, the increase in objective value, compared to the case in which no shunting feasibility is considered, is at most 0.95%. It is also apparent that the RLM is, in general, faster than the RFM at obtaining solutions within 1% of optimality and that the two approaches struggle on different instances. For the RFM, the BAP algorithm employed to solve the rolling stock problem is more computationally intensive, so may struggle on harder rolling stock instances, while the TUSP is a more difficult problem to solve when considering shunting feasibility. Monday appears to be more difficult from a rolling stock perspective, while Saturday seems to give rise to the hardest TUSP problems.

Looking at Table 7, the difference between the two methods is even more pronounced. With the exception of the Saturday instances, the RLM is able to solve all instances to optimality within the two hour limit. Two of the Saturday instances give rise to two very difficult TUSP instances. As a consequence, one of the Saturday instances terminates with no solution after two hours, while the other terminates with an optimality gap of 0.17%. The most difficult TUSP instances appear

		Runtime (s)			
Method	Instance	Total	Depot	Cuts	Δ
RFM	Fri	65.64	3.68	7.40	0.93%
	Mon	2015.43	31.56	101.20	0.95%
	Sat	96.84	25.36	121.20	0.63%
	Sun	9.08	1.48	5.80	0.49%
RLM	Fri	31.70	18.28	2.60	0.52%
	Mon	17.54	3.20	1.80	0.63%
	Sat	168.96	41.00	37.11	1.02%
	Sun	8.60	0.81	2.40	0.56%

Table 6: Summary of benchmark performed on all instances where the initial unit positions in the depots must be respected and an optimality gap of 1% is enforced. The columns respectively give the method, instance name, the total runtime, the time spent in the depot separation routine, the number of depot cuts generated, and finally the maximum percentage increase in the objective value relative to the values in Table 5 (Δ). Columns "Total", "Depot", and "Cuts" are averages over at most ten different runs.

					Runtime (s)			
Method	Instance	OPT	NR	TO	Total	Depot	Cuts	Δ
RFM	Fri	6	0	4	2923.81	117.42	552.90	0.30%
	Mon	5	0	5	4168.88	118.57	629.30	0.85%
	Sat	2	0	8	5859.51	400.30	6564.60	0.45%
	Sun	10	0	0	50.21	13.54	37.30	0.35%
RLM	Fri	10	0	0	40.60	24.38	6.80	0.22%
	Mon	10	0	0	28.40	7.11	3.20	0.43%
	Sat	8	1	1	1522.08	950.99	893.40	0.39%
	Sun	10	0	0	13.55	3.64	17.40	0.35%

Table 7: Summary of benchmark performed on all instances where the initial unit positions in the depots must be respected and optimality is required. In addition to the data in Table 6, we also report the number of instances solved to optimality (OPT), the number of instances for which no feasible solution was obtained (NR), the number of instances that time out (TO). Again, Columns "Total", "Depot", and "Cuts" are averages over at most ten different runs.

to be those that are almost feasible (or almost infeasible). Detecting infeasibility (or feasibility) in such cases can be extremely challenging. For the instance that times out without a solution, the MIP based approach for solving a single instance of the TUSP ultimately generates a large BAB tree with hundreds of thousands of unexplored nodes when the two hour time limit is reached. A better method for solving the TUSP would drastically improve the performance of the RLM. As a comparison, the RFM, with the exception of the Sunday instances, struggles and times out on many more instances. Consequently, the percentage increase in objective value is higher for the solutions in this case. The results do confirm that detecting feasibility/infeasibility of the depots with respect to the shunting movements is noticeably easier (per solve) with the RFM; however, this comes at the expense of a more computationally expensive algorithm for solving the RSP. For the RFM, the separation routine only makes up for between 3-25% of the total runtime on average, while for the RLM, this portion can be as high as 65% on average. For the cases that require most runtime with the RLM, the majority of runtime is spent in the separation routine.

Compared to the previous results in Table 5, the increase in objective is relatively small, in the region of 0.10–0.50% at most for the considered instances. This conclusion is based on the RLM only, as almost all of the instances are solved to optimality with this approach. We thereby know that this is precisely the increase in objective value expected in the worst case. The results do confirm the need for integration in this context; feasibility of the induced shunting movements is

not guaranteed without the separation routine. In the experiments a very large number of rolling stock schedules were identified in the BAB search trees. In some cases, in excess of 10,000 rolling stock schedules were considered), with the majority of these being infeasible. Finally, we note that swapping units (Section 4.3) is generally not necessary in order to get feasible solutions for the RFM and that the RFM, despite being heuristic in the absence of an exact swapping method, does not appear to compromise optimality for the considered instances, obtaining solutions with the same objective value as the RLM where such a comparison is possible.

6. Conclusions

In this paper we consider the problem of integrating rolling stock scheduling with train unit shunting, a problem which, to our knowledge has not been previously addressed. We have proposed two BAC procedures for solving this. Essentially the only difference between them is the formulation used to solve the RSP. This gives rise to different separation routines. For the RFM the separation routine is shown to entail solving the TAP. For this we also present a short computational study of different solution methods and, in doing so, highlight an error in the previously proposed column generation procedure of Freling et al. (2005). The results suggest that a solving a MIP formulation of the TAP without any decomposition, outperforms column generation approaches. However, given larger problem instances, it is unknown which method scales better. The results also show that, for the considered instances, dynamically separating violated LIFO constraints bears no advantage.

Computational results for the two proposed integrated frameworks highlight the need for integration on an operational planning level, where the problem is more constrained due to the initial positions of the units. We have demonstrated that in such situations the traditional approach of sequentially solving the RSP and then the TUSP can result in infeasibility. The integrated frameworks consider multiple rolling stock schedules, finding the best one from a shunting perspective. Our results show that the resulting objective is only slightly increased when integrating the problems compared to solving the RSP (ignoring the depot problem). We have also shown that both the RFM and the RLM are capable of quickly finding good solutions and that, if optimality is required, the RLM is superior. The results also indicate that if the integrated frameworks are allowed to determine initial unit positions, as would be the case on a tactical planning level, then integration of the two problems seems unnecessary. This logically makes sense. As long as the capacity of the depot is respected (which can be achieved in the RSP), it is almost always possible to reorder the units such that conflicting movements can be avoided.

In future work, it would be interesting to investigate different extensions of the RSP or the TUSP. Previous work in the literature includes some modelling choices with respect to shunting movements for entire train compositions as opposed to individual units. This addition has been omitted in this work and seems like an obvious choice for future research. Another interesting direction for future research includes train routing at the station and in the depots. Such considerations could be modeled in the separation routine presented in the proposed frameworks. Finally, maintenance constraints seem like an obvious extension which possibly motivates the RFM in preference to the RLM.

References

References

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., Vance, P. H., 1998. Branch-and-price: Column generation for solving huge integer programs. Operations Research 46 (3), 316–329.

Blasum, U., Bussieck, M. R., Hochstättler, W., Moll, C., Scheel, H.-H., Winter, T., 1999. Scheduling trams in the morning. Mathematical Methods of Operations Research 49 (1), 137–148.

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

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., Wagenaar, J., 2014. An overview of recovery models and algorithms for real-time railway rescheduling. Transportation Research Part B: Methodological 63, 15 – 37.

Desrosiers, J., Lübbecke, M., Solomon, M. M., 2005. Column generation. In: Desaulniers, G., Desrosiers, J., Solomon, M. M. (Eds.), A Primer in Column Generation. Springer: New York, Ch. 1, pp. 1–32.

Desrosiers, J., Lübbecke, M. E., 2010. Branch-price-and-cut algorithms.

Di Stefano, G., Koči, M. L., 2004. A graph theoretical approach to the shunting problem. Electronic Notes in Theoretical Computer Science 92 (0), 16 – 33, proceedings of ATMOS Workshop 2003.

Fioole, P., Kroon, L. G., Maróti, G., Schrijver, A., 2006. A rolling stock circulation model for combining and splitting of passenger trains. European Journal of Operational Research 174 (2), 1281–1297.

Føns, P., 2006. Desicion support for depot planning in the railway industry. Master's thesis, Technical University of Denmark.

Freling, R., Lentink, R. M., Kroon, L. G., Huisman, D., 2005. Shunting of passenger train units in a railway station. Transportation Science 39 (2), pp. 261–272.

Gallo, G., Miele, F. D., Jun. 2001. Dispatching buses in parking depots. Transportation Science 35 (3), 322–330.

Gatto, M., Maue, J., Mihalák, M., Widmayer, P., 2009. Shunting for dummies: An introductory algorithmic survey. In: Ahuja, R. K., Möhring, R. H., Zaroliagis, C. D. (Eds.), Robust and Online Large-Scale Optimization. Vol. 5868 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 310–337.

Haahr, J., Lusby, R., Larsen, J., Pisinger, D., 2014. A Branch-and-Price Framework for Railway Rolling Stock Rescheduling During Disruptions. DTU Management Engineering.

Haahr, J., Wagenaar, J. C., Veelenturf, L. P., Kroon, L. G., 2016. A comparison of two exact methods for passenger railway rolling stock (re)scheduling. Transportation Research Part E: Logistics and Transportation Review 91, 15 – 32.

Haijema, R., Duin, C., van Dijk, N. M., 2006. Train Shunting: A Practical Heuristic Inspired by Dynamic Programming. John Wiley & Sons, Inc., pp. 437–475.

Hamdouni, M., Desaulniers, G., Marcotte, O., Soumis, F., van Putten, M., 2006. Dispatching buses in a depot using block patterns. Transportation Science 40 (3), 364–377.

Hamdouni, M., Desaulniers, G., Soumis, F., Nov. 2007. Parking buses in a depot using block patterns: A benders decomposition approach for minimizing type mismatches. Comput. Oper. Res. 34 (11), 3362–3379.

Irnich, S., 2008. Resource extension functions: properties, inversion, and generalization to segments. OR Spectrum 30 (1), 113–148.

Irnich, S., Desaulniers, G., 2005. Shortest path problems with resource constraints. In: Desaulniers, G., Desrosiers, J., Solomon, M. (Eds.), Column Generation. Springer US, pp. 33–65.

Jacobsen, P. M., Pisinger, D., 2011. Train shunting at a workshop area. Flexible Services and Manufacturing Journal 23 (2), 156–180.

Kroon, L., Huisman, D., 2011. Algorithmic support for railway disruption management. In: Nunen, J. A., Huijbregts, P., Rietveld, P. (Eds.), Transitions Towards Sustainable Mobility. Springer Berlin Heidelberg, pp. 193–210.

Kroon, L. G., Lentink, R. M., Schrijver, A., 2008. Shunting of passenger train units: An integrated approach. Transportation Science 42 (4), 436–449.

Lentink, R. M., Fioole, P.-J., Kroon, L. G., van't Woudt, C., 2006. Applying Operations Research Techniques to Planning of Train Shunting. John Wiley & Sons, Inc., pp. 415–436.

Nielsen, L. K., Kroon, L., Maróti, G., 2012. A rolling horizon approach for disruption management of railway rolling stock. European Journal of Operational Research 220 (2), 496–509.

Padberg, M., , Rinaldi, G., Mar. 1987. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. Operations Research Letters 6 (1), 1–7.

Ryan, D. M., Foster, B. A., 1981. An integer programming approach to scheduling. In: Wren, A. (Ed.), Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling. North-Holland, pp. 269–280.

Schrijver, A., 1993. Minimum circulation of railway stock. CWI Quarterly 6, 205–217.

van Wezel, W., Riezebos, J., 2011. Case study: Advanced decision support for train shunting scheduling. In: Fransoo, J. C., Waefler, T., Wilson, J. R. (Eds.), Behavioral Operations in Planning and Scheduling. Springer Berlin Heidelberg, pp. 413–430.

Weide, O., Ryan, D., Ehrgott, M., 2010. An iterative approach to robust and integrated aircraft routing and crew scheduling. Computers & Operations Research 37 (5), 833 – 844, disruption Management.

Winter, T., Zimmermann, U. T., 2000. Real-time dispatch of trams in storage yards. Annals of Operations Research 96, 287–315.