Integrated Inbound Train Split and Load Planning in an Intermodal Railway Terminal

Bruno P. Bruck[a,*], Jean-François Cordeau[b], Emma Frejinger[c]

[a] *Centro de Informática, Universidade Federal da Paraíba, CEP 58059-900, João Pessoa, Brazil.*
[b] *Department of Logistics and Operations Management, HEC Montréal, Canada.*
[c] *Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada.*

Abstract

In the last decades the intermodal transportation of containers has become a key component of the entire international trade system, as it allows safe and efficient intercontinental door-to-door movement of freight by combining land and sea transportation services. Intermodal railway terminals are special components of these systems that allow container traffic to be consolidated from different sources and to be transported by train over long distances. This paper addresses a practical problem faced by a major North American railway. On a daily basis, terminal operators must make several decisions such as how inbound trains are split into sequences of railcars, on which tracks these railcars are parked for loading and off-loading operations, and how the railcars are assigned to outbound blocks so as to fulfill the demand of each block. We introduce an approach that incorporates all these decisions to obtain improved solutions. In a first step, a mixed integer linear programming (MILP) model is used to create a set of patterns that specify how each inbound train is split and how the railcars are assigned to the blocks. Then, in a second step, another MILP formulation decides which patterns to use and where to park the railcars. Tests on generated benchmark instances based on realistic data from the railway indicate that our approach is able to generate good quality solutions for all instances and can be used by the company to assist decision makers in generating less costly and more efficient plans.

Keywords: railway optimization, intermodal railway terminals, freight transportation.

*Corresponding author
Email addresses: bruno.bruck@ci.ufpb.br (Bruno P. Bruck), jean-francois.cordeau@hec.ca (Jean-François Cordeau), emma.frejinger@cirrelt.ca (Emma Frejinger)

1. Introduction

Intermodal freight transportation may be considered as one of the cornerstones of globalization, as it allows for the efficient intercontinental door-to-door movement of goods through multiple modes of land and sea transportation services that often involve several different carriers. In a classical example of an intermodal chain, loaded containers leave the initial shipper location by truck and are directed either to a port or to an intermodal railway terminal, from where a train will transport them to a port. A ship then moves the containers to another port, from where they are transported to the destination by one or a combination of several means of transportation (Crainic and Kim 2007).

These intermodal terminals are special transshipment nodes that are responsible for consolidating traffic and dispatching containers on trains destined to other nodes of the network, so that these containers can eventually reach their final destination. For a comprehensive survey of the literature on intermodal transportation we refer the reader to the surveys of Crainic and Kim (2007) and of SteadieSeifi et al. (2014).

Most of the intermodal traffic is containerized, which ensures a safer, cheaper and more reliable means of transportation without handling the cargo. Indeed, the North American market of intermodal transportation has performed remarkably well in the last decade or so, with annual growth rates of about 15% (SteadieSeifi et al. 2014). While the international market mainly follows the ISO standard and uses 20-, 40- and 45-ft containers, in North America there are also 48- and 53-ft containers, which are used for domestic traffic. Another feature of this market is that trains are usually double stacked and there are many types of railcars with different characteristics, e.g., number of platforms, platform length (40, 45, 48 or 53 feet long) and weight loading limit. This great variety of containers and railcar types has a significant impact on the difficulty of the *load planning*, which concerns the assignment of containers to railcar slots (see, e.g., Mantovani et al. 2018). Performing a proper matching between railcars and containers in the load planning is important to ensure the best usage of the available capacity of railcars but also the fuel efficiency of the train, because loading influences aerodynamic aspects.

In practice, the set of railcars that is made available to load containers is the result of decisions that derive from a block plan. The *block planning* is a tactical problem which is critical for the design of an efficient and profitable rail transportation system (Bodin et al. 1980). A *block* is defined as a group of railcars, with possibly different origins and destinations, that are moved as a single unit between terminals. The railcars of the same block do not need to be handled individually at intermediate terminals, which reduces handling costs.

In this paper we focus on an operational problem faced by a North American railway in the context of rail transportation of intermodal containers. In particular, we consider (1) how inbound trains are split (cut) into sequences of railcars after entering the intermodal terminal, (2) on which tracks those railcars are parked for loading and off-loading operations or even for temporary storage, and (3) the assignment of railcars to outbound blocks so as to fulfill the demand of each block. Figure 1 illustrates how these decisions are related and how inbound trains are processed and outbound trains assembled. Because we focus on operational decisions, we assume that the block plan is given. However, the choice of the individual cars that compose each block is optimized based on the available resources.

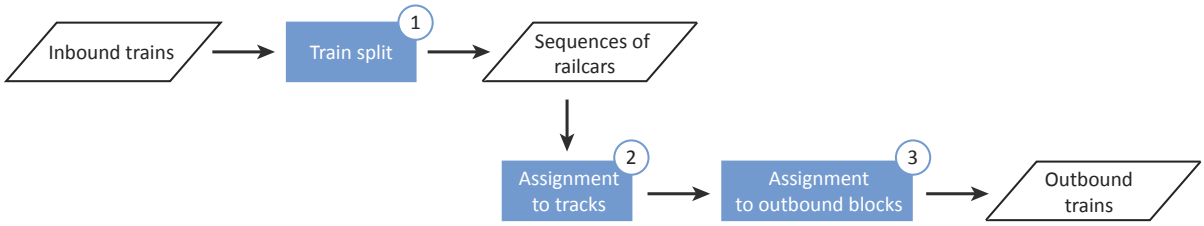


Figure 1: Diagram of the main decisions associated with our problem.

Our main contributions are to introduce a novel practical problem in intermodal transportation, and to propose an approach that incorporates all the aforementioned decisions to obtain better solutions than when a sequential decision process is used. This approach proceeds in two steps. The first one creates a set of patterns (called configurations) that specify how each inbound train is split and how railcars are assigned to the blocks. Therefore, this step deals with decisions (1) and (2) of Figure 1, which are heavily interconnected. Decision (3) is then handled in a second step, where we decide which configurations to use and where to park the railcars, making sure that it is possible to bring the railcars to their assigned track and to pull them out when they have to depart. To perform the first step we introduce two *mixed integer linear programming* (MILP) formulations that differ in the way we represent groups of consecutive railcars that are destined to the same block. The second step is solved by means of another MILP formulation which ensures that the railcars can be parked onto the tracks and that the outbound trains can be properly assembled. In order to test our algorithms we generated a set of realistic instances based on real data provided by the railway. Our experiments show that, although the problem is very complex, our solution approach is able to generate good quality solutions for all benchmark instances. These results also indicate that aspects such as the composition of the inbound trains and

the amount of traffic in the terminal play a larger role than the variations in the demand of blocks, in determining how difficult it is to find feasible solutions for a given instance.

The remainder of the paper is organized as follows. Section 2 presents a brief discussion of the related literature. Section 3 describes the problem and the notation used throughout the paper. Section 4 presents the solution approach, whereas Section 5 reports the results of the computational experiments. Finally, Section 6 provides some conclusions.

2. Related literature

Intermodal transportation is a broad and rich research topic which has evolved significantly in the last decades. In the literature, the associated problems are generally classified according to their planning horizon as strategic, tactical or operational problems (see Crainic and Kim 2007). The strategic level concerns long term decisions usually associated with investments in infrastructure, whereas at the tactical level one is concerned with the optimal utilization of a given infrastructure, for example, by designing train schedules and routes. Finally, the operational level deals with short term decisions such as real-time planning for orders and adjustments to schedules, or even daily operations at terminals.

Several studies in the literature concentrate on classification and shunting yards (e.g., Blasum et al. 1999, Dahlhaus et al. 2000 Boysen et al. 2012, Boysen, Emde, and Fliedner 2016, Adlbrecht et al. 2015, Shi and Zhou 2015). However, they should not be confused with intermodal terminals, as only the latter allow temporary storage, loading and off-loading of containers. Nevertheless, understanding the basic functioning of shunting yards is relevant, as some of their operations are somewhat similar to those performed in intermodal terminals. According to Boysen et al. (2012), shunting yards play an important role in rail freight transportation. They are used to disassemble inbound trains and to sort their railcars, which are then dispatched on outbound trains to other nodes of the network. Usually, these terminals are composed by three areas. The first one contains the so-called receiving tracks, where inbound trains first arrive and are inspected. Then, railcars are moved to the classification area either by a locomotive (flat yards), by being pushed to a hump (hump yards), or simply by gravity (gravity yards). The order in which railcars are moved to this second area, and the assignment to the classification tracks, define one of the core operational problems found in shunting yards, which is the sorting and rearranging of inbound railcars. Finally, once the railcars have been sorted, they are moved to a third area containing the departure tracks, where the outbound trains are assembled.

Boysen, Emde, and Fliedner (2016) classify the short term operational decisions of shunting yards according to the following hierarchy:

1. *Cut generation*: inbound trains arriving at the shunting yard must be assigned to the receiving tracks. As mentioned by Boysen et al. (2012), under certain conditions, inbound trains might be split into smaller sequences of railcars that can be parked on different tracks, especially if no receiving track can accommodate the entire train;

2. *Train makeup*: refers to the assignment of inbound railcars to outbound trains;

3. *Railcar classification*: decides the order in which inbound railcars parked on the receiving tracks are brought to the classification area and which locomotive pushes the railcars;

4. *Outbound track assignment*: decides on the departure track in which each outbound train is assembled.

Although our problem cannot be directly classified according to this hierarchy, some of the aspects we optimize are somewhat similar to those of shunting yards. In particular, the *cut generation* and *train makeup* problems in shunting yards are similar to the decisions on the inbound train split and assignment to outbound blocks found in our problem. However, the similarities are mostly with respect to the main idea of each aspect and, in practice, the operations differ significantly. For instance, because of their design, the operations in shunting yards usually follow a single direction. Inbound trains arrive from one side of the yard and are placed on the receiving tracks. Then, railcars are moved to the classification tracks and later to the departure tracks, from where they depart on outbound trains. Given that all operations in intermodal terminals are usually performed on the same set of tracks, it is possible that inbound trains arrive at the terminal from different directions. Similarly, outbound trains might depart from either side of the terminal. This adds another layer of complexity to the problem, as railcars that arrive later might block the departure of other cars that were already parked on the same track. Furthermore, in intermodal terminals there is significantly less flexibility in how railcars can be rearranged, given that sorting is usually very expensive in this environment.

Another related problem is the load planning, where given a set of containers stored in a terminal and a sequence of railcars one must determine the optimal assignment of containers to railcar slots while minimizing the costs (Mantovani et al. 2018). These costs are usually associated with the acquisition, renting and maintenance of locomotives, railcars and cranes, and the purchasing of fuel and employing of crews (see, e.g., Bouzaiene-Ayari, Cheng, and Das 2016).

Most of the literature on load planning focuses on single-stack trains and on different levels of detail. For instance, Corry and Kozan (2008) consider matching different types of containers and railcars, but do not take into account the weight of containers as in Bruns and Knust (2012). Others go even further and incorporate a number of practical constraints (see, e.g., Heggen, Braekers, and Caris 2016) or consider a robust optimization approach that is able to deal with several sources of uncertainty (Bruns et al. 2014). Usually, in the North American market there are also double-stack trains, in which case there are even more possible loading patterns and additional concerns, such as the center of gravity of the railcars (see, e.g., Mantovani et al. 2018) and aerodynamic drag (see, e.g., Lai, Barkan, and Önal 2008).

In our problem setting we assume that there can be double-stack railcars. However, given that the loading problem is itself already complex, we simplify it and consider only the aggregate matching of containers to railcars. In particular, we focus on the availability of slots for different container sizes in each railcar according to their loading capabilities described in the AAR Guide (Association of American Railroads 2014), which provides a complete description of the loading patterns for railcars. This aspect is discussed in detail in Section 3.

In the context of rail-rail transshipment yards, similar problems arise in terminals that have gantry cranes. According to Boysen, Jaehn, and Pesch (2011), these terminals face problems such as how to (i) schedule the sequence of trains that enter the tracks (e.g. Boysen, Jaehn, and Pesch 2012); (ii) decide on the assignment of containers to railcars (e.g. Corry and Kozan 2006); (iii) determine the fixed areas in which each crane is allowed to operate (e.g. Boysen and Fliedner 2010, Boysen, Fliedner, and Kellner 2010); (iv) decide on the train parking positions (e.g., Kellner, Boysen, and Fliedner 2012); and (v) decide on the sequence of movements for each crane in order to place the container onto the railcars (e.g., Alicke 2005, Zhang et al. 2002). While problems (ii) and (iv) seem closely related to some of the decisions in our setting, the presence of the gantry cranes significantly changes the underlying problem. For instance, in rail-rail transshipment terminals, trains usually enter the tracks in pulses (subset of trains that are processed in parallel) from a single direction and train splits are not allowed. Furthermore, the gantry cranes allow containers to be loaded onto railcars that are parked in a track that would be inaccessible to cranes that move along the terminal.

To the best of our knowledge, this is the first study to address an operational problem in the context of intermodal terminals that incorporates aspects of the load planning, cut generation and train makeup.

3. Problem description

Considering that the problem is complex and composed of multiple interconnected components, we divide its description into three subsections. The first one introduces some general notation and deals with the details associated with how inbound trains are split into sequences of railcars. Then, the second part focuses on aspects related to how railcars are parked in the terminal. Finally, the third subsection refers to the load plan and the assembly of the outbound trains.

3.1. Inbound train split

We are given an ordered set T^- of inbound trains and a sequence R_t of railcars arriving on each inbound train $t \in T^-$. Railcars that are destined to the terminal are said to be part of the *local* traffic. These railcars are represented by the sequence R'_t for each inbound train $t \in T^-$ and might be off-loaded and loaded again at the terminal. The remaining railcars are said to be part of the *pass-by* traffic and are destined to other terminals. Pass-by railcars cannot be off-loaded at the terminal and their assignment to outbound blocks is given a priori.

Each inbound train can be split into a number of *segments*. A segment is composed by a sequence of railcars that occupy consecutive positions in a certain inbound train. Therefore, a segment can be defined by the first and last positions occupied by the railcars forming this segment. For example, a train comprising 100 railcars could be split into three segments: one from positions 1 to 35, one from 36 to 85 and another one from 86 to 100. The set of all potential segments associated with each inbound train $t \in T^-$ is denoted by S_t.

Each segment can be further divided into a set of *sections* (or sub-segments) that will ultimately be assigned to different outbound blocks. For example, a segment with 35 railcars could be divided into a first section with 20 cars and a second one with 15 cars. The two sections will remain together until the departure of the outbound trains to which these sections are assigned. Figure 2 illustrates an example of a segment comprising two sections, each containing two railcars. The set of potential sections associated with a given segment $s \in S_t$, where $t \in T^-$, is denoted by E_s. Note that, for a given inbound train, the specification of which segments to create also determines how the train is physically split. However, the specification of sections might be seen as virtual cuts, which will be performed only when the sections are pulled out from the tracks to assemble the outbound trains.

3.2. Assignment to tracks

We assume that the given set of tracks P can be partitioned into a set of *storage* and *working* tracks. While both types can be used to move railcars through the terminal and

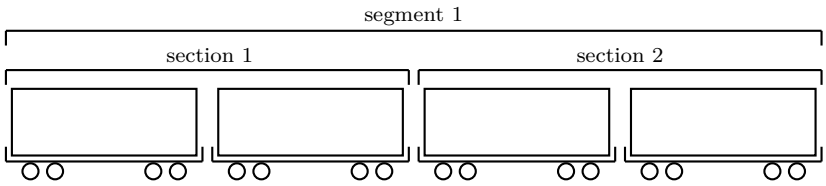


Figure 2: Example of a segment containing two sections.

for temporary storage, containers can only be loaded on and off-loaded from railcars parked on working tracks. To model the position of the railcars on these tracks, we discretize each track $p \in P$ into a sequence Q_p of *track positions* that represent portions of the track. For example, a 1000m track could be divided into 20 parts of 50m, thus a segment with a length of 500m assigned to position 1 would occupy the first 10 positions of this track. The sets of storage and working tracks can be further partitioned into a subset of single-ended tracks (where railcars enter and leave from just one end) and double-ended tracks (where railcars can enter and leave from both ends).

Figure 3 depicts an intermodal terminal with five double-ended tracks. As shown in this example, intermodal terminals are usually connected to the railway network through a main track, from which inbound trains arrive and outbound trains depart. This track may be used as a storage track for pass-by traffic (i.e., blocks that are not destined to this terminal) that remains in the terminal for a short period of time. We assume that the side of the main track from which the inbound trains arrive and the outbound trains depart is known beforehand and cannot be altered. This is an important aspect of the problem and determines from which end of the tracks the railcars of a certain train can enter or exit. For example, in Figure 3, an inbound train arriving from the left side of the main track could first go through the entire track, and then enter each track (tail first) from the right in order to park the railcars on their assigned tracks. If the locomotives are at the front end of the train, this strategy ensures that the locomotives are never in the way of the railcars being parked. A similar strategy is applied for inbound trains arriving from the right side. Furthermore, there can also be *crossovers* connecting different tracks. These crossovers might be used to move railcars from one track to the other and to reach portions of the connected track that would otherwise be unreachable due to the presence of other railcars parked between one end and the crossover. In this study, we assume that crossovers can only be used to pull out sections in order to assemble an outbound train.

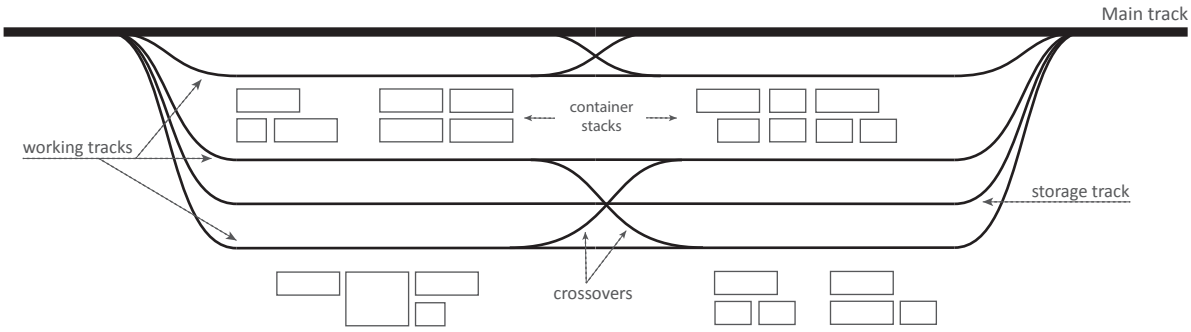


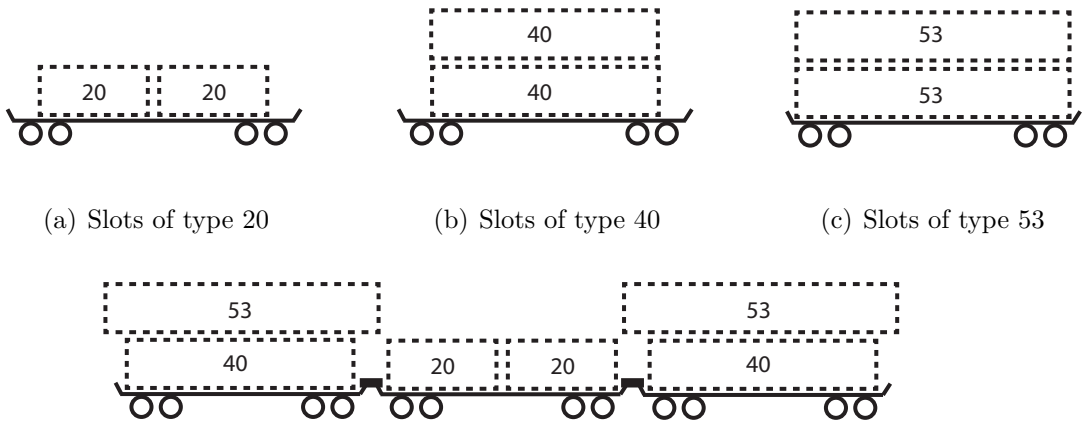
Figure 3: Example of an intermodal terminal with five tracks.

3.3. Load planning and makeup of outbound trains

Regarding the load planning, we assume that railcars have different loading capabilities depending on their length and their number of platforms, and whether or not they allow double stacking and heavy duty containers (i.e., those with weight greater than 52,900 pounds). In this study, we consider four types of containers, given by the set $F = \{20, 20h, 40, 53\}$. The types 20 and $20h$ both comprise 20-ft containers and are differentiated by the weight of the containers. While type 20 contain only containers with weight less than or equal to 52,900 pounds, type $20h$ is composed exclusively by heavy duty containers. Types 40 and 53 contain, respectively, 40-ft and 53-ft containers. Then, the loading capabilities of each railcar are specified by the number of available *slots* for each type of containers. Note that the number of slots of each type depends on the physical characteristics of the railcars. For instance, 20-ft containers might be loaded on any railcar, but only on the bottom of the platforms, as they cannot be securely stacked on top of other containers. Therefore, every railcar has two 20-ft slots for each platform. In case the railcar accepts heavy cargo it will also have a capacity of two slots for containers of type $20h$. For the other two types of slots, the availability depends on the stacking capabilities of the railcars, number of platforms and platform length.

It is important to notice that, for any given railcar, considering only the number of slots per type of container may lead to an overestimation of its loading capabilities. Figures 4-(a)(b)(c) illustrates an example of the loading capabilities, in terms of slots for containers of each type, of a railcar with a single 53-ft platform that can be double stacked. For simplicity, consider that this railcar cannot be loaded with containers of type $20h$. Then, this railcar has two slots of type 20, two of type 40 and also two of type 53. Although it is possible to load different types of containers in the same platform (e.g., two 20-ft and one

40-ft), some combinations are clearly infeasible (e.g., two 40-ft and two 53-ft containers), even though there are enough slots of each type. Because of this issue, we must also consider an aggregate number of slots per railcar, specifying an upper bound on the total number of containers that can be loaded. We refer to this bound as the number of *aggregate slots*, and evaluate it according to a simple rule: each platform contributes either one or two aggregate slots depending on whether it can be double stacked. Containers of type 40 and 53 require one aggregate slot, whereas containers of types 20 and $20h$ require only half of an aggregate slot, as two 20-ft containers can be loaded together in the bottom of a platform. The scenario becomes even more complex when multi-platform railcars are considered, as there are specific rules that only apply to such cars. For instance, a 40-ft railcar with 3 platforms may be loaded with 53-ft containers on the top slots as long as no two sequential platforms hold such containers. This can be clearly seen in the example illustrated in Figure 4-(d), where we are not able to place a 53-ft container on the top slot of the second platform.



(a) Slots of type 20 (b) Slots of type 40 (c) Slots of type 53

(d) A multi-platform 40-ft railcar with slots of type 20, 40 and 53

Figure 4: Example of the number of slots of each type for a railcar with a single 53-ft platform with double stacking capabilities (a)(b)(c) and a multi-platform 40-ft railcar with more complex loading capabilities (d).

The set of containers that must be loaded and dispatched on each outbound block is known beforehand and so are their specifications, such as dimensions and weight. Differently from Mantovani et al. (2018), in this study, we only incorporate a few key aspects of the associated load planning problem to ensure that minimum requirements are met. In particular, we consider that each block $b \in B$, where B is the set of outbound blocks, has a certain demand d_{bf} of slots for containers of type $f \in F$ and a demand d'_b of aggregate slots. Note that these demand values might consider not only the set of containers that must be loaded, but can also incorporate the requirements of other terminals for certain types of

railcars.

Furthermore, to speed up loading operations it is often the case that containers that are destined to the same outbound block are stacked in the same area of the terminal or in close vicinity of each other. As shown in Figure 3, the container stacks are usually concentrated in certain areas of the terminal and close to the working tracks. We assume that the location of the container stacks associated with each outbound block is known and so are the distances from the stacks to each portion $q \in Q_p$ of a track $p \in P$.

The ordered set of outbound trains is denoted by T^+, and each train $t \in T^+$ is assembled by combining sections assigned to one of the outbound blocks associated with train t, represented by the set $B_t \in B$. These sections may be parked on different tracks and may come from different segments. The expected length of each outbound train $t \in T^+$ is given by l_t and cannot be exceeded by more than $100\rho\%$, where ρ is a parameter, due to restrictions associated with the power of the locomotives. Note that ρ is used to allow a certain margin of flexibility in the length of the trains, which is important given that there are many types of railcars with different lengths and imposing a fixed value would be too restrictive in practice.

The order of the outbound blocks in each outbound train is defined by the *marshalling*, which is a fixed plan that can only be modified in exceptional situations. For certain trains, changes might be needed to achieve a better slot utilization or even to ensure that it is possible to generate a solution in cases where there is high traffic at the terminal and it would otherwise be impossible to generate a solution that fully respects the marshalling. In these cases, the order of the blocks could be readjusted by another terminal downstream.

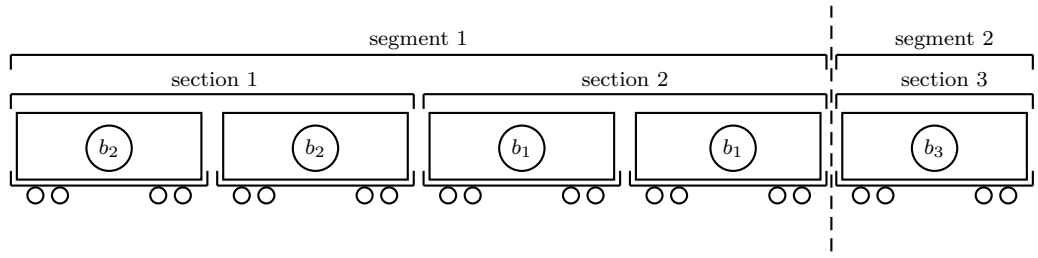
Furthermore, it is important to consider that terminals located in ports usually do not need 53-ft railcars because of the ISO standards for the containers sizes used in the international market. For this reason, we denote by $B' \subseteq B$ the subset of outbound blocks that are destined to terminals located in ports.

The problem then consists in deciding on (1) how each inbound train is split, i.e., which segments and sections are created, (2) on which track and track position each segment is parked, and (3) which sections of each segment are assigned to each outbound block. A feasible solution must meet the demand requirements for each outbound block, each segment must be able to reach its assigned track and track position, and there should exist a sequence of movements to pull each section out of the tracks when it is time to depart. In addition, the solution must ensure that containers can be reached from the location where the railcars are parked, the length of the outbound trains cannot exceed a certain threshold (associated with the power of its locomotives) and the order of the blocks in each outbound train should

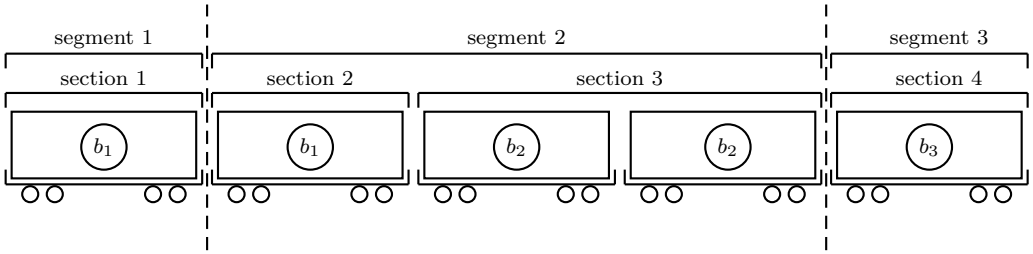
not violate the marshalling plan.

4. Solution method

Although it is theoretically possible to formulate an integrated model that incorporates all the aforementioned decisions and constraints, such a model would likely be intractable for instances of practical size. One way to potentially reduce the difficulty of the problem is to divide it in two parts that can be solved sequentially. Our approach follows this idea and proceeds in two steps. The first step is the generation of a set of potential configurations for each inbound train. A configuration specifies how a given inbound train is split into segments, which sections compose each segment and the assignment of sections to the outbound blocks. Thus, the configurations correspond to decisions (1) and (3). Figure 5 illustrates two possible configurations for an inbound train containing five railcars, each containing a single platform. In this example, the sequence of outbound blocks is $B = \{b_1, b_2, b_3\}$ and the vertical dashed lines indicate where the inbound trains are split. Note that, as shown in Figure 5-(b), it is possible to split even between railcars assigned to the same block. This can be beneficial given that it creates multiple segments that can be parked on different tracks, especially when the sequence of railcars is very long.



(a) Configuration 1



(b) Configuration 2

Figure 5: Example of two configurations for an inbound train with five railcars

The second step takes as input a list of possible configurations for each train and decides which configurations to use and where to place each segment in the terminal. Note that if all

possible configurations are generated in the first step, the algorithm is exact and produces an optimal solution to the full problem. In the following two sections, the solution method for each step is described in detail.

4.1. Step 1: Generation of configurations

The generation of configurations is perhaps the most critical and complex aspect of the problem. Given the combinatorial nature of this step, in practice, it is usually impractical to generate and consider all possible configurations for each inbound train. The objective then, is to generate a small set of configurations that is nevertheless rich enough to ensure that good feasible solutions can be found in the second step.

As mentioned in Section 3, railcars arriving on inbound trains are either part of the local or pass-by traffic. We assume that the segments and sections created from pass-by railcars are given a priori and so are the assignments of these sections to outbound blocks and the location where each pass-by segment is parked in the terminal. Therefore, the procedure for generating configurations initially considers only railcars from local traffic and later appends the pass-by segments and sections to the configurations generated.

There are also feasibility constraints that must be considered when generating configurations. The first constraint imposes a limit c_t on the number of physical cuts allowed for each inbound train $t \in T^-$ that are not associated with pass-by traffic. In other words, this constraint specifies the number of times an operator has to manually split the train by lifting a pin between the railcars of two consecutive segments. This constraint helps reducing the complexity of the operations and the time spent processing inbound trains. In addition, in certain terminals, the train split is done by the train crew, in which case this constraint might be needed in order to comply with rules imposed by collective agreements.

Another important aspect concerns the order of sections within the same segment and their assignment to outbound blocks. Consider a segment with three sections assigned to outbound blocks b_1, b_2 and b_3, respectively. Assume that blocks b_1 and b_2 are destined to an outbound train that departs at time 1, whereas block b_3 is destined to an outbound train that departs at time 2. In addition, suppose that sections assigned to blocks b_1 and b_2 must be pulled out of a track from its right side because of the direction from which the outbound train departs. Note that at time 1 sections e_1 and e_2 would have to be pulled out from the right of the track where they are parked. However, because section e_3 only departs later, this move would not be possible. In this example, a similar issue would occur if, according to the marshaling plan, railcars assigned to block b_1 cannot appear on the left of railcars assigned to block b_2 in the associated outbound train. Note that both issues could

be resolved by rearranging the railcars locally at the terminal. However, these operations are often forbidden or simply too expensive and time consuming. Therefore, we assume that configurations that contain segments with any of these issues are considered infeasible. To model this aspect of the problem, let $\phi(b_1, b_2)$ be a function that takes value 1 if and only if block $b_1 \in B$ cannot appear on the left of block $b_2 \in B$ in the same segment because of at least one of the aforementioned issues.

During preliminary computational experiments, we observed that some structural aspects of the configurations seem to play a large role in determining whether or not a configuration is likely to lead to a feasible solution in the second step. The most important aspects identified were the length of each section, the number of sections assigned to each outbound block, and the order of the assignments to outbound blocks. Including all these aspects in the same formulation has proved to be impractical due to the complexity and number of variables in the resulting formulation. Thus, in order to generate feasible configurations, we propose two MILP formulations, each focusing on different structural aspects of the configurations.

Note that each formulation only decides on a set of sections to be created for each inbound train and their assignment to outbound blocks. However, they do not specify which segments are actually created and only produce a list of which sections should be created for each inbound train. This not only reduces the number of variables in the formulations but also removes symmetry from the problem. A post-processing procedure then takes as input a solution of one of the formulations and, for each inbound train, generates a set of configurations by enumerating all the possible ways to create segments based on the list of sections of the respective train. For example, recall the configuration shown in Figure 5-(a). Given the same sequence of sections, a second valid configuration could be generated by assigning only section 1 to the first segment and sections 2 and 3 to the second one. Similarly, all three sections could be assigned to a single segment, generating another configuration.

In the following sections we present each of the proposed formulations and the post-processing procedure used to convert the solutions into sets of feasible configurations.

4.1.1. Formulation CG1.

Recall that each block $b \in B$ has a certain demand d_{bf} of slots for containers of type $f \in F$ and a demand d'_b of aggregate slots, and that R'_t specifies the sequence of local traffic railcars from inbound train $t \in T^-$. Given a sequence of railcars $[i, j] \in R'_t$, where $t \in T^-$ and $j \geq i$, we denote by β^f_{tij} and β'_{tij}, respectively, the total number of slots of type $f \in F$ and the total number of aggregate slots provided by the railcars in the sequence. In

addition, let λ^t_{ijb} denote the total number of hours that railcars in the sequence $[i,j]$ remain in the terminal if assigned to outbound block $b \in B$, μ^t_{ij} the number of 53-ft platforms in the sequence, and l^t_{ij} the total length of railcars from i to j. Then, let x^t_{ijb} be a binary variable that specifies whether the railcars in the interval $[i,j] \subseteq R'_t$, such that $j \geq i$ and $t \in T^-$, compose a section assigned to block $b \in B$. In other words, these variables determine which sections are created for each inbound train and their assignment to outbound blocks.

In addition, let us define a binary variable u_{tj} for each inbound train $t \in T^-$ and railcar $j \in [2, |R'_t|]$, that takes value 1 if two consecutive sections $e_1 = [i, j-1]$ and $e_2 = [j,k]$, where $i < j$ and $k \geq j$, are created and assigned to blocks b_1 and $b_2 \in B$ such that $\phi(b_1, b_2) = 1$. Intuitively, these variables count the number of potential conflicts that could prevent the generation of feasible configurations and are used by the model to impose a limit on the number of such conflicts.

Finally, let v_{tj} be a binary variable that takes value 1 if railcars $j-1 \in R'_t$ and $j \in R'_t$, from an inbound train $t \in T^-$, belong to sections $e_1 = [i, j-1]$ and $e_2 = [j,k]$, with $i < j$ and $k \geq j$, assigned to blocks b_1 and b_2 such that $T^+(b_1) \neq T^+(b_2)$, where $T^+(b)$ specifies the outbound train associated with block $b \in B$. These variables are based on the observation that, in practice, consecutive sections created from the same inbound train are often assigned to blocks that depart on the same outbound train. This usually reduces the complexity and time required by the operations that must be done to assemble the outbound trains. We are now ready to introduce the following MILP formulation:

$$\text{(CG1)} \qquad \min \; z_{CG1} = \sum_{t \in T^-} \sum_{\substack{i,j \in R'_t: \\ j \geq i}} \left(\sum_{b \in B} w_1 \; \lambda^t_{ijb} + \sum_{b \in B'} w_2 \; \mu^t_{ij} + \sum_{b \in B} w_3 \; \frac{|R'_t|}{j-i+1} \right) \; x^t_{ijb} + w_4 \sum_{t \in T^-} \sum_{i \in R'_t} v_{ti} \qquad (1)$$

subject to

$$\sum_{\substack{i,j \in R'_t: \\ r \in [i,j], j \geq i}} \sum_{b \in B} x^t_{ijb} = 1 \qquad \forall \; t \in T^-, r \in R'_t \qquad (2)$$

$$\sum_{t \in T^-} \sum_{i,j \in R'_t : j \geq i} \beta^f_{tij} \; x^t_{ijb} \geq d_{bf} \qquad \forall \; b \in B, f \in F \qquad (3)$$

$$\sum_{t \in T^-} \sum_{i,j \in R'_t : j \geq i} \beta'_{tij} \; x^t_{ijb} \geq d'_b \qquad \forall \; b \in B \qquad (4)$$

$$\sum_{t \in T^-} \sum_{\substack{i,j \in R'_t: \\ j \geq i}} \sum_{b \in B_k} l^t_{ij} \; x^t_{ijb} \leq (1+\rho) \; l_k \qquad \forall \; k \in T^+ \qquad (5)$$

$$u_{tj} \geq \sum_{i=1}^{j-1} x^t_{ijb_1} + \sum_{k=j}^{|R'_t|} x^t_{jkb_2} - 1 \qquad \forall\, t \in T^-, j \in [2, |R'_t|], \{b_1, b_2\} \in B, \phi(b_1, b_2) = 1 \tag{6}$$

$$v_{tj} \geq \sum_{i=1}^{j-1} x^t_{ijb_1} + \sum_{k=j}^{|R'_t|} x^t_{jkb_2} - 1 \qquad \forall\, t \in T^-, j \in [2, |R'_t|], \{b_1, b_2\} \in B, T^+(b_1) \neq T^+(b_2) \tag{7}$$

$$\sum_{i=2}^{|R'_t|} u_{ti} \leq c_t \qquad \forall\, t \in T^- \tag{8}$$

$$x^t_{ijb} \in \{0,1\} \qquad \forall\, t \in T^-, i, j \in R'_t, b \in B, j \geq i \tag{9}$$

$$u_{ti} \in \{0,1\} \qquad \forall\, t \in T^-, i \in R'_t \tag{10}$$

$$v_{ti} \in \{0,1\} \qquad \forall\, t \in T^-, i \in R'_t. \tag{11}$$

The objective function (1) minimizes the sum of four types of penalties, which are, respectively: (i) the total waiting time of railcars in the terminal, (ii) the number of 53-ft platforms sent to port terminals, (iii) a penalty associated with the number of railcars per section, which reduces with the number of railcars; and (iv) the number of times two consecutive sections are assigned to outbound blocks destined to different trains. The first penalty is especially important in settings like ours that consider a rather congested terminal, where it is crucial that railcars are dispatched from the terminal as soon as possible. The values w_1, w_2, w_3 and w_4 are the weights associated with each type of penalty and were tuned based on the feedback from the decision makers of the company. Then, constraints (2) specify that each railcar is assigned to exactly one section, whereas constraints (3) and (4) ensure that the demand of each outbound block is met. Constraints (5) specify an upper bound on the total length of each outbound train, such that it does not exceed a certain percentage of its expected length. The sets of constraints (6) and (7) are responsible for setting the values of the u and v variables, respectively. Note that each u_{ti} variable that takes value 1, where $t \in T^-$ and $i \in R'_t$, specifies that a physical cut is mandatory just before railcar i to ensure that the sections containing railcars i and $i-1 \in R'_t$ are part of different segments. Constraints (8) then impose a limit c_t to the number of cuts allowed for each inbound train $t \in T^-$.

Formulation $CG1$ can then be strengthened with the following valid inequalities:

$$u_{ti} \leq \sum_{b \in B} \sum_{k=i}^{|R'_t|} x^t_{ikb} \qquad \forall\, t \in T^-, i \in [2, |R'_t|] \tag{12}$$

$$v_{ti} \leq \sum_{b \in B} \sum_{k=i}^{|R'_t|} x^t_{ikb} \qquad \forall\, t \in T^-, i \in [2, |R'_t|]. \tag{13}$$

Constraints (12) specify that a certain variable u_{ti} can only take value 1 if railcars $i-1$ and i from set R'_t of the inbound train $t \in T^-$ are assigned to different sections. Constraints (13) apply the same reasoning to the v variables.

It is important to point out that, because the generation of configurations does not consider all the aspects of the problem, the optimal solution of formulation $CG1$ does not necessarily yield the best set of configurations for the second step. In fact, it might even yield a set of configurations that are not feasible for the second step. For this reason, including a few additional inequalities that possibly remove feasible solutions might be beneficial to reduce the search space and find solutions faster. This is especially interesting in large scenarios, where it can be quite difficult to find a single feasible solution in a reasonable amount of time. Let l_b denote the approximated expected length of outbound block $b \in B$. We thus propose the following two additional sets of constraints, which are based on practical observations of the problem:

$$\sum_{i,j \in R'_t : j \geq i} x^t_{ijb} \leq 1 \qquad \forall\ t \in T^-, b \in B \tag{14}$$

$$x^t_{ijb} = 0 \qquad \forall\ t \in T^-; i,j \in R'_t, b \in B, l^t_{ij} < \xi\ l_b. \tag{15}$$

Constraints (14) specify that for each inbound train there can be at most one section assigned to each outbound block. These constraints are based on the observation that having multiple sections from an inbound train assigned to the same block might require additional effort not only when moving the railcars on the tracks at their arrival, but also when pulling the sections out of the tracks to build an outbound train. In addition, these constraints remove some symmetry from the problem. Constraints (15) are used to impose a limit on the minimal length of sections relative to the expected length of the outbound block to which they are assigned. Therefore, an assignment of a sequence of railcars $[i,j]$, where $i,j \in R'_t$, $t \in T^-$ and $j \geq i$, to an outbound block $b \in B$ would be forbidden in case l_{ij} is less than $\xi\ l_b$. The main motivation for these constraints is to reduce the number of sections assigned to the same outbound block and also to remove solutions in which very small sections are created. From now on, we refer to the formulation composed by (1)-(15) as $CG1_{heur}$.

4.1.2. Formulation CG2.

One of the biggest downsides of formulation $CG1$ is the presence of variables u and v, which significantly increase the number of constraints. To mitigate this issue, in this section,

we model the generation of configurations as a network flow problem defined on a support graph.

In the support graph, each railcar $r \in R'_t$, where $t \in T^-$, is associated with a set of vertices $V_r = \{r_1, r_2, \ldots, r_{|B|}\}$, representing the assignment of railcar r to each outbound block. Let $R(i) \in R'_t$, where $t \in T^-$ and $B(i) \in B$ denote, respectively, the railcar and the outbound block associated with a vertex $i \in V_{R(i)}$. Then, if vertex i is visited, railcar $R(i)$ is assigned to block $B(i)$.

Formally, let $G = (V, A)$ be the support graph, such that $V = V' \cup \{0, n+1\}$ is the set of vertices, $V' = \bigcup_{t \in T^-, r \in R'_t} V_r$, and $\{0, n+1\}$ are two dummy vertices. The set of arcs A is generated as follows.

1. Create an arc $(0, i)$ for each vertex $i \in V_{r_1}$, such that $r_1 \in R'_t$ is the first railcar of train $t \in T^-$ and t is the first inbound train;
2. Create an arc (i, j) for each $i \in V_r$ and $j \in V_{r+1}$, such that $r \in [1, |R'_t| - 1]$ and $t \in T^-$;
3. Create an arc (i, j), where $i \in V_{|R_t|}$ is a vertex associated with the last railcar of train $t \in [1, |T^-| - 1]$, and $j \in V_r$, such that $r \in R'_{t+1}$ is the first railcar of train $t + 1$;
4. Create an arc $(i, n+1)$ for each vertex $i \in V_{|R_t|}$, such that $t \in T^-$ and t is the last inbound train.

As an illustrative example, consider Figure 6, which depicts a support graph created for a scenario with two inbound trains t_1 and t_2, each containing two railcars, and with $B = \{b_1, b_2, b_3\}$.

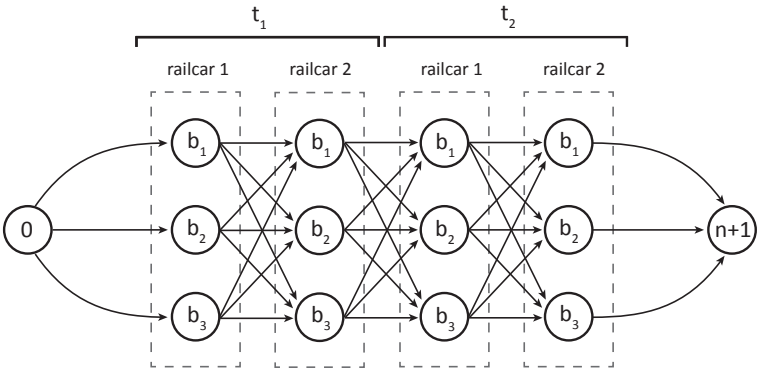


Figure 6: Example of the support graph created in a scenario with two inbound trains t_1 and t_2, each containing two railcars, and with $B = \{b_1, b_2, b_3\}$.

For a given vertex $i \in V'$ and its associated railcar $R(i)$, let us define λ_i as the time that the railcar remains in the terminal if assigned to block $B(i)$, μ_i the number of 53-ft

platforms in the railcar and l_i the length of the railcar. In addition, let γ_i^f and γ_i' specify, respectively, the number of slots for containers of type $f \in F$ and aggregated slots in the associated railcar. For any set $S \in V$, we define $\delta^+(S) = \{(i,j) \in A : i \in S, j \in V \setminus S\}$ and $\delta^-(S) = \{(i,j) \in A : i \in V \setminus S, j \in S\}$. Then, by setting g_{ij} as a binary variable that specifies whether arc $(i,j) \in A$ is traversed, and h_i as a binary variable that takes value 1 if vertex $i \in V$ is visited, we can define the following MILP formulation. Notice that variables h are only introduced to simplify the writing of the formulation, but are not needed for implementation purposes.

$$(CG2) \quad \min \; z_{CG2} = w_1 \sum_{i \in V'} \lambda_i \; h_i + w_2 \sum_{i \in V' : B(i) \in B'} \mu_i \; h_i + w_4 \sum_{t \in T^-} \sum_{\substack{i \in [0, |R_t'|-1]: \\ T^+(i) \neq T^+(i+1)}} g_{i,i+1} \tag{16}$$

subject to

$$g(\delta^+(0)) = 1 \tag{17}$$

$$g(\delta^+(i)) = h_i \qquad \forall \; i \in V' \tag{18}$$

$$\sum_{i \in V_r} h_i = 1 \qquad \forall \; t \in T^-, r \in R_t' \tag{19}$$

$$g(\delta^+(i)) - g(\delta^-(i)) = 0 \qquad \forall \; i \in V' \tag{20}$$

$$\sum_{i \in V_b} \gamma_i^f \; h_i \geq d_{bf} \qquad \forall \; b \in B, f \in F \tag{21}$$

$$\sum_{i \in V_b} \gamma_i' \; h_i \geq d_b' \qquad \forall \; b \in B \tag{22}$$

$$\sum_{i \in V' : B(i) \in B_t} l_i \; h_i \leq (1+\rho) \; l_t \qquad \forall \; t \in T^+ \tag{23}$$

$$\sum_{i,j \in V' : \phi(B(i), B(j)) = 1} g_{ij} \leq c_t \qquad \forall \; t \in T^- \tag{24}$$

$$g_{ij} \in \{0,1\} \qquad \forall \; (i,j) \in A \tag{25}$$

$$h_i \in \{0,1\} \qquad \forall \; i \in V'. \tag{26}$$

The objective function (16) minimizes the total dwell time of railcars in the terminal, the number of 53-ft platforms sent to port terminals, and the number of times a pair of consecutive railcars from the same inbound train are assigned to blocks destined to different outbound trains. Note that the penalty associated with weight w_3 is not incorporated into the objective function. This penalty is actually related to the number of railcars per section. However, formulation $CG2$ only creates the segments while the actual sections are generated later in a post processing procedure, which will be described in the next section.

Constraints (17) specify the degree of vertex 0, whereas constraint (18) link the g and h variables. Constraints (19) ensure that, for each railcar $r \in R'_t$ and train $t \in T^-$, a single vertex of V_r is visited. Set (20) are flow conservation constraints, while (21) and (22) force the minimum requirements of slots for each outbound block to be met. Finally, constraints (23) specify a maximum length for the composition of each outbound train, and constraints (24) impose a limit on the number of physical cuts allowed for each inbound train.

Notice that a feasible solution of formulation $CG2$ is a route from vertex 0 to $n+1$, visiting exactly one vertex of each set V_r, where $r \in R'_t$ and $t \in T^-$. The associated list of sections and their assignments to outbound blocks can be easily retrieved by a simple inspection of the solution.

Similar to what we described in Section 4.1.1, it is possible to impose additional constraints that might remove feasible solutions, but are also likely to lead to configurations with better quality. However, given the structure of this formulation, including sets of constraints similar to (14) and (15) is not as straightforward as in formulation $CG1$ and would require additional variables. Instead, in formulation $CG2$, we focus on a different structural aspect of configurations. We observed that, in practice, the sequence of assignments of sections to outbound blocks has a considerable impact on the likelihood of producing sets of configurations that result in feasible solutions in the second step of our approach. Let $M(b)$ specify the position of outbound block $b \in B_t$ in the marshalling plan of train $t \in T^+$. We propose the following set of constraints:

$$g_{ij} = 0 \quad \forall\, i,j \in V', T^+(i) = T^+(j) : M(B(j)) - M(B(i)) > 1 \text{ or } M(B(j)) - M(B(i)) < 0. \quad (27)$$

These constraints remove arcs that would lead to the generation of pairs of consecutive sections that are assigned to outbound blocks destined to the same train and that could violate the marshalling plan. For example, suppose that we have a set of outbound blocks $B = \{b_1, b_2, b_3\}$, which are all destined to the same outbound train. To simplify, also assume that the marshalling plan specifies that these blocks should appear in the train in the same order such that $M(b_1) = 1$, $M(b_2) = 2$ and $M(b_3) = 3$. Then, a solution such as the one depicted in Figure 7 would generate two sections, $e_1 = \{r_1\}$ and $e_2 = \{r_2, r_3, r_4\}$, assigned to blocks b_2 and b_1, respectively. In this scenario, there are two possible configurations. The first one would have both sections in the same segment, in which case it would be necessary to rearrange the order of these sections when the outbound train is assembled. In the second configuration, each section would be part of its own segment, which is likely to create issues in the second step or at least require additional effort to pick each section in the correct order. The post-processing procedure described in the next section would generate both

configurations of this example. From now on, we refer to the formulation composed by constraints (16)-(27) as $CG2_{heur}$.

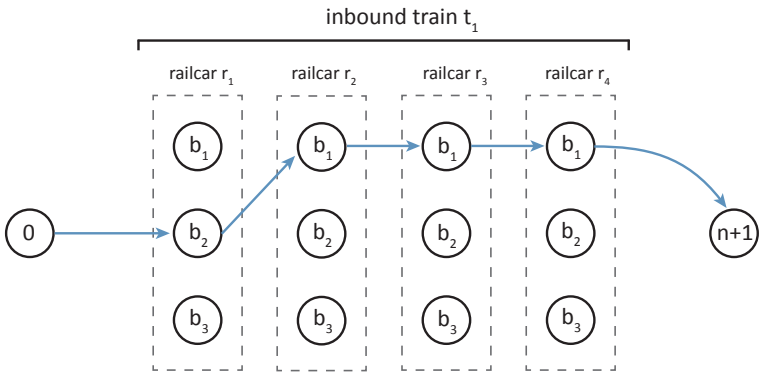


Figure 7: Example of a solution with consecutive sections that might lead to a violation of the marshalling plan in step two.

4.1.3. Post-processing procedure

As mentioned at the beginning of the section, the solution of formulation CG provides only a list of sections to be created and their assignment to outbound blocks. Given one such solution, a post-processing procedure enumerates all possible ways of creating segments based on the list of sections associated with each inbound train. This procedure is illustrated in Algorithm 1, which receives as input the values of the x and u variables, denoted by $\bar{x}$ and $\bar{u}$, respectively. Initially, the values of the x variables are analyzed by function *getListSectionsPerTrain* in order to generate a list of sections for each inbound train. Then, the list of sections for each inbound train $t \in T^-$ is processed by function *permutate*, which generates a set of configurations by trying all possible ways to combine the given sections into segments without violating the maximum number of cuts c_t. Note that cuts are enforced either because of the values of the u variables or because assigning two consecutive sections to the same segment would cause one of the two issues specified by ϕ described in Section 4.1. Therefore, not all permutations are accepted. The resulting output is a set of feasible configurations for each inbound train.

4.2. Step 2: Assignment of segments to tracks

We now propose an algorithm based on a second MILP formulation that takes as input a set of configurations generated in the first step, and decides which configurations to use and where to park each segment. As in the previous step, aspects such as the demand of outbound blocks and the length of the outbound trains are considered. However, in this

Algorithm 1 Post-processing algorithm to generate configurations

```
function generateConfigurations(x̄, ū)
    set_configurations ← ∅
    list_sections_per_train ← getListSectionsPerTrain(x̄)
    for each t ∈ T⁻ do
        set_configurations ← set_configurations ∪ permutate(list_sections_per_train[t], ū[t], c_t)
    return set_configurations
```

step, the proposed algorithm also considers that there must be a feasible set of movements to bring each segment to its assigned track and track position, and a set of movements capable of pulling out each section from the tracks to build the outbound trains.

Recall that each configuration is associated with a single inbound train and that the set of configurations, denoted by I, may contain multiple configurations for the same train. In this context, let I_t specify the subset of configurations associated with inbound train $t \in T^-$, and let $I(s)$ be a function that determines which configuration contains segment $s \in S$, while S_i is the set of all segments associated with configuration $i \in I$. Let l_{st} specify the total length of railcars from segment $s \in S$ that are part of sections assigned to blocks destined to outbound train $t \in T^+$. In addition, let γ^f_{sb} and γ'_{sb} specify, respectively, the number of slots for containers of type $f \in F$ and aggregate slots from railcars of segment $s \in S$ that belong to sections assigned to block $b \in B$.

Let y_{spq} be a binary variable that specifies whether segment $s \in S$ is assigned to track $p \in P$ in position $q \in Q_p$, and let z_i be a binary variable that takes value 1 if and only if configuration $i \in I$ is used. We are now ready to introduce the following formulation. For reasons of clarity, some notation associated with the objective function is introduced along with its description after the model.

$$\text{(AF)} \quad \min \; z_{AF} = w^d \sum_{s\in S}\sum_{p\in P}\sum_{q\in Q_p} \alpha_{spq}\; y_{spq} + \sum_{i\in I}(w^w\delta_i + w^p\sigma_i + w^m\theta_i)\; z_i \tag{28}$$

subject to

$$\sum_{i\in I_t} z_i = 1 \qquad \forall\; t \in T^- \tag{29}$$

$$\sum_{p\in P}\sum_{q\in Q_p} y_{spq} = z_{I(s)} \qquad \forall\; s \in S \tag{30}$$

$$\sum_{s\in S}\sum_{p\in P}\sum_{q\in Q_p} l_{st}\; y_{spq} \le (1+\rho)\; l_t \qquad \forall\; t \in T^+ \tag{31}$$

$$\sum_{k=0}^{q} \sum_{\substack{s \in S_i: \\ |s| > q-k}} y_{spk} \leq 1 \qquad \forall\ i \in I, p \in P, q \in Q_p \tag{32}$$

$$\sum_{s \in S} \sum_{p \in P} \sum_{q \in Q_p} \gamma^f_{sb}\ y_{spq} \geq d_{bf} \qquad \forall\ b \in B, f \in F \tag{33}$$

$$\sum_{s \in S} \sum_{p \in P} \sum_{q \in Q_p} \gamma'_{sb}\ y_{spq} \geq d'_b \qquad \forall\ b \in B, f \in F \tag{34}$$

$$y_{spq} \in \{0,1\} \qquad \forall\ s \in S, p \in P, q \in Q_p \tag{35}$$

$$z_i \in \{0,1\} \qquad \forall\ i \in I. \tag{36}$$

The objective function is composed by a set of four penalties:

- $\sum_{s \in S} \sum_{p \in P} \sum_{q \in Q_p} \alpha_{spq}\ y_{spq}$: evaluates the distance from the container stacks to the position on the tracks where the railcars in which these containers have to be loaded are parked. As mentioned in Section 3, containers destined to the same outbound block are usually stored in the same area in the terminal. Assuming that the position of each stack of containers is known, it is possible to evaluate the distance from these stacks to each track position. Therefore, for each segment $s \in S$, track $p \in P$ and track position $q \in Q_p$, we define α_{spq} as the sum of the distances (in meters) of each railcar and the location where the containers associated with its assigned outbound block are stacked.

- $\sum_{i \in I} \delta_i z_i$: evaluates the total dwell time of railcars in the terminal. Recall that a configuration specifies not only which segments and sections are created but also the assignment of sections to outbound blocks. Therefore, for any given configuration $i \in I$, the arrival and departure time of each railcar is known. Then, let us define δ_i as the sum of the number of hours that each railcar in configuration i remains at the terminal.

- $\sum_{i \in I} \sigma_i\ z_i$: evaluates the number of 53-ft platforms sent to ports. Because the international market mainly follows the ISO standards, which does not include 48- and 53-ft containers, terminal operators usually desire to minimize the number of railcars with 53-ft platforms that are sent to terminals located in ports. To incorporate this aspect into the model, we define σ_i as the number of 53-ft platforms in configuration $i \in I$ that are assigned to outbound blocks destined to ports.

- $\sum_{i \in I} \theta_i\ z_i$: evaluates the level of fragmentation of the segments that belong to configuration $i \in I$. A segment is considered fragmented if it contains consecutive sections

assigned to blocks destined to distinct outbound trains. For example, consider a certain configuration $i \in I$, which contains a segment s_1 composed by the sequence of sections $[e_1, e_2, e_3]$. Suppose that sections e_1 and e_3 are assigned to blocks destined to the same outbound train. However, section e_2 is destined to a different train. In this example, pairs $\{e_1, e_2\}$ and $\{e_2, e_3\}$ each contribute with value 1 to the evaluation of θ_i, which specifies the number of pairs of consecutive sections from configuration $i \in I$ that are assigned to blocks destined to different outbound trains. The motivation for this penalty is the same as the one presented for the creation of variables g in Section 4.1.

Notice that w^d, w^w, w^p and w^m are the weights of each penalty. Constraints (29) specify that exactly one configuration per inbound train is selected, whereas constraints (30) ensure that a segment can only be assigned if its respective configuration is chosen. Each outbound train has a maximum length that is determined by the power of its assigned locomotives (31). Constraints (32) specify that segments that originate from the same inbound train cannot overlap on the tracks. Note that overlaps between segments from different inbound trains are not possible, because in that scenario the segment that arrives later would push the other segment to a different position on the track. Constraints (33) ensure that there is a sufficient number of slots to meet the demand for each container type. As mentioned in Section 3, considering only the number of slots for container of each type might lead to an overestimation of the capacity of the railcars. To prevent this issue, constraints (34) impose an upper bound on the number of aggregate slots that can be used.

To ensure that a solution generated by formulation AF is feasible, we must verify that each segment can reach its assigned track and track position and that each section can be properly pulled out of its track when it is time to depart. In addition, we must consider that the arrival of each inbound train may cause other railcars that were already parked to be moved as well. To avoid burdening the formulation with additional variables and constraints, we opted to implement a branch-and-reject algorithm, where each integer solution is given to a feasibility checking procedure, which then verifies if the solution is indeed feasible. In the case where the procedure detects that the solution is infeasible, it simply rejects it.

The feasibility checking procedure is presented in Algorithm 2. It takes as input a solution $(\bar{y}, \bar{z})$ of formulation AF, and the initial state of the terminal, which represents the state of each track before the arrival of the first inbound train. In a first phase (lines 2-14), the procedure simulates how the arrival of each inbound train changes the state of the terminal. Starting with the first inbound train to arrive, the previous state of the terminal (in this case the initial state) is retrieved and copied (line 6). The values of the z variables are then

inspected and the set of segments created for train t are identified (line 7). At each iteration of the loop defined between lines 8-12, function *getAssignedTrackAndPosition* is called to find the track and position assigned to segment s. Then, function *simulateArrivalOfSegment* modifies the current state of the terminal by simulating segment s being moved to track $p \in P$ and position $q \in Q_p$. In this simulation, the segment enters track p from one of its ends, chosen based on the direction from which its inbound train arrives in the terminal, and any railcars that are in the way are pushed to other positions. To simplify, we assume that crossovers cannot be used in this case. Note that function *simulateArrivalOfSegment* returns *true* if moving segment s to its designated position does not result in other railcars being pushed over the limits of track p. Otherwise, the function returns *false* and the solution is infeasible (lines 11-12).

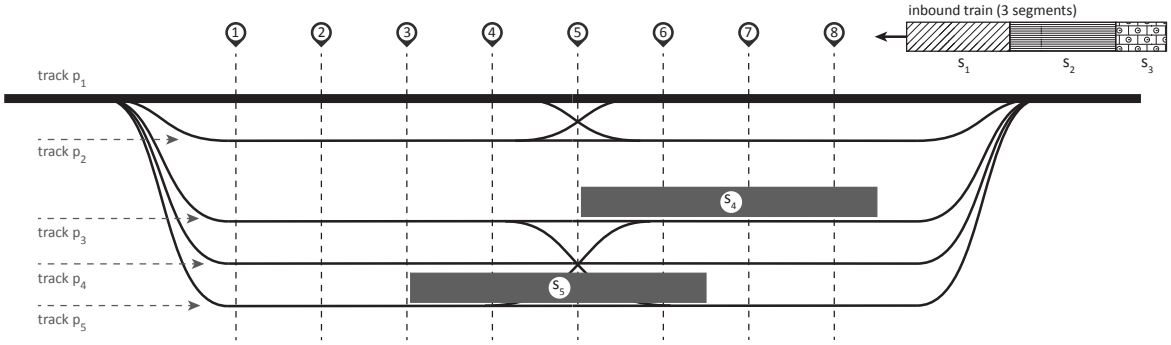
As an example of the simulation of an inbound train arrival, consider Figure 8-(a), which illustrates the state of a terminal before the arrival of an inbound train containing segments s_1, s_2 and s_3. The vertical dashed lines represent the positions of the tracks. Notice that the inbound train is arriving from the right side of the terminal, and that there are already two segments (s_4 and s_5) at the terminal, on tracks p_3 and p_5. Recall from Section 3 that segments from an inbound train that arrives from the right side of the terminal are brought to their assigned positions from the opposite side. Then, assume that segment s_1 is assigned to track p_3 and position 6, segment s_2 to track p_5 and position 3, and segment s_3 should be parked in track p_4 at position 2. Figure 8-(b) illustrates the state of the terminal after these segments are brought to their assigned positions. Considering the order of the segments on the inbound train, segment s_3 is the first to be brought to its position. Because there are no railcars parked on track p_4, this operation is done normally. Then, in order to bring segment s_2 to its assigned position, segment s_5 has to be pushed to the right. Similarly, segment s_1 pushes segment s_4 to the right in order to reach its assigned position. However, by doing so, segment s_4 is pushed over the limit of track p_3 (indicated by the vertical red line), which makes the solution infeasible.

In the second phase of Algorithm 2 (lines 16-23), in case the solution has not already been found infeasible, the algorithm verifies if all sections can be pulled out from their tracks when it is time to depart. For each outbound train $t \in T^+$ the last simulated terminal state before the departure of t is retrieved. Then, the procedure finds all sections assigned to outbound blocks destined to the outbound train t (line 19). For each section, function *isSectionAbleToDepart* checks all possible alternatives to pull out the section, either from its own track or via a crossover (line 21). This verification is performed iteratively for each connected track and considering the side from which the respective outbound train departs,

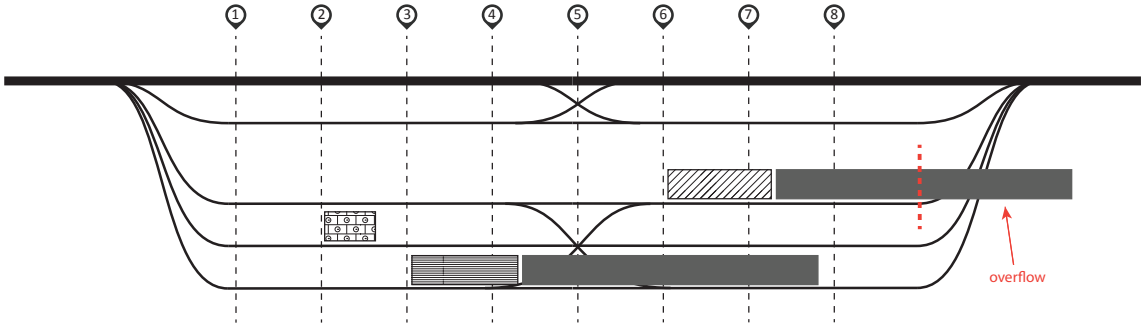
Algorithm 2 Feasibility checking procedure

```
1: function isSolutionFeasible(ȳ, z̄, initial_state)
2:     list_terminal_states ← ∅
3:     previous_state ← initial_state
4:     in_trains_ordered ← orderInboundTrainsByArrivalTime()
5:     for each t ∈ in_trains_ordered do
6:         current_state ← copy(previous_state)
7:         segments ← getSegmentsCreated(in_trains_ordered[t], z̄)
8:         for each s ∈ segments do
9:             (p, q) ← getAssignedTrackAndPosition(s, ȳ)
10:            is_feasible ← simulateArrivalOfSegment(s, p, q, current_state)
11:            if not is_feasible then
12:                return false
13:        list_terminal_states.insert(current_state)
14:        previous_state ← current_state
15:
16:    out_trains_ordered ← orderOutboundTrainsByDepartureTime()
17:    for each t ∈ out_trains_ordered do
18:        current_state ← getLastTerminalStateBeforeDeparture(t,
       list_terminal_states)
19:        sections ← getSectionsAssignedToOutboundTrain(t, ȳ)
20:        for each e ∈ section do
21:            is_feasible ← isSectionAbleToDepart(e, current_state)
22:            if not is_feasible then
23:                return false
24:    return true
```

which defines from where the section must be pulled out. If at least one of the alternatives is feasible (i.e., there is nothing in the way), then we consider that the section can be pulled out and we proceed to the next section on the list. Otherwise, the algorithm classifies the solution as infeasible and terminates (lines 22-23). As an example, refer to Figure 9, which depicts the state of the terminal just before the departure of an outbound train containing sections e_1, e_2 and e_3, departing from the right side. Note that sections e_4 and e_5 only depart later. A call to function *isSectionAbleToDepart* from Algorithm 2 for section e_1 successfully verifies that it is indeed possible to pull out this section from the same track it is parked. Similarly, function *isSectionAbleToDepart* would verify that section e_3 is not able to exit from track p_5 due to the presence of section e_5, but it can depart either from track p_3 or p_4 using the crossover. However, when verifying the departure of section e_2, the function would return *false*, because section e_4 is blocking its departure. Note that section e_2 cannot be pulled from the left side of the terminal, as the assembly of its outbound train occurs on



(a) The state of the terminal before the arrival of the inbound train.



(b) The state of the terminal after the arrival of the inbound train

Figure 8: Example of the arrival of an inbound train containing 3 segments.

the right side.

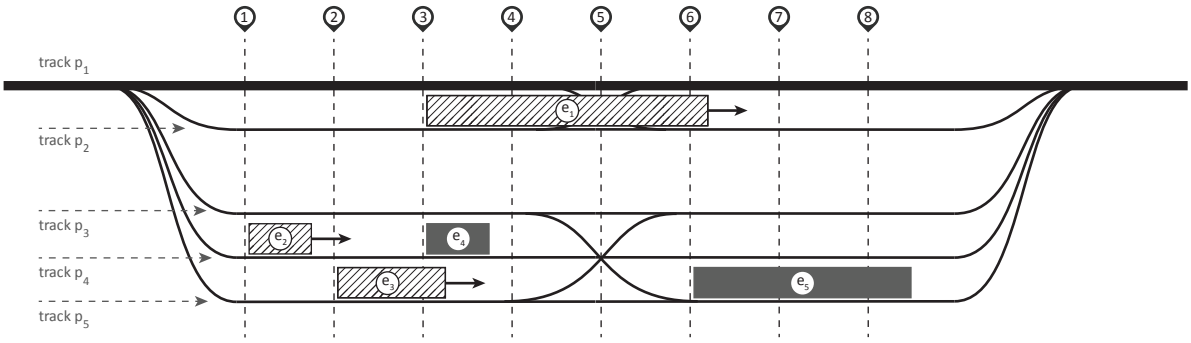


Figure 9: Example of the state of the terminal just before the departure of sections e_1, e_2 and e_3 from the right side. Sections e_4 and e_5 only depart later. While sections e_1 and e_3 can be pulled out from their tracks, section e_2 is blocked by section e_4.

5. Computational experiments

Our algorithms were coded in C++ and the computational experiments were run on a PC with an Intel Xeon X5650 2.67 GHz processor with 48 Gb of RAM. We have used CPLEX 12.7.1 with default options and 4 threads to solve the formulations. The values for the weights in all three formulations were chosen with guidance from the company to ensure that the results are meaningful and are in line with the company's needs. They have been selected so that the model produces solutions that are deemed implementable and that reflect trade-offs made by the decision makers.

To test our algorithms we initially derived a set of 10 benchmark instances from realistic data of terminal operations during 10 consecutive days. Therefore, each instance represents the operations of a terminal in a single day. Table 1 summarizes the dimension of each instance by providing the number of inbound and outbound trains (*# in trains* and *# out trains*, respectively), the number of inbound railcars (*# in railcars*) and the number of outbound blocks (*# out blocks*). The considered terminal layout is fairly similar to the one shown in Figure 3, which has a total of five tracks, ranging from 6800 ft to 8400 ft in length.

Table 1: Summary of the benchmark instances

instance	# in trains	# in railcars	# out trains	# out blocks
i1	3	235	4	9
i2	4	316	6	18
i3	3	233	3	13
i4	3	221	5	14
i5	3	232	4	9
i6	3	239	3	11
i7	5	351	7	21
i8	3	236	5	12
i9	3	285	3	10
i10	4	257	5	13
avg	3.4	260.5	4.5	13.0

5.1. Experiments on realistic instances

Initially, we ran an extensive round of tests for Step 1, using a total of 75 and 87 different combinations of parameter values for formulations $CG1$ and $CG2$, respectively. A time limit of one hour was set for each of these tests. Approximately half of the combinations were set to use the heuristic version of the formulations. Note that we ended up running more experiments with formulation $CG2$ given that this formulation proved to be more sensitive

to parameter values than $CG1$. Table 2 presents the results of these tests in terms of average number of configurations generated per instance (column *# avg*), along with the standard deviation (column *std dev*). Although this table presents high-level results, it still provides useful insights on the level of difficulty of the instances. We observe that, usually, the more configurations are generated in Step 1, the more flexible is the scenario in terms of how the outbound blocks can be built, which in turn usually means that it is easier to find a feasible solution in Step 2.

Table 2: Average results from Step 1 - Generation of configurations

	$CG1$		$CG2$	
instance	# avg	std dev	# avg	std dev
i1	1200.91	2360.11	344.37	295.96
i2	60.16	169.63	4.20	10.24
i3	611.40	1821.95	48.78	240.84
i4	151.29	364.89	17.24	31.12
i5	1402.12	2317.11	696.63	755.88
i6	32.05	53.89	119.32	247.85
i7	5.68	24.69	0.00	0.00
i8	26.11	78.34	43.86	71.32
i9	124.09	325.92	31.66	74.90
i10	349.99	700.76	275.44	855.01

An important detail about the terminal operations is that when traffic is too high and the terminal is congested, it is possible to call a special operator that uses an extra locomotive to move railcars around the terminal without the constraints that are normally imposed. We refer to this locomotive as a *switcher*. This means that with the switcher at the terminal almost any placement of railcars on the tracks is feasible with respect to the assembly of the outbound trains, given that railcars that are on the way of others are simply moved to another track by the switcher to allow outbound trains to be assembled with ease. The downside is that using a switcher is time-consuming and expensive, so it is only used when strictly needed. Among our test cases, there are two instances in which the switcher was needed to ensure that the terminal could function properly, which are $i7$ and $i10$. We initially tried to use our solution approach to find solutions for these instances that did not require a switcher, but we could not find any. Therefore, all the results for instances $i7$ and $i10$ that are presented in this section, including those of Table 2, refer to experiments performed by disabling the simulation of how the outbound trains are assembled, thus simulating the presence of the switcher.

In general, tuning the parameters for Step 1 is a difficult task, given that analyzing the number of generated configurations is not a proper measure of quality and neither is the objective function used in this step, which is used mostly to guide the model towards features that are usually found in good quality solutions. For example, the results presented in Table 2 suggest that instances $i6$, $i7$ and $i8$ are difficult ones, as the number of configurations generated is rather small, especially for formulation CG1. While this is a correct assumption for the latter two, our experiments have shown that our approach easily finds solutions for instance $i6$.

In order to properly analyze which combinations of parameters lead to the best results, we also run Step 2 with each set of configurations and get complete solutions for the problem. These solutions can then be compared directly by their objective function values. Figures 10 and 11 depict the results of these experiments with the configurations generated by, respectively, formulations $CG1$ and $CG2$, and allow us to directly compare each combination of parameters. In the x axis we show an identifier of each combination of parameters, including only those that yielded at least one feasible solution in the second step. The primary y axis (on the left) is associated with the orange line and shows the average improvement over the solution found by the company, while the secondary y axis is associated with the vertical bars and indicates the number of instances for which each combination of parameters found a feasible solution. Therefore, the best combinations of parameters should be those with the highest bars and that achieved the largest improvements.

According to Figure 10, the best combinations of parameters for formulation $CG1$ seem to be $CG1\_74$ and $CG1\_75$. Interestingly, in both cases the weights w_1 and w_2 were set to zero, which seems to indicate that formulation $CG1$ performs better (in terms of quality of the generated configurations) when the penalties associated with the waiting time of railcars in the terminal and the number of 53-ft railcars sent to port terminals are turned off. In fact, from the results of the individual experiments it is clear that whenever a larger weight was given to w_1 in Step 1, the resulting set of configurations was, in most of the cases, unable to yield a single feasible solution, and when a solution was found, the improvements were significantly lower than in the other cases. One possible explanation for this is that considering that Step 1 does not take into account the assignment of the railcars to the tracks, when it tries to optimize the waiting time of railcars in the terminal it ends up grouping railcars in such a way that it is not optimal for the placement of railcars in Step 2. Indeed, from a practical perspective and considering the layout of the terminal, usually the traffic is quite intense and there is not much room for concentrating the efforts in optimizing this criterion.

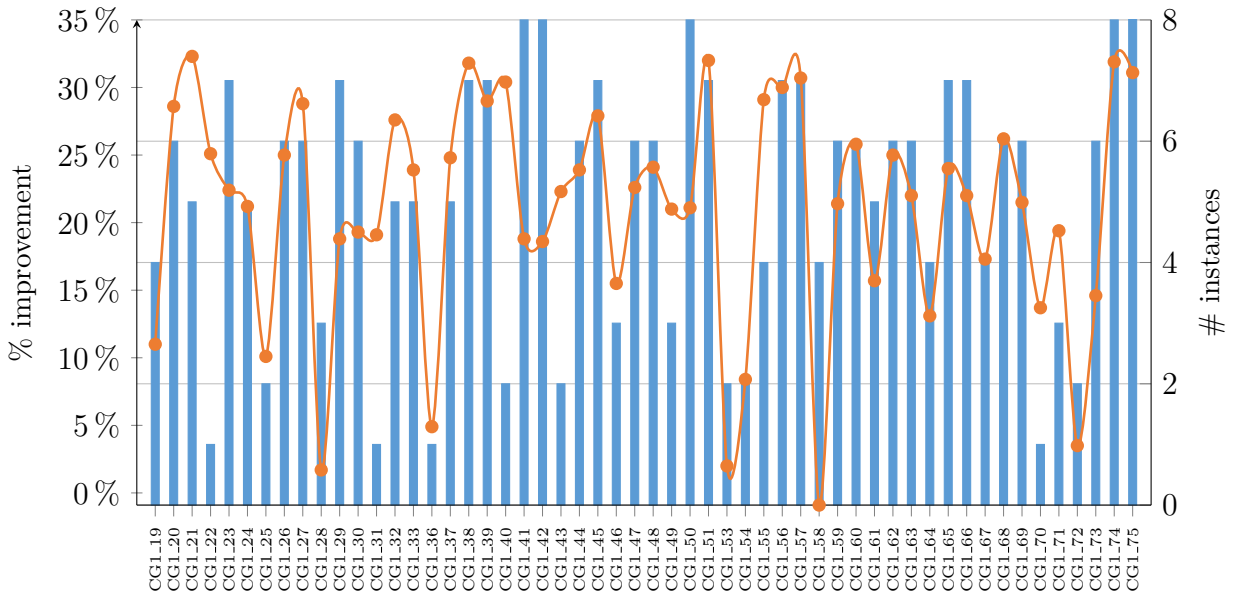


Figure 10: Results from Step 2 based on the configurations generated by $CG1$ with different combinations of parameters

Figure 11 presents a similar analysis for formulation $CG2$. Note that this formulation behaves very differently and is more sensitive to parameter values. While $CG1$ rarely resulted in solutions that were worse than those implemented by the company, in this case, there are combinations of parameters that lead to considerably worse solutions. In the case of $CG2$ it is also not as clear which combinations are the best. However, good candidates are $CG2\_27$, $CG2\_29$, $CG2\_68$, $CG2\_78$, $CG2\_83$ and $CG2\_84$. Although none of these combinations of parameters provided the highest improvements, they were able to yield feasible solutions for the highest number of instances (8 or 9 out of 10), while also providing considerable improvements ranging, on average, from 18% to 20%. An interesting observation about these scenarios is that in all of them the weight w_4 was set to a significantly higher value than the other ones. Recall from Section 4.1.1 that w_4 is the weight of the penalty associated with the number of times two consecutive sections are assigned to outbound blocks destined to different trains. These results were somehow expected considering that when this penalty is set to a higher value the model is driven towards finding configurations that have consecutive sections assigned to the same outbound train. This, in turn, can greatly reduce the complexity of the operations required to assemble the outbound trains on the tracks, increasing the likelihood of finding feasible solutions. As for $CG1$, we also observed that in all of the aforementioned scenarios, except for $CG2\_78$, the penalty w_1 was set to

zero. Both observations seem to imply that in Step 1 it is best to focus on the penalties that are associated with structural aspects, which may lead to an easier placement of the railcars on the actual tracks and simplify the assembly of the outbound trains.

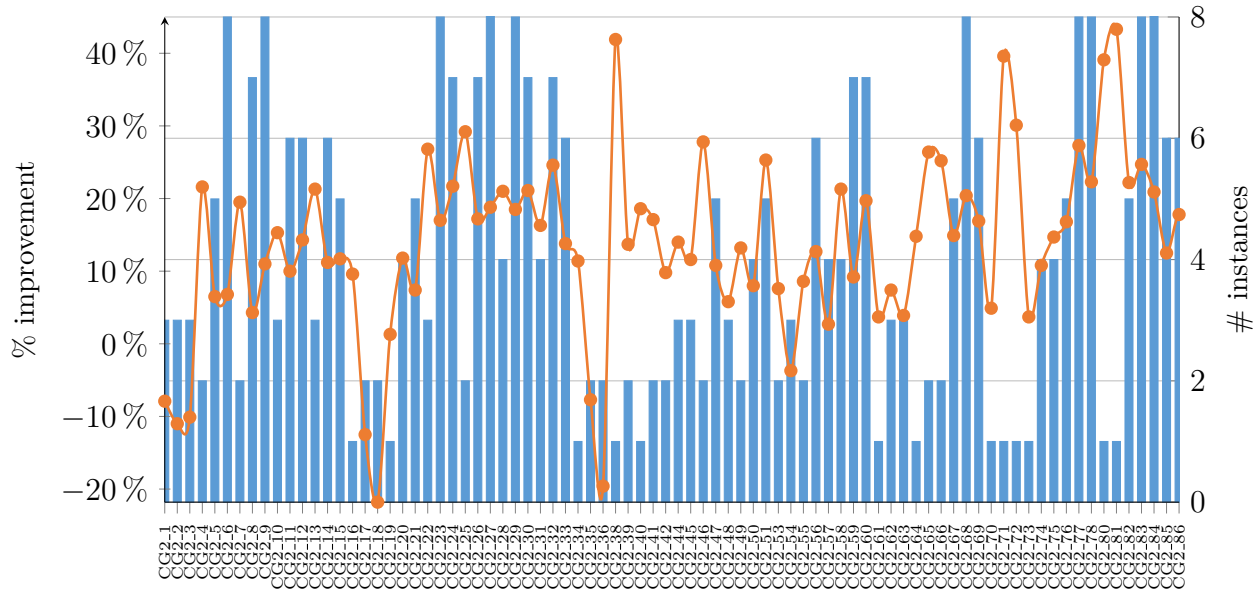


Figure 11: Results from Step 2 based on the configurations generated by $CG2$ with different combinations of parameters

Figure 12 presents a boxplot of the results found in Step 2 per instance, considering the sets of configurations generated by $CG1$ and $CG2$. The value of the company solution for each instance is depicted in the figure as a blue line. From this perspective we can clearly see how our solution approach performs for each instance and the variation of the solution values that were found. Note that, with the configurations generated by $CG1$ we were unable to find a feasible solution for instance $i8$, while with configurations from $CG2$ we were unable to find solutions only for instance $i2$. Furthermore, these results suggest that the configurations generated by $CG1$ are more stable and usually lead to good solutions with a small range of variation if compared with $CG2$. This is especially true for instance $i6$, where the solutions found by $CG1$ are significantly better.

Given that the objective function of formulation AF from Step 2 is composed by four terms, a more in depth analysis is necessary to understand the contribution of each term. To this end, Figures 13 and 14 present the percentage improvement per objective function term over the solution proposed by the company using configurations generated by $CG1$ and $CG2$, respectively. The biggest improvements, especially for the experiments performed with

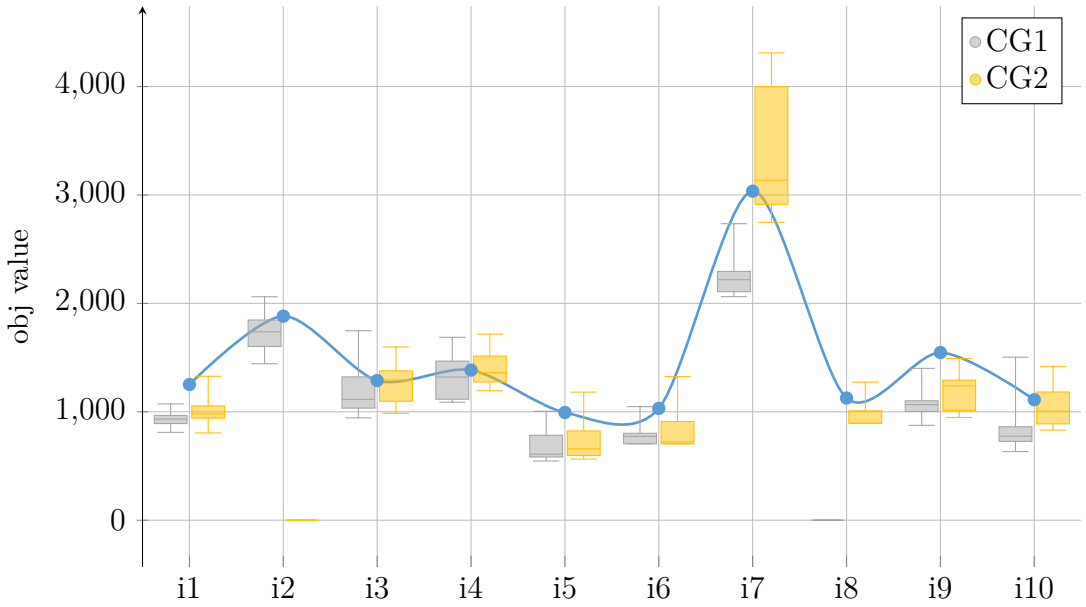


Figure 12: Boxplot presenting the results of Step 2 per instance.

configurations from $CG1$, seem to come from the penalties associated with the distance from container stacks to the location of the railcars and the number of 53-ft railcars sent to port terminals. This is an interesting result given that placing railcars closer to their associated container stacks is critical to improve the efficiency of the loading operations. Furthermore, reducing the number of 53-ft railcars sent to the port terminals allows to better use the train capacity. Indeed, there are no 53-ft containers at the ports (it is a size used exclusively in the North American domestic market) and 53-ft railcars take up more train capacity than 40-ft ones. On the other hand, having low improvements (or even a slightly negative impact) on the railcar dwell time is less important as costs associated with the dwell of railcars are much lower than those associated with train capacity and loading operations. In fact, optimizing the dwell time in our problem is only a preemptive measure to try to ensure that the terminal has free space on the tracks, which, as a consequence, gives more options to park incoming railcars.

Concerning the fragmentation aspect, according to Figure 13, the improvements achieved using configurations from $CG1$ vary widely depending on which set of configurations that is used. In this case we cannot find a clear pattern in the results that would determine why sometimes we get significant improvements and in other cases the solution is considerably worse. The results obtained with the configurations from $CG2$, shown in Figure 14, are more interesting with respect to the fragmentation. For instance, the configurations used in the scenarios from $CG2\_1$ to $CG2\_18$ were generated by formulation $CG2$ with $w_4 = 0$,

i.e., with the penalty associated with the level of fragmentation disabled. As a result, most of the solutions that were found using these configurations were worse with respect to the fragmentation. When the fragmentation was allowed, the results were much better.

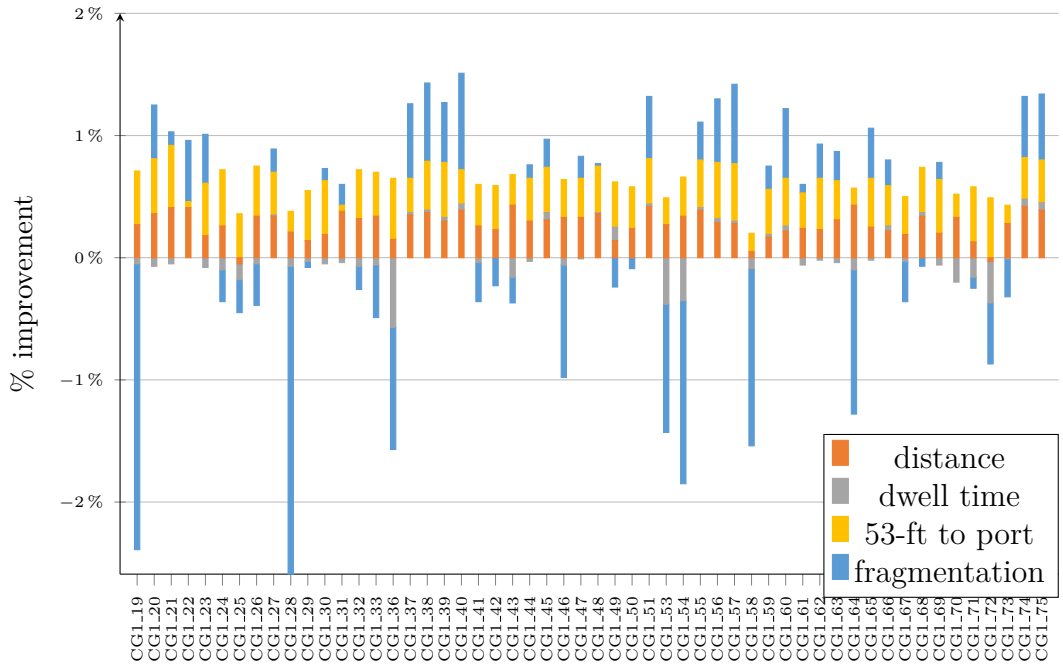


Figure 13: Improvements from Step 2 per objective function term based on the configurations generated by $CG1$

In general, we observed that using $CG1$ usually leads to better solutions and, on average, the improvements are higher. Although $CG2$ also performs well, it is more unstable and prone to generating solutions that are actually worse than those created manually by the company. Given that none of the formulations is able to consistently provide solutions for all instances, in practice, a good strategy would be to run $CG1$ and $CG2$ in parallel and take advantage of the strength of both formulations.

5.2. Sensitivity analyses

To understand how aspects such as the demand for the outbound blocks and the composition of the inbound trains impact on the efficiency of the method and in the complexity of the scenarios, we modified the original instances to generate three new benchmark sets. Usually, 53-ft railcars are spread out on the inbound trains instead of being grouped. We suspected that this heterogeneous composition of the trains could considerably affect the complexity of the scenarios, given that 53-ft railcars are not supposed to be sent to port terminals and this is considered when generating configurations. In this sense, for the first

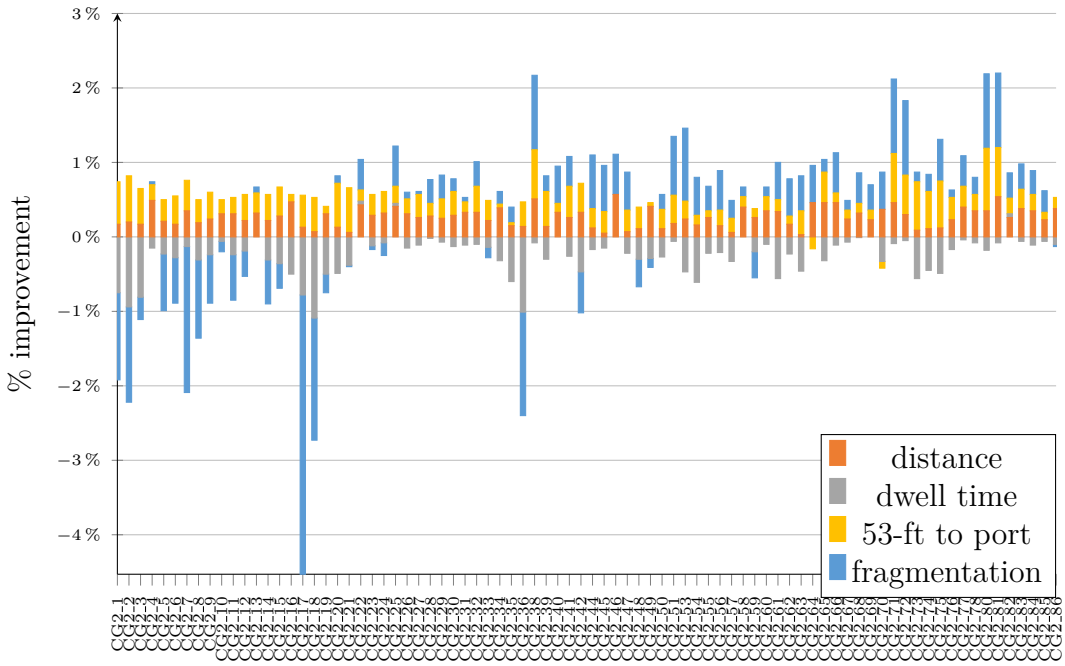


Figure 14: Improvements from Step 2 per objective function term based on the configurations generated by $CG2$

new set, named $N53$, each instance was modified by transforming 53-ft railcars into 40-ft ones and demand for 53-ft containers into demand for 40-ft containers. The two other new sets of instances are intended to test how variations in the demand for the outbound blocks might impact the overall performance. These sets – $D50$ and $D75$ – are thus composed by instances where the demand for outbound blocks is set, respectively, to 50% and 75% of the original values.

Table 3 presents the results of the experiments with the new sets of benchmark instances along with the results of the original instances for comparison. Column *# sets* specifies the number of sets of configurations that were generated in Step 1, which is also the number of times Step 2 was run. That is, once for each set. Column *# sol* specifies the number of tests that resulted in a feasible solution, while columns *avg obj*, *avg # configs* and *avg time* present, respectively, the average solution value, number of configurations used in the experiments and computational time required. Note that from the results of set N53, indeed, not having 53-ft railcars does seem to improve the efficiency of the method, given that we were able to find feasible solutions for all instances with configurations generated by both $CG1$ and $CG2$. For most cases, there were no significant changes in the number of sets of configurations that were found in Step 1. This suggests that allowing $CG1$ and $CG2$ to focus more on optimizing structural aspects leads to configurations that contain

segments and sections that are more easily handled by the second step and do not cause too many problems with respect to how they are placed in the tracks and later moved when the outbound trains are assembled.

According to the results shown in Table 3 for sets D75 and D50, where the demand for the outbound blocks was reduced by 25% and 50%, respectively, scenarios with lower demand do not necessarily lead to improvements in the performance of the method. Although for scenarios like $i8\_D50$ and $i8\_D75$ the results were significantly better, for the other instances there were no considerable improvements that came from reducing the demand for the outbound blocks. This might be explained by the fact that, in instance $i8$, the demand for the outbound blocks was very tight with respect to the total number of slots that were available to be assigned.

In practice, a decision support system that implements the proposed approach could run, in parallel, multiple instances of the algorithm with the best parameter settings to maximize the chances of finding feasible solutions. The terminal plans are built one day before they are put in practice, which could even allow the DSS to run the algorithms for longer, increasing even further the number of solutions that could be found. In this setting, the algorithm could also be used in a rolling horizon fashion to account for railcars that remained in the terminal as a result of the solutions implemented on the previous days.

6. Conclusions

We have addressed a practical problem faced by a major North American railway concerning rail transportation of intermodal containers. Our approach incorporates all relevant decisions and constraints to obtain good quality solutions. The first step creates a set of configurations (patterns) which specify how each inbound train can be split into sequences of railcars and the assignment of outbound blocks to each railcar. The second step then decides which configurations to use and how to park the railcars, ensuring that not only it is possible to bring the railcars to their assigned track, but also that they can be properly pulled out from the tracks when it is time to depart. Computational experiments on benchmark instances based on realistic data from the company show that our approach is able to find good quality solutions for all instances. In practice, the methods and tools developed might prove valuable to the company not only to optimize their operations, but also as a means to automate the whole process of generating the plans, which at the moment is performed manually.

As future work we are mainly focused on implementing a rolling horizon approach to simulate the performance of our algorithm in longer time periods. This is important because,

Table 3: Results of the experiments with the new sets of benchmark instances: N53, D50 and D75.

	*CG*1					*CG*2				
	# sets	# sol	avg obj	avg # configs	avg time	# sets	# sol	avg obj	avg # configs	avg time
Oct_20	52	29	**939.2**	397.4	3049.4	84	33	1004.5	292.5	2271.9
Oct_21	34	28	**1727.6**	111.6	1032.6	29	0	—	—	—
Oct_22	42	24	**1182.8**	302.1	2899.8	47	35	1281.3	47.8	2328.7
Oct_23	35	12	**1326.3**	181.7	1774.4	47	24	1397.1	31.8	2135.8
Oct_24	48	48	**673.9**	406.0	955.6	71	71	726.9	353.6	1051.3
Oct_25	45	45	**775.4**	53.4	172.0	63	62	821.5	125.3	241.6
Oct_26	11	11	**2240.9**	43.0	405.6	19	18	3405.9	53.6	933.7
Oct_27	19	0	—	—	—	60	30	**962.8**	60.6	90.7
Oct_28	34	33	**1063**	180.1	781.8	48	27	1187.5	51.8	121.1
Oct_29	43	43	**835.3**	262.4	33.1	38	38	1043.9	178.6	370.1
Oct_20_N53	55	20	**655.8**	465.1	3343.3	86	22	688.4	364.7	2990.4
Oct_21_N53	32	24	**1288.8**	124.3	1696.0	2	2	1610.0	10.5	868.8
Oct_22_N53	46	31	**755.9**	389.0	3105.9	53	46	867.4	113.0	2596.5
Oct_23_N53	33	14	**957.7**	213.5	2182.7	49	27	966.7	88.8	2581.1
Oct_24_N53	51	51	**444.9**	398.7	1368.4	81	81	521.6	361.2	1397.3
Oct_25_N53	50	48	**601.2**	53.0	217.5	57	55	614.6	88.6	175.2
Oct_26_N53	18	18	**1651.5**	46.9	766.3	18	18	2804.7	50.0	980.7
Oct_27_N53	35	28	**918.1**	159.6	378.4	70	38	925.6	87.6	398.3
Oct_28_N53	47	46	**696.9**	300.1	586.8	66	60	802.1	225.0	476.4
Oct_29_N53	43	43	**530.5**	406.1	21.7	42	42	680.9	124.1	8.9
Oct_20_D50	51	21	**944.8**	475.2	3424.4	84	29	994.9	350.1	2608.1
Oct_21_D50	34	27	**1679.6**	138.1	1343.9	2	0	—	—	—
Oct_22_D50	45	33	**1018.1**	412.8	3347.1	59	49	1145.2	154.3	2539.0
Oct_23_D50	39	18	1324.6	209.7	2057.2	49	35	**1307.1**	97.8	2288.3
Oct_24_D50	50	50	**680.3**	403.8	1333.0	73	73	713.9	384.6	1130.5
Oct_25_D50	48	48	**782.3**	61.5	93.2	68	64	836.6	83.6	189.1
Oct_26_D50	17	17	**2268.3**	48.8	395.7	33	32	3334.7	86.6	1497.7
Oct_27_D50	34	21	1046.0	154.4	854.7	70	38	**971.2**	112.6	999.7
Oct_28_D50	43	40	**965.1**	347.5	1813.1	63	58	1024.5	239.0	370.1
Oct_29_D50	44	44	**696.4**	368.8	17.8	39	39	1035.8	86.2	8.3
Oct_20_D75	53	27	**940.8**	426.7	3282.0	86	26	1015.4	293.9	2407.9
Oct_21_D75	33	26	**1723.1**	128.6	1370.1	4	2	1833.4	6.3	2160.2
Oct_22_D75	46	35	**1112.9**	372.2	2977.1	53	42	1250.6	81.8	2375.4
Oct_23_D75	36	14	**1262.3**	175.1	1952.6	51	28	1352.0	55.3	2033.1
Oct_24_D75	50	50	**671.9**	387.6	1030.5	76	75	733.2	360.0	1033.9
Oct_25_D75	47	47	**769.7**	63.8	246.6	60	58	833.8	82.4	229.3
Oct_26_D75	16	16	**2295**	33.4	360.9	30	29	3325.2	72.7	1522.8
Oct_27_D75	35	18	1057.8	140.8	882.7	70	39	**957.7**	122.3	584.6
Oct_28_D75	37	34	**958.4**	348.6	1646.7	64	59	1030.2	283.7	280.6
Oct_29_D75	43	43	**761.6**	315.6	28.0	36	36	1132.7	58.0	370.1

at the moment, our solutions correspond to plans that only include the inbound trains that arrive on a given day. However, there are usually several railcars that remain in the terminal for more than one day, as they might be assigned to blocks from outbound trains that depart on a day outside the time horizon that we consider. It would be interesting to analyze the algorithm's performance if the initial state of the terminal is determined by the solution

generated for the previous day.

Acknowledgments

We gratefully acknowledge the close collaboration with personnel from the Canadian National Railway Company (CN), the funding through the CN Chair in Optimization of Railway Operations at Université de Montréal and funding from the National Sciences and Engineering Council of Canada (NSERC) through a Collaborative Research and Development grant.

References

Adlbrecht JA, Hüttler B, Zazgornik J, Gronalt M, 2015 *The train marshalling by a single shunting engine problem. Transportation Research Part C: Emerging Technologies* 58:56–72.

Alicke K, 2005 *Modeling and optimization of the intermodal terminal mega hub. Container Terminals and Automated Transport Systems*, 307–323 (Springer).

Association of American Railroads, 2014 *Loading Capabilities Guide*. Available at: `https://www.aar.org/StatisticsAndPublications/Publications?title=Loading Capabilities Guide`.

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

Bodin LD, Golden BL, Schuster AD, Romig W, 1980 *A model for the blocking of trains. Transportation Research Part B: Methodological* 14(1):115 – 120.

Bouzaiene-Ayari B, Cheng C, Das S, 2016 *From single commodity to multiattribute models for locomotive optimization: A comparison of optimal integer programming and approximate dynamic programming. Transportation Science* 50(2):366–389.

Boysen N, Emde S, Fliedner M, 2016 *The basic train makeup problem in shunting yards. OR Spectrum* 38(1):207–233.

Boysen N, Fliedner M, 2010 *Determining crane areas in intermodal transshipment yards: The yard partition problem. European Journal of Operational Research* 204(2):336–342.

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.

Boysen N, Fliedner M, Kellner M, 2010 *Determining fixed crane areas in rail–rail transshipment yards. Transportation Research Part E: Logistics and Transportation Review* 46(6):1005–1016.

Boysen N, Jaehn F, Pesch E, 2011 *Scheduling freight trains in rail-rail transshipment yards. Transportation science* 45(2):199–211.

Boysen N, Jaehn F, Pesch E, 2012 *New bounds and algorithms for the transshipment yard scheduling problem. Journal of Scheduling* 15(4):499–511.

Bruns F, Goerigk M, Knust S, Schöbel A, 2014 *Robust load planning of trains in intermodal transportation. OR Spectrum* 36(3):631–668.

Bruns F, Knust S, 2012 *Optimized load planning of trains in intermodal transportation. OR Spectrum* 34(3):511–533.

Corry P, Kozan E, 2006 *An assignment model for dynamic load planning of intermodal trains. Computers & Operations Research* 33(1):1–17.

Corry P, Kozan E, 2008 *Optimised loading patterns for intermodal trains. OR Spectrum* 30(4):721–750.

Crainic TG, Kim KH, 2007 *Intermodal transportation.* Barnhart C, Laporte G, eds., *Handbooks in Operations Research and Management Science*, volume 14, chapter 8, 467–537 (Elsevier).

Dahlhaus E, Horak P, Miller M, Ryan J, 2000 *The train marshalling problem. Discrete Applied Mathematics* 103(1-3):41–54.

Heggen H, Braekers K, Caris A, 2016 *Optimizing train load planning: review and decision support for train planners. International Conference on Computational Logistics*, 193–208 (Springer).

Kellner M, Boysen N, Fliedner M, 2012 *How to park freight trains on rail–rail transshipment yards: the train location problem. OR spectrum* 34(3):535–561.

Lai YC, Barkan C, Önal H, 2008 *Optimizing the aerodynamic efficiency of intermodal freight trains. Transportation Research Part E: Logistics and Transportation Review* 44(5):820–834.

Mantovani S, Morganti G, Umang N, Crainic TG, Frejinger E, Larsen E, 2018 *The load planning problem for double-stack intermodal trains. European Journal of Operational Research* 267(1):107 – 119.

Shi T, Zhou X, 2015 *A mixed integer programming model for optimizing multi-level operations process in railroad yards. Transportation Research Part B: Methodological* 80:19–39.

SteadieSeifi M, Dellaert NP, Nuijten W, Van Woensel T, Raoufi R, 2014 *Multimodal freight transportation planning: A literature review. European Journal of Operational Research* 233(1):1–15.

Zhang C, Wan Yw, Liu J, Linn RJ, 2002 *Dynamic crane deployment in container storage yards. Transportation Research Part B: Methodological* 36(6):537–555.