Université de Montréal

Load Sequencing for Double-Stack Trains

par

William Perrault

Département d'informatique et recherche opérationelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures et postdoctorales
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Informatique

december 2018

Sommaire

Les trains à empilement double sont une composante majeure du réseau de transport ferroviaire pour les conteneurs intermodaux dans certains marchés comme celui de l'Amérique du Nord. Le séquençage du chargement représente un problème opérationnel auquel font face les opérateurs de grues dans les cours de chargement lorsqu'ils ont pour tâche de placer les conteneurs sur un train. Le séquençage du chargement consiste à trouver une séquence de mouvements permettant d'extraire les conteneurs des piles dans lesquels ils sont entreposés afin de les placer sur le train. Le séquençage du chargement est interrelié avec la planification du chargement, processus dans lequel des conteneurs sont assignés à des placements spécifiques sur les wagons, afin de former un plan de chargement pour guider le séquençage.

Le travail dans ce mémoire s'articule autour d'un article scientifique sur l'optimisation du séquençage du chargement pour les trains à empilement double. Dans cet article sont présentés des algorithmes basés sur la programmation dynamique, ainsi qu'une stratégie tirant avantage de plans de chargement développés afin de solutionner le séquençage pour des instances de chargement réalistes. Les résultats montrent que les heuristiques suggérées fonctionnent bien même pour des instances de grande taille. Ces dernières présentent une légère perte en qualité des solutions mais un temps d'exécution nettement inférieur aux méthodes exactes faisant défaut pour des instances de grande taille. L'analyse démontre également que l'utilisation de plans de chargement plus flexibles permet d'améliorer la qualité des solutions avec toutes les méthodes, ceci se faisant au coût d'un temps d'éxecution supérieur et l'absence d'une garantie de solution pour les heuristiques. Finalement, la planification et le séquençage simultané sont comparés avec l'approche successive utilisant les algorithmes developpés afin d'évaluer la performance relative des deux approches.

Mots-clés: Train, empilement double, conteneurs, programmation dynamique, séquençage du chargement, terminal intermodal, plan de chargement, heuristique

Summary

Double-stack trains are an important component of the railroad transport network for containerized cargo in specific markets such as the North American one. The load sequencing is an operational problem commonly faced in rail terminals by crane operators when tasked with loading containers on the railcars of a train. The load sequencing problem aims to find an efficient sequence of container retrievals in the storage yard, where containers are stored in piles while awaiting departure by train. Load sequencing is interrelated with load planning, the assignment of containers to specific locations on the train, forming a load plan which guides the load sequencing.

The work in this thesis is centered around a scientific paper on the optimization of load sequencing for double-stack trains. This paper proposes algorithms based on dynamic programming and a strategy leveraging the load plans, and assesses their performance in terms of computing time, tractability and solution quality on realistic instance sizes. The results show that the heuristics suggested to solve the load sequencing scale well for realistic instance size, managing to achieve a significantly reduced computing time with a small loss in solution quality compared to exact methods, which would often falter for larger instances. The analysis also illustrates how using a flexible load plan in the load sequencing significantly improves solution quality at the cost of greater computing requirements and lack of guaranteed solution for the heuristics. Finally, the paper compares the performance resulting from the successive application of load planning and sequencing with jointly performing the load planning and sequencing.

Keywords: Intermodal rail terminals, containers, load planning, load sequencing, dynamic programming, heuristics, double-stack, trains.

Contents

List of tables

List of figures

List of abbreviations

AAR	Association of American Railroad
BRP	block relocation problem
CN	Canadian National Railway Company
DP	dynamic programming
DT	double touches
E	exact
Ei	exact initialized
FO	found optimal
FS	found solution
FSLP	flexible ship loading problem
G	greedy
GC	gantry crane
ILP	integer linear programming
LBDC	lower bound of downstream cost
LBTC	lower bound of total cost
LCA	label-correcting algorithm
LIFO	last-in first-out
LP	load planning
LS	load sequencing
MIP	mixed integer programming
QC	quay crane
RS	reach stacker
S	semi-greedy
Si	semi-greedy initialized

SO	shown optimal
SP	stowage planning
SPP	ship planning problem
TC	transfer crane
TT	total touches

Acknowledgements

First and foremost, I would like to express my deep gratitude to my supervisor, Emma Frejinger, for allowing me the opportunity to work under her guidance. Her direction and support were invaluable in the outcome of this research, but her enthusiasm made working with her a thoroughly enjoyable experience. Thank you for everything.

I would like to extend special thanks to Éric Larsen, whose availability, patience and advice were invaluable during the entirety of my graduate research. I am truly grateful for his involvement. This thesis is very much a product of his help.

To the people at CIRRELT and Université de Montréal always willing to assist the graduate students, I express sincere thanks for his help.

To my family and friends, I cannot thank you enough for your encouragements and support. To my parents, Hanne and Louis, my brother Victor and Laurence, I am thankful for you always being there for me.

To David, my office partner, you were the sunshine of our windowless office. I truly appreciated our many discussions and your presence made the countless hours spent at CIRRELT profoundly pleasant.

Finally, to my friend Philippe Lacaille. Going through our Master's degree together was a blast and I am genuinely grateful for your friendship. See you on the course !

Chapter 1

Introduction

Trains are a widely used transport method for containerized cargo. By nature, they can carry much more containers inland than trucks while being cost effective compared to airplanes. Relative to single-stack trains, double-stack trains have an increased carrying capacity as they allow the placement of two containers, one on top of the other, rather than a single one. Double-stack trains are less common worldwide but are are extensively used in some large markets, like the North American one.

Load sequencing (LS) comes into play in railroad terminals when loading the containers on the railcars that form the departing train. Loading a train can be separated in two related problems, load planning (LP) and LS. LP aims to find an assignment of stored containers to specific positions on railcars. LS aims to find an efficient sequence in which to retrieve containers from the storage and place them on the train. LS is a complex task as there are significant constraints regarding the retrieval of containers from the piles in which they are stored. Only the topmost containers in piles are accessible and there may be additional constraints depending on the type of crane used. Solving LS to optimality is notably harder for double-stack trains as there are additional precedence constraints when loading containers on the railcars.

The current chapter provides a broad overview of LS for double-stack trains and the methods used in this research. It is structured as follows: Section 1.1 presents the context and relevant notions necessary to understand the LS problem, Section 1.2 situates the LS among the operations research literature it relates to, Section 1.3 provides an overview of technical concepts used in the proposed solution methods and Section 1.4 details the research contributions of the following work. The research paper on the LS for double-stack trains on

which this thesis is based is presented in Chapter 2. The research was focused on the North American context and motivated by a collaboration with an industrial partner in the field of railroad transportation. The paper presents methods that have been developed to solve LS for realistic instances, details the computation results and a comparative assessment of the methods' performance. Chapter 3 summarizes on the thesis' findings and suggests future research direction for the LS for double-stack trains.

1.1. Problem description

Containerization has established itself as the dominant means of carrying goods: "Roughly ninety percent of the world's goods are transported by sea with over seventy percent as containerized cargo" (Castonguay and Stone). Containers are particularly suitable for the transportation of cargo worldwide because they allow the goods to transition seamlessly between transport modes without additional handling. The sustained growth in the volume of transported containerized goods (United Nations Conference on Trade and Development, 2018) has been accompanied by a greater strain on the current shipping infrastructure to scale their operations with the increasing demand for intermodal freight transport. A direct consequence of this growth is the necessity for greater operational efficiency in intermodal terminals, the node where containers transition between the various transport methods.

Container ships are the predominant vector of transport because of their ability to carry a tremendous amount of containers, but they lack the penetration of the railway system or the trucking industry to efficiently forward the containers inland. All three means of transportation share similar operational challenges but they also present some unique characteristics that requires adapted solutions to their circumstances. For instance, trains in the North American market have the particularity of allowing the stacking of containers on trains for an increased transport capacity, an uncommon practice elsewhere. The scarcity of double-stack trains outside of North America means their operational concerns have not been widely studied, despite North America being an important market for railroad container transport.

To this end, this research focuses on one such operational challenge: the LS problem for double-stack trains.

1.1.1. Intermodal terminals

1.1.1.1. *Containers*

The underlying principle of *containerization* is the standardization of the containers. Those standards facilitate the shipping as the global transportation network aligns its practice accordingly. A container is defined by a few main characteristics: its dimension (length and height), weight, type and content. The following standardized dimensions are used worldwide (length - width - height, in ft. World Shipping Council):

- 20 standard (20' - 8' - 8'6")
- 40 standard (40' - 8' - 8'6")
- 40 high (40' - 8' - 9'6")
- 45 high (45' - 8' - 9'6")

In the North American market, two additional container dimensions are also used: the 48 high (48' - 8' - 9'6") and 53 high (53' - 8' - 9'6") containers. Dry cargo is the most common content type found in containers. Containers carrying dry cargo are referred to as *dry containers*. Some special containers can be used whose cargo has to be temperature controlled. These have to be placed close to a power source when transported. Specific containers are used for liquids or dangerous goods. The nature of their content forces such containers to be stored and handled with additional restrictions compared to others. The dimension of the different containers remains the same across content types. Most containers can be stacked as is the case on ships, in storage yard and some trains, but soft-walled containers or containers carrying dangerous goods are often exceptions in that regard. The complexity behind the management of the storage and transitions of containers between modes of transport largely contributes to the operational challenges faced in intermodal terminals.

1.1.1.2. *Terminal types*

The most common intermodal terminals are either inland terminals, the interface between two land-based transportation modes (generally trains and trucks), or maritime intermodal terminals, connecting a water-based transportation method with any other. Intermodal terminals vary according to their layout, the equipment used, the volume of traffic they can accommodate, but they share the following components: (i) a container yard where

containers are stored before moving on to the next mode of transportation and (ii) dedicated areas for the loading/unloading of containers on the trains, trucks, ships or others.

The review by Crainic et al. (2005) extensively describes intermodal transportation and covers most of of the challenges faced in intermodal terminals. The five following literature reviews explain a wide range of processes related to the transit of containers in the terminal and illustrate the complexity of operations in intermodal terminals: Carlo et al. (2014a) examine container transport in the yard. They detail the specialized equipment used, describe the different problems related to container movement and a classification of these problems. They also perform an extensive review of papers on the different subjects. The same authors also provide similar overviews on the subjects of seaside operations in the intermodal terminal (Carlo et al., 2015) and storage operations in the yard (Carlo et al., 2014b). Lehnfeld and Knust (2014) formulate an overview of operational problems for the storage and movement of containers in the yard, and Gharehgozli et al. (2015) review the latest technologies and operations research models used in terminal operations.

1.1.1.3. *Storage yard*

The storage yard is a dedicated area where containers are stored while awaiting transit to the next mode of transportation. There, containers are stored in series of piles, named *stacks*, where containers are piled one on top of each other to a maximum height. The height at which a container is stored in the stack is called a *tier* or *level* and a series of stacks placed in a row is called a *lot*. The general practice is to group containers in lots by length. Grouping by length allows container to be efficiently placed in lots since it allows for uniform stacks. In railroad terminals, containers are also grouped by *train block* in the lots. A train block is a sequence of railcars that travel as one between a given origin and destination pair. It is advantageous to regroup containers by block since intermodal terminals are venues where containers are consolidated, i.e. smaller shipments are regrouped into bigger ones. A train will normally be composed of multiple blocks. Having containers regrouped by block and designated as such on the train avoids the need for containers to be classified at every yard the train passes. It is only required at the destination.

Fig. 1.1. Segment of double-stack train composed of multiple railcars and stacked containers

1.1.1.4. *Railcars*

A train is inherently defined by railcars, its constitutive components. Like containers, they present their own differences and capabilities. Railcars are divided in platforms whose number range, in North America, between one and five. The number of platforms as well as the weight holding capacity, length and tare weight of each platform define a railcar's ability to accommodate containers. A *slot* corresponds to a space on which a container can be placed on a railcar platform. On double-stack trains, two containers can be stacked so each platform presents two slots, a *bottom* and a *top* slot. An example of a double-stack train formed of railcars with stacked containers is shown in Figure 1.1.

1.1.1.5. *Cranes*

Cranes are omnipresent in terminals due to their carrying capabilities but their type vary greatly between terminals. *Reach stackers* (RS) are special types of truck/crane hybrids. They are used to retrieve and load containers in the stacks but have the ability to move within the yard while carrying containers and load them directly on the train. This differs from overhead cranes, such as gantry cranes, that simply load and unload containers in the stacks. When using an overhead crane, the container is generally moved within the yard with the help of another vehicle. Both types of cranes can only retrieve the topmost container in a stack, but RS have a limited reach compared to overhead cranes. An example of a RS can be seen in Figure 1.2 while Figure 1.3 shows an illustration of a gantry crane.

Fig. 1.2. Reach stacker adjacent to multiple lots

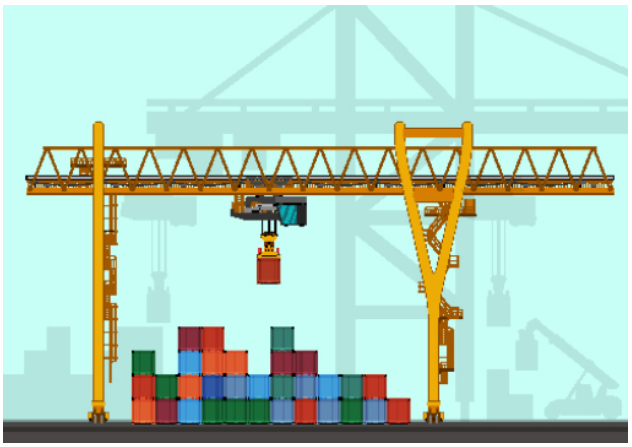

Fig. 1.3. Gantry crane retrieving a container in stacks

The ability of RS to access and handle containers in the stacks is a function of the distance from the base of the crane to the container, the weight of the target container as well as any obstruction. In practice, the weight capacity of a RS diminishes with increasing distance between the base of the reach stacker and the container. Some stacks configuration can restrict the access of containers with a RS as opposed to an overhead crane if the target container is not visible to the crane operator. A common case of a visual obstruction happens when a target container is the top container of a stack two containers high, but another two-container-high stack is placed between the target and the base of the reach stacker, as illustrated in Figure 2.2 by scenario II.

1.1.2. Load planning and load sequencing

1.1.2.1. *Load planning*

From the storage yard, departing containers are designated by the LP. In railroad terminals, the LP consists in designating departing containers and matching them with locations on the railcars that form the departing train. The operations research literature presents multiple papers on the LP as it is a common but complex problem also applicable to ship loading. In the LP, the output is a load plan where containers are assigned to only one or a set of slots.

The LP is a complex task since it is subject to a large number of technical restrictions. A technical document called the AAR guide (*Association of American Railroad guide*) presents every possible railcar and their feasible container loading schemes. These possible container configurations are called load patterns. For railcars with multiple platforms, the number of loading patterns can be overwhelming. The placement of temperature-controlled or dangerous containers involves specific placement considerations. Temperature-controlled containers generally have to be placed in specific platforms to be next to generators while dangerous goods need to be spaced out on the train. Double-stack trains present specific restrictions in regard to the relative weight of the stacked containers. To ensure the balance of the railcar, the center-of-gravity of the containers has to be under a certain height. This constraint is always satisfied if the heavier container of the two is positionned in the bottom slot. (See Mantovani et al., 2018, for a detailed breakdown of LP for double-stack container trains).

1.1.2.2. *Load sequencing*

The present section introduces the load sequencing, the problem this thesis and the research paper presented on Chapter 2 focus on.

The output of the LP, the load plan, assigns each departing container to at least one slot of a railcar from the departing train. Then, the LS problem corresponds to the task faced by crane operators: finding an efficient sequence of operations to load all containers on the train according to the specified load plan. LS is a complex problem for double-stack trains as there are precedence constraints when loading containers. A container designated for a bottom slot on a specific train platform has to be placed before the container destined to

the top slot. Moreover, some retrievals can be more favorable than other depending on the yard layout. Finally, admissible retrieval operations depend on the type of cranes used.

For single stack trains, every topmost accessible container in the stacks can be loaded directly on a train. For the double-stack trains, a container can only be loaded on a train in a single retrieval operation if both the container is accessible by the crane (for RS: no obstruction, admissible weight-distance pair and topmost container in a stack) and either the container is to be placed on a bottom slot, or is placed in a top slot with the bottom slot already filled on its destination platform. The challenge is therefore that the currently available moves, at least in terms of productive movements, depend both on the organization of containers in the stack and on the railcars rather than only the former. Unproductive movements, called *double handling* in this problem, arise when a top-destined container has to be retrieved before the bottom destined container because of availability issues. Two movements of the top container are then necessary in that case: one to move the container from the yard to the ground, and then from the ground to the train when the corresponding bottom has been loaded. Some double handling are unavoidable. When a top container is placed directly over the bottom container bound for the same platform, the top container has the be loaded in two touches. While double handling a container is always a possibility, some double handling can be avoided by loading containers on the train in the right order. As the efficiency of the loading depends on the number of touches, finding these sequences that bypass avoidable double touches is key to the LS.

Another factor that complexifies the LS is that there can be *intruding* containers in the lots. Containers unassigned in the loading process or awaiting a later departure can obstruct the retrieval of outbound containers. Moving an unassigned container, called *move-over*, is an unproductive movement that should be avoided if possible.

It is common for lots to be placed perpendicular to the train tracks in railroad terminals, meaning one access side to the containers is closer to the train than the other. To efficiently load containers on the train, it is preferable to favor the access side closest to the train to reduce the distance traveled by the RS. All the aforementioned factors explain why coordinating an efficient LS can be challenging, especially for a large number of containers.

1.2. Literature review

LS for double-stack trains represents a highly specific component of global shipping operations. The following literature review is meant to situate the LS and related problems in the broader context of container transportation. For a detailed review of the literature specifically related to this research, refer to the paper's literature review in Section 2.3.

1.2.1. Maritime terminals

Maritime terminals present several challenges similar to what is encountered in intermodal or railroad terminals. The review by Gharehgozli et al. (2015) presents a wide range of topics, technologies and problems related to maritime terminals. Specifically, they introduce the stowage planning (SP), the equivalent of the LP for container ships, as well as challenges related to the movement of containers: (i) the container relocation problem, also known as *block relocation problem* (BRP), (ii) the container stacking problem and (iii) the container pre-marshalling problem. Gharehgozli et al. (2015) also present various crane-specific technologies and problems such as the crane scheduling, which can be treated jointly in the stowage planning.

A detailed presentation of the problem and the challenges in the SP problem is presented by Ambrosino et al. (2015). Monaco et al. (2014) present many of the important contributions to the SP and classify multiple SP papers on the basis of the considered characteristics, objectives and settings. In the SP, the aim is to match departing containers with locations on the ship. A key difference in the SP versus the LP is that ships have significant requirements for balancing the many *holds* of the ship. Holds are dedicated compartments where the containers are stored in piles. This is a necessary requirement as ships present significant center-of-gravity loading constraints to ensure their stability during travel.

The *multi-port master bay planning problem* (MP-MBPP) encompasses the SP in a larger problem (Ambrosino et al., 2015). Container ships usually have a cyclical itinerary in which they visit multiple ports, where at every port they deliver and load containers destined for other ports in the route. To minimize the time spent in the different ports, it is important for containers to have been stored taking into account the ship's route. If not, containers bound for a later destination may have to be unloaded simply to free blocked containers having reached their destination and then reloaded, leading to unproductive movements. Those

unproductive rehandles are the main concern in the stowage planning. The complexity of this problem is increased by the scale of the instances, as ships transport very large number of containers.

LS is also a relevant problem for maritime terminals when loading ships, albeit with some distinctive differences. The papers by Kim et al. (2004), Álvarez Serrano (2006), Bian et al. (2016), Shen and Zhang (2015) and Ji et al. (2015) address the LS, focusing on maritime terminals. This subject matter is covered in-depth in the literature review in Section 2.3.

1.2.2. Stack management

Of the three aforementioned problems regarding the movement of containers in stacks (Gharehgozli et al., 2015), we first consider the BRP. At first glance, the BRP (Gharehgozli et al., 2015; Caserta et al., 2012; Lehnfeld and Knust, 2014) is very close to the LS. They both occur in a similar setting where multiple containers are stacked in a storage yard and have to be retrieved out of it. In both, only topmost containers can be picked up, although in the BRP the containers moved to free up a target container are placed in another stack rather than set in a specific area. The aim of the BRP is to optimize the retrieval sequence so as to minimize the number of moves, but when retrieving the containers in a specific order. This is an important distinction as the retrieval order is not set in LS. In the LS, availability of each container shifts as other containers get placed on the railcars because of the precedence constraints between top and bottom containers. Nevertheless, the BRP is related to this research as it concerns terminal management and optimization in stacks.

The loading problem (Lehnfeld and Knust, 2014; Gharehgozli et al., 2015) is centered around the placement of incoming items that have to be stored in a warehouse, yard or other storing space. The objective of the loading problem is to minimize the number of expected reshufflings. As the items to be retrieved are generally unknown, the loading seeks to construct the storage so as to avoid future unnecessary movements when accessing the items. The objective is to gradually organize the stacks efficiently so that future retrievals will be facilitated. In contrast, the BRP only considers outgoing objects.

The pre-marshalling problem (Lehnfeld and Knust, 2014; Gharehgozli et al., 2015) corresponds in the reorganization of objects in the storage space to avoid future reshuffles. The difference is that no incoming or outgoing items are considered during the shuffling process.

Generally, problems are tackled as mixed problems combining some or all of the above. Lehnfeld and Knust (2014) also detail different variants and constraints of the BRP, unloading and pre-marshalling as seen in the literature. They classify and review a large number of articles by problem, and bring forth the solution method and distinctive quality of each.

1.2.3. Railroad terminals

Trains are extensively covered by the literature on LP (Heggen et al., 2016), although there are fewer studies on double-stack trains. Lang et al. (2011), Upadhyay et al. (2017) Mantovani et al. (2018) consider the LP for double-stack trains. Mantovani et al. (2018) is of particular importance to this research as the LP used here originates from their work and they also consider the North American context. In terms of the LS for trains, Ambrosino et al. (2011) and Ambrosino et al. (2013) cover the simultaneous loading and sequencing of the containers for single stack trains but with marked differences. Finally, Ruf et al. (2018) tackle the joint loading and sequencing problem for double-stack trains for the North American context but they cannot solve the large realistic instances of interest in this thesis.

1.3. Methodological foundations

The following section presents notions important to understand the work in the thesis. First, Section 1.3.1 introduces *dynamic programming* (DP), why DP based methods are applicable to the LS and the variants on which the solution methods presented in 2.4.1 are based. Section 1.3.2 presents the integer linear programming (ILP) formulation for the LP presented in Mantovani et al. (2018) and used in the paper presented in Chapter 2.

1.3.1. Dynamic programming

The solution methods applied in the paper to solve the LS are based on dynamic programming, as discussed in Section 2.4.1. DP is a methodology that can used to solve problems that can be modeled as Markov decision processes (e.g. Bertsekas, 2017). A Markov decision process can be defined by four components: a state space, an action space, state transition functions and a cost function (or objective function). The LS can be formulated based on these components. The states of the system correspond to the different possible configurations of container locations in the stacks and on the railcars. The actions correspond to the movements of containers. Actions are defined according to a number of rules that represent

the constraints of the crane and allowed actions depend on the state of the system. For every state, the state transition functions supply the probabilities of transition to the next states for every action. The cost function provides the cost associated with the transition between states as a consequence of an action. Here, the cost function is defined which the objective of minimizing the number of container moves.

In this thesis, we consider the case of a deterministic LS so the state transition probabilities are degenerate. There is no uncertainty in the system, meaning that the outcome of every action at any given state is known. As a result, the deterministic LS can be viewed as a shortest-path problem, where nodes of the graph are the states of the LS problem and the distance corresponds to the cost between transitions. Finding the shortest route from the origin, where all containers are in the stacks, to the terminal state, where all the departing containers have been loaded on the railcars, is the optimal solution to the LS.

The solution methods applied in the research paper are based on label-correcting methods (Bertsekas, 2017, p.78) as presented in Algorithm 1 of Section 2.4.1. The idea behind this method is to assign labels to nodes as they are encountered while exploring the graph. The label corresponds to the cost from the origin to the node and the parent node along this shortest path is kept in memory. At first, all nodes are unlabeled and the labels are updated as nodes (previously labeled or not) are encountered. As the exploration through alternate paths proceeds, a label is updated if the new distance to the origin is smaller than the previous label indicated.

The methods presented in Section 2.4.1 use various node selection variants when deciding which nodes to select next when exploring the graph. One of these variants is equivalent to the A* algorithm (Bertsekas, 2017, p.87). At every node, a conservative estimate of the distance from the current node to the terminal node is generated. If the distance to the current node plus the conservative estimate is larger than the shortest known path from origin to terminal state, the exploration from this node is abandoned. This lower bound on the distance to the terminal node can also be kept in memory and used to select promising nodes for further exploration.

1.3.2. An integer linear programming formulation for load planning

This section presents the model and methodology based on the work by Mantovani et al. (2018) used to solve the LP. The necessary background information to understand the ILP model and how it relates to the LP is presented without technical details.

The importance of LP in this thesis stems from the interdependence between the LS and LP. In principle, the LP and LS may be solved simultaneously (Ruf et al., 2018). In practice, they are usually solved successively with the LP preceding the LS. It is the case in the research paper presented in Chapter 2. In this paper, when load plans are generated for the LS instances, they are based on the LP framework of Mantovani et al. (2018). They examine the LP problem for North American double-stack trains in a realistic setting that conforms to the objectives of the present research.

Mantovani et al. (2018) formulate the LP problem by means of an ILP. This ILP model formalizes the constraints and the objectives of the LP. In their model, unit costs are assigned to containers left behind and to railcars that are used. The objective is therefore to minimize the total cost of leaving containers behind and of using railcars. The ILP formulation specifies a number constraints:

(1) Each container is assigned to a maximum of one slot.
(2) A single loading pattern is assigned to each railcar.
(3) The number and types of containers assigned on each railcar has to correspond to the assigned loading pattern.
(4) The total weight of the containers loaded on each platform cannot exceed its maximum loading capacity.
(5) The weight of containers in top slots does not exceed an upper bound dependent on the weight and height of the bottom loaded container so as to respect the constraint regarding the centre-of-gravity.
(6) The total length of the container(s) in the bottom slot of a railcar must not exceed the platform length.
(7) Some containers are forbidden to be placed at certain positions in a sequence of railcars.
(8) Some containers have to be loaded on railcars where the carrying capacity exceeds a minimum value.

(9) Some containers cannot be placed in top slots.

(10) Some containers cannot be stacked or placed in top slots.

(11) Some containers have to be assigned to a platform within a certain distance of a specific container.

Constraints 7-11 are know as the technical loading restrictions. They encapsulate constraints relevant to the North American market that cover special situations, for instance containers with temperature-controlled cargo or dangerous content.

A commercial solver (in their case CPLEX) is used to solve the problem in reasonable time for realistic instances. In the generated load plan, containers are assigned to at most one slot on which they are to be placed. Such a load plan is said to be *specific*. In this work, we propose to use *general* load plans, mapping a container to possibly several slots, derived from the specific plan. These additional assignments must individually satisfy all relevant constraints. They are generated from the specific plan by exploring permutations of containers and verifying that the new placement still verifies the weight and center-of-gravity constraints.

1.4. Thesis contributions

This section outlines the contributions of the thesis.

Several contributions originate from the graduate student work on the LS problem leading to the production of this thesis. This work has been realized under the umbrella of the *CN Chair in Optimization of Railway Operations* whose aims are to improve solutions to the LP and LS problems and to assess the applicability of the resulting methods in a real industrial setting. The contributions originating from the activities during the graduate research can be divided in two categories, the work on the research paper presented in Chapter 2 and contributions related to the work on a technology transfer with the industrial partner. While this document focuses on the research component, the technology transfer component is also mentioned in the following.

Contributions were made in the form of facilitating the technology transfer to the industrial partner in the context of a pilot project. This consisted in assisting in the automatization of a data pipeline to process field information, both to match a specific input format to work with the requirement of the LP and LS framework and to match the output to the field

requirements of the operators. Contributions were also made by assisting other graduate students in using the framework and understanding the specifics of the LP and LS projects.

In regard to the paper presented in Chapter 2, there are multiple research contributions to be outlined. LS for double stack trains has not been covered extensively in the literature while being critical to important markets in railroad transportation, like the North American one. The paper provides methods to solve LS for instances featuring realistic size and constraints in accordance with the North American context. Five algorithms (i.e., two exact, two semi-greedy, one greedy) based on dynamic programming are formulated and the results are studied in-depth. Three node selection methods are considered for each algorithm. The results show that three proposed heuristics manage to solve realistic instances with a small penalty and greatly reduced computing time compared to the exact methods. Furthermore, these heuristics are shown to solve large instances that are intractable with exact methods. The methods are also compared with a joint solution approach for the LP and LS as opposed to the successive LP and LS suggested in the paper to assess the tractability, solution quality and relative computing time requirements between the two strategies. Moreover, the paper quantifies the gain resulting from leveraging the structure of the LP so as to improve the solution quality by using general load plans.

Chapter 2

Load sequencing for double-stack trains

Author's contribution

The first author's contribution to the paper spans all its components. A significant contribution was made by building on an existing project for the LP and LS. Most algorithms variants presented in the paper were added to the existing framework in addition to various data processing and analysis methods. An extensive and thorough literature review of intermodal operations was an integral component of the research. The testing instances presented in Section 2.5.1 were specifically designed for the paper and elaborate work was involved in adapting the joint LP and LS solution comparison instances from Ruf et al. (2018) presented in Section 2.5.2. Generating the necessary results and ensuring the integrity of the solutions was a crucial part of the research work. A major contribution also came in the form of writing the paper in an iterative process with the guidance and invaluable input of the other authors. Moreover, the presentation of the results and their subsequent analysis is a key aspect of the research contribution. As of right now, the paper is still awaiting additional modifications before being submitted.

Load sequencing for double-stack trains

by William Perrault, Eric Larsen and Emma Frejinger

This paper presents multiple heuristics based on dynamic programming (DP) to solve the load sequencing (LS) problem for double-stack trains. Finding the sequence in which to load the containers is significantly more complex for double-stack trains than the usual single stack. We also consider components of load planning (LP), the assignment of containers to the train, as it directly relates to LS. Solving LS is intractable for large and realistic instances. We model the LS problem and propose greedy, semi-greedy and exact resolution methods based on a label-correcting algorithm to solve LS instances with realistic constraints and number of containers. The multiplicity of optimal load plans, the assignment of containers to locations on the trains preceding the LS, is also leveraged to solve the instances on the basis of both specific and general plans.

The suggested solution methods are compared to a joint solution approach to the LP and LS, as opposed to the successive LP than LS, to assess the differences. Our results show that the greedy and semi-greedy variants are viable solutions to solve large instances quickly with little loss in solution quality when exact solutions falter. Leveraging the general load plans enhances the solution quality significantly but at the cost of a solution guarantee for the non-exact methods. We also quantify the loss when solving the LP and LS successively rather than jointly.

Keywords: Intermodal rail terminals, load planning, double-stack, trains, load sequencing, reach stacker, dynamic programming, heuristics

2.1. Introduction

Containerized transportation is a significant component of any transport system and essential to the national and global supply chains. Intermodal terminals have a key role in this context, needing to ensure an efficient and cost effective transfer between different modes of transportation.

This paper focuses on the optimization of intermodal rail terminal operations. More precisely, we consider the load sequencing (LS) problem for double-stack trains. Double-stack trains allow the stacking of two containers in height, a widespread practice in the North America and some other parts of the world. The LS aims to find the precise loading

sequence that minimizes the handling of containers out of the storage yard, where they are stacked in piles, and onto the train according to a predetermined load plan. The load plan matches containers from the yard with positions on the trains and is the output of the load planning (LP) problem.

Both the LS and the LP are noticeably more complex in the double stack case compared to single stack. An efficient sequencing entails minimizing the number of retrievals and the distance traveled by the crane, while accounting for the varying availability of containers in the piles based on the containers precedence in the load plan and crane restrictions. We consider a single crane, a reach stacker (RS) with realistic constraints in the LS. Also, we do not optimize the stacking of containers in the stacks when they cannot be loaded directly to the train. As this is the first approach to the LS for double-stack trains, a problem known to be hard, these assumptions are a first step to the more general case. Our objective is to solve realistic large instances in short computing time (seconds to minutes) so that the solutions can be used by crane operators.

The optimization of intermodal terminal operations is a well studied topic. While maritime terminals specifically have been highly represented in the literature (Carlo et al., 2015), the challenges they face regarding container handling and storage yard operations (Carlo et al., 2014a,b) share similarities with rail terminals (Boysen et al., 2013). Closest to our work are Kim et al. (2004), Álvarez Serrano (2006), Bian et al. (2016), Shen and Zhang (2015) and Ji et al. (2015) focusing on the LS problem in maritime terminals. Moreover, Ruf et al. (2018), to the best of our knowledge, is the sole study in the literature on the LP and LS for double-stack trains. They solve the joint loading and sequencing with an exact formulation but cannot solve instances of the large size and restricted computing budget we focus on. This paper addresses this gap by proposing multiple heuristics to quickly solve the successive LP and LS for realistic large instances.

In order to expand and explore the available trade off between computational speed, tractability on the one hand and quality of solutions on the other hand, we consider a setting where the LP and the LS are solved successively rather than jointly. First, solving the LP problem (using Mantovani et al., 2018) outputs a load plan generated on the basis of available railcars and containers with respect to realistic loading constraints. Then, this load plan is used in combination with the containers' precise location in the yard to solve

the LS. The structure of the problem is such that there can be multiple optimal load plans for a given set of railcars and containers. We leverage this fact to generate both specific and general load plans when solving the LP. A specific load plan specifies a single location for each container while a general plan allows for multiple possible placement options per container when applicable. For the LS, we propose a dynamic programming (DP) approach in the form of a label-correcting algorithm (LCA) to solve the problem. Using the LCA as a basis, we develop 5 forms of the algorithm with 3 variants each, for a total of 15 solution methods to tackle the LS. All 15 are used to solve instances presenting a specific and general load plans and of these 15, 9 are greedy based heuristics.

This paper makes several contributions. We propose three heuristics totalling 9 non-exact variants to solve the LS very quickly. We focus on the North American market and present results on realistic instances defined in collaboration with our industrial partner, Canadian National Railway Company. Our work presents extensive results on such realistic instances comparing the performance of 15 variants of the exact, greedy and semi-greedy methods based on the label-correcting algorithm. We also compare the successive LP and LS to the joint approach as seen in Ruf et al. (2018). The results show that leveraging a flexible load plan can greatly benefit the quality of the LS and the suggested heuristics manage to solve large instance in a very short time.

The remainder of the paper is structured as follows. Section 2.2 gives a detailed description of the problem and Section 2.3 presents an overview of the literature. In Section 2.4 we outline the proposed methodology and the numerical results are reported in Section 2.5. Finally, Section 2.6 concludes and gives directions for future research.

2.2. Problem description

Intermodal terminals are a key component of the worldwide container transport system. These terminals act as the link between different modes of transportation for containers. There, incoming containers are unloaded, stored and later forwarded to the the next leg of their travel. Operationally speaking, this presents a significant amount of complexity and coordination to ensure the terminals run efficiently. The most common types of transport modes at intermodals terminals are trucks, trains and ships. In this study, we focus on intermodal rail terminals. More precisely, we focus on the problem of moving containers from

a storage area and loading them onto double-stack railcars, a crucial problem for intermodal rail transportation in the North American market.

The problem corresponds to the task faced by crane operators when loading containers from piles in the storage area onto departing trains, the so-called LS problem. Specifically, crane operators seek to minimize the number of retrieval operations needed to successfully place all designated departing containers on the train. The key difficulty here being that the double-stack nature of the train forces that some containers are loaded before others on the train.

In this section we first present a typical yard layout with its relevant components. The characteristics of the cranes and their operations are then presented to exemplify their limitations and how these relate to the problem at hand. Finally, we present the LP of the train and the loading of containers from the storage to the train.

2.2.1. Containers, yard layout and railcars

Containers are defined by a few main characteristics: their dimension (in length and height), weight, type and content. Containers have standardized dimensions and can therefore be most commonly found in 4 variety (high containers having an extra foot in height compared to the standard 8'6"): 20 ft standard, 40 ft standard, 40 ft high, 45 ft high (World Shipping Council). The North American market also makes use of 48 ft high and 53 ft high containers. While containers are mostly *dry containers* (used to transport general cargo) some are used to carry cargo that has to be temperature-controlled, liquid cargo, or dangerous goods and are therefore subject to additional restrictions. These restrictions can affect their storage in the yard and the sequence in which they are loaded relative to standard cargo. In addition to the physical characteristics of the container and cargo, each container in the yard has a destination, an arrival time at the terminal and a due date at the destination.

The container yard is the area where containers are stored. It is generally placed strategically in the terminal as to limit the distance traveled between the different transport modes. To use the space in the yard economically, containers are stored on top of each other. Containers are grouped in a series of piles, named *stacks*, where containers of the same size are piled to a maximum height. The height at which a container is stored in the stack is called

a *level* (or tier) and a series of stacks in a same row is called a *lot*, see Figure 2.1 for an illustration.

Containers can be sorted into different lots according to various characteristics. Typically the containers have the same length. In rail terminals, containers can also be sorted according to train *block*. A block is a group of rail cars that are consolidated according to destination and are not split up (classified) until the destination is reached. Similarly, containers can be consolidated according to destination and loaded on railcars for a given block.

As seen in Figure 2.1, railcars in each block are placed on tracks adjacent to the lots. A railcar is characterized by its number of platforms. The term *slot* is used to designate a location where a container can be placed on a railcar platform. Since we consider double-stack trains, railcars with platforms that can accommodate two stacked containers have two slots, a *bottom* and *top* slot (20 ft containers fill a half slot each and are only placed on a bottom slots). Each railcar and platform has a given weight holding capacity, length and tare weight that characterizes a railcars' ability to accommodate containers. There are many different types of railcars in the North American market, all listed in the AAR guide, *Association of American Railroad guide*, which provides the instructions for feasible loadings (see Mantovani et al., 2018, for further details).

While Figure 2.1 presents a layout in which the lots are adjacent to a single track, some layouts allow the loading of railcars to be done on multiple adjacent tracks. In this case study, we consider railcars placed on a single track adjacent to the lots with the outbound containers. If the yard layout allows it, lots can be accessed both from the front side, next to the track where railcars are loaded, or the rear side. Obviously, the rear access is less preferable because the distance to collect and load the container on the railcars is greater.

2.2.2. Cranes, container handling and loading

To move the containers around the yard, stack them and load them on trucks, trains or ships, it is necessary to use specialized equipment. Cranes exist in many variety (e.g., Carlo et al., 2014a,b), the relevant distinction between them being that some cranes will only retrieve or deposit containers in stacks, while others are a sort of truck-crane hybrid and can also move containers around in the yard.

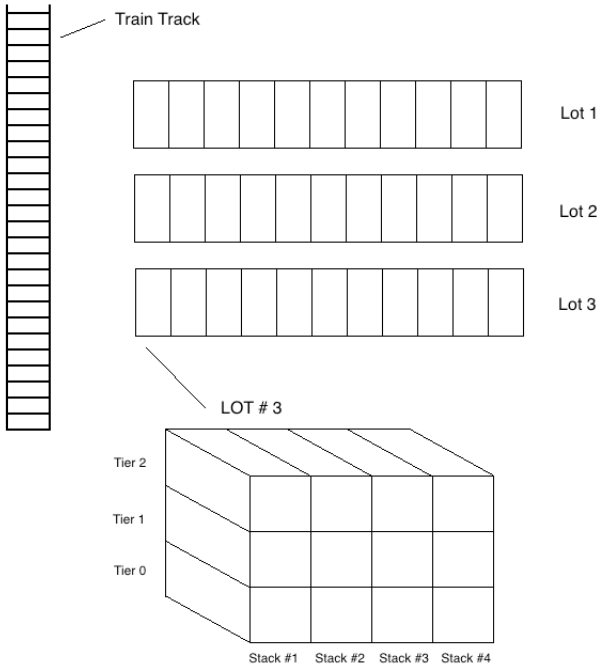

Fig. 2.1. Top view of three adjacent lots with a side view cut of a lot showing 3 tiers high and 4 stacks across

Gantry cranes (GC) are of the first type, they are usually placed over lots of containers and can access any stack in a lot, but are not adapted to move containers through the yard. Reach stackers belong to the second type and possess an arm like structure enabling them to reach in stacks to grab or place containers and can then carry containers within the terminal. Unlike GC, RS have a limited reach because of constraints such as the weight-distance from the reach stacker to the container. The further a container is in the lot, the lesser in weight it must be in order for the reach stacker to successfully retrieve it. Visibility also proves to be a challenge for RS since other containers can obstruct the vision of the crane operator when reaching in the lots from the side. What is true for both types of cranes is that they may only access the topmost containers within a stack. Figure 2.2 showcases typical scenarios of container visibility obstructions for reach stackers when reaching in the stacks. In scenario I, the topmost container in every stack is accessible. In scenarios II and IV, the reach stackers can't lift the topmost container in a further stack (2) when a closer stack (1) is higher or has the same height. The only exception to this obstruction is illustrated in the scenario III. For stacks of a single container, it is possible for the reach stacker to grab a the target container from a further stack (2), then lift it over a closer stack of same height (1).

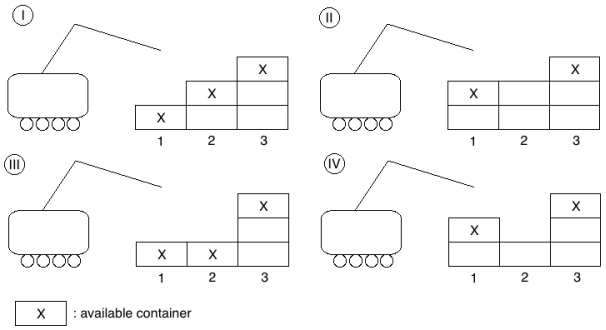

Fig. 2.2. Side view of a stacks showing 4 different scenarios of reach stacker and container obstructions.
X: indicates accessible containers provided that weight constraints are satisfied

For a container to be loaded on a railcar in a single retrieval operation, the container must be accessible in the stack by the crane and the container is to be assigned on a bottom slot, or a top slot with the bottom slot already filled on the destination platform. Unproductive movements, i.e., *double handling*, arise when a top destined container has to be retrieved before its bottom counterpart because of availability issues. Two movements of the top container are then necessary in that case, one to move the container from the stack to the ground, and then from the ground to the railcar when the corresponding bottom has been loaded. Unavoidable double handling occur when a top container is placed directly over the bottom container bound for the same platform. Many double handling can be avoided by judiciously choosing the retrieval order of the containers. For example, accessing a lot by the rear side can be a way to avoid double handling if no single touch retrieval can be attempted from the front side.

In this study we assume that the loading problem is solved in two successive steps: the LP and then the LS. The LP assigns outbound containers in the yard to slots on outbound railcars. These assignment decisions are subject to a plethora of operational and technical loading constraints (Mantovani et al., 2018). In this work we will focus on the constraints pertaining to container weights. The AAR guide, *Association American Railroad guide*, stipulates that the total weight on a platform cannot exceed its given weight holding capacity, moreover, the center of gravity of every loaded platform of the railcar has to be below or equal to 98 inches from the top of the rail (we refer to Mantovani et al., 2018, for more details).

We focus on a North American case study. In order to define the LS we make a number of assumptions deemed realistic in this context. First, we consider that a singe yard crane is used to load the containers for a given outbound block, moreover, the containers are sorted according to blocks in the yard. Therefore, we do not consider a crane scheduling problem and potential interference between cranes. Second, we consider that type of cranes used are reach stackers. Third, if a double touch occurs or if a container is moved to the side for loading on another block, we assume it is placed in an appropriate location in the yard, e.g., close to the outbound railcar or in another lot. That is, we do not optimize the stacking of containers in the yard.

In the following section we review literature relevant to the LS problem.

2.3. Literature review

Intermodal terminal operations have been extensively studied. We focus this literature review on LS and related problems and we refer the reader to dedicated literature reviews for broader overviews. For example, Carlo et al. (2014a) deal with different problems related to container handling in a yard, Carlo et al. (2015) cover seaside operations in intermodal terminals, Carlo et al. (2014b) focus on storage operations and Boysen et al. (2013) present railway yards. Most of the literature focuses on intermodal maritime terminals so we start by reviewing the related literature before focusing on the specificities of rail terminals.

Three different problems that deal with the movement of containers in stacks of maritime terminals are presented in a review by Lehnfeld and Knust (2014): (i) *pre-marshalling*, (ii) *loading*, and (iii) *unloading* also known as the *block relocation problem* (BRP). In the BRP containers are unloaded from stacks in a specific order while minimizing the number of retrieval operations. The pre-marshalling problem consists in the reorganization of objects in the storage space to avoid future reshuffles in subsequent retrievals. The difference is that no incoming or outgoing items are considered during the process. The loading problem concerns the placement of incoming containers in the stacks as to limit later retrieval operations, basically building the stacks with incoming containers. All of these as well as the LS can be considered specific cases of the block world problem from Gupta and Nau (1992) and Slaney and Thiébaux (2001).

Upon first examination, the BRP is very similar to the LS but there are a few key differences. In the BRP, there is a fixed retrieval order for the containers or class of containers. Blocking containers are also reshuffled in the stacks. In the LS, there is a dynamic quality to the retrieval order since top destined containers can only be loaded when a corresponding bottom destined container is placed on their common platform. Also, containers are not reshuffled in the stacks as in the BRP.

The *ship planning problem* (SPP) as described in the literature relates directly to the loading and sequencing of trains. Iris and Pacino (2015) present an overview of the different parts of the ship planning problem: (i) *operational stowage planning*, (ii) *load sequencing*, (iii) *equipment assignment* and (iv) *equipment scheduling*. One or several are addressed in most studies of maritime terminals. Iris et al. (2018) tackle the complete ship planning problem, which they call the *flexible ship loading problem* (FLSP), but the load sequencing is not explicitly optimized. Monaco et al. (2014) combine an overview as well as methodical approach to the classification of papers on stowage planning problems. The stowage planning problem is analog to the LP but with marked differences. The major concern in stowage planning are placing the containers so as to ensure the stability of the ship and minimizing reshuffling at the different ports on a ships' journey due to placing containers bound for different destinations stacked. Imai et al. (2006) study the SPP and include the minimization of the estimated number of rehandles in their objective. While this seems similar to the LS and the LP for trains, their loading sequence is derived directly from the placement of the containers on the ship and imposes a sequential retrieval like in the BRP. From there, the rehandles are estimated rather than directly minimized.

In regards to trains, research has focused on the LP (Heggen et al., 2016), which has been examined quite extensively. Lang et al. (2011), Upadhyay et al. (2017) and Mantovani et al. (2018) are alone in considering double-stack trains but they focus on the LP rather than the LS. Our work is based on the integer linear programming formulation (ILP) of Mantovani et al. (2018).

Ambrosino et al. (2011) cover simultaneous LP and LS for single stack trains but with significant differences to our work. They consider three scenarios for loading the train: (i) allowing double handling of containers in the yard but loading the railcars sequentially, (ii) no double handling of containers but with non-sequential loading of containers and (iii)

the sequential loading with double handling. Ambrosino et al. (2013) extend the work of Ambrosino et al. (2011) with a specific emphasis on the influence of storing strategies in lots and on the handling costs in the yard for the joint loading/sequencing problem. The setting of these papers creates a problem that differs from that addressed in the present case study. Their system creates a cost associated with not loading a container, rehandling a container in the yard and with a backtracking of the crane to load a wagon previously kept empty in the sequential loading. More importantly, they consider single-stack trains, which are easier to load as opposed to double-stack ones because of the absence of predecessors in the placement of containers (top after bottom). Unlike us, they consider GC whereas we consider the more complicated case of RS. The use of GC can be considered a variant on the use of RS with relaxed constraints as there are less restrictions on the allowed actions for gantry cranes.

Closest to our work are the following: Ruf et al. (2018) study the joint LP and LS for double-stack trains, Kim et al. (2004) and Álvarez Serrano (2006) tackle the joint placement-sequencing of containers on ships while Bian et al. (2016), Shen and Zhang (2015) and Ji et al. (2015) focus on the LS problem in maritime terminal. In the following, we describe these studies one-by-one in more detail.

Ruf et al. (2018) propose multiple ILP formulations to simultaneously solve the LP and LS for double-stack trains. Their research is based on the LP restrictions as detailed in Mantovani et al. (2018) and they consider reach stackers for the container loading, as in the present paper. They also compare their RS results with GC and present a model that includes the distance traveled by the cranes. The exact method cannot solve instances of the size we focus on here in reasonable time.

Kim et al. (2004) tackle the joint placement and sequencing of containers aboard ships, plus they consider container retrieval from lots with transfer cranes (TC) and quay crane (QC) scheduling jointly. TC are used for the container retrieval from the stacks while QC load the outbound containers on the ship. The authors present a realistic setting of maritime terminals with a broad range of practical constraints and objectives. The layout they consider divides the yard by bay holding containers of similar types. Their solution approach simultaneously solves two nested problems: (i) use filtered beam search to determine the number of containers to be picked up in each yard-bay, (ii) use standard beam search to determine the sequence of individual containers picked up in each yard-bay. The empirical

performance of this method is compared with metaheuristics. Contrarily to our work, there are no exact methods to act as a benchmark for the solution performance. Their representation of the yard also differs significantly with the reality of railroad terminals we consider. Finally, we explore the trade off in terms of computing time and solution quality of the successive versus joint LP and LS problem with multiple suggested algorithms specific to double-stack trains.

Álvarez Serrano (2006) consider the joint placement and sequencing problem in maritime terminals for RS. They aim to minimize a mixed criterion objective composed notably of the number of rehandles, ship stability, crane interference and crane travels. To solve this problem, they suggest to use a multi-start tabu algorithm and compare their results with an mixed integer programming (MIP) benchmark. Their work presents similarities to ours, especially the fact that they consider RS and their constraints. Note that their suggested MIP formulation is unable to solve instances large enough to present a meaningful comparison. The setting of maritime terminals and its constraints creates a different problem as opposed to the railroad terminals we consider.

Bian et al. (2016) aim to find a sequence in which to load the containers to minimize the number of rehandles starting from a specific stowage plan and yard layout. This specific version of the problem is the same as the LS for a specific load plan. They use a DP approach with two phases to solve the load sequencing. The first phase moves directly all containers that can be loaded in one retrieval (no blocking containers). Then, a DP algorithm is used to solve the rest of the problem with varying heuristic rules to determine the container relocations when blocking containers are encountered. The solution methodology shares some similarities with ours but unlike this study that focuses on GC, we consider the more complicated case of RS. Moreover, we generalize the load plan leveraging the specific structure of the double-stack train loading problem.

Shen and Zhang (2015) focus on the same problem as Bian et al. (2016) but they use a greedy randomized adaptive search procedure. They consider the so-called eyesight blockage, a visibility constraint related to the inability of the crane operator to effectively place containers on the ship when his vision is obstructed by other loaded containers. They also take into account setup times for cranes. Ji et al. (2015) solve the same problem as Bian et al. (2016) but they consider the relocation of blocking containers in the stacks and schedule both

single and *multiple* cranes to solve this problem making use of a genetic algorithm. Three different relocation strategies are also considered (nearest stack, lowest stack, "optimal") when using the genetic algorithm used to solve the problem. We focus on a more restricted version of the problem and make use of DP based heuristics hence avoiding black-box optimization algorithms.

In summary, except for Ruf et al. (2018), there is no other study in the literature that deal with the LP and LS for double-stack intermodal trains. Unlike Ruf et al. (2018) who propose an exact method, we focus on heuristics that can solve large problems in very short computing time. For this purpose we propose solving the LP and LS successively while allowing some flexibility in the load plan so as to improve the sequencing solution. We outline the methodology in detail in the following section.

2.4. Methodology

As explained above, the LS aims at finding a retrieval order for containers in a storage yard so as to place them onto a departing train. The state of the problem, X_k, can be completely described by the containers' current position at stage k in the yard and on the railcars within a block. The position of a container in the yard is defined by its respective lot, stack and level (tier) whereas its position on the railcar is defined by the respective slot in which it is placed. The state space for this problem increases significantly with the number of containers as the possible solution space increases dramatically. The origin state, X_0, corresponds to having no containers loaded on the railcars. The terminal state, X_N, is reached when all containers are loaded or there are no more free slots on the railcars. Note that N does not have a definitive value as it depends on the solution found. Containers are indexed by i and slots on the train by q.

Admissible actions are denoted $a_k \in A(X_k)$ and consist of the next available container moves. Thus, a container may be carried (i) directly to its destination slot (this slot is either in the lower position or in the upper position if the slot immediately below has already been filled), (ii) temporarily to ground, next to its destination platform, so as to be placed in its slot later on (if its slot is in the upper position and the slot immediately below has not been filled yet) and (iii) permanently on ground, where it will not interfere with future moves (if it is not bound to a slot and was simply interspersed with other containers in the lot).

The cost of each action is $c(a_k,X_k)$. The problem here is to minimize the total cost of the sequence of retrieval actions used to load all containers on the train:

$$\min \sum_{k}^{N} c(a_k,X_k) \quad \text{over all stages k}$$

This can be formulated as a dynamic programming problem as follows:

$$\begin{aligned} J(X_k) &= \min_{a_k \in A(X_k)} \left\{ c(a_k,X_k) + J(X_{k+1}) \right\} \\ X_{k+1} &= f(a_k,X_k) \end{aligned} \tag{2.4.1}$$

where $J(X_k)$ is the cost to go and f the deterministic transition function

The LS problem is specified by a number of initial settings that will condition the admissible actions at each step of the loading process. Among them, the load plan V_{iq} specifies in advance the possible destination slot(s), q, on the train for each container, i. The load plan, along with the position of the containers X_k, defines the allowed actions $A(X_k)$. In a *specific* load plan, each container is assigned to a single slot. In a *general* load plan, each container may be assigned to more than one slot where the set of allowed slot(s), Q_i, is determined in advance according to weight and center-of-gravity restrictions as in Mantovani et al. (2018). To derive the general load plan from the specific load plan, permutations of containers part of a same weight range are generated to define multiple possible slots for each containers, if applicable. The range of weights for the containers are defined as to ensure that the permutations will respect center of gravity constraints and weight limits on the railcars. More precisely, the AAR guide, *Association American Railroad guide* stipulates that the center of gravity of every loaded platform of the railcar has to be below or equal to 98 inches from the top of the rail. In practice this condition is always satisfied when the heaviest of two containers is placed in the bottom slot on the railcar platform. Therefore, containers can be interchangeable between slots if their respective weights still satisfy the weight constraints after permutation.

The specific load plan assigns a single slot for each departing container even though containers can possibly be assigned differently and still form an optimal load plan. The general load plan is based on the fact that there can be multiple possible optimal load plans

from a single set of containers and railcars. The alternative slots for each containers in a general plan are generated on the basis of a specific load plan by identifying, for each slot, other containers they could be matched with while respecting the center of gravity constraints and weight limits on the railcar. This creates an important flexibility that can be used to further optimize the retrieval process in the LS, as containers are not necessarily bound as top or bottom, or to a specific platform as in the specific load plan. For instance, situations that create unavoidable double handling, having a top destined container over its corresponding bottom in a stack for a specific load plan, may be avoided if the general plan allows an alternative placement of these relative containers.

A particularity of the general loading plan is that the alternative container placement is defined unilaterally. Meaning, when an alternative slot is designated for a container, it ensures that this container only will respect the constraints in the new configuration. It does not identify direct permutations between two containers, so for two pairs (container A, slot A) and (container B, slot B), the pair (container B, slot A) could be allowed while it is not the case for (container A, slot B). The general plan is an expanded form of the specific load plan, so exact solution methods can guarantee that, at worst, the general approach will load as many container as the specific one as it will consider all possibilities. This is not the case for heuristics so for the same instance, these solution methods may lead to loading less containers with a general plan than a specific plan depending on the chosen assignment. Note that the multiplicity of possible destination slots for each container in the general load plan is another contributor to the expansion of the action space with the growing number of containers.

The characteristics of the crane also influence the admissible actions in conjunction with the load plan and the state of the system. These characteristics concern reach and weight limits, as described in Section 2.2. For reach stackers, they translate into weight capacity as a function of elevation and horizontal distance to the base of the crane. For instance, the maximum allowed weight for a container move decreases with this horizontal distance. Additional restrictions concern access to containers that are hidden behind other containers. As an example, reaching for the top container in a further and lower stack than a closer one could cause reach issues and prevent the retrieval of the container in question.

2.4.1. Load sequencing algorithms

To solve problem (2.4.1), we apply variants of the *label-correcting algorithm* (e.g. Bertsekas, 2017, p.82). This algorithm iterates forward in a representative graph from the initial node where all containers are stored in their lot to a terminal node where all containers have been loaded. Essentially, a node comprises the state, a reference to its current parent, the action leading to it from its parent node and the minimal total cost upstream from the initial node. Note that the parent node is designated as the precursor along the current path with the lowest total cost upstream. The LCA is equipped with a set of open nodes *setOfOpenNOdes* and a set of visited nodes *setOfVisitedNodes*. At each iteration of the LCA, a node, *selectedNode*, is selected in *setOfOpenNodes* and a number of admissible actions leading to children nodes are explored. Promising children nodes are deposited in *setOfOpenNodes* and *setOfVisitedNodes* is updated if necessary. If a child node is terminal, then the best terminal node *bestSolutionAvailable* is updated if necessary. The LCA terminates when *setOfOpenNodes* is empty.

In the *exact* version of the LCA, every child resulting from an admissible action departing from the selected node is explored. While this method is theoretically guaranteed to lead to an optimal solution, the sizes of the state space and the set of admissible actions may be too large to allow for a solution in reasonable time, even for specific load plans and smaller instances. This problem is magnified with general load plans that exponentially increase the size of the action space because of the multiple positions for the loading of the containers. The reality of container yards being one of many containers and little time to plan the LS, it thus becomes necessary to find solutions efficiently, even if they are sub-optimal.

A *greedy* heuristic variant of the LCA is used to solve the LS much faster, at the possible expense of solution quality compared to the exact method. The greedy heuristic uses a logic very similar to the exact method. While the exact method explores all possible moves from the selected node, the greedy approach conducts a hierarchical review of all possible actions until an admissible action is found and explores only the latter, repeating this process until a solution is found. This hierarchy is meant to reflect the relative costs of the different actions built-in the problem. For instance, ceteris paribus, it is less costly to load containers from the side of the lot nearest to the train, the so-called front side, rather than from the farthest

side, which we call the rear side, and it is generally preferable to load containers slated for the bottom slots first as well.

The hierarchy of actions reviewed at each iteration of the greedy variant of the LCA is as follows:

(1) Shortest reach required from front side among direct moves to bottom slots.
(2) Shortest reach required from rear side among direct moves to bottom slots.
(3) Shortest reach required from front side among direct moves to top slots.
(4) Shortest reach required from rear side among direct moves to top slots.
(5) Shortest reach required from front side among containers bound to the upper slot of a platform whose lower slot has not been filled (container is moved to ground next to destination platform).
(6) Shortest reach required from rear side among containers bound to the upper slot of a platform whose lower slot has not been filled (container is moved to ground next to destination platform).
(7) Shortest reach required from front side among containers not bound to a slot (container is *moved over* to clear the way of the crane).
(8) Shortest reach from rear side among containers not bound to a slot (container is said to be *moved over* to clear the way of the crane).

The semi-greedy variant of the LCA aims to combine the efficiency of the greedy method with improved solution quality of the exact. This hybrid algorithm uses the same hierarchy as the greedy algorithm while exploring a range of actions. The difference here lies in the fact that the greedy algorithm explores only the children of the first possible action of the considered action type in the hierarchy, while the semi-greedy explores all children resulting from the possible actions from that action type. For example, the greedy method would explore only the children resulting from the first possible action for *containers to be placed on a bottom slot while accessing the containers from the front side of the lot* ((1) in the aforementioned list), while the semi-greedy would explore all the children resulting from the admissible actions that conform to this action type.

Building on the three main variants outlined above, initialization can also be used to assist the exact and semi-greedy algorithms when solving the LS. We consider that the algorithm finds a solution when all slots on the train have been filled. This means that

the heuristic methods can technically lead to unfeasible solutions if the assignment based on the general load plan causes some slots on the railcars to be left empty. Whereas the application of the greedy or the semi-greedy algorithms are not guaranteed to find a solution when paired with a general plan, solving with a specific load plan first and then initializing *setOfOpenNodes*, *setOfVisitedNodes* and *bestSolutionAvailable* with the resulting solution can guarantee at least one solution for the general plan for both methods. In our case, this initialization method with the solution resulting from the application of the greedy algorithm to the specific load plan was used in conjunction with the semi-greedy and exact algorithms for both the specific and general load plans.

For each algorithm, the selection of *selectedNode* at each step from *setOfOpenNodes* comes in 3 possible ways: (i) last-in first-out (LIFO), (ii) least lower bound of downstream cost (LBDC) as evaluated by an heuristic and (iii) least lower bound of total cost (LBTC) where the latter is equal to the sum of actual cost upstream from the initial node and LBDC. When each node is created while exploring the graph, in addition to the state, the reference to the parent, the action from the parent and the total cost upstream, a lower bound from this node to the final state is calculated by a heuristic and used to determine a lower bound of downstream cost and a lower bound of total cost. These can then be used as described above in the selection of *selectedNode* from *setOfOpenNodes*. Notice that in the exact method with LIFO, the children nodes are deposited in decreasing order of cost to explore the most promising first.

Algorithm 1 describes the LCA generically. Table 2.1 lists the main variants of the LCA that are considered. Each one of the latter is successively equipped with each one of the node selection methods listed in Table 2.2. The initialization as well as the children exploration process vary depending on the exact variant in Table 2.1. If the variant chosen is not initialized, *setOfOpenNodes* and *setOfVisitedNodes* are empty at first as well as *bestSolutionAvailable*. If it is, the latter sets are based on a greedy solution arising from a specific load plan. The nodes that form the solution are simply added to *setOfOpenNodes* and *setOfVisitedNodes* while *bestSolutionAvailable* describes the solution itself. Combining the possibilities from both the selection process and the algorithm, there are 15 variants overall (5 algorithms with 3 node selection processes) whose performance is presented below.

Notice that variant E equipped with node selection LBTC is equivalent to the algorithm known as A* (Bertsekas, 2017, p.91).

```
Set initialNode ;
Initialize setOfVisitedNodes & setOfOpenNodes;
Initialize bestSolutionAvailable;
while setOfOpenNodes not empty do
    pick selectedNode from setOfOpenNodes according to NodeSelect;
    from selectedNode generate children according to ChildGen;
    for each child of selectedNode considered do
        compute lowerBoundTotalCost = exact cost upstream of child + lower bound
         on cost downstream to final state of child;
        if lowerBoundTotalCost < cost of bestSolutionAvailable then
            search in setOfVisitedNodes for comparableNode whose state is identical
             with that of child ;
            childSelected = false;
            if comparableNode found then
                if exactCostUpStream of child found < exactCostUpStream of
                 comparable node then
                    remove comparableNode from setOfVisitedNodes;
                    add child to setOfVisitedNodes ;
                    childSelected = true ;
                end
            else
                add child to setOfVisitedNodes;
                childSelected = true ;
            end
            if child is terminal then
                if exactCostUpStream of child found < bestSolutionAvailable then
                    bestSolutionAvailable = child;
                end
            else if childSelected then
                add child to setOfOpenNodes;
            end
        end
    end
end
```

Algorithm 1: Generic Label-Correcting Algorithm (LCA)

2.5. Numerical results

This section assesses the empirical performance of the sequencing algorithms. Two settings are considered. First, we synthesize loading instances of increasing sizes and compare

Tab. 2.1. Variants of the LCA

Algorithm	Description	Generation of child node(s) (*ChildGen* in Algorithm 1)
E	exact	Child node generated for each admissible action.
G	greedy	Actions are reviewed hierarchically until an admissible action is found. Child node generated only for the latter.
S	semi-greedy	Actions are reviewed hierarchically until an admissible action is found. Child nodes generated for all similar actions.
Ei	exact initialized	Same as for exact.
Si	semi-greedy initialized	Same as for semi-greedy.

Tab. 2.2. Node selection from *setOfOpenNodes*

Selection method	Description (*NodeSelect* in Algorithm 1)
LIFO	Last node deposited in *setOfOpenNodes* is first taken out (LIFO).
LBDC	Least lower bound on downstream cost (LBDC) to terminal node. Lower bounds are evaluated with an heuristic.
LBTC	Least lower bound on total cost (LBTC) from initial to terminal node (exact cost upstream + LBDC).

the quality of the solutions resulting from the 15 variants of the sequencing algorithm. Second, we compare the quality of the sequencing solutions resulting from jointly solving the LP and LS problem as in Ruf et al. (2018) with that achieved when solving the LP and LS problems successively and applying the proposed variants of sequencing algorithms.

2.5.1. Comparative performance of load sequencing algorithms

The comparative performance of the 15 variants of the LCA described above was evaluated on testing instances with both general and specific load plans. Instances of cardinality 24, 40, 76, 120 containers were created with containers of length 40 ft and 53 ft for a 100% slot utilization of the railcars. An instance corresponds to a load plan paired with data detailing the position of the containers in the yard in terms of lot, stack and level. From one parent instances, 30 additional instances are generated by keeping the same load plan, but randomly permuting the locations of the containers 30 times, for 31 total instances per instance size.

Tab. 2.3. Testing instances in Section 2.5.1

# containers	# lots	max stack height	# stacks for each lot (40 ft, 53 ft)
24	1	3	(10)*
40	2	3	(5, 9)
76	2	3	(5, 21)
120	2	3	(9, 32)

* The 40 ft and 53 ft containers share a single lot. They are grouped by length in the other instances.

Tab. 2.4. Common cost structure in Section 2.5.1 for the testing instances as a function of access side and distance between reach stacker and container

side	front side			rear side			double touch front			double touch rear		
distance	0	1	2	0	1	2	0	1	2	0	1	2
cost	1.0	1.05	1.1	1.2	1.25	1.3	2.0	2.05	2.1	2.2	2.25	2.3

The 100% slot utilization version of the problem is the hardest version where all the containers in the lots are to be loaded. The slot utilization does not have an impact on the solution methods but when all slots are matched with containers, at the very least there are as many containers bound to top slots than for a lesser slot utilization. Having more containers destined for top slots complicates the LS as they impose the precedence constraint of having a container placed on the same platform in the bottom slot first to be loaded in one touch. In comparison, when not all containers from the lots are to be loaded on the train, unassigned containers can complexifiy the LS by blocking containers to be loaded, but the outcome of the LS also becomes more dependant on the LP. This is the case considered in Section 2.5.2.

The test instances are broken down as presented in Table 2.3. The instances of size 24 are composed of a single lot with both 40 and 53 ft containers while the other instances present two lots, one for each container size. In the column *# stacks*, the first number corresponds to the number of stacks for the lot with the 40 ft containers while the other pertains to the 53 ft.

The Table 2.4 presents the cost structure applied in the application of the sequencing algorithms. Single touches from the front side are less costly ([1.0, 1.05 ,1.10] from closest to farthest from the reach stacker, respectively), followed by rear side touches ([1.2, 1.25, 1.30]) and then double touches from the front side ([2.0, 2.05, 2.10]) and rear side ([2.2, 2.25, 2.3]). In the test instances (24-, 40-, 76-, 120-container), all containers in the yard are to be

placed on the train with no containers left in the lots after the sequencing, hence the 100% slot utilization. Therefore, the instances do not impose any *move over* operations. That is, moving a container from the stacks not intended to be placed on the train. This will generally happen when containers departing on different trains are placed in the same lot where the intruding containers have to be moved if they block departing containers.

Before analyzing the results, we note that the Java programming language was used for processing the data and for running and post processing the solutions on an Intel(R) Core(TM) i5-5300U, 2.30 GHz CPU processor equipped with 32 GB of RAM. The ILP optimization model for the LP was solved using a 32-bit version of the IBM ILOG CPLEX 12.6 solver.

In the current section, the results are shown in Tables 2.5, 2.6, 2.7, 2.8 and 2.9 presenting an identical structure. The two first columns indicate the type of algorithm and the node selection variant for the 15 considered options. The data relative to the sequencing time is shown by 4 metrics: the average, the 5^{th}, 50^{th} and 95^{th} percentile over the 31 instances considered per instance size. The cost is presented in the same way with the addition of the percentage gap between the optimal cost and the actual cost for each variant, averaged over the 31 instances. Then, *FS*, *FO* and *SO* represent respectively the number of instances for which a solution was found, for which an optimal solution was found and for which the solution found was shown to be optimal (this being only possible for exact methods). Afterwards, *TT*, *F*, *R* and *DT* show the average number of total touches, front side touches, rear side touches and double touches. Finally, *MNodes* represent the average maximum number of nodes in the open set while *VNodes* corresponds to the average number of unique visited nodes (respectively *setOfOpenNodes* and *setOfVisitedNodes* in Algorithm 1).

In the following analysis, the solutions of each algorithm variant is analyzed on the merit of the essential relevant characteristics to the realistic LS problem. Empirically, the number of touches and their access side represent the most important metrics and are therefore mentioned accordingly.

The 24-container instances are of particular interest because for these, the available resources were sufficient to use all algorithm variants, both with general and specific load plans. The results for the 24-container instances are presented in Table 2.5 when using a specific load plan and Table 2.6 with a general plan. All methods with the specific load

plan solve the instances very quickly on average. While none of the greedy (G), semi-greedy (S) and semi-greedy initialized (Si) variants solve to optimality, the loss in solution quality for these methods represents less than a full touch, as seen by the average total number of touches (TT) in Table 2.5. Compared with the results for the general load plan, even the worst variant of the algorithm performs, on average, better than the exact (E) and exact initialized (Ei) methods for the specific load plans. The G and S methods still find a solution very quickly and manage to improve by almost two the average number of double touches (DT in Table 2.6), increasing the number of front side touches, the most efficient move, compared to when solved with a specific load plan.

The greater computing time of the E and Ei methods with a general plan is apparent in the increased average run time and the fact that some instances are unable to be solved for the E variants. This is shown in Table 2.6 by the average time as well as the number of found solutions. The G methods finding a solution for only 11 of the 31 instances (see FS in Table 2.6) illustrates the lack of guarantee when using non-exact methods with a general load plan. Even without this guarantee, the S variants find as many solutions as the E methods at a much lower computing cost and for a fairly small penalty in solution quality. From Table 2.6, the S and E method both present 29 found solutions but with a two order of magnitude difference in average time in favour of the S and approximately a 0.2 touches difference in total touches averaged over all instances (0.82% additional total touches). The E and Ei methods can guarantee an optimal solution by exploring all possibilities with the general load plans. This does not apply to the non-exact methods and illustrates how the order of the list by which the general load plans are presented to the non-exact algorithms has a direct impact on the feasibility and/or the solution found. Note how this relates to the distinction between *found solution*, *found optimal* and *shown optimal*. As stated above, by nature, the exact methods can guarantee that the solution found is the optimal solution by exhausting all the possibilities (emptying *setofOpenNodes*) and finding the solution with the lowest cost. This does not apply to the non-exact methods so while they may find a solution that is optimal, hence found optimal, they can never guarantee that the solution found is optimal, meaning a shown optimal solution, like the exact methods.

Tab. 2.5. 24-container, specific load plan, 31 instances (300 s time limit)

algo		sequencing time (s)				cost				gap (%)	FS	FO	SO	TT	F	R	DT	MNodes	VNodes
		avg	5%	50%	95%	avg	5%	50%	95%										
E	LBDC	1.92E-02	3.00E-03	1.30E-02	4.15E-02	2.76E+01	2.51E+01	2.79E+01	2.92E+01	0	31	31	31	26.65	17.45	3.90	2.65	4.48E+01	4.37E+02
E	LBTC	2.04E-02	3.00E-03	1.10E-02	5.40E-02	2.76E+01	2.51E+01	2.79E+01	2.92E+01	0	31	31	31	26.65	17.45	3.90	2.65	2.26E+02	5.04E+02
E	LIFO	2.60E-02	6.00E-03	1.50E-02	1.00E-01	2.76E+01	2.51E+01	2.79E+01	2.92E+01	0	31	31	31	26.65	17.45	3.90	2.65	-	4.72E+02
Ei	LBDC	1.78E-02	3.00E-03	1.00E-02	5.85E-02	2.76E+01	2.51E+01	2.79E+01	2.92E+01	0	31	31	31	26.65	17.45	3.90	2.65	2.78E+01	4.26E+02
Ei	LBTC	1.57E-02	2.00E-03	8.00E-03	4.40E-02	2.76E+01	2.51E+01	2.79E+01	2.92E+01	0	31	31	31	26.65	17.45	3.90	2.65	1.79E+02	4.38E+02
Ei	LIFO	2.10E-02	4.50E-03	1.30E-02	6.55E-02	2.76E+01	2.51E+01	2.79E+01	2.92E+01	0	31	31	31	26.65	17.45	3.90	2.65	-	4.89E+02
S	LBDC	<0.001	<0.001	1.00E-03	2.00E-03	2.84E+01	2.56E+01	2.86E+01	3.17E+01	2.75	31	0	0	27.42	17.39	3.19	3.42	4.45E+00	2.98E+01
S	LBTC	<0.001	<0.001	1.00E-03	2.00E-03	2.84E+01	2.56E+01	2.86E+01	3.17E+01	2.78	31	0	0	27.42	17.32	3.26	3.42	3.48E+00	2.94E+01
S	LIFO	<0.001	<0.001	1.00E-03	2.00E-03	2.84E+01	2.56E+01	2.86E+01	3.17E+01	2.77	31	0	0	27.39	17.26	3.35	3.39	4.32E+00	3.46E+01
Si	LBDC	<0.001	<0.001	1.00E-03	2.00E-03	2.84E+01	2.56E+01	2.86E+01	3.17E+01	2.75	31	0	0	27.42	17.39	3.19	3.42	2.50E+01	3.31E+01
Si	LBTC	<0.001	<0.001	1.00E-03	2.00E-03	2.84E+01	2.56E+01	2.86E+01	3.17E+01	2.75	31	0	0	27.42	17.39	3.19	3.42	2.50E+01	3.31E+01
Si	LIFO	1.06E-03	<0.001	1.00E-03	2.50E-03	2.84E+01	2.56E+01	2.86E+01	3.17E+01	2.74	31	0	0	27.39	17.32	3.29	3.39	2.51E+01	3.37E+01
G	LBDC	<0.001	<0.001	<0.001	1.00E-03	2.84E+01	2.56E+01	2.86E+01	3.17E+01	3.00	31	0	0	27.48	17.26	3.26	3.48	1.48E+00	2.07E+01
G	LBTC	<0.001	<0.001	1.00E-03	1.00E-03	2.84E+01	2.56E+01	2.86E+01	3.17E+01	3.00	31	0	0	27.48	17.26	3.26	3.48	1.48E+00	2.07E+01
G	LIFO	<0.001	<0.001	<0.001	1.00E-03	2.84E+01	2.56E+01	2.86E+01	3.17E+01	3.00	31	0	0	27.48	17.26	3.26	3.48	1.48E+00	2.07E+01

[1] FS : found solution [2] FO : found optimal solution [3] SO : shown optimal solution [4] TT : avg. of total touches [5] F : avg. of front side touches
[6] R : avg. of rear side touches [7] DT : avg. double touches [8] MNodes : avg. maximum number of nodes in open set [9] VNodes : avg. number of unique visited nodes

Tab. 2.6. 24-container, general load plan, 31 instances (300 s time limit)

algo		sequencing time (s)				cost				gap (%)	FS	FO	SO	TT	F	R	DT	MNodes	VNodes
		avg	5%	50%	95%	avg	5%	50%	95%										
E	LBDC	8.12E+00	1.85E-02	5.05E-01	3.03E+02	2.50E+01	2.40E+01	2.48E+01	2.67E+01	0	28	28	28	24.43	21.43	2.14	0.43	2.93E+02	1.52E+05
E	LBTC	6.69E+01	3.45E-02	2.94E+00	3.60E+02	2.48E+01	2.40E+01	2.48E+01	2.62E+01	0	29	29	28	24.38	22.17	1.45	0.38	1.99E+04	2.40E+04
E	LIFO	2.39E+00	3.90E-02	5.67E-01	7.38E+01	2.49E+01	2.40E+01	2.48E+01	2.67E+01	0	29	29	29	24.41	21.52	2.07	0.41	-	3.25E+04
Ei	LBDC	1.80E+00	3.45E-02	3.58E-01	5.35E+00	2.49E+01	2.40E+01	2.48E+01	2.66E+01	0	31	31	31	24.39	21.68	1.94	0.39	1.92E+02	3.08E+04
Ei	LBTC	8.50E+01	2.20E-02	2.70E+00	3.60E+02	2.52E+01	2.40E+01	2.48E+01	2.94E+01	0.73	31	29	28	24.81	21.68	1.52	0.81	2.78E+04	3.31E+04
Ei	LIFO	2.15E+00	5.30E-02	3.68E-01	7.74E+00	2.49E+01	2.40E+01	2.48E+01	2.66E+01	0	31	31	31	24.39	21.68	1.94	0.39	-	2.65E+04
S	LBDC	2.31E-02	2.00E-03	1.10E-02	5.70E-02	2.54E+01	2.41E+01	2.52E+01	2.75E+01	2.12	29	0	0	24.69	21.24	2.07	0.69	3.42E+01	1.17E+03
S	LBTC	2.03E-02	2.00E-03	8.00E-03	6.80E-02	2.54E+01	2.42E+01	2.52E+01	2.75E+01	2.15	29	0	0	24.69	21.24	2.07	0.69	2.39E+02	9.95E+02
S	LIFO	2.42E-02	2.50E-03	1.30E-02	6.95E-02	2.54E+01	2.41E+01	2.52E+01	2.75E+01	2.12	29	0	0	24.69	21.24	2.07	0.69	3.45E+01	1.17E+03
Si	LBDC	3.52E-02	5.00E-03	1.90E-02	1.08E-01	2.55E+01	2.42E+01	2.52E+01	2.86E+01	2.60	31	0	0	24.81	21.03	2.16	0.81	3.64E+01	2.06E+03
Si	LBTC	3.35E-02	3.00E-03	1.10E-02	1.34E-01	2.55E+01	2.42E+01	2.52E+01	2.86E+01	2.62	31	0	0	24.81	21.03	2.16	0.81	2.78E+02	1.16E+03
Si	LIFO	2.61E-02	4.00E-03	1.50E-02	6.60E-02	2.55E+01	2.42E+01	2.52E+01	2.86E+01	2.60	31	0	0	24.81	21.03	2.16	0.81	5.73E+01	1.32E+03
G	LBDC	<0.001	<0.001	<0.001	1.00E-03	2.63E+01	2.43E+01	2.57E+01	3.02E+01	5.08	11	0	0	25.64	19.91	2.45	1.64	1.55E+00	2.01E+01
G	LBTC	<0.001	<0.001	<0.001	1.00E-03	2.63E+01	2.43E+01	2.57E+01	3.02E+01	5.08	11	0	0	25.64	19.91	2.45	1.64	1.55E+00	2.01E+01
G	LIFO	<0.001	<0.001	<0.001	1.00E-03	2.63E+01	2.43E+01	2.57E+01	3.02E+01	5.08	11	0	0	25.64	19.91	2.45	1.64	1.55E+00	2.01E+01

The average maximum number of nodes in the set of open nodes confers some insight in the relative computing time of LBDC versus LBTC. A direct correlation can be seen between the runtime and the average maximum number of nodes in *open* (i.e., corresponds to *SetOfOpenNodes* as used in Algorithm 1). The accumulation of nodes in the set increases the sorting time for the extraction of the subsequent node according to the selection process at each step. This explains for LBTC the larger average computing time compared to the LBDC. Note that this doesn't concern LIFO as there is no sorting involved in this node selection process.

The greater state space size when using a general load plan is apparent in the form of a lack of results. This approach was not used for the instances with more than 24 containers as the available resources failed to allow the exact methods and most semi-greedy methods to solve due to memory constraints. For this reason, Table 2.7, Table 2.8 and Table 2.9, for the 40-container, 76-container and 120-containers instances respectively, only present results for the G, S and Si methods. The additional instance sizes confirm the observations drawn from the 24-container instances. For 40 containers, E-LBTC and Ei-LBTC are the only exact method that cannot show the optimality for all 31 instances. For the three instance sizes, {40, 76, 120}, we observe a small growth in the delta of average number of total touches between the G and S methods, respectively 1.03, 2.25 and 3.97 (2.40%, 2.57%, 2.84% additional total touches). For 120 containers, the non-exact method variants solve all instances with an average time on the order of the milliseconds for the greedy and in seconds for the semi-greedy variants.

Tab. 2.7. 40-container, specific load plan, 31 instances (360 s time limit)

algo		sequencing time (s)				cost				gap (%)	FS	FO	SO	TT	F	R	DT	MNodes	VNodes
		avg	5%	50%	95%	avg	5%	50%	95%										
E	LBDC	3.42E+00	1.72E-01	1.63E+00	1.35E+01	4.43E+01	4.17E+01	4.43E+01	4.64E+01	0	31	31	31	42.90	31.45	5.65	2.90	1.48E+02	3.59E+04
E	LBTC	1.66E+02	2.47E+00	1.38E+02	3.60E+02	4.43E+01	4.17E+01	4.43E+01	4.64E+01	0	31	31	22	42.90	31.35	5.74	2.90	3.37E+04	5.31E+04
E	LIFO	5.19E+00	3.96E-01	3.74E+00	1.56E+01	4.43E+01	4.17E+01	4.43E+01	4.64E+01	0.0	31	31	31	42.90	31.42	5.68	2.90	-	3.62E+04
Ei	LBDC	3.37E+00	2.22E-01	1.98E+00	1.14E+01	4.43E+01	4.17E+01	4.43E+01	4.64E+01	0	31	31	31	42.90	31.45	5.65	2.90	9.78E+01	3.66E+04
Ei	LBTC	1.55E+02	1.51E+00	1.40E+02	3.60E+02	4.46E+01	4.17E+01	4.44E+01	4.73E+01	0.68	31	26	24	42.97	30.19	6.84	2.97	3.11E+04	5.03E+04
Ei	LIFO	5.42E+00	3.50E-01	4.03E+00	1.52E+01	4.43E+01	4.17E+01	4.43E+01	4.64E+01	0	31	31	31	42.90	31.42	5.68	2.90	-	3.81E+04
S	LBDC	4.68E-03	1.00E-03	3.00E-03	1.15E-02	4.52E+01	4.26E+01	4.52E+01	4.79E+01	2.08	31	0	0	42.97	28.45	8.58	2.97	1.65E+01	2.65E+02
S	LBTC	3.84E-03	1.50E-03	3.00E-03	9.00E-03	4.52E+01	4.26E+01	4.52E+01	4.79E+01	2.12	31	0	0	42.97	28.29	8.74	2.97	2.75E+01	1.94E+02
S	LIFO	9.65E-03	2.00E-03	7.00E-03	2.60E-02	4.52E+01	4.26E+01	4.52E+01	4.79E+01	2.07	31	0	0	42.97	28.42	8.61	2.97	1.55E+01	3.19E+02
Si	LBDC	5.39E-03	2.00E-03	4.00E-03	1.30E-02	4.52E+01	4.26E+01	4.52E+01	4.79E+01	2.10	31	0	0	42.97	28.42	8.61	2.97	4.10E+01	2.78E+02
Si	LBTC	4.97E-03	2.00E-03	4.00E-03	1.25E-02	4.52E+01	4.26E+01	4.52E+01	4.79E+01	2.11	31	0	0	42.97	28.32	8.71	2.97	5.09E+01	2.07E+02
Si	LIFO	6.94E-03	2.00E-03	5.00E-03	1.65E-02	4.52E+01	4.26E+01	4.52E+01	4.79E+01	2.07	31	0	0	42.97	28.42	8.61	2.97	4.56E+01	2.61E+02
G	LBDC	<0.001	<0.001	1.00E-03	1.00E-03	4.65E+01	4.39E+01	4.61E+01	4.98E+01	4.92	31	0	0	44.00	26.29	9.71	4.00	1.61E+00	3.45E+01
G	LBTC	<0.001	<0.001	1.00E-03	1.00E-03	4.65E+01	4.39E+01	4.61E+01	4.98E+01	4.92	31	0	0	44.00	26.29	9.71	4.00	1.61E+00	3.45E+01
G	LIFO	<0.001	<0.001	<0.001	1.50E-03	4.65E+01	4.39E+01	4.61E+01	4.98E+01	4.92	31	0	0	44.00	26.29	9.71	4.00	1.61E+00	3.45E+01

[1] FS : found solution [2] FO : found optimal solution [3] SO : shown optimal solution [4] TT : avg. of total touches [5] F : avg. of front side touches
[6] R : avg. of rear side touches [7] DT : avg. double touches [8] MNodes : avg. maximum number of nodes in open set [9] VNodes : avg. number of unique visited nodes

Tab. 2.8. 76-container, specific load plan, 31 instances (300s time limit)

algo		sequencing time (s)				cost				FS	TT	F	R	DT	MNodes	VNodes
		avg	5%	50%	95%	avg	5%	50%	95%							
S	LBDC	7.87E-02	1.30E-02	5.60E-02	1.71E-01	9.06E+01	8.49E+01	9.03E+01	9.43E+01	31	87.61	54.26	10.13	11.61	3.01E+01	1.77E+03
S	LBTC	6.00E-02	9.50E-03	4.70E-02	1.38E-01	9.06E+01	8.51E+01	9.03E+01	9.43E+01	31	87.65	54.10	10.26	11.65	9.92E+01	1.41E+03
S	LIFO	2.79E-01	3.75E-02	1.78E-01	6.31E-01	9.06E+01	8.49E+01	9.02E+01	9.43E+01	31	87.65	54.39	9.97	11.65	2.75E+01	1.00E+04
Si	LBDC	6.89E-02	1.60E-02	5.40E-02	1.64E-01	9.06E+01	8.49E+01	9.03E+01	9.43E+01	31	87.61	54.26	10.13	11.61	7.70E+01	2.43E+03
Si	LBTC	5.86E-02	1.55E-02	4.70E-02	1.32E-01	9.06E+01	8.51E+01	9.03E+01	9.43E+01	31	87.65	54.10	10.26	11.65	1.38E+02	1.44E+03
Si	LIFO	1.99E-01	2.25E-02	1.42E-01	4.46E-01	9.06E+01	8.49E+01	9.02E+01	9.43E+01	31	87.65	54.39	9.97	11.65	9.16E+01	8.02E+03
G	LBDC	2.39E-03	1.00E-03	2.00E-03	4.00E-03	9.28E+01	8.77E+01	9.32E+01	9.81E+01	31	89.90	52.32	9.77	13.90	1.65E+00	6.08E+01
G	LBTC	2.16E-03	1.00E-03	2.00E-03	6.00E-03	9.28E+01	8.77E+01	9.32E+01	9.81E+01	31	89.90	52.32	9.77	13.90	1.65E+00	6.08E+01
G	LIFO	2.03E-03	1.00E-03	2.00E-03	3.50E-03	9.28E+01	8.77E+01	9.32E+01	9.81E+01	31	89.90	52.32	9.77	13.90	1.65E+00	6.08E+01

Tab. 2.9. 120-container, specific load plan, 31 instances (300s time limit)

algo		sequencing time (s)				cost				FS	TT	F	R	DT	MNodes	VNodes
		avg	5%	50%	95%	avg	5%	50%	95%							
S	LBDC	1.07E+00	2.09E-01	7.97E-01	2.92E+00	1.44E+02	1.38E+02	1.44E+02	1.49E+02	31	139.68	87.06	13.26	19.68	5.05E+01	1.41E+04
S	LBTC	1.63E+00	1.38E-01	6.61E-01	7.47E+00	1.44E+02	1.39E+02	1.44E+02	1.49E+02	31	139.71	87.03	13.26	19.71	6.43E+02	1.24E+04
S	LIFO	6.33E+00	1.04E+00	4.09E+00	1.41E+01	1.44E+02	1.38E+02	1.44E+02	1.49E+02	31	139.68	87.26	13.06	19.68	4.33E+01	1.55E+05
Si	LBDC	1.05E+00	1.50E-01	8.49E-01	2.97E+00	1.44E+02	1.38E+02	1.44E+02	1.49E+02	31	139.68	87.03	13.29	19.68	1.21E+02	2.41E+04
Si	LBTC	1.72E+00	1.62E-01	7.26E-01	7.67E+00	1.44E+02	1.39E+02	1.44E+02	1.49E+02	31	139.71	87.03	13.26	19.71	6.89E+02	1.24E+04
Si	LIFO	5.92E+00	5.84E-01	3.99E+00	1.35E+01	1.44E+02	1.38E+02	1.44E+02	1.49E+02	31	139.71	87.48	12.81	19.71	1.51E+02	1.44E+05
G	LBDC	6.84E-03	3.00E-03	4.00E-03	2.65E-02	1.48E+02	1.40E+02	1.49E+02	1.55E+02	31	143.68	84.03	12.29	23.68	1.84E+00	9.43E+01
G	LBTC	4.94E-03	3.00E-03	4.00E-03	1.30E-02	1.48E+02	1.40E+02	1.49E+02	1.55E+02	31	143.68	84.03	12.29	23.68	1.84E+00	9.43E+01
G	LIFO	5.03E-03	3.00E-03	4.00E-03	1.20E-02	1.48E+02	1.40E+02	1.49E+02	1.55E+02	31	143.68	84.03	12.29	23.68	1.84E+00	9.43E+01

[1] FS : found solution [2] FO : found optimal solution [3] SO : shown optimal solution [4] TT : avg. of total touches [5] F : avg. of front side touches
[6] R : avg. of rear side touches [7] DT : avg. double touches [8] MNodes : avg. maximum number of nodes in open set [9] VNodes : avg. number of unique visited nodes

The experiments on the set of testing instances presented above grant insight on the trade-off between computing time, solution quality and tractability when solving the LS with both specific and general plans. The results discussed above show that the proposed heuristics are able to solve the LS problem with a small decrease in solution quality and a much smaller computing time compared to the exact methods. The heuristic methods remain tractable even for larger instances with a specific plan where the exact methods exhaust the available computing resources, showing the heuristics' scalability. The general plan is shown to improve the solution quality at the cost of greater computing time for all exact and semi-greedy methods. A drawback to the general plan approach is that it nullifies the guarantee of a solution for the non-exact methods. In conjunction with the exact methods, the node selection methods rank as follows in regards to computing time, tractability and solution quality: LBTC clearly being the worst performing, LIFO and then LBDC by a narrow margin. This distinction does not generalize when applied to the other variants. Finally, the initialization does not clearly improve the solution method in terms of computing time or quality but it guarantees a solution.

2.5.2. Comparing with joint planning and sequencing

In Section 2.5.1, we compared the performances of the 15 variants of the LCA in the LS problem when supplied with predetermined specific and general load plans. Decoupling LP and LS in this fashion saves computational resources and may circumvent the intractability of the joint LP and LS that may occur even for loading problems of modest size (Ruf et al., 2018). In this section, we assess the loss in sequencing performance incurred when the LP and LS solutions are performed successively rather than jointly, whenever the latter is feasible. Hence, we reexamine the instances whose LP and LS could be solved jointly by Ruf et al. (2018). For each of these instances, we first generate specific and general load plans with the computational apparatus developed in Mantovani et al. (2018). Then, based on a sequencing cost structure identical to that in Ruf et al. (2018), we apply the 15 variants of the LCA described in Section 2.4.1 to these load plans and compare the sequencing performance resulting from this separate application of LP and LS with that achieved through a joint application in Ruf et al. (2018).

Tab. 2.10. Yard layout properties for Section 2.5.2 instances

train length (ft)	# containers	# lots	max stack height	# stacks
200	15	2	3	(3, 2)
667	50	6	3	(3, 3, 3, 3, 3, 2)

Tab. 2.11. Common cost structure in Section 2.5.2 as a function of access side

side	single touch		double touch		move over	
	front side	rear side	front	rear	front	rear
cost	0	60	80	140	80	140

The testing instances obtained from Ruf et al. (2018) result from crossing 5 different train compositions and 10 different sets of containers and locations in the yard. This is done for trains of 200 ft and 667 ft lengths and thus results in a total of 100 distinct problem instances. (The 5 train composition give rise in the 200 ft case to 5 load plans assigning respectively {8, 10, 15, 10, 10} containers. For the 667 ft case: {30, 32, 30, 38, 30}.) These instances are used to compare the gap between solving jointly the LP and LS compared to solving them successively. The LCA variants are used to solve the LS in this two part process and are compared here with solutions from an ILP optimization model for the joint problem. While the containers in the yard change location and characteristics among the 10 containers files mentioned above, the yard layout is fixed as shown in Table 2.10. Among the 50 instances of a given length, the differences in numbers of loaded containers stem from the 5 different train composition since different railcars combinations can accommodate more or less cargo. In these instances, there are more containers in the lots than slots on the train so some instances might require move over operations. This is a significant disadvantage to the successive solution since some containers selected to be placed on the train can be located in configurations that are difficult to access in the lots.

The Tables 2.12, 2.13 and 2.14 present the results for the 200 ft testing instance and Tables 2.15, 2.16 and 2.17 for the 667 ft. Their structure is similar to the results tables presented in Section 2.5.1 with the addition of three elements : Mo, the average number of move over operations, LPT, the load planning time (avg. time to generate the load plans) and TTm, the average total time which corresponds to the sum of the sequencing time and the load planning time. Most importantly, the aforementioned tables present an additional algorithm, the joint solution ($JSol$), as shown in the first row.

It is important to note that these experiments present a different cost structure as opposed to instances from the previous section. The Table 2.11 presents the cost structure for the testing instances originating from Ruf et al. (2018). There is a significant penalty for rear side touches (60), double touches (front: 80, rear: 140) and move-overs (front: 80, rear: 140) compared to the front side touches (0).

The joint solution was obtained using an ILP optimization model to jointly solve the LP and LS. This method is advantageous since it leverages the position of the container in the loading decision while this is absent in the successive process. In the successive method, the time to generate load plans in both the general and specific case is very short, but not insignificant in regard to the sequencing time for most of the methods.

For the sake of clarity, the results for the 200 ft and 667 ft instances seen in Tables 2.12, 2.13, 2.14, 2.15, 2.16, 2.17 display the cost as in the previous Section 2.5.1. However, the following analysis is done on the basis on the number of touches, their access side (front or rear) and their nature (standard, double touch, move over). This is the case as they are the relevant metrics for the realistic LS while the cost structure from Ruf et al. (2018) instances were chosen to elicit a certain type of behaviour from the algorithm but without a clear operational interpretation.

We first consider Table 2.12, presenting the results for all variants of the LCA using specific plans derived from the optimal solutions of the joint solution. First, they attest to the proper alignment of the solution methods from the LS and the joint solution as the exact methods present the same cost and breakdown of touches as the joint solution. Second, these results confirm the conclusions from Section 2.5.1 in regards to the performance of the heuristics compared with the exact methods. Table 2.12 clearly displays the small penalty in terms of total touches when using G, S, Si with a maximum delta of 0.04 touches (0.38% additional total touches) between these and the E, Ei solutions. All methods find the optimal solution in 45 out of 50 instances or more, except for S-LBTC and G-LBTC who find the optimal solution in 35 and 33 cases respectively. Interestingly, the initialization here benefits the semi-greedy variant, increasing the number of found optimal solutions.

Tables 2.13 and 2.14 summarize the same information for the successive LP and LS. Those results confirm the hypothesis that there is significant loss in solution quality when solving successively compared to jointly. Precisely, this difference is worth a minimum of 3.31

total touches on average when comparing the exact methods with the joint solution using a specific plan, half a touch less with a general plan (31.1% and 26.4% additional touches respectively). It is noteworthy that the exact methods show how 2.5 of these additional 3.31 average touches are caused by move-overs when compared to the joint solution. The load planning time is very short for both types of plans but still constitutes the majority of the total solution time for the heuristics. The heuristics perform very quickly for both types of plans solving all instances. Again, there is a small penalty in the average number of touches compared to the exact methods when using the heuristics. This is at the expense of a much worst cost though since the heuristics incur more penalized moves compared to the exact methods (i.e., rear-side touches and double-touches). The general plan proves beneficial to the solution quality for all variants but with a significant computing time penalty for the exact methods.

Tab. 2.12. 200 ft instances from Section 2.5.2, load sequencing using optimal load plans from joint solution

algo		sequencing time (s)				cost				FS	FO	SO	TT	F	R	DT	Mo	MNodes	VNodes	LPT	TTm
		avg	5%	50%	95%	avg	5%	50%	95%												
JSol	-	-	-	-	-	8.20E+00	-	-	-	50	50	50	10.66	10.56	0.02	0.02	0.04	-	-	-	4.89E+00
E	LBDC	1.54E-03	<0.001	1.00E-03	4.55E-03	8.20E+00	1.00E+00	2.00E+00	4.59E+01	50	50	50	10.66	10.56	0.02	0.02	0.04	2.59E+01	3.88E+01	-	-
E	LBTC	1.20E-03	<0.001	1.00E-03	3.00E-03	8.20E+00	1.00E+00	2.00E+00	4.59E+01	50	50	50	10.66	10.56	0.02	0.02	0.04	2.63E+01	3.80E+01	-	-
E	LIFO	1.14E-03	<0.001	1.00E-03	3.00E-03	8.20E+00	1.00E+00	2.00E+00	4.59E+01	50	50	50	10.66	10.56	0.02	0.02	0.04	-	4.09E+01	-	-
Ei	LBDC	<0.001	<0.001	1.00E-03	2.55E-03	8.20E+00	1.00E+00	2.00E+00	4.59E+01	50	50	50	10.66	10.56	0.02	0.02	0.04	1.17E+01	1.55E+01	-	-
Ei	LBTC	<0.001	<0.001	1.00E-03	2.00E-03	8.20E+00	1.00E+00	2.00E+00	4.59E+01	50	50	47	10.66	10.56	0.02	0.02	0.04	1.33E+01	1.59E+01	-	-
Ei	LIFO	<0.001	<0.001	1.00E-03	2.00E-03	8.20E+00	1.00E+00	2.00E+00	4.59E+01	50	50	47	10.66	10.56	0.02	0.02	0.04	-	1.77E+01	-	-
S	LBDC	<0.001	<0.001	<0.001	1.00E-03	1.78E+01	1.00E+00	2.50E+00	1.23E+02	50	47	0	10.66	10.40	0.18	0.02	0.04	3.92E+00	1.27E+01	-	-
S	LBTC	<0.001	<0.001	<0.001	1.00E-03	3.22E+01	1.00E+00	3.00E+00	1.23E+02	50	35	0	10.66	10.16	0.42	0.02	0.04	3.62E+00	1.14E+01	-	-
S	LIFO	<0.001	<0.001	<0.001	2.00E-03	1.78E+01	1.00E+00	2.50E+00	1.23E+02	50	47	0	10.66	10.40	0.18	0.02	0.04	3.70E+00	1.31E+01	-	-
Si	LBDC	<0.001	<0.001	1.00E-03	2.00E-03	8.20E+00	1.00E+00	2.00E+00	4.59E+01	50	50	0	10.66	10.56	0.02	0.02	0.04	1.16E+01	1.43E+01	-	-
Si	LBTC	<0.001	<0.001	1.00E-03	2.00E-03	1.42E+01	1.00E+00	2.50E+00	8.10E+01	50	47	0	10.66	10.46	0.12	0.02	0.04	1.20E+01	1.38E+01	-	-
Si	LIFO	<0.001	<0.001	1.00E-03	1.55E-03	1.42E+01	1.00E+00	2.50E+00	8.10E+01	50	47	0	10.66	10.46	0.12	0.02	0.04	1.17E+01	1.40E+01	-	-
G	LBDC	<0.001	<0.001	<0.001	1.00E-03	2.10E+01	1.00E+00	2.50E+00	1.44E+02	50	45	0	10.70	10.36	0.18	0.06	0.04	1.38E+00	9.82E+00	-	-
G	LBTC	<0.001	<0.001	<0.001	1.00E-03	3.54E+01	1.00E+00	3.00E+00	1.44E+02	50	33	0	10.70	10.12	0.42	0.06	0.04	1.38E+00	8.80E+00	-	-
G	LIFO	<0.001	<0.001	<0.001	1.00E-03	2.10E+01	1.00E+00	2.50E+00	1.44E+02	50	45	0	10.70	10.36	0.18	0.06	0.04	1.36E+00	9.66E+00	-	-

[1] FS : found solution [2] FO : found optimal solution [3] SO : shown optimal solution [4] TT : avg. of total touches [5] F : avg. of front side touches [6] R : avg. of rear side touches
[7] DT : avg. double touches [8] MNodes : avg. maximum number of nodes in open set [9] VNodes : avg. number of unique visited nodes [11] LPT : avg. load planning time
[12] TTm : avg. total time time (LPT + sequencing time) [13] JSol : joint solution

Tab. 2.13. 200 ft instances from Section 2.5.2, successive load planning and sequencing, specific load plans

algo		sequencing time (s)				cost				FS	FO	SO	TT	F	R	DT	Mo	MNodes	VNodes	LPT	TTm
		avg	5%	50%	95%	avg	5%	50%	95%												
JSol	-	-	-	-	-	8.20E+00	-	-	-	50	50	50	10.66	10.56	0.02	0.02	0.04	-	-	-	4.89E+00
E	LBDC	8.18E-03	1.00E-03	5.00E-03	2.86E-02	3.53E+02	1.21E+02	3.32E+02	5.62E+02	50	0	0	14.04	8.54	1.18	0.88	2.56	3.16E+01	3.29E+02	1.58E-01	1.66E-01
E	LBTC	1.37E-02	1.45E-03	1.00E-02	4.36E-02	3.53E+02	1.21E+02	3.32E+02	5.62E+02	50	0	0	13.98	8.50	1.22	0.88	2.50	2.09E+02	4.23E+02	1.58E-01	1.72E-01
E	LIFO	6.22E-03	2.00E-03	5.00E-03	1.79E-02	3.53E+02	1.21E+02	3.32E+02	5.62E+02	50	0	0	14.04	8.54	1.18	0.88	2.56	-	3.32E+02	1.58E-01	1.64E-01
Ei	LBDC	6.56E-03	1.45E-03	5.00E-03	1.66E-02	3.53E+02	1.21E+02	3.32E+02	5.62E+02	50	0	0	14.04	8.54	1.18	0.88	2.56	2.02E+01	3.20E+02	1.58E-01	1.65E-01
Ei	LBTC	1.21E-02	1.45E-03	8.50E-03	3.68E-02	3.53E+02	1.21E+02	3.32E+02	5.62E+02	50	0	0	14.04	8.54	1.18	0.88	2.56	1.92E+02	3.91E+02	1.58E-01	1.70E-01
Ei	LIFO	6.46E-03	1.45E-03	4.00E-03	1.56E-02	3.53E+02	1.21E+02	3.32E+02	5.62E+02	50	0	0	14.04	8.54	1.18	0.88	2.56	-	3.42E+02	1.58E-01	1.64E-01
S	LBDC	<0.001	<0.001	1.00E-03	1.55E-03	4.47E+02	1.60E+02	4.33E+02	7.14E+02	50	0	0	13.92	7.02	2.74	0.84	2.48	5.62E+00	3.51E+01	1.58E-01	1.58E-01
S	LBTC	<0.001	<0.001	1.00E-03	2.00E-03	4.49E+02	1.69E+02	4.43E+02	7.14E+02	50	0	0	13.92	6.98	2.76	0.86	2.46	7.40E+00	3.51E+01	1.58E-01	1.58E-01
S	LIFO	<0.001	<0.001	1.00E-03	2.00E-03	4.47E+02	1.60E+02	4.33E+02	7.14E+02	50	0	0	13.92	7.02	2.74	0.84	2.48	5.78E+00	3.81E+01	1.58E-01	1.58E-01
Si	LBDC	<0.001	<0.001	1.00E-03	2.00E-03	4.48E+02	1.60E+02	4.43E+02	7.14E+02	50	0	0	13.92	7.00	2.74	0.86	2.46	1.47E+01	3.67E+01	1.58E-01	1.58E-01
Si	LBTC	<0.001	<0.001	1.00E-03	1.55E-03	4.48E+02	1.60E+02	4.43E+02	7.14E+02	50	0	0	13.92	7.00	2.74	0.86	2.46	1.58E+01	3.69E+01	1.58E-01	1.58E-01
Si	LIFO	<0.001	<0.001	1.00E-03	2.00E-03	4.47E+02	1.60E+02	4.33E+02	7.14E+02	50	0	0	13.92	7.02	2.74	0.84	2.48	1.58E+01	3.47E+01	1.58E-01	1.58E-01
G	LBDC	<0.001	<0.001	<0.001	1.00E-03	5.39E+02	1.69E+02	5.52E+02	9.63E+02	50	0	0	14.92	6.54	2.70	1.36	2.96	2.20E+00	1.66E+01	1.58E-01	1.58E-01
G	LBTC	<0.001	<0.001	<0.001	1.00E-03	5.39E+02	1.69E+02	5.52E+02	9.63E+02	50	0	0	14.94	6.52	2.70	1.38	2.96	2.16E+00	1.65E+01	1.58E-01	1.58E-01
G	LIFO	<0.001	<0.001	<0.001	1.00E-03	5.39E+02	1.69E+02	5.52E+02	9.63E+02	50	0	0	14.94	6.52	2.70	1.38	2.96	2.26E+00	1.65E+01	1.58E-01	1.58E-01

[1] FS : found solution [2] FO : found optimal solution [3] SO : shown optimal solution [4] TT : avg. of total touches [5] F : avg. of front side touches [6] R : avg. of rear side touches
[7] DT : avg. double touches [8] MNodes : avg. maximum number of nodes in open set [9] VNodes : avg. number of unique visited nodes [11] LPT : avg. load planning time
[12] TTm : avg. total time time (LPT + sequencing time) [13] JSol : joint solution

Tab. 2.14. 200 ft instances from Section 2.5.2, successive load planning and sequencing, general load plans

algo		sequencing time (s)				cost				FS	FO	SO	TT	F	R	DT	Mo	MNodes	VNodes	LPT	TTm
		avg	5%	50%	95%	avg	5%	50%	95%												
JSol	-	-	-	-	-	8.20E+00	-	-	-	50	50	50	10.66	10.56	0.02	0.02	0.04	-	-	-	4.89E+00
E	LBDC	2.77E+00	9.15E-03	1.87E-01	2.32E+00	2.72E+02	2.00E+00	3.12E+02	4.82E+02	49	8	8	13.24	9.65	0.78	0.08	2.65	1.22E+02	7.54E+04	1.68E-01	2.94E+00
E	LBTC	3.88E+01	9.15E-03	3.32E+00	1.96E+02	2.61E+02	2.00E+00	2.92E+02	4.82E+02	48	9	9	13.15	9.69	0.85	0.08	2.44	1.44E+04	2.09E+04	1.68E-01	3.89E+01
E	LIFO	1.12E+01	3.68E-02	6.42E-01	8.59E+01	2.72E+02	2.00E+00	3.12E+02	4.82E+02	49	8	8	13.24	9.65	0.78	0.08	2.65	-	3.19E+05	1.68E-01	1.14E+01
Ei	LBDC	3.31E-01	7.45E-03	1.06E-01	1.63E+00	2.67E+02	2.00E+00	3.12E+02	4.82E+02	50	9	9	13.28	9.76	0.76	0.08	2.60	7.56E+01	1.45E+04	1.68E-01	4.99E-01
Ei	LBTC	4.69E+01	1.10E-02	3.69E+00	3.00E+02	2.80E+02	2.00E+00	3.12E+02	5.62E+02	50	9	9	13.26	9.46	0.98	0.16	2.50	1.60E+04	2.38E+04	1.68E-01	4.71E+01
Ei	LIFO	2.25E+00	3.22E-02	3.82E-01	8.05E+00	2.67E+02	2.00E+00	3.12E+02	4.82E+02	50	9	9	13.28	9.76	0.76	0.08	2.60	-	7.21E+04	1.68E-01	2.42E+00
S	LBDC	2.10E-03	<0.001	1.00E-03	8.10E-03	3.66E+02	2.00E+00	4.42E+02	6.63E+02	50	9	0	13.44	8.20	2.06	0.34	2.50	2.02E+01	1.53E+02	1.68E-01	1.70E-01
S	LBTC	2.70E-03	<0.001	1.00E-03	9.00E-03	3.90E+02	6.20E+01	4.42E+02	6.63E+02	50	1	0	13.44	7.78	2.48	0.34	2.50	4.27E+01	1.88E+02	1.68E-01	1.70E-01
S	LIFO	2.84E-03	<0.001	2.00E-03	1.27E-02	3.66E+02	2.00E+00	4.42E+02	6.63E+02	50	9	0	13.44	8.18	2.08	0.34	2.50	2.10E+01	2.72E+02	1.68E-01	1.71E-01
Si	LBDC	2.28E-03	<0.001	2.00E-03	5.55E-03	3.59E+02	2.00E+00	4.03E+02	6.54E+02	50	9	0	13.34	8.16	2.14	0.30	2.44	1.87E+01	1.72E+02	1.68E-01	1.70E-01
Si	LBTC	2.84E-03	<0.001	2.00E-03	8.00E-03	3.86E+02	6.20E+01	4.13E+02	6.54E+02	50	1	0	13.36	7.72	2.56	0.32	2.44	5.43E+01	1.94E+02	1.68E-01	1.71E-01
Si	LIFO	3.00E-03	<0.001	2.00E-03	1.09E-02	3.62E+02	2.00E+00	4.03E+02	6.54E+02	50	8	0	13.34	8.14	2.16	0.30	2.44	3.12E+01	2.59E+02	1.68E-01	1.71E-01
G	LBDC	<0.001	<0.001	<0.001	1.55E-03	3.95E+02	2.00E+00	4.32E+02	6.98E+02	50	4	0	13.18	7.74	2.68	0.18	2.40	4.70E+00	3.20E+01	1.68E-01	1.68E-01
G	LBTC	<0.001	<0.001	<0.001	1.00E-03	3.98E+02	6.20E+01	4.32E+02	6.98E+02	50	1	0	13.18	7.70	2.72	0.18	2.40	6.56E+00	2.75E+01	1.68E-01	1.68E-01
G	LIFO	<0.001	<0.001	<0.001	1.55E-03	3.95E+02	2.00E+00	4.32E+02	6.98E+02	50	4	0	13.18	7.76	2.66	0.18	2.40	4.76E+00	4.16E+01	1.68E-01	1.68E-01

[1] FS : found solution [2] FO : found optimal solution [3] SO : shown optimal solution [4] TT : avg. of total touches [5] F : avg. of front side touches [6] R : avg. of rear side touches
[7] DT : avg. double touches [8] MNodes : avg. maximum number of nodes in open set [9] VNodes : avg. number of unique visited nodes [11] LPT : avg. load planning time
[12] TTm : avg. total time time (LPT + sequencing time) [13] JSol : joint solution

The results for the 667 ft testing instances follows the same structure as before. Table 2.15 confirm the proper alignment of the joint and successive solution approach and the conclusions from Section 2.5.1. Tables 2.16 and 2.17 summarize the results of the successive LP and LS with a specific and general load plan respectively. Both these tables present only the results for the heuristics because of insufficient resources to compute solutions using the exact methods for the 667 ft testing instances. The difference in average number of total touches between the suggested methods and the joint solution ranges between 14.4 to 22.5 touches with a specific plan (between 45% and 70.3% additional total touches) and 13.23 to 17.18 using a general plan (between 41.3% and 53.7% additional total touches). Note that we neglect the S-LBTC and Si-LBTC in those estimates since both perform terribly with general plans. While Si-LBTC solves 50 out of 50 instances, 48 of these solutions stem from the initialization with a greedy method using a specific plan. Again, it is noteworthy that between the general and specific load plan, the number of move-over remains largely the same while the reduction of touches is due to lesser double touches. Also, move over operations are largely responsible for the difference in total touches between the joint solutions and our proposed algorithms. Therefore, the LS is constrained by the load plan when the planning and sequencing are decoupled compared to the joint approach.

Tab. 2.15. 667 ft instances from Section 2.5.2, load sequencing using optimal load plans from joint solution

algo		sequencing time (s)				cost				FS	FO	SO	TT	F	R	DT	Mo	MNodes	VNodes	LPT	TTm
		avg	5%	50%	95%	avg	5%	50%	95%												
JSol	-	-	-	-	-	6.40E+00	-	-	-	50	50	50	32	32	0	0	0	-	-	-	1.03E+03
E	LBDC	3.85E-02	2.29E-02	3.10E-02	8.98E-02	6.40E+00	5.00E+00	7.00E+00	7.00E+00	50	50	50	32.00	32.00	0.00	0.00	0.00	3.16E+02	3.49E+02	-	-
E	LBTC	3.36E-02	2.30E-02	3.00E-02	6.06E-02	6.40E+00	5.00E+00	7.00E+00	7.00E+00	50	50	50	32.00	32.00	0.00	0.00	0.00	3.16E+02	3.49E+02	-	-
E	LIFO	3.32E-02	1.90E-02	2.80E-02	5.69E-02	6.40E+00	5.00E+00	7.00E+00	7.00E+00	50	50	50	32.00	32.00	0.00	0.00	0.00	-	3.65E+02	-	-
Ei	LBDC	7.00E-03	2.00E-03	5.50E-03	1.40E-02	6.40E+00	5.00E+00	7.00E+00	7.00E+00	50	50	50	32.00	32.00	0.00	0.00	0.00	4.36E+01	8.14E+01	-	-
Ei	LBTC	7.76E-03	2.00E-03	6.00E-03	2.00E-02	6.40E+00	5.00E+00	7.00E+00	7.00E+00	50	50	50	32.00	32.00	0.00	0.00	0.00	7.77E+01	8.93E+01	-	-
Ei	LIFO	8.76E-03	2.00E-03	6.00E-03	2.41E-02	6.40E+00	5.00E+00	7.00E+00	7.00E+00	50	50	50	32.00	32.00	0.00	0.00	0.00	-	1.00E+02	-	-
S	LBDC	6.38E-03	2.00E-03	5.00E-03	1.47E-02	2.08E+01	5.00E+00	7.00E+00	1.26E+02	50	44	0	32.00	31.76	0.24	0.00	0.00	3.71E+01	8.33E+01	-	-
S	LBTC	4.30E-03	2.00E-03	3.00E-03	9.65E-03	5.20E+01	5.00E+00	6.50E+01	1.27E+02	50	24	0	32.00	31.24	0.76	0.00	0.00	3.62E+01	5.77E+01	-	-
S	LIFO	4.48E-03	1.45E-03	4.00E-03	9.55E-03	2.08E+01	5.00E+00	7.00E+00	1.26E+02	50	44	0	32.00	31.76	0.24	0.00	0.00	2.81E+01	6.37E+01	-	-
Si	LBDC	6.72E-03	2.00E-03	5.50E-03	1.46E-02	6.40E+00	5.00E+00	7.00E+00	7.00E+00	50	50	0	32.00	32.00	0.00	0.00	0.00	3.92E+01	7.29E+01	-	-
Si	LBTC	5.76E-03	2.00E-03	5.00E-03	1.16E-02	2.08E+01	5.00E+00	7.00E+00	6.70E+01	50	40	0	32.00	31.76	0.24	0.00	0.00	4.90E+01	6.13E+01	-	-
Si	LIFO	4.76E-03	2.00E-03	4.00E-03	1.01E-02	1.48E+01	5.00E+00	7.00E+00	6.60E+01	50	43	0	32.00	31.86	0.14	0.00	0.00	4.19E+01	5.79E+01	-	-
G	LBDC	<0.001	<0.001	<0.001	1.00E-03	3.28E+01	5.00E+00	7.00E+00	1.27E+02	50	40	0	32.00	31.56	0.44	0.00	0.00	1.52E+00	2.57E+01	-	-
G	LBTC	<0.001	<0.001	1.00E-03	1.55E-03	5.20E+01	5.00E+00	6.50E+01	1.27E+02	50	24	0	32.00	31.24	0.76	0.00	0.00	1.52E+00	2.23E+01	-	-
G	LIFO	<0.001	<0.001	1.00E-03	1.00E-03	3.28E+01	5.00E+00	7.00E+00	1.27E+02	50	40	0	32.00	31.56	0.44	0.00	0.00	1.52E+00	2.57E+01	-	-

[1] FS : found solution [2] FO : found optimal solution [3] SO : shown optimal solution [4] TT : avg. of total touches [5] F : avg. of front side touches [6] R : avg. of rear side touches
[7] DT : avg. double touches [8] MNodes : avg. maximum number of nodes in open set [9] VNodes : avg. number of unique visited nodes [11] LPT : avg. load planning time
[12] TTm : avg. total time time (LPT + sequencing time) [13] JSol : joint solution

Tab. 2.16. 667 ft instances from Section 2.5.2, successive load planning and sequencing, specific load plans

algo		sequencing time (s)				cost				FS	FO	SO	TT	F	R	DT	Mo	MNodes	VNodes	LPT	TTm
		avg	5%	50%	95%	avg	5%	50%	95%												
JSol	-	-	-	-	-	6.40E+00	-	-	-	50	50	50	32	32	0	0	0	-	-	-	1.03E+03
S	LBDC	3.91E+00	1.15E-01	1.47E+00	1.24E+01	1.71E+03	1.04E+03	1.67E+03	2.38E+03	50	0	0	47.66	22.40	6.42	3.18	12.48	2.01E+02	1.50E+05	1.66E-01	4.08E+00
S	LBTC	5.46E+01	<0.001	5.61E+00	1.54E+02	1.64E+03	0.00E+00	1.40E+03	2.35E+03	38	0	0	46.53	22.53	7.32	2.47	11.74	1.05E+04	5.35E+04	1.66E-01	5.47E+01
S	LIFO	8.87E+00	9.11E-02	1.46E+00	1.01E+01	1.54E+03	1.06E+03	1.48E+03	2.09E+03	50	0	0	46.42	24.56	5.90	1.54	12.88	7.68E+01	2.23E+05	1.66E-01	9.03E+00
Si	LBDC	2.44E+00	9.03E-02	1.12E+00	6.78E+00	1.75E+03	1.16E+03	1.67E+03	2.42E+03	50	0	0	48.06	22.12	6.30	3.58	12.48	1.85E+02	1.07E+05	1.66E-01	2.61E+00
Si	LBTC	1.14E+02	7.18E-01	5.04E+01	3.00E+02	1.80E+03	1.13E+03	1.75E+03	2.53E+03	50	0	0	48.67	22.10	6.43	3.51	13.12	1.82E+04	7.78E+04	1.66E-01	1.14E+02
Si	LIFO	7.14E+00	9.36E-02	1.49E+00	9.60E+00	1.53E+03	1.06E+03	1.48E+03	2.09E+03	50	0	0	46.38	24.56	5.90	1.54	12.84	1.09E+02	2.20E+05	1.66E-01	7.30E+00
G	LBDC	2.34E-03	<0.001	1.00E-03	4.55E-03	2.29E+03	1.77E+03	2.35E+03	2.75E+03	50	0	0	54.50	19.94	6.28	5.78	16.72	5.82E+00	1.17E+02	1.66E-01	1.68E-01
G	LBTC	1.66E-03	1.00E-03	1.00E-03	4.00E-03	2.29E+03	1.77E+03	2.36E+03	2.84E+03	50	0	0	54.32	19.92	6.48	5.60	16.72	6.60E+00	1.05E+02	1.66E-01	1.67E-01
G	LIFO	2.38E-03	<0.001	1.00E-03	6.55E-03	2.27E+03	1.77E+03	2.35E+03	2.75E+03	50	0	0	54.24	20.10	6.38	5.52	16.72	6.26E+00	1.34E+02	1.66E-01	1.68E-01

Tab. 2.17. 667 ft instances from Section 2.5.2, successive load planning and sequencing, general load plans

algo		sequencing time (s)				cost				FS	FO	SO	TT	F	R	DT	Mo	MNodes	VNodes	LPT	TTm
		avg	5%	50%	95%	avg	5%	50%	95%												
JSol	-	-	-	-	-	6.40E+00	-	-	-	50	50	50	32	32	0	0	0	-	-	-	1.03E+03
S	LBDC	2.46E+02	1.16E+00	3.00E+02	3.46E+02	1.55E+03	3.81E+02	1.53E+03	2.11E+03	47	0	0	45.23	23.15	8.17	0.81	12.30	6.52E+02	3.68E+06	2.84E-01	2.46E+02
S	LBTC	1.02E+02	<0.001	<0.001	<0.001	1.03E+03	0.00E+00	0.00E+00	0.00E+00	2	0	0	38.00	23.00	9.00	0.00	6.00	2.54E+04	9.59E+04	2.84E-01	1.02E+02
S	LIFO	2.13E+02	<0.001	3.64E+01	3.31E+02	1.48E+03	0.00E+00	1.13E+03	2.12E+03	32	0	0	45.72	24.41	7.06	1.22	11.81	2.86E+02	3.67E+06	2.84E-01	2.13E+02
Si	LBDC	2.49E+02	1.54E+01	3.01E+02	3.51E+02	1.57E+03	1.03E+03	1.53E+03	2.25E+03	50	0	0	45.80	23.26	7.36	1.38	12.42	6.04E+02	4.06E+06	2.84E-01	2.49E+02
Si	LBTC	2.93E+02	3.00E+02	3.00E+02	3.00E+02	2.24E+03	1.67E+03	2.35E+03	2.75E+03	50	0	0	53.86	19.98	6.40	5.62	16.24	8.72E+04	1.93E+05	2.84E-01	2.94E+02
Si	LIFO	2.53E+02	4.91E+00	3.20E+02	3.59E+02	1.82E+03	1.03E+03	1.73E+03	2.75E+03	50	0	0	48.80	22.66	6.44	2.90	13.90	3.13E+02	4.28E+06	2.84E-01	2.53E+02
G	LBDC	1.53E-01	3.00E-03	1.80E-02	9.59E-01	1.89E+03	1.29E+03	1.93E+03	2.34E+03	50	0	0	49.06	23.06	8.12	0.82	16.24	1.61E+01	3.43E+03	2.84E-01	4.37E-01
G	LBTC	1.41E+00	2.00E-03	1.30E-02	3.61E+00	1.89E+03	1.29E+03	1.93E+03	2.34E+03	50	0	0	49.18	23.14	8.04	0.82	16.36	6.53E+02	2.84E+03	2.84E-01	1.70E+00
G	LIFO	1.71E-01	3.00E-03	3.05E-02	7.96E-01	1.89E+03	1.29E+03	1.93E+03	2.34E+03	50	0	0	49.06	23.06	8.12	0.82	16.24	1.44E+01	4.14E+03	2.84E-01	4.55E-01

[1] FS : found solution [2] FO : found optimal solution [3] SO : shown optimal solution [4] TT : avg. of total touches [5] F : avg. of front side touches [6] R : avg. of rear side touches
[7] DT : avg. double touches [8] MNodes : avg. maximum number of nodes in open set [9] VNodes : avg. number of unique visited nodes [11] LPT : avg. load planning time
[12] TTm : avg. total time time (LPT + sequencing time) [13] JSol : joint solution

2.6. Conclusions and future work

In this paper, we studied the LS problem for double-stack trains using RS based on a North American case study. Double-stack trains allow the stacking of two containers in height on certain railcars. The LP, finding a match between available containers and positions on the railcars that form a departing train, generally precedes and complements the LS. The LS corresponds to the task of ordering the retrieval sequence of piled containers in the terminal yard according to the load plan and crane characteristics. The LP and LS is harder for double-stack trains due to additional loading constraints when generating a load plan as well as the precedence constraints between the containers stacked on the train. RS are mobile cranes that can retrieve containers in move them in the yard but they suffer from a limited reach compare to overhead cranes.

Our research aimed to explore the trade-off between solution quality, tractability and computing time by developing heuristics that could solve realistic sized instances of the LS with small loss in solution quality. We also wanted to verify the impact of leveraging the multiplicity of possible load plans on the performance of the LS. Finally we wished to quantify the loss of solving the LP and LS successively as opposed to jointly (Ruf et al., 2018).

We proposed 15 variants of a label-correcting algorithm to solve the LS problem. Of these 15 variants, we denote 5 distinct algorithms (i.e., 2 exact, 2 semi-greedy, 1 greedy) with 3 node selection variants. We generated instances of size 24, 40, 76 and 120 containers to assess the performance of the suggested solution methods for the LS using both specific and general load plans. The performance of these 15 variants for the successive LP and LS was compared with the results from the joint solution as seen in Ruf et al. (2018).

Our results show that the suggested heuristics are scalable to realistic instance sizes where exact methods falter, while presenting a small loss in solution quality with short computing time. We also illustrate the benefit of using general load plans to improve the solution quality, at the cost of additional computing time and lack of solution guarantees for the non-exact methods. Finally we quantify the significant loss in solution quality when decoupling the LP and LS as opposed to the joint approach.

We envision multiple promising future work directions:

- In an attempt to close the gap in performance between the joint and successive solution methods, we consider generating a multiplicity of possible optimal load plans, solving them by leveraging the low computing cost of the greedy methods and choosing the best solution.
- Similarly, the influence from the order of the list by which the general plans are presented to the non-exact algorithms could be exploited by generating permutations of the list, solving with the non-exact algorithms and retaining the best solution.
- The mathematics underlying the construction of general load plans could be parameterized so as to generate ranges of general load plans. This additional freedom could then be leveraged by using the inexpensive greedy method to solve the LS with the aforementioned ranges of plans and keeping the best solution found.
- Develop a hybrid approach combining dynamic programming and integer linear programming to solve the LS.
- Examining the influence of the yard layout on the LS performance.
- Our approach considers indirectly the distance in the cost structure. A refinement would be to optimize the distance explicitly.
- Consider multiple cranes and multiple types of cranes.
- Develop finer approximations for the downstream cost in an attempt to increase solution quality or computing time.

Acknowledgments

We gratefully acknowledge the close collaboration with personnel from the Canadian National Railway Company (CN). This research was funded by the CN Chair in Optimization of Railway Operations at Université de Montréal and a Collaborative Research and Development Grant from the Natural Sciences and Engineering Research Council of Canada (CRD-477938-14). We also acknowledge the support of Fonds de recherche du Québec through their infrastructure grants. While working on this project, the third author was holding the CN Chair in Optimization of Railway Operations.

Chapter 3

Conclusion and future research directions

The thesis situated the LS problem for double-stack trains in the overall context of containerized transport and introduced the relevant literature and methodology to support the research paper presented in Chapter 2. There are several contributions resulting from this research. The goal of the paper was to study the trade-off between solution quality, tractability and computing time for a range methods, exact and heuristics, to solve realistic-sized instances of the LS. The paper presented fifteen variants of a label-correcting algorithm used to solve the LS: five distinct algorithms (i.e. two exact, two semi-greedy heuristics, one greedy heuristic) with three node selection variants for each. The impact of using general loading plans for LS was also explored as a strategy to enhance solution quality compared with only using specific loading plans. Additionally, the loss in solution quality when solving the LP and LS successively as opposed to jointly as seen in Ruf et al. (2018) was investigated.

The results illustrated that the suggested heuristics are scalable to large, realistic instance sizes when the solution was intractable with exact methods. The heuristics were shown to solve the LS with small loss in solution quality and greatly reduced computing time compared to the exact methods. The greedy method is significantly quicker than all other methods and scales well with the instance size. The semi-greedy methods present a compromise between computing and solution quality as a middle-ground between the exact and greedy variants. For the semi-greedy methods, the difference in solution quality with the greedy method increases with the instance size, but semi-greedy methods present a more pronounced increase in computing time as instance size rises. Still, semi-greedy methods remain viable for large instances when using specific plans.

Using general loading plans was confirmed to improve the solution quality across all methods. However, a drawback of using general loading plans outlined by the results is that it increases the action space of the system greatly, which quickly renders exact methods intractable for even small instance sizes. Non-exact methods also lose the ability to guarantee a solution when using a general plan as opposed to specific plans.

Finally, the results illustrated the significant loss in solution quality when decoupling the LP and LS as opposed to the joint approach. However, solving the LP and the LS jointly comes at a significant computing cost which makes an exact method hardly applicable for realistic instance sizes compared to the decoupled strategy using the proposed methods.

The several future research directions envisioned for the LS of double-stack trains have been detailed in Section 2.6. The most important of these are the following:

- In an attempt to close the gap in performance between the joint and successive solution methods, generating a multiplicity of possible optimal specific loading plans, solving them by leveraging the low computing cost of the greedy methods and choosing the best solution.
- The influence from the order of the list by which the general plans are presented to the non-exact algorithms could be exploited by generating permutations of the list, solving with the non-exact algorithms and retaining the best solution.
- Our approach considers indirectly the distance in the cost structure. A refinement would be to optimize the distance explicitly.
- Consider multiple cranes and multiple types of cranes.

The scope of the research on LS could be broadened in the following directions:

- Implement a dynamic version of the LS where the structure of the storage evolves to reflect the progressive arrival of new containers during loading operations.
- Building on the dynamic LS, implement a stochastic LS reflecting the uncertainty in arrival times of containers within the stacks.
- Considering both the loading and unloading of the double stack trains would be a natural addition to the current framework. This would jointly solve sequencing operation from train to stack and from stack to train.
- Compare performance of the suggested heuristic methods with that of other methods from the literature.

Bibliography

Álvarez Serrano, J. F. A heuristic for vessel planning in a reach stacker terminal. *Journal of Maritime Research*, 3(1):3–16, 2006.

Ambrosino, D., Bramardi, A., Pucciano, M., Sacone, S., and Siri, S. Modeling and solving the train load planning problem in seaport container terminals. In *2011 IEEE International Conference on Automation Science and Engineering*, pages 208–213, Aug 2011.

Ambrosino, D., Caballini, C., and Siri, S. A mathematical model to evaluate different train loading and stacking policies in a container terminal. *Maritime Economics & Logistics*, 15 (3):292–308, 2013.

Ambrosino, D., Paolucci, M., and Sciomachen, A. A mip heuristic for multi port stowage planning. *Transportation Research Procedia*, 10(Supplement C):725 – 734, 2015. 18th Euro Working Group on Transportation, EWGT 2015, 14-16 July 2015, Delft, The Netherlands.

Bertsekas, D. P. *Dynamic Programming and Optimal Control*, volume I. Athena Scientific, 4rd edition, 2017.

Bian, Z., Shao, Q., and Jin, Z. Optimization on the container loading sequence based on hybrid dynamic programming. *Transport*, 31(4):440–449, 2016.

Boysen, N., Fliedner, M., Jaehn, F., and Pesch, E. A survey on container processing in railway yards. *Transportation Science*, 47(1):312–329, 2013.

Carlo, H. J., Vis, I. F. A., and Roodbergen, K. J. Seaside operations in container terminals: literature overview, trends, and research directions. *Flexible Services and Manufacturing Journal*, 27(2):224–262, 2015.

Carlo, H. J., Vis, I. F., and Roodbergen, K. J. Transport operations in container terminals: Literature overview, trends, research directions and classification scheme. *European Journal of Operational Research*, 236(1):1 – 13, 2014a.

Carlo, H. J., Vis, I. F., and Roodbergen, K. J. Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research*, 235(2):412 – 430, 2014b. Maritime Logistics.

Caserta, M., Schwarze, S., and Voß, S. A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research*, 219 (1):96 – 104, 2012.

Castonguay, J. and Stone, L. International shipping: Globalization in crisis. `http://www.visionproject.org/images/img_magazine/pdfs/international_shipping.pdf`. Accessed: 2018-06-21.

Crainic, T., Kim, K., and Centre for Research on Transportation (Montréal, Q. *Intermodal Transportation*. Publication (Centre for Research on Transportation (Montréal, Québec)). Centre de recherche sur les transports (C.R.T.), 2005.

Gharehgozli, A., Roy, D., and De Koster, R. *Sea container terminals: New technologies and OR models*, volume 18. 04 2015.

Gupta, N. and Nau, D. S. On the complexity of blocks-world planning. *Artificial Intelligence*, 56(2):223 – 254, 1992.

Heggen, H., Braekers, K., and Caris, A. *Optimizing Train Load Planning: Review and Decision Support for Train Planners*, pages 193–208. Springer International Publishing, 2016.

Imai, A., Sasaki, K., Nishimura, E., and Papadimitriou, S. Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. *European Journal of Operational Research*, 171(2):373 – 389, 2006.

Iris, C. and Pacino, D. A survey on the ship loading problem. In Corman, F., Voß, S., and Negenborn, R. R., editors, *Computational Logistics*, pages 238–251. Springer International Publishing, 2015.

Iris, C., Christensen, J., Pacino, D., and Ropke, S. Flexible ship loading problem with transfer vehicle assignment and scheduling. *Transportation Research Part B: Methodological*, 111:113 – 134, 2018.

Ji, M., Guo, W., Zhu, H., and Yang, Y. Optimization of loading sequence and rehandling strategy for multi-quay crane operations in container terminals. *Transportation Research Part E: Logistics and Transportation Review*, 80:1 – 19, 2015.

Kim, K. H., Kang, J. S., and Ryu, K. R. A beam search algorithm for the load sequencing of outbound containers in port container terminals. *OR Spectrum*, 26(1):93–116, Jan 2004.

Lang, M., Przybyla, J., and Zhou, X. Loading containers on double-stack cars: Multi-objective optimization models and solution algorithms for improved safety and reduced maintenance cost. *Unpublished, Consulted online on 25 June 2017, https://pdfs.semanticscholar.org/ddda/ab5c087e9ce762f6ded44e9102826df5d00d.pdf*, 2011.

Lehnfeld, J. and Knust, S. Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research*, 239(2):297 – 312, 2014.

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

Monaco, M. F., Sammarra, M., and Sorrentino, G. The terminal-oriented ship stowage planning problem. *European Journal of Operational Research*, 239(1):256 – 265, 2014.

Ruf, M., Frejinger, E., Cordeau, J.-F., and CIRRELT. Load planning and sequencing for double stack trains. *Unpublished, Working paper*, 2018.

Shen, Y. and Zhang, C. Loading sequencing with consideration of container rehandling. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1237–1241, Dec 2015.

Slaney, J. and Thiébaux, S. Blocks world revisited. *Artificial Intelligence*, 125(1):119 – 153, 2001.

United Nations Conference on Trade and Development. Handbook of Statistics 2018 - Maritime Transport. `https://unctad.org/en/PublicationChapters/tdstat43_FS15_en.pdf`, 2018. Accessed: 2019-02-04.

Upadhyay, A., Gu, W., and Bolia, N. Optimal loading of double-stack container trains. *Transportation Research Part E: Logistics and Transportation Review*, 107:1 – 22, 2017.

World Shipping Council. Dry Cargo Containers. `http://www.worldshipping.org/about-the-industry/containers/dry-cargo-containers`. Accessed: 2019-02-05.