UNIVERSITÄT OSNABRÜCK

Optimization of operative planning in rail-road terminals

Dissertation von

Dipl. Systemwiss. Florian Bruns
Fachbereich 6, Mathematik und Informatik
Universität Osnabrück

florian.bruns@uos.de

30. June 2014

Contents

List of Figures

Danksagung

In beiden Promotions-Ratgebern, die ich konsultiert habe, waren jeweils Familiengründung und Eigenheimerwerb als nicht ratsame “Nebenbeschäftigungen” parallel zur Promotion aufgelistet. Dafür, dass ich von vielen Menschen unterstützt worden bin, wodurch ich so trotz meiner “Nebenbeschäftigungen” promovieren konnte, möchte ich mich bedanken.

Zuallererst möchte ich meiner Frau Julia danken, die mit mir durch alle Höhen und Tiefen der letzen Jahren gegangen ist und mich dabei immer unterstützt hat.

In allen Phasen meiner Promotion hat mich auch meine Betreuerin Sigrid Knust stets unterstützt und mir Rückhalt gegeben.

Dank gebührt auch meinen Eltern und meiner Patentante für die Unterstützung auf allen Ebenen auf meinem bisherigen Lebensweg.

Im weiteren möchte ich den Personen und Organisationen danken, die mich in der Anfangsphase der Promotion unterstützt haben. Besonders geholfen hat mir die geduldige und offene Art von Manfred Schuster und Chris Fanter vom Terminal Hamburg Billwerder. In der Arbeitsgruppe haben mir die vielen Diskussionen und Anregungen mit und von Christian Viergutz sehr geholfen. Zudem möchte ich mich für die finanzielle Unterstützung von der Hans Mühlenhoff-Stiftung bedanken.

Sehr geholfen hat mir auch die gute Zusammenarbeit mit den Firmen Berghof Automationstechnik, der DB Netz sowie der Deutsche Umschlaggesellschaft Schiene-Straße im Rahmen mehrerer Aufträge. In diesem Zusammenhang möchte ich besonders Herrn Wallstein, Gerald Klein, Herrn Hetzel, Kathrin Altmann, Marco Lenk und Jonathan Neubert danken.

Für die anregenden und produktiven Kooperationen auf wissenschaftlicher Ebene bedanke ich mich bei Marc Goerigk, Anita Schöbel, Natalia Shakhlevich sowie Stefan Waldherr, David Meignan, Jana Lehnfeld und Xuan Thanh Le aus meiner eigenen Arbeitsgruppe. Zudem danke ich Nils Boysen für viele stets interessante und motivierende Gespräche auf Konferenzen.

Last but not least, waren mir meine Kinder Linus und Jonathan immer wieder eine schöne, wichtige und wunderbare Ablenkung von der Arbeit an der Dissertation. Oft hat mir der Abstand sehr geholfen, die Gedanken die Arbeit betreffend zu sortieren.

1 Introduction

In this thesis we consider the operational planning processes in intermodal rail-road terminals. The principle of intermodal transportation is that goods are transported in standardized transport boxes (e.g. containers) using different transportation modes like rail, road or ship. When changing the transportation mode the goods do not leave the transport boxes, but change the transportation mode inside the boxes. Rail-road terminals operate as chain links in intermodal transportation, where containers, swap bodies and trailers (which are all called load units in the following) are transferred from trains to trucks and vice versa. In rail-road terminals continental and maritime traffics are handled. While continental intermodal traffic only contains the transportation modes rail and road, in maritime traffic deep sea ships are also contained. In maritime traffic almost only containers occur, but in continental traffic also swap bodies and trailers of various sizes have to be handled.

We consider three subproblems of the operational planning process at rail-road terminals that terminal operators are facing in their daily operations. These are the optimization problems storage planning, load planning and crane planning.

- The aim of storage planning is to determine load unit storage positions for a set of load units in a partially filled storage area. Here, different restrictions like non-overlapping of stored load units have to be respected. The objective of storage planning is to minimize the total transportation costs and the number of load units that are not stored at the ground level.
- For the load planning we assume a scenario of overbooked trains. So, the aim of load planning is to assign a subset of the load units that are booked on a train to feasible positions on the wagons such that the utilization of the train is maximized and the costs for the handling in the terminal are minimized. For the feasible positioning of load units length and weight restrictions for the wagons and the train have to be respected. For the load planning of trains we consider a deterministic version and a robust approach motivated by uncertainty in the input data.
- The last considered optimization problem is the crane planning. The crane planning determines the transfer of the load units by crane between the different transportation modes. For each crane a working plan is computed which contains a subset of the load units that have to be handled together with individual start times for the transfer operations. For the load units which have to be transfered in the terminal, storage and load planning compute destination positions (inside the terminal). These destination positions are part of the input for the crane planning. The main objective of crane planning is to minimize the total length of the empty crane moves that have to be performed between successive transports of load units by the cranes.

Note, that we do not consider planning problems on the tactical level like the compositions and destinations of the trains or the spatial policy organization of the terminal storage areas.

According to rail-road terminals relatively few literature exists, which has been overviewed recently by Boysen et al. [24]. But even earlier also Bontekoning et al. [20] saw an emerging research field in the optimization of rail-truck intermodal transportation.
On the other hand, there is a relatively large number of publications concerning optimization problems in maritime intermodal transportation (see e.g. Stahlbock and Voß [82] for an overview). Due to the different settings the optimization approaches in maritime transportation and terminals cannot easily be used for the optimization in rail-road terminals.

Compared to fully automated yard areas in maritime terminals, rail-road terminals are operated kind of old-fashioned. Indeed large rail-road terminals nowadays work with IT-systems (cf. Altmann et al. [4]). But only few optimization is integrated in the IT-systems so far (for example, see Hetzel [55]). The motivation of this work is to describe and implement optimization methods for the operative rail-road terminal optimization problems described above. On the one hand, these are new scientific approaches, as there exists only very few and not adequate works on load and storage planning. On the other hand, we want to propose optimization methods that can improve commercial rail-road terminal optimization software. Therefore, we need to model real world constraints and optimize the real world objectives. In addition, especially for the storage and the crane planning our practical motivation leads to the requirement that the algorithms provide good solutions after a short runtime. This is the case, as the optimization takes place during operation that can not be interrupted. As our motivation is to provide basic optimization methods for practical problems, we test the developed algorithms using real world data obtained from German rail-road terminals.

The main contributions of this thesis concern load and storage planning. For the deterministic load planning we provide the first model that represents all practical constraints including physical weight restrictions. For the load planning we furthermore present robustness approaches for different practical uncertainties. For the storage planning we provide complexity results for different variants. For the practical setting we developed a heuristic which is able to compute solutions of high quality in a small amount of runtime.

The remainder of this thesis is organized as follows. After introducing intermodal transportation more detailed in Section 2 there are sections for the four considered optimization problems. In Section 3 storage planning is considered. After introducing a three-field notation for storage planning some complexity results are presented. The main part of Section 3 are optimization methods for the practical rail-road terminal storage planning problem. We introduce different MIP-models and a heuristic based on list scheduling and compare computational results for these optimization approaches. The deterministic load planning is presented in Section 4. Here we introduce different MIP-models and compare their results and runtimes. Subsequently, a robust load planning approach for different uncertainties is presented in Section 5. In that section we show how one MIP-model of Section 4 can be extended to treat the robust problem. The last subproblem crane planning is treated in Section 6. To describe the problem exactly we present a MIP-model. Afterwards, again a heuristic is proposed to provide solutions for problems of practical size and computational results are compared. Finally, we give a conclusion of our results and some outlook to further research in Section 7.

2 Intermodal transportation

Intermodal transport uses different transportation modes like roads, rails or ships. The goods are transported in load units (containers, swap bodies or trailers) and change their transportation modes at bi- or trimodal terminals. The goods are transported in one and the same load unit from the dispatching customer to the receiving costumer. A typical transport chain consists of three chain links: pre-, long- and end-haulage. Note that there might also be more than three chain-links. The aim of intermodal transportation is to bundle transportations amounts on the long-haulage to a efficient transportation mode like rail, deep-sea vessel or barge.

Figure 2.1 shows intermodal terminals in Germany. Bimodal terminals operated by the DUSS company (Deutsche Umschlagsgesellschaft Schiene Straße) are indicated by red circles while hinterland terminals of other companies are indicated by grey pentagons. Trimodal terminals with deep sea vessel access are marked by a red ship. Further terminals are planned terminals (red circle containing a blue circle) and terminals for roll-on roll-off ("Rollende Landstraße") which are indicated by yellow rectangles with a truck inside.

We consider two types of intermodal transportation modes which use rail-road terminals: rail-road transport and hinterland transportation.

In intermodal rail-road transport the pre-haulage is the road transport from the dispatching customer to the nearest (source) rail-road terminal, the long-haulage is the rail transport to the (sink) terminal and the transportation from the (sink) terminal to the receiving customer is called end-haulage. Between the source and the sink terminal some load units have to change trains at intermediate hub terminals. Such a transportation mode is called gateway traffic. In European intermodal rail-road transportation the pre-haulage typically takes place in the evening, the long-haulage by train is overnight and the end-haulage is processed in the morning.
An example for a rail-road transport is a transport from a company close to Munich by truck to the terminal München Riem, then by train to the terminal Hamburg Billwerder and again by train to a company close to Hamburg.

In hinterland transportation containers are delivered to the deep sea ports by vessels and then transported by train to a rail-road terminal in the hinterland. The post haulage is then again by truck to the customer. Note that the long haulage is divided in a transportation by deep-sea vessel and one by train.
Examples for hinterland transports are transports from the maritime terminals (e.g. in Hamburg or Bremerhaven) to a rail-road terminal in the southern part Germany like Kornwestheim (close to Stuttgart) or München Riem.

Intermodal transportation is a growing market. A study of the German government assumes a mean growth rate of 6.3% for hinterland transportation in Germany and the Netherlands till 2025 [31]. The total amount of hinterland transportation would then be 235 Mio. tons in 2025 in this region. The mean growth rate for the German ports is assumed to be even 6.5%.
The total intermodal transport that uses trains had growth rates of 8.7% in 2010, 9.4%

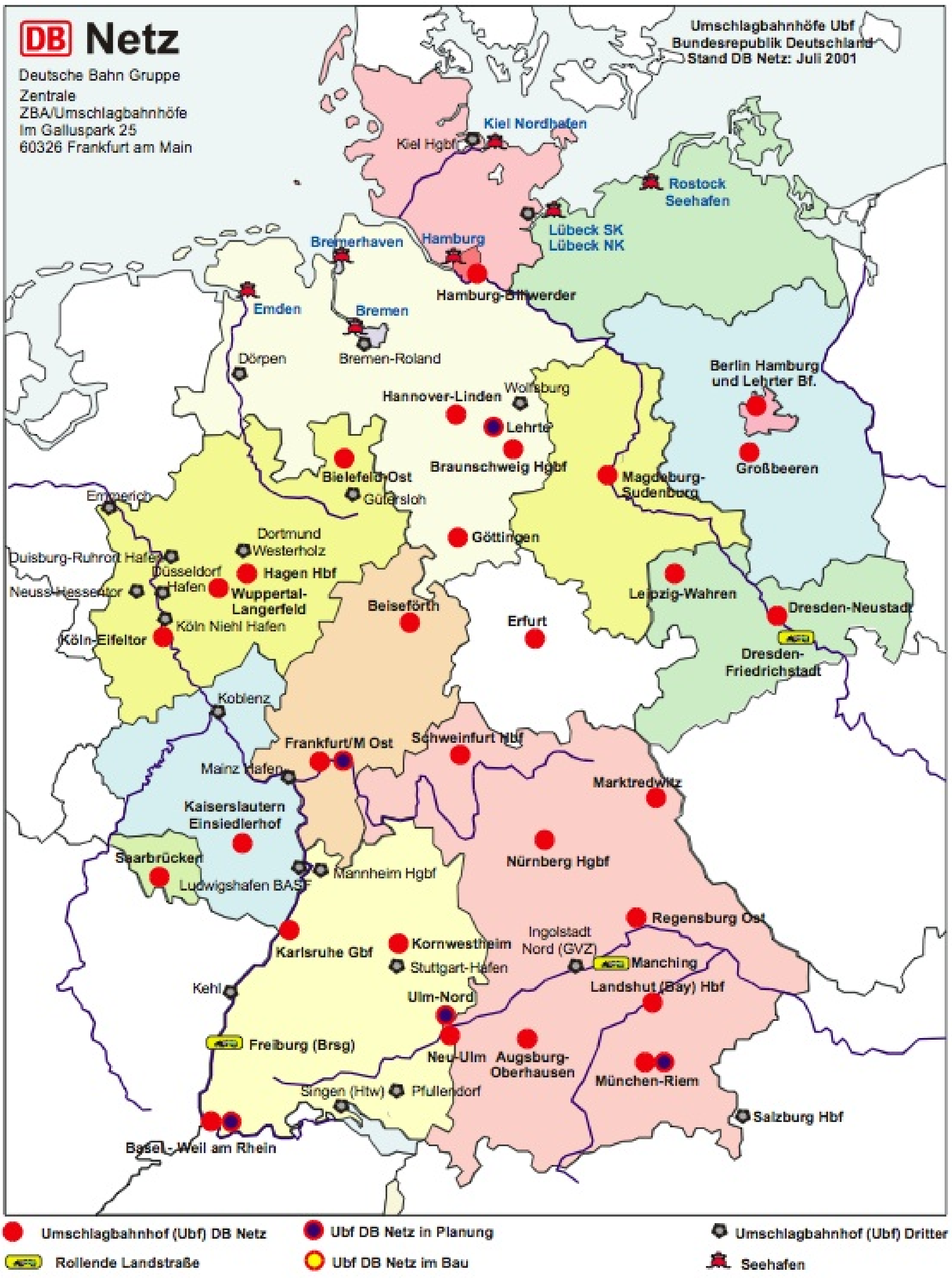


Figure 2.1: Map showing intermodal terminals in Germany [30].

in 2011 and 5% in 2012 [29]. So, intermodal transportation increased in contrast to the stagnating volume of overall rail transport.

Most of the research in intermodal transport has been carried out in the field of maritime container terminals (cf. Stahlbock and Voss [82]), whereas fewer research concerns rail-road or hinterland terminals. But the huge transportation volume leads also to an emerging research field of intermodal rail-truck freight transport within the area of operations research (cf. Bontekoning et al. [20]). Recently, the first overview article on the optimization of rail-road and rail-rail terminals has been published by Boysen et al. [24].

This section is structured as follows. In Section 2.1 rail-road terminals are described in more detail. Furthermore, properties of load units (Section 2.2), trucks (Section 2.3), intermodal trains (Section 2.4) and the storage area (Section 2.5) of terminals are described. At the end of this section the optimization problems considered in this work are introduced in Section 2.6 .

2.1 Rail-road-terminals

A rail-road-terminal is the exchange point between the chain links in intermodal rail-road transportation. In a rail-road-terminal load units are lifted by rail mounted gantry cranes (RMGs) or reach stackers from trucks to trains and vice versa. If a terminal is operated as a hub, load units are also interchanged between trains. To enlarge flexibility of terminals, storage areas for load units are integrated, so not every corresponding (load unit exchanging) truck-train pair or pair of trains needs to have overlapping time windows of visiting the terminal.

A typical terminal module layout with rail, road and storage areas as well as overspanning rail mounted gantry cranes is shown in Figure 2.2. The sizes of the different areas are about four tracks for trains, a driving and a loading lane for trucks and up to five rows for storage. The cranes operate on a common pair of special tracks. This implies that crane interferences occur because the cranes can not pass each other. Huge terminals like Hamburg Billwerder, Köln Eifeltor or München Riem consist of more than one module. Different modules of a terminal can be treated independently on an operational planning level.

To describe a position we use 3D poses. We define the x-axis (continuous) along the tracks. The y-axis is orthogonal to the x-axis at ground level. For y-positions a discrete number of values is possible, e.g. rows one to ten where the first four rows are tracks, rows five and six are driving lanes and the last four rows represent the storage. The z-axis describes the elevation of a position. The z-position is just relevant for the storage because load units are not stacked on top of trains and trucks.

The RMGs are able to lift containers at the upper corner castings and swap bodies as well as trailers by garblers at the bottom of the load units. Furthermore, RMGs are able to turn load units if necessary.

Terminals with low utilization are operated in the “Standverfahren”. This means all trains are parked under the RMGs after entering the terminal until the trains continue

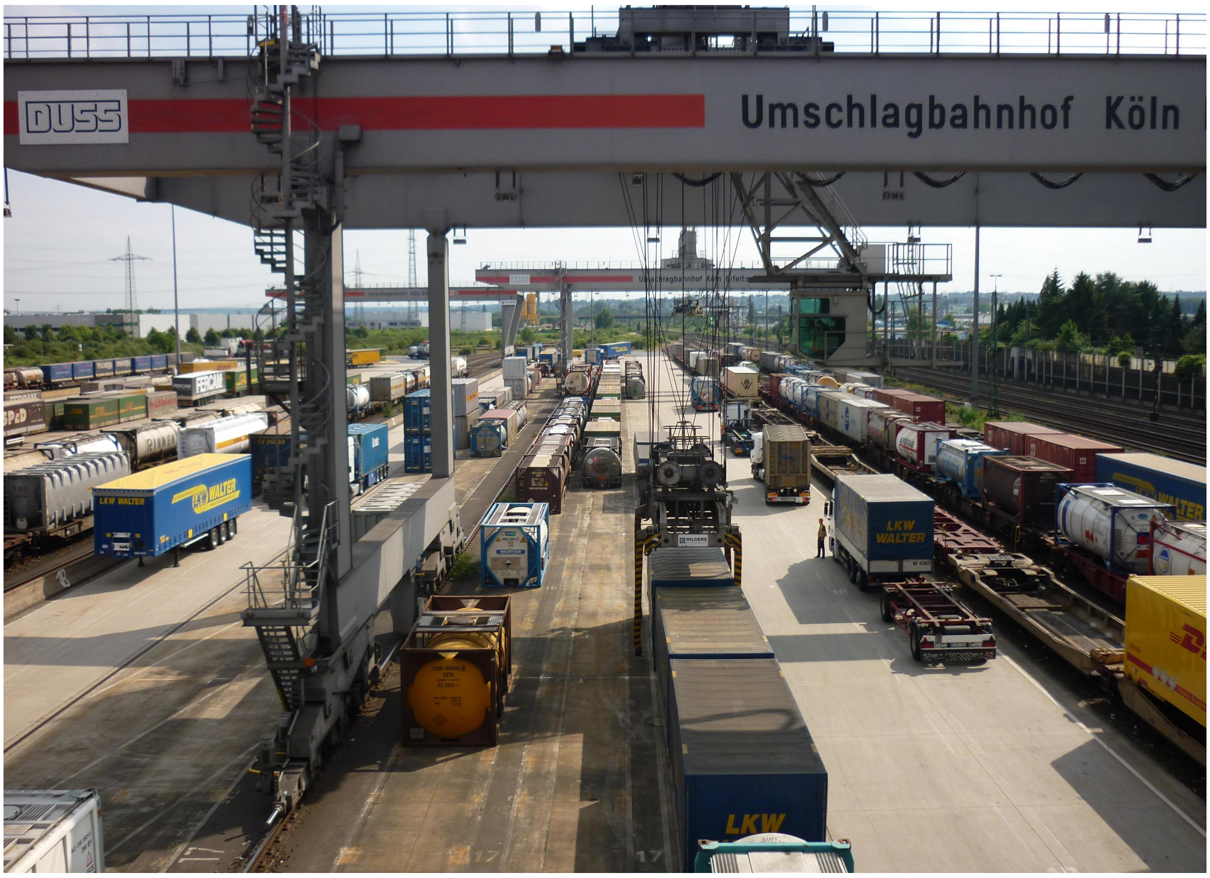


Figure 2.2: This picture shows a rail-road terminal (Köln Eifeltor) with rail, road and storage areas and overspanning cranes.

to an other terminal. In the following we describe the typical daily operations of an European rail-road terminal that is operated in the "Standverfahren": Most trains arrive in the early morning and leave the terminal in the evening. Most trucks pick up load units in the morning while more load units are delivered to a terminal by truck in the early evening. This causes two peaks of the workload in many rail-road terminals. One when most of the trains arrive in the morning and lot of trucks collect load units. The other peak is in the evening close before the deadlines for train loading.

If a terminal has a high utilization, it is operated in the "Fließverfahren". This means that trains have two stays in the terminal. The first stay is directly after the arrival at the terminal. The second visit is just before the train leaves to an other terminal. Between the two visits of the terminal the trains are parked on sidings. It is common that trains are completely unloaded during the first stay because otherwise it would not be possible to access the load units on the trains while they stay on the sidings. The main loading operations of the trains take place during the second visit. Terminals with high utilization do not have clear peaks of workload.

For each terminal we categorize the load units handled in the terminal in import-, export- and gateway load units. An import load unit reaches the corresponding terminal by truck and leaves by train, for an export load unit it is vice versa (enter by train, leave on truck). A gateway load unit enters and leaves the terminal by train. Note that load units change their category at least once. At the starting terminal the load unit is an export load unit and at the destination terminal it is categorized as import load unit. Additionally, the load unit may be a gateway load unit at one or several intermediate terminals. But more than one train-train transfer is not typical.

In the best case load units are directly transferred between the different transportation modes (trains and trucks). Often this is not possible, e.g. due to non-overlapping time-windows or missing information of truck arrival times. If a direct handling of load units is not possible, these load units have to be stored inside the storage area.

Rail-road terminals are quite different to maritime terminals. In rail-road terminals all relevant working areas are spanned by the RMGs. Maritime terminals have different areas that are treated by different transportation facilities. These areas are the berth or quay where cranes load and unload the ships, the yard where containers are stored, and often rail interfaces where trains are loaded and unloaded by RMGs [82]. Furthermore, the load unit diversity is quite different. In maritime terminals almost all load units are either 20 feet or 40 feet containers while in rail-road terminals up to 24 different load unit types have to be considered.
The most similar areas in maritime terminals are the rail interfaces. They are like rail-road terminals without storage areas and truck lanes but also handled by several RMGs.

A special type of hinterland terminals are rail-rail or hub terminals which are built to forward load units from trains to trains (cf. Rotter [78]). These terminals work as hub terminals in a hub and spoke network. The advantage of these terminals is that connections between small hinterland terminals with low volumes can be established without the necessity for time-consuming shunting. A rail-rail terminal that is actually built in Hannover Lehrte is composed like a rail-road terminal (see Figure 2.2) with an additional rail mounted load unit transport system for long moves in x-direction (this transport system is loaded and unloaded by the RMGs) (cf. Alicke [3]). So, actually rail-rail terminals are often built like rail-road terminals and the operations in the terminal are comparable (excluding additional handling facilities like transport systems).

2.2 Load units

The load units are divided into three different categories: containers, swap bodies, and trailers. Containers and swap bodies are three-dimensional boxes that have four so-called corner castings which are one part of the connector between containers and trucks, trains or ships. The other part of the connection in rail and truck transport are flexible pins on the wagons or truck chassis frames. The distances in between the corner castings (L2 in Figure 2.3) are standardized to 20 feet (5853 $\pm$ 3 mm), 30 feet (8918 $\pm$ 4 mm) and 40 feet (11985 $\pm$ 5 mm). Note that the corner castings do not have to be at the corners of a container, some containers and many swap bodies have an overhang above the corner castings. This is e.g. the case for the container in Figure 2.3 with the overhangs L3 and L4.

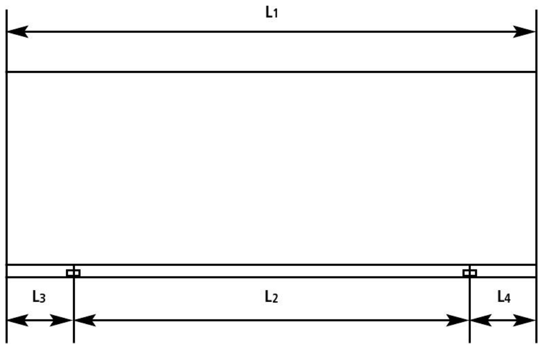


Figure 2.3: This figure shows a schematic representation of a container or swap body [63].

We use two different ways to categorize containers and swap bodies into types: the lengths of the load units (length-type) and the corner casting distances (fixation-type). While with each length-type exactly one specific corner casting distance is associated, containers and swap bodies with the same fixation-type may have different lengths. Therefore, for the fixation-types in addition the overhangs to both sides have to be considered.

The length-type definition is equivalent to the load unit categorization of the International Union of Railways (UIC) [63], which distinguishes 23 different container types. All these types are shown in Table 2.1. Each row corresponds to a length-type, the columns contain the length-type number, the fixation-type number, the maximum length, the maximum left overhang, the corner casting distance and the maximum right overhang (all lengths are measured in millimetres). While in the first part of the table the symmetric containers are shown (i.e., left and right overhangs are equal), in the second part the asymmetric containers are listed. Note that for the symmetric containers the length-type number is equal to the length of the container in feet, which is not the case for the asymmetric containers. The maximum length of a container belonging to a certain type is equal to the sum of the distance in between the corner castings and the maximum overhangs above these castings to both sides. Trailers are considered as one additional load unit type (corresponding to one length-type and one fixation-type). Thus, in total 24 length- and four fixation-types exist.

Most containers and some swap bodies are stackable. Some load units contain dangerous goods and have to be treated in a special way. For some groups of dangerous goods safety distances have to be kept on the train and inside the storage.

2.3 Truck properties

Truck arrival times are not communicated to the terminal operator before the truck arrives at the terminal. Some trucks deliver and collect load units during a terminal visit but trucks are not able to carry more than two load units together.
For the truckage companies a major terminal quality measure is the time of truck circulation. This is the time between entering the terminal and leaving the terminal. The

Length-type	Fixation-type	Max. length (mm) (L1)	Max. left overhang (L3)	Corner casting distance (L2)	Max. right overhang (L4)
20	1	6058	102.5	5853 ± 3	102.5
21	1	6250	198.5	5853 ± 3	198.5
22	1	7150	648.5	5853 ± 3	648.5
23	1	7450	798.5	5853 ± 3	798.5
24	1	7820	983.5	5853 ± 3	983.5
26	1	8150	1148.5	5853 ± 3	1148.5
30	2	9125	103.5	8918 ± 4	103.5
31	2	9300	191.0	8918 ± 4	191.0
40	3	12192	103.5	11985 ± 5	103.5
42	3	12500	257.5	11985 ± 5	257.5
44	3	13100	557.5	11985 ± 5	557.5
45	3	13716	865.5	11985 ± 5	865.5
60	1	8543	1190.0	5853 ± 3	1500.0
81	2	9275	103.5	8918 ± 4	253.5
82	2	9330	103.5	8918 ± 4	308.5
84	2	10040	103.5	8918 ± 4	1018.5
85	2	10200	103.5	8918 ± 4	1178.5
86	2	10900	103.5	8918 ± 4	1878.5
91	3	12500	103.5	11985 ± 5	411.5
94	3	12750	103.5	11985 ± 5	661.5
95	3	13200	257.5	11985 ± 5	957.5
96	3	13600	715.0	11985 ± 5	900.0
97	3	14040	715.0	11985 ± 5	1340.0

Table 2.1: The 23 different container types of the UIC [63]

truck circulation time can be seen as an unproductive time for the truckage companies because driver and truck can not be assigned to any other jobs but wait to be processed in the terminal.

2.4 Intermodal trains

Intermodal trains are normally run as shuttle trains (see Ballis and Golias [9]). A fixed group of wagons is used for transportation from terminal A to terminal B and vice versa. The trains have fixed timetables but delays may occur. Trains have a length of up to 700 m and consist of about 30 wagons. A lot of wagon types are used. These wagon types have a length between 14 and 36 m [36] and they are not built for double stacking. Wagon types can be distinguished into flatcars and pocket wagons. While flatcars can only carry containers and swap bodies, pocket wagons can also carry trailers. For each wagon type a number of physical configurations define feasible loadings. For each configuration the number of load unit slots on the wagon as well as the length and the weight of these load units are restricted. Furthermore, the total payload of the whole train is limited. We assume that trains are usually overbooked, so a subset of load units has to be chosen to be loaded on the train. The load units that are not chosen to be loaded just stay at the terminal.

2.5 Storage area

The storage area is used to store load units that can not be transferred directly. The time load units are stored differs between some hours and a month or even longer. As the load units are assessed by the RMGs, the stacking height is limited. Usually it is allowed to stack containers and special swap bodies up to a height of three load units. The weight of all container stacks is limited due to the carrying capacity of the storage ground. If the utilization of the stock is low, load units are not stacked. Stacking implies the risk of reshuffles when a load unit needs to be accessed that has other load units stacked on top.
In many terminals safety distances between different groups of dangerous goods have to be respected. Load units that contain explosive materials are not allowed to be stored close to load units with content that is harmful to the environment.

Nowadays, storage areas are often organized in grids. As a policy load units are not allowed to exceed the borders of these grids. This grid concept shall help the human crane operators to easily find load units when they have to be delivered. There are no technical restrictions that motivate grids. For the future it is planned to equip the cranes with positioning systems (see Hetzel [55]). Then grids are no longer necessary as the cranes can be positioned exactly at any position of a stored container that has to be delivered.

Typical grid lengths are 15m or 25m. Grids of 15m are motivated by the fact that the longest load units have a length of about 14m to 14.5m. So, a 15m grid allows to place at least one of each load unit type inside. For load units of at most 7.5m (equivalent to 23 feet) two load units can be positioned in one grid. Contrary, 25m grids allow more combinations of load unit types. Consider for example a grid where a 30 feet container (9.125m) is placed at the beginning. If the grid has a length of 15m, no other load unit can be placed in the grid as the shortest load units have a length of 6.058m (20 feet container). If the grid has a length of 25m, different combinations of load units can be placed inside the grid in addition to the 30 feet container.

2.6 Rail-road-terminal optimization of operative planning

In this section we describe the operative terminal situation where the considered optimization is done. Afterwards, we formally introduce the optimization problems that are treated in this thesis. At the and of this section we will give a short overview on commercial software for rail-road terminal optimization.

We assume a fixed train schedule with fixed tracks and also fixed parking positions for the trains which is usual for German rail-road terminals where shuttle trains are operated. The schedules are fixed as the train companies book fixed time slots on the tracks they run their trains. So, when delays are ignored it is possible to work with a fixed assignment of tracks to trains for the terminal operator. The parking positions of trains are also fixed. This enables to store load units that enter the terminal before the train arrives at positions that are close to wagons of the train, when load units have to be stored and later on transshipped to a wagon of the train. If parking positions are not fixed long transportation by cranes alongside the tracks may be necessary to transship stored load units to the closest "end" of the train.

In this work we will treat the storage planning of the storage area, the load planning of trains and the crane planning of the RMGs. The main objective of the optimization is to minimize the total length of the moves of the RMGs. For this purpose we decompose the problem in two phases. The first phases determines sink positions of load units that have to be transported. The objective is to minimize the total length of load unit transports by the RMGs. In the first phase storage and load planning are done. In the second phase the crane planning aims to minimize the total length of the empty crane moves. In the following we will introduce the three considered optimization problems in more detail.

For all import load units that are not directly transferred to a truck, storage positions have to be determined. In addition, also for some export and gateway load units that are not transferred directly, storage positions have to be determined (this may also be the case for load units that have to be reshuffled, i.e. load units which are stored on top of a load unit that has to be delivered).
For the storage planning (treated in Section 3) we know the current positions of the load units that have to be stored due to the fixed tracks and parking positions of the trains. In addition, we know positions that are occupied by already stored load units. The aim is to find feasible storage positions for all load units while the total length of the load unit transportation by the RMGs and the risk of reshuffles has to be minimized. Feasible storage positions means that load units are not allowed to overlap. Load units may have the same position expect the level when they are stacked. But for stacked load units further conditions have to be fulfilled.

For the load planning (treated in Section 4) a set of export and gateway load units that should be loaded onto the train is given. For these load units their position in the terminal is given as we assume that they are stored inside the storage area. Since the trains are used to be overbooked, not all load units may fit on the train. The wagons of the trains can be used in specific configurations that define how the wagon can be loaded. Furthermore, the train entered the terminal loaded and therefore the wagons are in specific configurations that have been used at the travel to the terminal. The load planning decides which load units are assigned to the train and to which slot onto one of the wagons they are assigned. The aim of load planning is to maximize train utilization while again the total length of the load unit transports has to be minimized. Furthermore, the number of wagons where the configurations are changed has to be minimized as changing configurations is time-consuming. As load units are often delivered to the terminal just before the deadline for train loading, the information on some load units is highly uncertain. By the prebooking systems of the rail companies the hauler announces the information on the load units. But the actually delivered load units may differ in size and weight. In addition, some wagons may have a bug and not be loaded at all. So it is useful to treat these uncertainties. We will do this by a robust optimization approach (in Section 5).

The crane planning (see Section 6) can be done with preexisting source and sink positions for the crane transports in the second phase as the sink positions have been determined in the storage and load planning (first phase). The crane plan assigns each transport to one of the cranes and defines starting times for each transport. Note, that the starting times are based on the sequences of the transports for each crane as travel times for empty crane moves have to be respected. The major goal of the crane planning is to minimize the length of the empty crane moves as well as violations of transport jobs due dates and again set up costs (for the crane spreaders). The transport jobs due dates are mainly caused by the closing for cargo of the trains.

After the proposed optimization problems are solved, positions on the trains as well as in the storage are determined and sequences for all cranes are fixed. So, the crane operations are fixed and can be processed by the human crane operator without the need for further decisions. So, the proposed optimization is a step towards automation in terminals.

There are two motivations to minimize the total length of the crane moves: The crane spends a lot of energy when moved and on the other hand the length of the maintenance interval is reciprocal to the total length of the crane moves. Spending energy and having a lot of maintenance both cause costs.

The optimization of operative planning of rail-road-terminals is a multicriterial optimization problem. The objective is composed of the following main elements:

- Maximize train utilization (L)
- Minimize energy consumption and maintenance of the RMGs (L,S,C)
- Minimize time consuming handling/situations (L,S)
- Minimize violations of closing for cargo deadlines (C)
- Minimize times of truck circulations (C,S,L)

In brackets the capitals L, S and/or C denote that load planning (L), storage planning (S) or crane planning (C) affects the component of the objective.

Currently, the operative planning in German and European rail-road terminals is mainly done without optimization software. But e.g. the DUSS who operates most terminals in Germany uses a software that mainly organizes and controls the data transfer which is called control system for container-loading stations in intermodal transport (BLU)[1]. With the extension BLU-OPTI a first optimization is included to minimize the total length of the crane moves.
The IT-company INFORM offers a software called SyncroTESS[2] that also organizes the data management and optimizes the transshipment of containers and also the load planning of trains.

[1]`http://www.berghof-gruppe.de/Automation/en/Products/Process+Engineering/Railway+engineering+and+Control+systems/BLU.html`

[2]`http://www.inform-software.de/produkte/syncrotess`

3 Storage planning

Storage planning defines storage positions in a potentially partially filled storage area that consists of one or multiple storage lanes for a set of load units that have to be stored. Load units can be stored at the ground level and some load units (containers and some swap bodies) can also be stacked on top of other load units. But the stacking height is limited due to the cranes.
In rail-road terminals the storage planning has to be done when trains are unloaded or trucks deliver load units for trains that are not yet in the terminal.

Contributions. As there are many variants of the storage planning problem, we start with introducing a classification scheme to distinguish them. For several theoretical variants of the storage planning problem we analyze the complexity and provide polynomial algorithms if possible. The main focus of this section is a practical storage planning problem that arises in German and European rail-road terminals. For this problem we provide different MIP models and a heuristic approach. We analyze and compare results for these methods. For smaller instances we can show that our heuristic is able to find solutions that are close to optimality for most instances. For larger instances our heuristic is able to provide feasible solutions for more instances and solutions with better quality than CPLEX could do using the MIP-models.

Overview. We start with introducing the storage planning problem in Section 3.1 and a short literature overview is presented in Section 3.2. A classification scheme is introduced in Section 3.3. The complexity of different storage planning problems is analyzed in Sections 3.4 to 3.6. The following sections treat the practical storage planning problem. The practical setting is described in Section 3.7 and MIP models are presented and discussed in Section 3.8. A heuristic for the practical problem is introduced in Section 3.9 while in Section 3.10 computational results for real world instances are presented. Finally, a list of storage planning notations is given in Section 3.11.

3.1 Problem definition

Given is a three-dimensional rectangular storage area (cf. Figure 3.1) with m storage lanes that normally all have the same length L (discretized into decimetre positions). The stacking height is limited by a parameter b. There is a set $\mathcal{C} = \{1, \dots, n\}$ of available load units (arriving by one or several trains or trucks) which have to be placed into the storage area. Note, that the load units are transported on trains or trucks with empty spaces in between them. This can either mean that load units are transported with small gaps in between them or that one or several positions on the train are empty. According to the arrival time of the trains (and possibly also according to some control process of the load units after arriving in the terminal) the load units $i \in \mathcal{C}$ have arrival times a_i. Each load unit $i \in \mathcal{C}$ is characterized by a length ℓ_i and an origin (left corner) position O_i on the train. Note, that widths are the same for all load units and we normally assume that all load units have the same height. We assume that no load unit on a train (or a truck) exceeds the borders of the terminal when they have to be transported by a crane, which means that $0 \leq O_i \leq L - \ell_i \ \forall i \in \mathcal{C}$. So, if a train is longer than L, it has to be split to enter the terminal. Additionally, some load units $\mathcal{C}^{fix}$ may already be stored in the area. For each $i \in \mathcal{C}^{fix}$ besides its length ℓ_i a 3-tuple (lane, starting position in the lane, stacking level) describing the coordinates of its position in the storage area is known. We assume that the load units of the set $\mathcal{C}^{fix}$ will not be moved at all, so they stay at their fixed 3-tuple positions. Furthermore, the set of all load units $\mathcal{M}$ is composed of the load units to be stored $\mathcal{C}$ and the load units that are initially stored in the terminal $\mathcal{C}^{fix}$ ($\mathcal{M} = \mathcal{C} \cup \mathcal{C}^{fix}$).

We call a potential storage position (again a 3-tuple of lane, starting position and level) for one load unit a location. Without further restrictions we have $m \cdot b \cdot L$ locations (where L is measured in decimetres). Note, that these locations cannot be used all at the same time as load units have a length of more than 1dm and load units are not allowed to overlap.

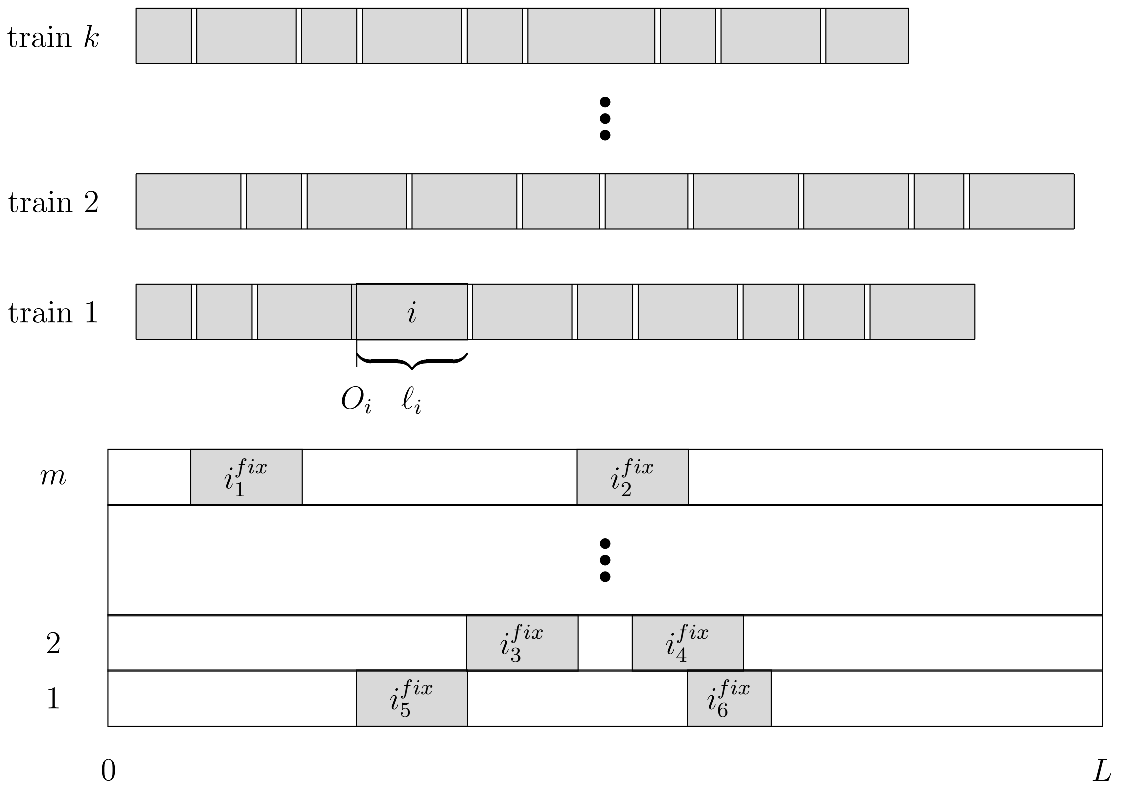


Figure 3.1: Schematic representation of a storage area with gray rectangles representing load units (stacking is not represented)

The goal is to assign each load unit $i \in \mathcal{C}$ to a location $\alpha(i)$ in the storage area such that no two load units overlap and a certain objective function is optimized. If c_{ip} denotes the assignment costs for placing load unit i to location p, the total assignment costs $AC = \sum_{i=1}^{n} c_{i\alpha(i)}$ may be minimized. Special situations arise if the costs c_{ip} are transportation costs based on a metric where the triangle inequality holds. Such transportation costs may e.g. be given as Euclidean or Manhattan distances. We consider four types of transport costs. $TC^1(x, y, z)$ denotes that the sum of the transportation costs in x-, y-

and z- direction are considered, while $TC^2(x, y, z)$ denotes the Euclidean distance and $TC^\infty(x, y, z)$ is the maximum of the distances. If an arbitrary cost function is considered, this is indicated by $TC(x, y, z)$. Note, that also a cost function on a lower dimensional space can be considered. E.g. $TC(x, y)$ means that only moves in x- and y-direction are accounted, while $TC(x)$ indicates that only the x-direction is considered for the transport costs.
Additionally, the total number $\#SI^{>1}$ (stacked items) of load units stacked in levels above the ground level may be minimized. In this situation we assume that the departure times of the load units are not known, so stacking always implies the risk of unproductive load unit handling or so-called reshuffling. If additionally departure times d_i are known for all $i \in \mathcal{C} \cup \mathcal{C}^{fix}$, another possible objective is the number of unordered stackings:

- for $\#US$ we count all pairs of load units (i, j) were load unit i is on top of load unit j (with no other load unit between them) and the departure time of load unit j is earlier than the departure time of load unit i ($d_j < d_i$). Note, that $\#US$ is just a lower bound for the number of reshuffles, but more reshuffles may be necessary to unload a storage area.

Consider, for example, the situation with two stacks in Figure 3.2 (load units are numbered according to the sequence of departures (d_i), i.e. at first load unit 1 has to be retrieved, then load unit 2, etc.). The value $\#US$ is one for the left stack as only the stacking of load unit 7 on top of load unit 3 counts. For the right stack the value $\#US$ is two for 6 on top of 5 and 5 on top of 1.

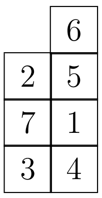


Figure 3.2: Example for calculating unordered stackings

According to the stacking conditions we assume that exactly one load unit may be stacked on top of another. Furthermore, we consider four situations:

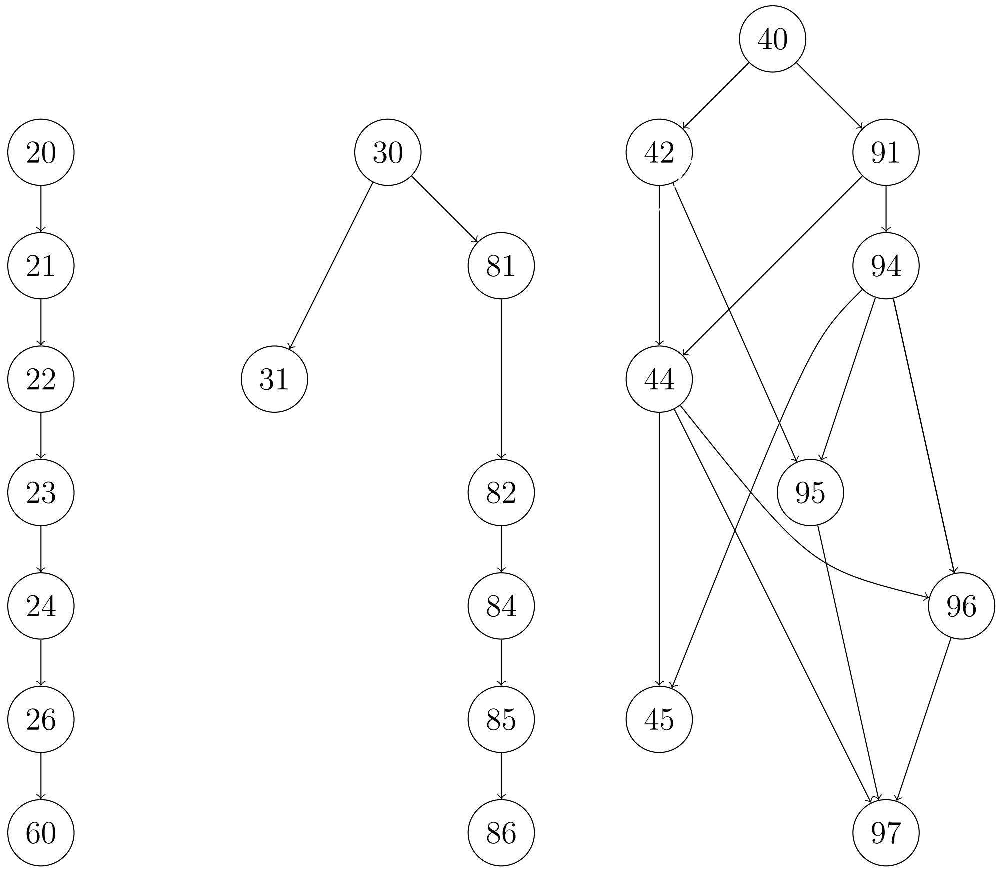
- Arbitrary stacking constraints have to be respected (e.g. certain load units may not be stacked). In this case for all pairs of load units $i, j \in \mathcal{M}$ the binary parameter s_{ij} equals 1 iff load unit i can be stacked on top of load unit j (i.e. if $s_{ij} = 0$, it is not allowed to stack i on top of j) These stacking constraints are the most general stacking constraint as the stacking restrictions are defined for pairs of load units without any further restrictions.

- A special situation is to consider transitive stacking conditions. In the situation of transitive stacking conditions, if load unit i can be stacked on top of j and j can be stacked on top of load unit h, then also i can be stacked on top of h (transitivity). For the stacking parameter s_{ij} this means, if $s_{ij} = s_{jh} = 1$ then also $s_{ih} = 1$. Transitive stacking constraints can be motivated by length, weight or temporal restrictions. Length restrictions can be that the upper load unit length is not allowed to exceed the length of any lower load unit in the stack below or restrictions according to the overhangs above the corner castings. Analogously for weight restrictions the weight of the upper load unit may not exceed the weight of any lower load unit. Temporal constraints may be caused by arrival and departure times of load units. They may state that the interval that the upper load unit stays at the terminal is contained in the stay interval of the lower load unit.

- For practical stacking restrictions in addition to the overall length ℓ_i the distance in between the corner castings $\hat{\ell}_i$ for $i \in \mathcal{M}$ is given as well as the left and the right overhang. All these length values are listed in Table 2.1 for UIC load units and the structure of a container (or swap body) is visualized in Figure 2.3. According to the notation in the figure $\hat{\ell}_i = L2$ and the overhangs are $L3$ and $L4$.

  Then the practical stacking restrictions state that for stacked load units the corner casting distances $\hat{\ell}$ have to be the same. Additionally, the lengths of the overhangs of the upper load unit are not allowed to exceed the lengths of those of the lower load unit. So, if load unit i is stacked on top of load unit j: $\hat{\ell}_i = \hat{\ell}_j$ and $L3_i \leq L3_j$ and $L4_i \leq L4_j$. Furthermore, the parameter s_i for $i \in \mathcal{M}$ states if load unit i is stackable. If $s_i = 1$, load unit i may be stacked on top of another load unit and another load unit may be stacked on top of load unit i (if also the conditions above are fulfilled). On the other hand, if $s_i = 0$, load unit i must be stored at the ground level, and no other load unit may be stacked on top of it. Load units that are not stackable are most swap bodies and all trailers. For the UIC load units listed in Table 2.1 Figure 3.3 shows a graph representation of the stacking restrictions. Each group of equal size load units (with equal corner casting distance and length of the overhangs) is represented as a node in the graph. So, each node potentially represents several load units. If the load units represented by node i can be stacked onto the load units of another node j, an arc goes from the node i to the node j. This means that between nodes an arc exists, if for the represented load units $s_{ij} = 1$. Furthermore, the graph is transitive, but we did not draw all the transitive arcs to keep the graph clear. The practical stacking restrictions are a special type of transitive stacking restrictions.

- If for transitive stacking restrictions all pairs of load units are comparable, the stacking restrictions define a total order. Pairs of load units i, j are comparable, if they can be stacked in at least one direction ($s_{ij} + s_{ji} = 1$ for all $i, j \in \mathcal{M}$). So, either i can be stacked onto j or j can be stacked onto i but it can also be the case that both stacking directions are feasible. Stacking restrictions which define a total order on the load units can be motivated by single length or weight restrictions. E.g. if the length (or the weight) of the upper load unit is not allowed to exceed the length (or the weight) of the lower load unit. This is the case if we only consider symmetric load units with the same corner casting distance. The symmetric load units with the corner casting distances of $20, 30$ and 40 feet are listed in the upper half of Table 2.1). Note, that a total order is a special case of the practical stacking restrictions.



(a) 20 feet corner casting (b) 30 feet corner casting (c) 40 feet corner casting

Figure 3.3: Graph representation of stacking restrictions for the 23 UIC load units

The assumption that a load unit may only be stacked on a single other one in the level below is not motivated by physical real-world restrictions. In reality, larger containers can be stacked on top of smaller containers as long as all the corner castings of the upper containers are exactly positioned above corner castings of any lower container, e.g. a 40 feet container can be stacked on top of two 20 feet containers that are placed beside each other as close as possible. Nevertheless, stacking one container on top of two smaller ones enlarges the risk of reshuffling, as the upper container has to be reshuffled if one of the lower containers has to be delivered. For this reason, stacking a container on two smaller ones is not allowed in German rail-road terminals.

Furthermore, if two load units are stacked, they must have the same starting x-position within the corresponding lane (i.e. the load unit above is moved to the left as much as possible). In practice this is not completely correct, since the corner castings of stacked load units must have the same position. This may have an impact on transportation costs.

According to the spatial organization of the storage lanes one can look at two special cases where the terminal is organized in grids:

- The storage area is organized as fixed grid, i.e. consists of r fixed subareas with predefined lengths $g_1, \dots, g_r$. In this case the load units must be placed in the subareas such that they do not exceed their borders.

- In a special situation it may only be allowed to place at most one load unit in each subarea. This case especially applies if all load units have the same length ℓ. Then a grid consisting of subareas with length l can be introduced dividing the storage area into fixed slots for the load units. Even if the load units have different lengths, a policy of the terminal might be that at most one load unit may be placed inside each subarea (independent of the unused space left).

An additional terminal policy may be, that every time a new load unit is placed in a lane, it must adjoin another load unit that is already placed in the lane (except the case where the lane is empty). This policy can be relaxed by allowing to place a load unit in a lane if it either adjoins another load unit or a minimum distance to other load units is not undercut. The adjoin policy aims to minimize unused gaps between load units that are too small to position any load unit inside. This is done as unused gaps worsen the utilization of the storage area.

As special cases of the storage planning problem the following additional constraints may be considered:

- Safety distances must be respected between certain load units that are loaded with dangerous goods.
- The load units have different heights and there is an additional height limit for the stacks (besides the maximum number of load units b to be stacked).
- The load unit weights are known and there is a maximum weight limit for load unit stacks (caused by the ground stability).
- Some load units may only be positioned at special locations that e.g. provide a power plug or provide special safety systems for dangerous goods.

3.2 Related literature

In this section we give a short overview on the literature concerning storage planning. We start summarizing optimization approaches in rail-road and maritime terminals and continue with some complexity results.

An overview of optimization approaches in rail-road terminals has recently been provided by Boysen et al. [24]. In this context of rail-road terminals, Jaehn [57] is the only publication that considers a storage loading problem. Here, the storage area consists of parallel lanes that are organized in fixed grids of different lengths. Due to the low utilization of the terminal, containers are mostly stored at the ground level and stacking is not considered in detail. It is assumed that a grid subarea may only be (partially) covered by one load unit (independent of its length), but since some grids are smaller than the maximum load unit length, a load unit may also cover up to two subareas. Jaehn shows that the storage planning problem is NP-hard in the strong sense if a storage area consisting of a single lane is partially filled at the beginning of the loading process and stacking is not allowed or the number of load units which are not stored at the ground level has to be minimized in a setting without grid policy. For a storage area which is initially empty the problem is NP-hard in the ordinary sense, if the storage area consists of two lanes with identical length. To solve the planning problem the author provides a MIP-model and provides and tests different heuristics for the grid based storage planning problem. The aim of Jaehn's study is to determine the distribution of different subarea lengths and their positions in the storage lanes.

There are many papers treating container storage or stacking problems in maritime terminals (see [82] for an overview). The situation of maritime terminals differs from the situation in rail-road terminals in two major points. In maritime terminals two [68] or three [37] different container types are considered and according to the small number of different containers the storage area is mostly organized in fixed grids. Furthermore, there is no literature in the area of maritime terminals that treats stacking restrictions according to load unit length as they occur in rail-road terminals.
For seaport container terminals stacking rules are proposed by e.g. [37] or [21]. Here, additionally departure times are given and the aim is to avoid stacks that cause reshuffles. Reshuffles are necessary if a container is stacked on a container with an earlier departure time. A reshuffle then means to move the upper container with later departure time away in order to access the container below. Beside the minimization of the number of reshuffles, also spatial aspects of the storage (mainly the distance to the quay-side) are relevant. However, stacking constraints according to load unit length are not considered in these studies.

Storage planning problems can be categorized in storage loading, unloading and pre-marshalling problems (see [69]). Storage loading describes that a set of load units has to be stored in a stacking area. Unloading a storage area means to provide a set or a sequence of load units from the storage area at an exit of the storage area. While in storage unloading load units are immediately delivered when they can be accessed, in pre-marshalling all load units stay in the terminal but have to be ordered to enable a fast unloading of the storage area afterwards (e.g. pre-marshalling takes place in the night and the next day load units are delivered).

As observed in Lehnfeld & Knust [69], up to now almost no complexity results for storage loading problems based on stacks have been published. On the other hand, for storage unloading problems or combined loading/unloading problems some results are known (cf. [69]). For example, Caserta et al. [33] deal with a basic unloading problem (the so-called "blocks relocation problem") where containers are stored in stacks and have to be retrieved in a given order. It is shown that minimizing the number of reshuffles is NP-hard. In the so-called "container stowage problem", a vessel visits several ports consecutively. At each port, a set of containers has to be retrieved from the vessel and another set of containers has to be stored on it. Avriel et al. [7] derived complexity results by relating the problem to coloring of circle graphs. For the problem with unlimited b and at most 3 stacks it can be decided in polynomial time whether a solution without relocation exists. On the other hand, for every fixed number of stacks greater than 3 this problem is NP-complete.

Delgado et al. [38] consider a loading problem where a container ship has to be loaded with a set of containers. According to the vessel's stability, weight and height restrictions are given for each stack. Since some containers need a power plug, there are so called location restrictions as well. Location restrictions state that not all locations may be feasible for some load units. But for each load unit a set of feasible locations is specified. The objective is to minimize a weighted sum of four components penalizing unordered stackings, storing different kinds of containers in the same stack, opening a stack and storing non-reefer containers in a power plug location. By a reduction from the bin packing problem it is shown that the subproblem of minimizing the number of newly opened stacks is already NP-hard.

Kim et al. [61] deal with a loading problem where a sequence of containers has to be stored in a storage yard. It is assumed that each container belongs to one of three weight groups but at the time of arrival the weight group is not known. The objective is to minimize the expected number of reshuffles. A dynamic programming approach has been developed based on the probability of the weight group of the next arriving container. Kang et al. [59] tackle a similar problem where the incoming containers have uncertain weights. Based on a probability distribution, a simulated annealing algorithm tries to find a good stacking strategy regarding different stack types of different weight combinations.

Note, that the storage planing problem also has similarities to some loading problems that appear in other practical applications like container ships, warehouses, steel yards or tram depots.

3.3 $\alpha|\beta|\gamma$-notation

In this section we introduce a classification scheme for storage planning problems occuring in rail-road terminals. The classification we propose is based on the idea of an $\alpha|\beta|\gamma$ three-field notation. The first three-field notation was introduced for scheduling problems by [53]. The motivation for our notation is to analyze the complexity of different storage planning problems and to propose a classification for enabling easy and clear descriptions of storage planning problems in rail-road terminals. The notation has some similarities to the notation of [69], some basic assumptions are skipped and specific details of the situation in rail-road terminals are considered. In the notation of [69] it is assumed that for the storage planning problems items of equal length are considered and that the storage area organized in fixed stacks. So, the location x- and y-positions are substituted by stacks as no positioning of the load units in lanes has to be done. A organization of the terminal in stacks can be modeled in our notation but is not a basic assumption. Furthermore, we consider load units of different length and special stacking restrictions according to load units length.
In our notation (as in [69]) α describes properties of the storage area. While the load unit properties are listed in the β-field, the objective is described in the last field γ.

In the field α characteristics of the storage area are described, e.g. about storage lanes, the maximum stacking height, stored load units, grid organizations of the lanes and

possible adjoin-policies.
The first element α_1 considers the number of storage lanes.

$$\alpha_1 = \begin{cases} m = m', & \text{the number } m \text{ of storage lanes is fixed to } m' \in \mathbb{N} \\ \circ, & \text{the number } m \text{ of storage lanes is part of the input} \end{cases}$$

The next element α_2 considers the maximum stacking height.

$$\alpha_2 = \begin{cases} b = b', & \text{the maximum stacking height } b \text{ is fixed to } b' \in \mathbb{N} \\ b = \infty, & \text{the stacking height is not limited} \\ \circ, & \text{the maximum stacking height } b \text{ is part of the input} \end{cases}$$

The element α_3 provides information about the initial storage situation.

$$\alpha_3 = \begin{cases} \mathcal{C}^{fix} \neq \emptyset, & \text{fixed load units are stored in the storage area} \\ \mathcal{C}^{fix} = \emptyset, & \text{the storage area is initially empty} \\ \circ, & \text{the storage area may be empty or contain fixed load units} \end{cases}$$

The next element distinguishes four different grid policies.

$$\alpha_4 = \begin{cases} \textit{fixed-grid}(g_1, \dots, g_r), & \text{the area is organized in a fixed grid of } r \text{ subareas with} \\ & \text{the lengths } g_1, \dots, g_r \\ \textit{fixed-grid-}1(g_1, \dots, g_r), & \text{additionally means that at most one load unit may be} \\ & \text{positioned within each subarea} \\ \textit{fixed-grid}(g), & \text{the area is organized in a fixed grid where all subareas} \\ & \text{have the same length } g \\ \textit{fixed-grid-}1(g), & \text{additionally the number of load units per subarea is at} \\ & \text{most one} \\ \circ, & \text{load units may be positioned at arbitrary dm-positions} \end{cases}$$

Another storage policy that only occurs in combination with $\textit{fixed-grid}(g_1, \dots, g_r)$ or $\textit{fixed-grid}(g)$ is the adjoin-policy, here we consider two variants in the element α_5.

$$\alpha_5 = \begin{cases} \textit{adjoin}, & \text{load units have to adjoin other load units or a grid border} \\ \textit{adjoin}(e), & \text{load units have to adjoin other load units or a grid border unless} \\ & \text{they have a minimum distance to other load units of at least } e \\ \circ, & \text{load units may be stored at arbitrary locations} \end{cases}$$

Furthermore, the element α_6 may specify a weight limit for load unit stacks.

$$\alpha_6 = \begin{cases} \textit{weight-limit}, & \text{there is a weight limit for load unit stacks} \\ \circ, & \text{the weight of load unit stacks is not limited} \end{cases}$$

If *weight-limit* is specified, the weight of load unit stacks is limited to G and for each load unit g_i denotes the load unit weight (for $i \in \mathcal{C} \cup \mathcal{C}^{fix}$).
Similar, in element α_7 an additional height limit can be described

$$\alpha_7 = \begin{cases} \textit{height-limit}, & \text{there is an additional height limit for load unit stacks} \\ \circ, & \text{the stack height is not limited (except } b) \end{cases}$$

If *height-limit* is specified, the height of load unit stacks is limited to H and for each load unit h_i denotes the load unit height (for $i \in \mathcal{C} \cup \mathcal{C}^{fix}$).

The field β describes parameters of the load units, e.g. about the length of load units, the corner casting distances, the stacking restrictions, safety distances between load units or the number of trains.

The first element β_1 of the field β considers the length of the load units.

$$\beta_1 = \begin{cases} \ell_i = \ell, & \text{all load units have the same length } \ell \\ \circ, & \text{load units have arbitrary lengths} \end{cases}$$

The next element β_2 considers stacking conditions.

$$\beta_2 = \begin{cases} s_{ij}, & \text{an arbitrary matrix } (s_{ij}) \text{ is given} \\ s_{ij}^{trans}, & \text{a transitive matrix } (s_{ij}) \text{ is given} \\ s_{ij}(\hat{\ell}), & \text{stacking restrictions are motivated by practical corner casting} \\ & \text{distances and overhangs} \\ s_{ij}^{to}, & \text{matrix } (s_{ij}) \text{ defines a total order} \\ \circ, & \text{it is allowed to stack load units arbitrarily} \end{cases}$$

β_3 specifies arrival times of the load units.

$$\beta_3 = \begin{cases} a_i, & \text{the load units arrive at different times at the terminal} \\ \circ, & \text{all load units arrive at the terminal at the same time} \end{cases}$$

The element β_4 considers safety distances of load units.

$$\beta_4 = \begin{cases} safety\text{-}distance, & \text{safety distances have to be kept between load units} \\ & \text{containing dangerous goods} \\ \circ, & \text{no safety distances have to be respected} \end{cases}$$

The element β_5 specifies the number of trains to unload.

$$\beta_5 = \begin{cases} q = q', & q' \text{ trains have to be unloaded} \\ \circ, & \text{the number of trains to be unloaded is part of the input} \end{cases}$$

Finally, the field γ describes the objective function, e.g. transport or stacking costs.

- $-$: only feasibility is important
- AC: total assignment costs
- $TC^1(x, y)$: total transportation costs as sum of the costs in x- and y-direction
- $TC^2(x, y)$: total euclidean transportation costs in x- and y-direction
- $TC^\infty(x, y)$: total maximum transportation costs of x- and y-direction

- $TC(x, y)$: one of the transportation costs $TC^1(x, y)$, $TC^2(x, y)$ or $TC^\infty(x, y)$
- $TC(x)$: total transportation costs in x-direction
- $\#SI^{>1}$: total number of load units that are not stored at the ground level
- $\#US$: total number of unordered load unit pairs that are in adjacent levels (here for all load units departure times d_i are given for all $i \in \mathcal{C} \cup \mathcal{C}^{fix}$)

The different elements of the objective may be combined. For example, $w_1 \cdot TC(x, y) + w_2 \cdot \#SI^{>1}$ means that a weighted sum of the transportation cost and the number of stacked load units is the objective or $Lex(TC(x, y), \#SI^{>1})$ means that the transport costs and the number of stacked load units have to be optimized lexicographically.

3.4 Problems with uniform load unit lengh and fixed stacks

In this section we analyze the complexity of storage loading problems with uniform load unit length ($\ell_i = \ell$) where the storage area is organized in fixed grids with subareas capable to hold load units of the length ℓ ($fixed\text{-}grid(\ell)$). Note, that for $fixed\text{-}grid(\ell)$ the grid subareas can be interpreted as fixed stacks. Recall from Section 3.3 that the policy $fixed\text{-}grid(\ell)$ implies that one cannot decide where to position a load unit in the area, but which stack to use for putting a load unit. For example, the considered predefined arrangement of stacks is often faced in maritime settings of yards or ships, where mostly 40 feet containers have to be handled.

For this case of $fixed\text{-}grid(\ell)$, we assume that the length of the terminal L is a multiple of ℓ. We introduce the parameter $S = \frac{L \cdot m}{\ell}$ which is the number of stacks in the terminal and $n^{fix} = |\mathcal{C}^{fix}|$ which is the number of load units that are initially stored in the terminal. We assume that $S < n + n^{fix}$ holds which implies that not all load units can be positioned at the ground level but some have to be stacked. Otherwise, we are not treating a real stacking problem. The results in this section are part of Bruns et al. [28].

In the field α for all instances considered in this section $fixed\text{-}grid(\ell)$ is listed. Furthermore, we assume equal length load units, so $\ell_i = \ell$ is listed in field β. Note, that it does not make a difference if we consider the situation $fixed\text{-}grid(\ell)$ or $fixed\text{-}grid\text{-}1(\ell)$ as due to the uniform load unit lengths also for $fixed\text{-}grid(\ell)$ only one load unit can be positioned per level in each subarea. All results of this section are based on $fixed\text{-}grid(\ell)$, nevertheless without transportation costs and with $\mathcal{C}^{fix} = \emptyset$ the results can be generalized to the case without grid policy, as the grid policy offers the maximum number of stacks. For transportation costs it is feasible and may be better to exceed subarea borders if fixed grids do not have to be respected.
We structure our analysis in sections for problems with stacking limit $b = 2$ (Section 3.4.1), for stacking limit $b = 3$ (Section 3.4.2) and for problems with arbitrary stacking limit b (Section 3.4.3).

3.4.1 Problems with stacking limit $b = 2$

In this section we consider problems where the stacking limit is $b = 2$. Such a situation is typical, for example, for rail-road container terminals, where often at most 2 containers may be stacked. We consider arbitrary stacking constraints s_{ij} and the objective functions $\#SI^{>1}$ and $\#US$. The storage area may be empty or some fixed load units $\mathcal{C}^{fix}$ may be contained in it.

Theorem 3.1. *The feasibility problem $b = 2, fixed\text{-}grid(\ell), \mid \ell_i = \ell, s_{ij} \mid -$ with arbitrary stacking constraints s_{ij} can be solved as a maximum cardinality matching problem in $\mathcal{O}\left((n + n^{fix})^{2.5}\right)$ time. A solution minimizing $\#SI^{>1}$ can be derived from a feasible solution in $\mathcal{O}(n)$ time.*

Proof. We introduce the undirected graph $G = (V, E)$, in which the nodes V correspond to the load units $\mathcal{M} = \mathcal{C} \cup \mathcal{C}^{fix}$ and edges $E = \{\{i, j\} \mid i, j \in V\}$ connect pairs of nodes corresponding to stackable load units. For incoming load units $i, j \in \mathcal{C}$ this means that i must be stackable on j or j must be stackable on i (i.e. $s_{ij} + s_{ji} \geq 1$). For load units $i \in \mathcal{C}$ and $j \in \mathcal{C}^{fix}$ load unit i must be stackable on top of j (i.e. $s_{ij} = 1$).

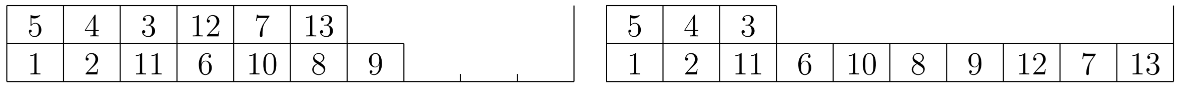
By finding a matching of maximum cardinality in the graph G, we get a solution with the largest number of stacks containing two load units. Additionally, the load units that are not matched have to be stored at the ground level. A feasible solution exists if the number of edges in the matching plus the number of unmatched nodes is not larger than the number of stacks S. Since the number of nodes is $n + n^{fix}$, a maximum cardinality matching can be computed in $\mathcal{O}\left((n + n^{fix})^{2.5}\right)$ time (cf. Even & Kariv [41]).

The optimization problem $b = 2, fixed\text{-}grid(\ell) \mid \ell_i = \ell, s_{ij} \mid \#SI^{>1}$ minimizing the number of load units stacked above the ground level can be solved as follows. We start with a feasible matching solution and re-allocate as many load units of the set $\mathcal{C}$ as possible to the ground (i.e. until all stacks contain at least one load unit, see Example 3.1). $\square$

The problem without fixed stacks can be solved with the same approach if the storage area is initially empty ($\mathcal{C}^{fix} = \emptyset$). In our notion this is the problem $b = 2, \mathcal{C}^{fix} = \emptyset \mid \ell_i = \ell, s_{ij} \mid \#SI^{>1}$. The case without fixed grids but with a partially filled storage area is different, as load units may already be positioned such that they violate grid borders which implies spatial restrictions.

Example 3.1. In the following, we illustrate how a solution minimizing $\#SI^{>1}$ can be constructed from a feasible solution to the feasibility problem. We consider an example with $S = 10$ stacks and $n = 13$ load units (numbered from 1 to 13). We assume that in the maximum matching the edges $\{1, 5\}$, $\{2, 4\}$, $\{3, 11\}$, $\{6, 12\}$, $\{7, 10\}$, $\{8, 13\}$ are chosen. Furthermore, load unit 9 is not matched at all.

The solution to the matching problem can be interpreted as the stacking solution shown in Figure 3.4(a). In this solution three stacks are left empty, so three stacked load units can be moved to the ground level. This results, for example, in the solution shown in Figure 3.4(b) where the number of load units stacked above the ground level is $3 = 13 - 10$. Obviously, for $S = 10$, $b = 2$ and $n = 13$ this is optimal. $\square$



(a) A feasible solution (b) An optimal solution minimizing $\#SI^{>1}$

Figure 3.4: Constructing an optimal solution from a feasible one (Example 3.1)

Next, we consider the same problem setting but with the objective to minimize the number of unordered stackings. For this problem additionally departure times d_i are given for all load units $i \in \mathcal{M}$.

Theorem 3.2. *Problem $b = 2, fixed\text{-}grid(\ell) \mid \ell_i = \ell, s_{ij} \mid \#US$ with arbitrary stacking constraints s_{ij} can be solved as a minimum-weight perfect matching problem in $\mathcal{O}(n^3)$ time.*

Note, that for the case of an initially empty storage $\mathcal{C}^{fix} = \emptyset$ we do not have to impose $fixed\text{-}grid(\ell)$ because fixed stacks would not make a difference.

Proof. At first we introduce $2S - n - n^{fix}$ dummy load units in order to get $2 \cdot S$ load units in total. These dummy load units are used to represent empty positions in a solution and can be stacked with any other load unit (real or dummy) in any direction.

We introduce the undirected graph $G = (V_1 \cup V_2, E_1 \cup E_2)$ where V_1 and V_2 represent real ($i \in \mathcal{M}$) and dummy load units, respectively, $|V_1| = n + n^{fix}$, $|V_2| = 2S - n - n^{fix}$. Edges E_1 connect pairs of nodes corresponding to stackable real load units, edges $E_2 = \{\{i, j\} : i \in V_1 \cup V_2, j \in V_2\}$ contain all pairs involving at least one dummy load unit. For an incoming load unit $i \in \mathcal{C}$ and a fixed load unit $j \in \mathcal{C}^{fix}$ an edge $\{i, j\} \in E_1$ exists if i can be stacked on j (i.e. if $s_{ij} = 1$). For two incoming load units $i, j \in \mathcal{C}$ an edge $\{i, j\} \in E_1$ exists if i can be stacked on j or vice versa (i.e. if $s_{ij} + s_{ji} \geq 1$).

The costs c_{ij} for edges $\{i, j\} \in E_1$ with $i \in \mathcal{C}, j \in \mathcal{C}^{fix}$ are set to

$$c_{ij} := \begin{cases} 0, & \text{if } d_i \leq d_j \\ 1, & \text{otherwise.} \end{cases}$$

The costs c_{ij} for edges $\{i, j\} \in E_1$ with $i, j \in \mathcal{C}$ are set to

$$c_{ij} := \begin{cases} 0, & \text{if } (s_{ij} = 1, d_i \leq d_j) \text{ or } (s_{ji} = 1, d_j \leq d_i) \\ 1, & \text{otherwise.} \end{cases}$$

The costs for two real load units of the set V_1 are 0 if they can be stacked without inducing an unordered stacking. If i and j can only be stacked with an unordered stacking, the costs c_{ij} are equal to 1. Furthermore, we set $c_{ij} := 0$ for all edges $\{i, j\} \in E_2$, i.e. all dummy load units can be stacked at no cost. Such an edge corresponds to the situation that a stack is completely empty (two dummy load units) or a stack contains a real load unit stored at the ground level without a stacked load unit on top (one real and one dummy load unit).

Consider the minimum-weight perfect matching problem defined for the graph G. Clearly, if a perfect matching does not exist, then also no feasible stacking solution exists. If a perfect matching exists, this minimum-weight perfect matching will match all load units with the minimum possible number of unordered stackings. Indeed, due to the definition of c_{ij}, each unordered stack has cost 1, while stacks without unordered load units or those containing only one load unit have cost 0.

Based on the solution to the minimum-weight perfect matching problem, we define a solution to the storage loading problem. For edges $\{i, j\} \in E_1$ in the matching we define the order of i, j in the stack as follows. If $i \in \mathcal{C}, j \in \mathcal{C}^{fix}$ or $i, j \in \mathcal{C}$ are only stackable in one direction (i.e. $s_{ij} + s_{ji} = 1$), then they are stored in the unique possible way respecting the stacking constraints (i.e. if $s_{ij} = 1$, load unit i is stacked on load unit j). On the other hand, if load units $i, j \in \mathcal{C}$ are stackable in both directions (i.e. $s_{ij} = s_{ji} = 1$), then j is placed at the ground level and i is stacked on top if $d_i \leq d_j$; otherwise the alternative order is selected.

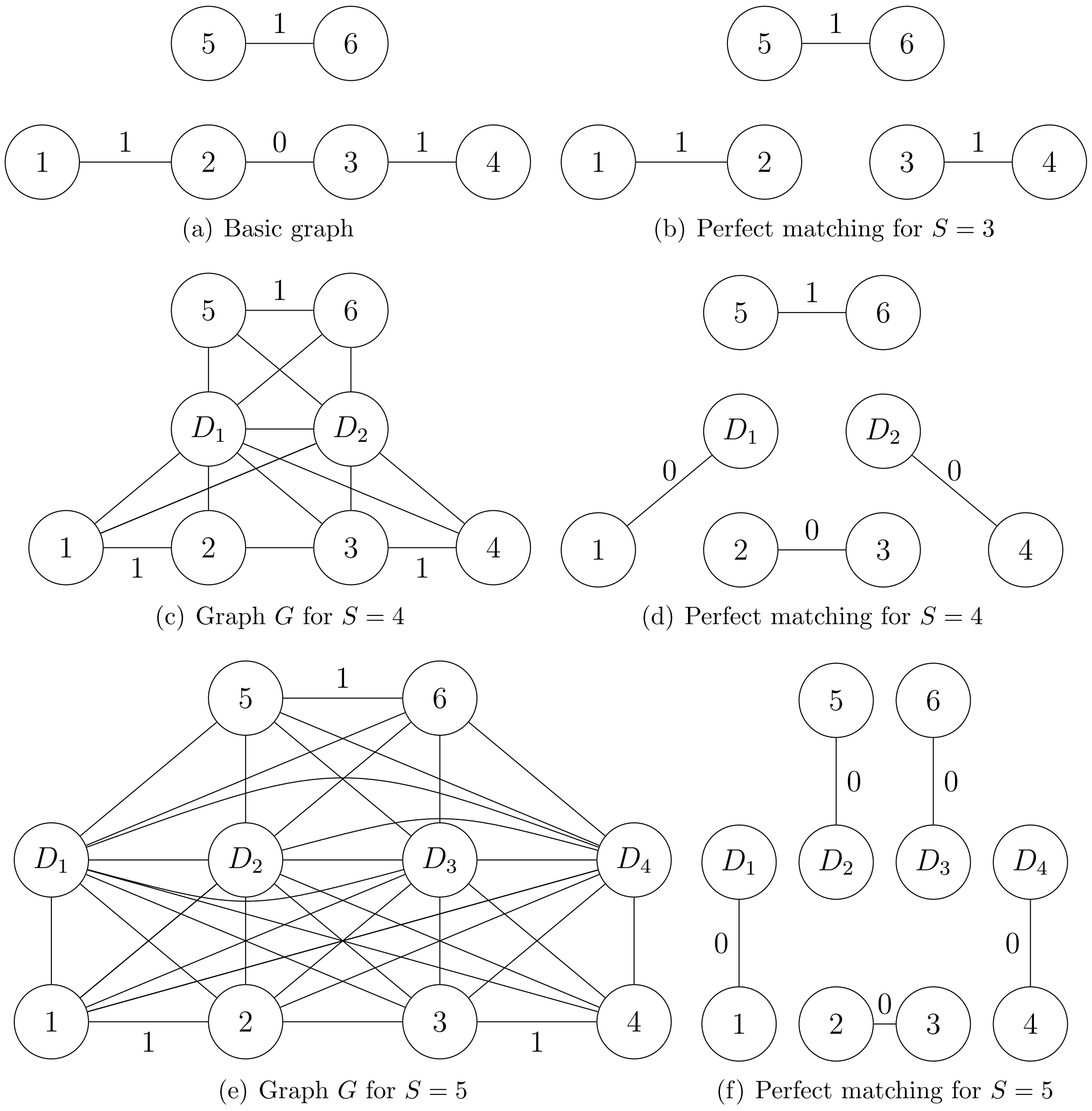
For edges $\{i, j\} \in E_2$ in the matching involving two dummy load units, we introduce a completely empty stack; for edges $\{i, j\} \in E_2$ with one dummy and one real load unit, we place the real load unit on the ground level (without any load unit on top).

Since the number of nodes is $2 \cdot S$ and we assume that $S < n + n^{fix}$, a minimum-weight perfect matching can be computed in $\mathcal{O}((n + n^{fix})^3)$ time (cf. Gabow [45] and Lawler [67]). □

Example 3.2. Consider three instances with $n = 6$ load units and $S \in \{3, 4, 5\}$, respectively. Let $s_{12} = s_{23} = s_{32} = s_{34} = s_{56} = 1$ and all other s_{ij} be zero. The delivery times are $d_1 = 4$, $d_2 = d_4 = 1$, $d_3 = d_6 = 2$ and $d_5 = 3$. In the basic graph shown in Figure 3.5(a) load units are connected if they can be stacked. Furthermore, the edges between two load units have cost 0 if the load units can be stacked without an unordered stacking and cost 1 if an unordered stacking occurs by stacking the load units. In the case of Figure 3.5(a) only load unit 2 can be stacked on load unit 3 without an unordered stacking.

The basic graph shown in Figure 3.5(a) is the graph that has to be considered if $S = 3$. Due to $2 \cdot S = n$ no dummy nodes have to be introduced and a minimum-weight perfect matching with cost 3 is shown in Figure 3.5(b). This matching can be transformed into the stacking solution shown in Figure 3.6(a). For each stack the ordering of load units is unique as they can only be stacked in one direction.

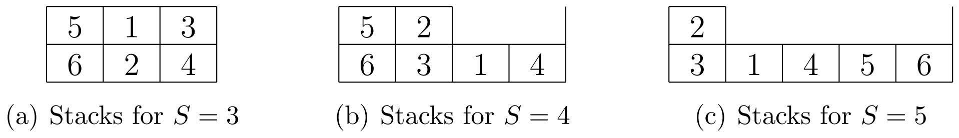
For $S = 4$ we introduce 2 dummy nodes D_1 and D_2 which are connected by edges of cost 0 to the 6 nodes corresponding to real load units. Additionally, D_1 and D_2 are also connected, which enables empty stacks. The resulting graph is presented in Figure 3.5(c). Here, we do not label edges with 0 cost for clarity. In this situation a perfect matching with cost 1 exists, see Figure 3.5(d). This matching can be interpreted as the stacking solution shown in Figure 3.6(b). Item 5 has to be stacked on top of load unit 6 since they are in one stack and $s_{65} = 0$, i.e. load unit 6 cannot be stacked on top of load unit 5. For the second stack, load units 2 and 3 are stackable in both directions, but putting 3 on top of 2 would imply an unordered stacking due to the delivery times ($d_2 < d_3$). Thus load unit 2 should be stacked on top of 3.


(a) Basic graph (b) Perfect matching for $S = 3$

(c) Graph G for $S = 4$ (d) Perfect matching for $S = 4$

(e) Graph G for $S = 5$ (f) Perfect matching for $S = 5$

Figure 3.5: Graphs and matchings of Example 3.2



(a) Stacks for $S = 3$ (b) Stacks for $S = 4$ (c) Stacks for $S = 5$

Figure 3.6: Stacking solutions of Example 3.2

For $S = 5$ we introduce the 4 dummy nodes D_1 to D_4 which are again connected by 0-cost edges to all nodes corresponding to real load units; they are also interconnected by 0-cost edges. The graph is shown in Figure 3.5(e) and a perfect matching with cost 0 is presented in Figure 3.5(f). The corresponding stacking solution is shown in Figure 3.6(c). The only stacked load units are 2 and 3, where again 2 is on top of 3 as this direction does not induce an unordered stacking. □

Up to now we assumed that one set $\mathcal{C}$ of incoming load units is given and arbitrary stacking constraints s_{ij} have to be respected. If arbitrary arrival times a_i have to be considered, the corresponding problems can be solved by the same algorithms as in Theorems 3.1 and 3.2. We simply use a modified stacking matrix $S' = (s'_{ij})$ with

$$s'_{ij} := \begin{cases} 1, & \text{if } s_{ij} = 1 \text{ and load unit } i \text{ does not arrive before load unit } j \ (a_i \geq a_j) \\ 0, & \text{otherwise} \end{cases}$$

which guarantees that a load unit arriving at a later time is not placed underneath a load unit that arrives earlier.

3.4.2 Problems with stacking limit $b = 3$

In this section we show that several problems with stacking limit $b = 3$ are strongly NP-complete.

Theorem 3.3. *The feasibility problem $b = 3, \mathcal{C}^{fix} = \emptyset, fixed\text{-}grid(\ell) \mid \ell_i = \ell, s_{ij}^{trans} \mid -$ is strongly NP-complete even for transitive stacking constraints s_{ij}.*

Proof. We prove NP-completeness by a reduction from the strongly NP-complete problem EXACT COVER BY 3-SETS (X3C), see [47]. The idea of the proof is similar to that used in [47] (Section 3.2.2) for proving NP-completeness of PARTITION INTO TRIANGLES.

An instance of X3C is given by a finite set X with $|X| = 3q$ for some integer q as well as a collection I of three-element subsets of X. The decision problem asks whether I contains an exact cover of X, i.e. a subcollection $I' \subseteq I$ where each element of X is contained in exactly one element of I'.

For an instance of X3C we construct an instance of the stacking problem with $S = q+3|I|$ stacks, stack limit $b = 3$ and $n = 3\,(q + 3|I|)$ incoming load units $\mathcal{C}$. There are no load units $\mathcal{C}^{fix}$ stored in the stacks. Note that $n = 3S$ and therefore in each feasible solution all positions in the stacks have to be filled with load units. We specify the instance of the stacking problem by a directed graph $G = (V, A)$, in which nodes V correspond to the load units $\mathcal{C}$, $|V| = n$, and arcs A define pairs of stackable load units: $(i, j) \in A$ means that i can be stacked onto j (i.e. $s_{ij} = 1$).

There are two types of nodes in V: the main nodes, one node per each element from X, and auxiliary nodes, 9 nodes per each triple from I. Each triple $c_i = \{u_i, v_i, w_i\}$ defines a so called *substitution graph* with 3 main nodes $\{u_i, v_i, w_i\}$, 9 auxiliary nodes $\{a_{i1}, \ldots, a_{i9}\}$ and collection A_i of 11 arcs shown in Figure 3.7. Note that the main nodes $\{u_i, v_i, w_i\}$

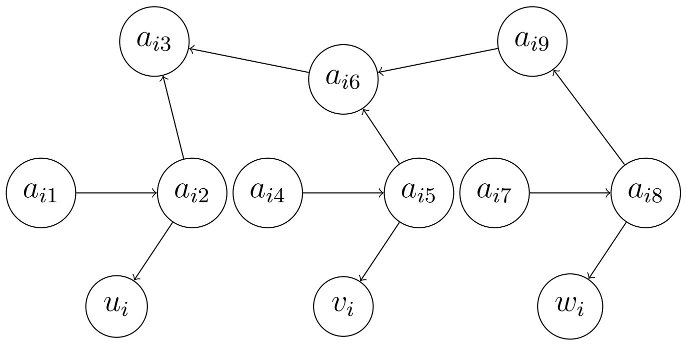


Figure 3.7: Substitution graph

may belong to several substitution graphs, but there are no arcs between the a_{ij}-nodes belonging to different substitution graphs.

Thus for the graph $G = (V, A)$ we have

$$V = X \cup \bigcup_{i=1}^{|I|} \{a_{ij} | 1 \leq j \leq 9\} \text{ and } A = \bigcup_{i=1}^{|I|} A_i.$$

In the following we show that I contains an exact cover of X if and only if the stacking problem has a feasible solution.

"⇒": Assume that X3C has an exact cover $I' \subseteq I$. If $c_i = \{u_i, v_i, w_i\} \in I'$, then in the stacking problem we build four stacks (each containing $b = 3$ elements): (a_{i1}, a_{i2}, u_i), (a_{i4}, a_{i5}, v_i), (a_{i7}, a_{i8}, w_i), and (a_{i9}, a_{i6}, a_{i3}). The corresponding paths within the corresponding substitution graph are drawn as thick red lines in Figure 3.8. Note that the load units corresponding to main nodes of c_i are "in" the stacks and cannot be used in any other stack.

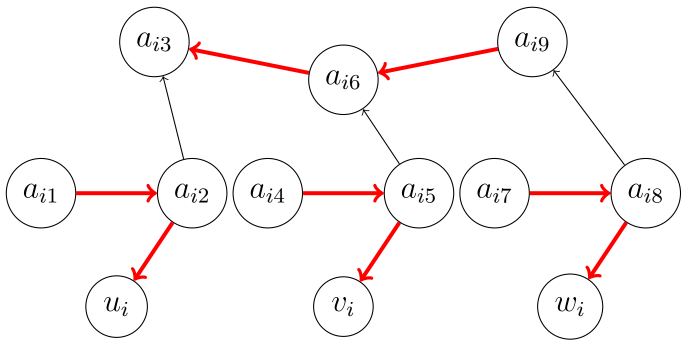


Figure 3.8: Stacks if c_i is part of the cover

If on the other hand, c_i is not part of the cover I', the load units in the corresponding substitution graph are stacked according to the thick red arcs shown in Figure 3.9. Here, the three stacks (a_{i1}, a_{i2}, a_{i3}), (a_{i4}, a_{i5}, a_{i6}) and (a_{i7}, a_{i8}, a_{i9}) are built. In this case the load units corresponding to main nodes of c_i are not included in any stack, while the load units corresponding to all auxiliary nodes are in the stacks.

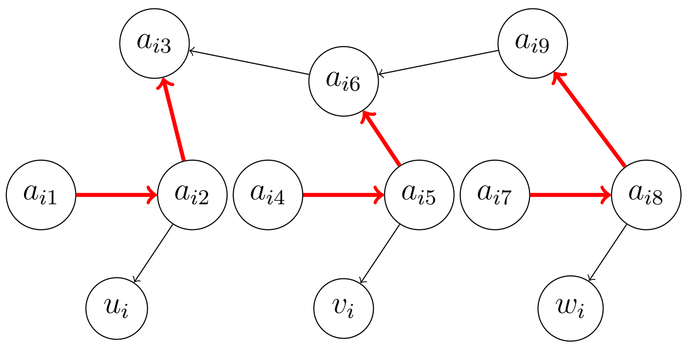


Figure 3.9: Stacks if c_i is not part of the cover

"$\Leftarrow$": Assume that conversely a feasible solution to the stacking problem exists. We construct a partition into triples which defines a solution to X3C.

First we note that due to the stacking constraints imposed by arcs A in any feasible solutions all main load units are stacked at level 1, and the load units at levels 2 and 3 correspond to auxiliary nodes.

Consider a stack with a main load unit at the ground level and an auxiliary load unit at level 2. Suppose that these load units are u_i, a_{i2}; the cases of v_i, a_{i5} and w_i, a_{i8} are similar. Let G_i be the substitution graph that contains a_{i2}. We show that in this case there are four stacks

$$(a_{i1}, a_{i2}, u_i),\ (a_{i4}, a_{i5}, v_i),\ (a_{i7}, a_{i8}, w_i),\ (a_{i9}, a_{i6}, a_{i3}) \tag{3.1}$$

for the substitution graph G_i, and thus we include $c_i = \{u_i, v_i, w_i\}$ in the exact cover.

(i) The stack with u_i and a_{i2} at levels 1 and 2 is of the form (a_{i1}, a_{i2}, u_i); thus there are 7 remaining auxiliary load units of G_i, not counting a_{i1},a_{i2}.

(ii) If the load unit stacked on top of v_i does not belong to G_i and the same is true for the load unit stacked on top of w_i, then 7 auxiliary load units of G_i cannot form full stacks containing $b = 3$ load units.

(iii) If the load unit stacked on top of v_i does not belong to G_i, while the load unit stacked on top of w_i does, then (a_{i7}, a_{i8}, w_i) forms a stack in G_i, and the remaining 5 auxiliary nodes of G_i cannot form full stacks.

(iv) If the load unit stacked on top of v_i belongs to G_i, while the load unit stacked on top of w_i does not, then (a_{i4}, a_{i5}, v_i) forms a stack in G_i, and the remaining 5 auxiliary nodes of G_i cannot form full stacks.

Thus, in each of the above cases we get a contradiction to the assumption that there exists a feasible solution to the stacking problem, and the only feasible stacking is given by (3.1).

In the following we prove that already the special case of transitive stacking constraints s_{ij} is NP-complete by showing that the reduction from X3C also holds if the graph

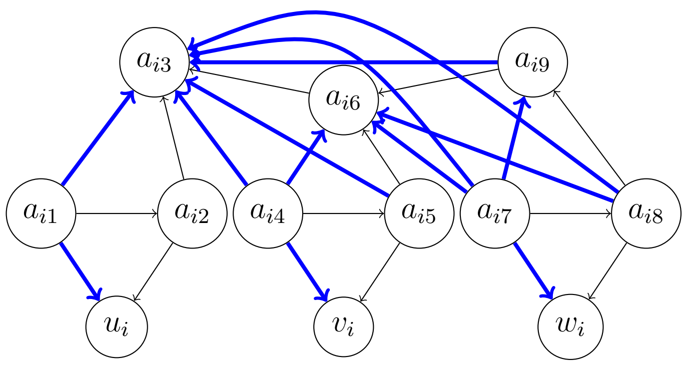


Figure 3.10: Transitive substitution graph

$G = (V, A)$ is transitive. For this purpose we consider the transitive substitution graph shown in Figure 3.10, where the transitive arcs are added as thick blue arcs.

We use similar arguments as before. Consider a load unit corresponding to the main node u_i included in a stack together with load unit a_{i2} (notice that there does not exist a stack containing 3 load units with load unit a_{i1} at level 2). Again it can be shown that there are four stacks (a_{i1}, a_{i2}, u_i), (a_{i4}, a_{i5}, v_i), (a_{i7}, a_{i8}, w_i) and (a_{i9}, a_{i6}, a_{i3}) for the substitution graph G_i, which means that $c_i = \{u_i, v_i, w_i\}$ should be included in the exact cover. Notice that properties (i) and (ii) remain the same; in (iii) we need an additional observation that a_{i7} cannot be stacked at level 2, immediately on top of w_i; similarly in (iv) we observe that a_{i4} cannot be stacked at level 2, immediately on top of v_i. $\square$

Note, that we again do not have to impose *fixed-grid*(ℓ). Without the fixed grids the storage has to be organized in the same way to hold $n = 3\,(q + 3|I|)$ load units for the situation where $\frac{L \cdot m}{\ell} = (q + 3|I|)$ and $b = 3$.

In the following we will show that if we have no stacking constraints s_{ij}, but an additional weight or height limit per stack, the problem is also strongly NP-complete.

Theorem 3.4. *The feasibility problems $b = 3, \mathcal{C}^{fix} = \emptyset$, fixed-grid$(\ell)$ $\mid \ell_i = \ell$, weight-limit $\mid -$ with a weight limit and $b = 3, \mathcal{C}^{fix}$, fixed-grid(ℓ) $\mid \ell_i = \ell$, height-limit $\mid -$ with a height limit per stack are strongly NP-complete.*

Proof. We prove NP-completeness for the first problem (the second problem can be tackled similarly) by a reduction from the strongly NP-complete problem 3-PARTITION (3-PART), see [47]. An instance of 3-PART is defined by a set $A = \{1, \ldots, 3k\}$ of $3k$ elements and a bound $B \in \mathbb{N}$. Associated with the elements $i \in A$ are numbers a_i with $\sum_{i=1}^{3k} a_i = kB$ and $B/4 < a_i < B/2$. The decision problem asks whether A contains a partition $S_1, \ldots, S_k$ such that $\sum_{i \in S_\lambda} a_i = B$ for $\lambda = 1, \ldots, k$. Note, that by the definition of a_i each set S_λ must contain exactly three elements.

For an instance of 3-PART we construct an instance of the stacking problem with $|A| = 3k$ load units and weights a_i for $i = 1, \ldots, 3k$. We introduce $S = k$ stacks with stacking limit $b = 3$ and weight limit B per stack. We show that 3-PART has a feasible solution iff a feasible solution for the stacking problem exists.

"$\Rightarrow$": Assume that 3-PART has a feasible solution. Then the $k = S$ subsets can be interpreted as S stacks, each containing three load units. These stacks are feasible since each stack meets the weight limit B and the stacking limit $b = 3$.

"$\Leftarrow$": Assume that conversely a feasible solution to the stacking problem exists. The $S = k$ stacks have to be filled by three load units per stack because the number of load units is $3k$. Since the maximum weight of a stack is limited to B and the sum of the load unit weights is $B \cdot k$, each stack has a weight of B. Thus, the load units of a stack can be interpreted as subsets with cardinality three for the partition problem, where for each subset the sum of the weights is B. □

As we again assume that the storage area is initially empty ($\mathcal{C}^{fix} = \emptyset$) the problems are also NP-complete if the grid policy is not imposed ($b = 3, \mathcal{C}^{fix} = \emptyset \mid \ell_i = \ell, \textit{weight-limit} \mid -$ and $b = 3, \mathcal{C}^{fix} \mid \ell_i = \ell, \textit{height-limit} \mid -$).

3.4.3 Problems with arbitrary stacking limit b

In this section we consider an arbitrary stacking limit $b \geq 2$ which is given as part of the input. First, we consider a storage planning problem where it is possible to assign each load unit by an easy rule to a subarea with lowest possible transportation costs. This situation occurs if only one train has to be unloaded. Additionally, we show that the problem without stacking constraints is polynomially solvable if the objective is to minimize the transportation costs $TC(x, y)$, the number of load units stacked above the ground level $\#SI^{>1}$, or a weighted sum of them. Furthermore, we prove that the problem $\textit{fixed-grid}(\ell) \mid \ell_i = \ell, s_{ij}^{to} \mid \#SI^{>1}$ is polynomially solvable in the case that the stacking constraints s_{ij} define a total order.

We start with the problem where each load unit can be directly assigned to a grid subarea with lowest transportation costs.

Theorem 3.5. *The problem $m = 1, \mathcal{C}^{fix} = \emptyset,\ \textit{fixed-grid}(\ell) \mid \ell_i = \ell, s_{ij}, q = 1 \mid TC(x, y)$ where a single train has to be unloaded in a grid setting without stacking, can be solved in $\mathcal{O}(n)$ time.*

Proof. In the following we define different cases to determine the subareas load units are assigned to according to the load units origin position. Note, that we do not need to stack load units to solve the considered problem.
For each load unit we distinguish three cases according to the source x-position:

- If $O_i \mod \ell < \frac{\ell}{2}$, the load unit has to be assigned to location $p = \lfloor \frac{O_i}{\ell} \rfloor$ with transport costs of $O_i \mod \ell$.
- If $O_i \mod \ell > \frac{\ell}{2}$, the load unit has to be assigned to location $p = \lfloor \frac{O_i}{\ell} \rfloor + 1$ with transport costs of $\ell - O_i \mod \ell$.

- If $O_i \mod \ell = \frac{\ell}{2}$ the load unit can be assigned to one of the optimal locations $p_1 = \lfloor \frac{O_i}{\ell} \rfloor$ or $p_2 = \lfloor \frac{O_i}{\ell} \rfloor + 1$ with transport costs of $\frac{\ell}{2}$ for both locations. If there is a load unit j with $O_j = O_i + \ell$, load units i and j have both the location $p_2 = \lfloor \frac{O_i}{\ell} \rfloor + 1$ as one of the optimal locations (if load unit j' has the position $O_{j'} = O_i - \ell$, the same is the case for location p_1). To resolve these conflicts we can decide for all load units with $O_i \mod \ell = \frac{\ell}{2}$ that they are transported to the left. It would also be possible to transport all to the right, we just have to do it in the same direction for all load units.

To show that the solution generated by the above described rule is feasible, we have to show that there cannot be any best location conflicts for load units i with $O_i \mod \ell \neq \frac{\ell}{2}$. We show that no load unit to the left and no load unit to the right for a load unit i with $O_i \mod \ell < \frac{\ell}{2}$ may have such a conflict:

- Assume load unit j has a conflict with i and is to the left. Then the worst case is that $O_j = O_i - \ell$ as the load units do not overlap on the train. But then the best location for j is $p_j = \lfloor \frac{O_j}{\ell} \rfloor = \lfloor \frac{O_i - \ell}{\ell} \rfloor = \lfloor \frac{O_i}{\ell} \rfloor - 1$ which is a different location. So, no conflict exists.
- Assume load unit j has a conflict with i and is to the right. Then the worst case is that $O_j = O_i + \ell$ as the load units do not overlap on the train. But then the best location for j is $p_j = \lfloor \frac{O_j}{\ell} \rfloor = \lfloor \frac{O_i + \ell}{\ell} \rfloor = \lfloor \frac{O_i}{\ell} \rfloor + 1$ which is a different location. So, again no conflict exists.

The proof is analog for load units with $O_i \mod \ell > \frac{\ell}{2}$. On the other hand, it is easy to see that the generated solution is optimal as all load units are positioned with the lowest transportation cost for the single load units. Note, that the transportation cost in y-direction is a constant value for all load units, as all load units have to be transported from the train to the only storage lane. So, we have an easy rule to assign the load units that ensures that each load unit is assigned to an optimal position which means that we constructed an optimal plan.

As we just have a single calculation for all load units, the complexity of the above described rule is $\mathcal{O}(n)$. □

Note, that we can skip the restriction of a single train if the circumstances allow to position each load unit to an optimal position. This is the case for the problems $m = 1, b = \infty,\ \textit{fixed-grid}(\ell) \mid \ell_i = \ell, \mid TC(x, y)$ with no stacking height limit and without stacking restrictions and for $b = 1, \mathcal{C}^{fix} = \emptyset,\ \textit{fixed-grid}(\ell) \mid \ell_i = \ell, q \leq m \mid TC(x)$ where we have more lanes than trains to be unloaded. Note, that we can not treat transportation cost in y-direction as these costs are no longer constant. The problem with $TC(x, y)$ is treated in Theorem 3.6.

Next we consider the problem $\textit{fixed-grid}(\ell) \mid \ell_i = \ell \mid w_1 TC(x, y) + w_2 \#SI^{>1}$ where no stacking restrictions exist and a weighted sum of the transportation costs and the number of load units stacked above the ground level has to be minimized.

Theorem 3.6. *Problem $fixed\text{-}grid(\ell) \mid \ell_i = \ell \mid w_1 TC(x,y) + w_2 \#SI^{>1}$ can be solved in $\mathcal{O}((n + S \cdot b)^3)$ time for any stacking limit b.*

Proof. This problem can be solved as a minimum-weight matching problem in the bipartite graph $G = (V_1 \cup V_2, E)$ where V_1 corresponds to the set of incoming load units $\mathcal{C}$ and V_2 contains all free positions in the storage area (as positions we consider pairs (stack,level)). If the storage area is empty, we consider Sb positions. If, on the other hand, the storage area is partially filled, we do not consider those positions that are already occupied with load units from the set $\mathcal{C}^{fix}$.

We introduce an edge $\{i, p\}$ with $i \in V_1$ and $p \in V_2$. The weight c_{ip} is defined as the cost of transporting load unit i from its origin O_i to the grid subarea (also called stack) of position p multiplied by w_1. Additionally, if the position p is not at the ground level, we add the unit cost for a stacked load unit multiplied by w_2.

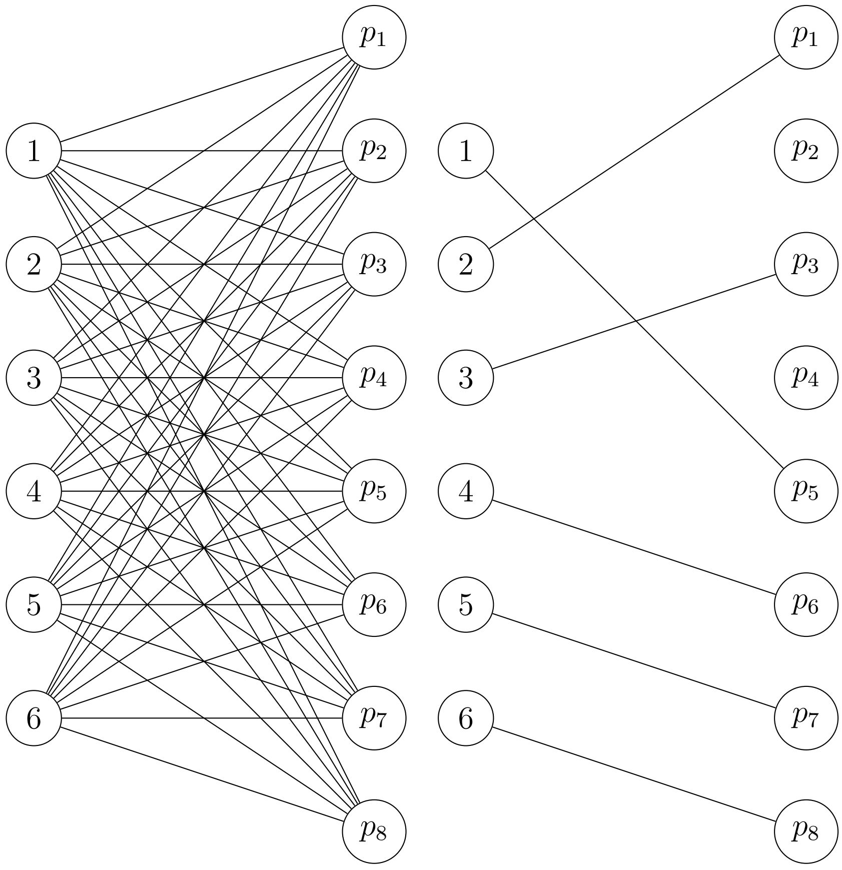
The bipartite graph G has at most $n + Sb$ nodes, and the minimum-weight bipartite matching problem for it can be solved in $\mathcal{O}((n + S \cdot b)^3)$ time (cf. Gabow [46]). Since in practical settings the stacking limit b is usually smaller than the number of load units n, this complexity is polynomial.

In an optimal solution to the matching problem it can occur that an item is stored in the k-th level without an item stored in the $(k-1)$-th level in the same stack. If $w_2 > 0$, then in an optimal solution such k satisfies $2 \leq k \leq b$; if $w_2 = 0$, then $1 \leq k \leq b$. In either case in the corresponding actual solution to the stacking problem, we can just lower all items that are stacked in the "air" until they are stacked on other items without increasing the costs. □

Note, that the matching does not have to be done if the storage setting allows to position each load unit to the stack with the lowest transportation cost. So, if the stacking height is unlimited, for problem $b = \infty, fixed\text{-}grid(\ell) \mid \ell_i = \ell \mid TC(x,y)$ we directly assign each load unit to the subarea with lowest transportation cost. This is also the case for an empty storage and a stacking limit of the number of load units to be stored n $(b = n,\ \ fixed\text{-}grid(\ell), \mathcal{C}^{fix} = \emptyset \mid \ell_i = \ell \mid TC(x,y))$.

Example 3.3. Consider an instance with $S = 2$ stacks with stacking limit $b = 4$ and $n = 6$ load units. The vertex sets have cardinalities $|V_1| = 6$ for load units and $|V_2| = 8$ for free positions. The corresponding bipartite graph is shown in Figure 3.11(a). Item vertices are numbered from 1 to 6, while position vertices are numbered from p_1 to p_8. Note, that the positions p_1 to p_4 belong to the first stack, while positions p_5 to p_8 belong to the second stack. Positions p_1 and p_5 represent the ground level positions, the other positions represent non-ground levels for stacked load units.

Assume that an optimal matching is the one shown in Figure 3.11(b). All load units are assigned and positions p_2 and p_4 stay empty. This solution can be interpreted as the stacking solution shown in Figure 3.12(a). In that solution, in the first stack level three is filled but level two is empty, which is not feasible. However, the solution can be repaired by lowering all load units that are stored in the "air". In the solution shown in Figure 3.12(b) the level of load unit 3 is changed from 3 to 2. The latter stacking solution



(a) Bipartite graph (b) An optimal matching

Figure 3.11: Bipartite graph and an optimal matching for Example 3.3

has the same cost as the one shown in Figure 3.12(a) since the costs for assigning a load unit to the second or third level in a stack are the same. □

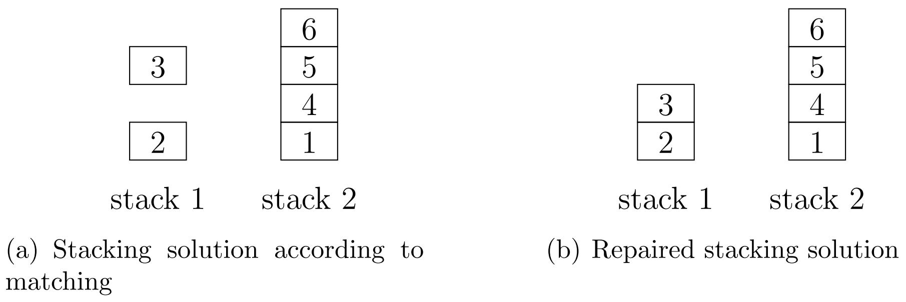
Now we consider the situation of special stacking constraints s_{ij} which define a total order on all load units (i.e. s_{ij} is transitive and for all $i \neq j$ we have $s_{ij} = 1$ or $s_{ji} = 1$). For example, this condition is satisfied if the s_{ij} are based on weight or length restrictions of the load units.

Theorem 3.7. *Problem fixed-grid(ℓ) $\mid \ell_i = \ell, s_{ij}^{to} \mid \#SI^{>1}$, where the stacking constraints s_{ij} define a total order, can be solved in $\mathcal{O}(n \log n + n^{fix} \log n^{fix})$ time.*

Proof. Since the s_{ij} define a total order, all load units can be compared. We define a comparator $\mathrel{\hat{=}}$ for load units i and j based on the stacking restrictions s_{ij} as follows

$$i \mathrel{\hat{=}} j := \begin{cases} i \approx j, & \text{if } s_{ij} = 1 \text{ and } s_{ji} = 1 \\ i \prec j, & \text{if } s_{ij} = 1 \text{ and } s_{ji} = 0 \\ i \succ j, & \text{if } s_{ij} = 0 \text{ and } s_{ji} = 1. \end{cases}$$

At first we consider the situation that the storage area is empty (i.e. $\mathcal{C}^{fix} = \emptyset$). By sorting the load units non-increasingly according to the defined comparator $\mathrel{\hat{=}}$, the list



(a) Stacking solution according to matching

(b) Repaired stacking solution

Figure 3.12: Original and repaired stacking solution for Example 3.3

of load units can always be transformed into a feasible stacking solution by filling up one stack after another according to the sorted list. This means that the first (the "largest") load unit is placed at the ground level in the first stack and all load units up to the b-th load unit are stacked on top. The $(b+1)$-st load unit is at the ground level of the second stack and so on. In the following, we show that the stacking constraints are respected by this approach. Whenever a load unit i is stacked on another load unit j, we have $i \preceq j$ which implies that $s_{ij} = 1$ (for $i \preceq j$ we have to distinguish the situations $i \prec j$ and $i \approx j$, but according to the definition of the comparator for both cases $s_{ij} = 1$ holds). Since $s_{ij} = 1$ for all load units i that are stacked on another load unit j, the stacking constraints are not violated by processing the ordered list.

If the storage area is not empty (i.e. $\mathcal{C}^{fix} \neq \emptyset$), we first fill up all partially filled stacks starting with the stack that has the smallest load unit on top (according to the comparator $\stackrel{\circ}{=}$). We fill up this stack up to the limit b with the smallest feasible load units of the set $\mathcal{C}$ that have to be stored. We then proceed with the stack with the second smallest load unit on top and so on. This strategy guarantees that incomplete stacks are filled as much as possible, up to the maximum level b. However, in the resulting solution still partly filled stacks may exist if there are not enough load units smaller than the topmost load unit in a partial stack. Having processed all stacks containing load units of $\mathcal{C}^{fix}$, the problem reduces to the one with an empty storage area discussed in the beginning of the proof. As result, we either get a feasible solution or demonstrate that none exists.

If a feasible solution is found, in order to minimize the number $\#SI^{>1}$ of load units stacked above the ground level, we fill all empty stacks. This can again be done by moving as many load units of the set $\mathcal{C}$ as possible from levels above the ground level to the ground.

Sorting the load units of $\mathcal{C}$ and sorting the topmost load units of $\mathcal{C}^{fix}$ (in the case that this set is not empty) requires $\mathcal{O}(n \log n + n^{fix} \log n^{fix})$ time, while filling up the stacks can be implemented in $\mathcal{O}(n)$ time. Thus, the problem can be solved in $\mathcal{O}(n \log n + n^{fix} \log n^{fix})$. □

Note that the ordered list in the proof defines a hamiltonian path through the directed graph of the load units with arcs (i, j) for $s_{ij} = 1$. If the s_{ij}-values do not define a total order, but the graph induced by s_{ij} contains a hamiltonian path, this path can be transformed into a feasible stacking solution, adopting the same strategy as in the proof of Theorem 3.7. However, the existence of a total order or a hamiltonian path is only a sufficient but not a necessary condition for the existence of a feasible stacking solution. For a feasible solution the graph has to be partitioned into at most S chains of length at most b, where each load unit is contained in exactly one chain.

Example 3.4. As a small real-world example we consider three types of load units: 40, 42 and 45-ft containers, where the number denotes the length of the container in feet. For this example, 40-ft containers can be stacked on top of all container types, 42-ft container can be stacked on top of 42 and 45-ft containers, but 45-ft containers can only be stacked on 45-ft containers. By sorting the containers according to non-increasing lengths, we get a sequence of containers starting with 45-ft containers, continuing with 42-ft ones and ending with 40-ft containers. The order of containers of the same type can be arbitrary. By splitting this sequence into stacks of height b, the stacking constraints are respected for all stacks as load units are stacked within their groups and potentially a 42-ft container is stacked on top of a 45-ft container and a 40-ft container is stacked on a 42-ft container, which is feasible. □

Theorem 3.7 can be generalized to handle the situation where two incoming sets $\mathcal{C}_1$ and $\mathcal{C}_2$ are given. Here, we assume that $\mathcal{C}_1$ arrives first and that the temporal intervals within the sets of incoming load units have to be stored are non-overlapping ($d_i < a_j$ for all $i \in \mathcal{C}_1$ and $j \in \mathcal{C}_2$). So, no load unit $i \in \mathcal{C}_1$ can be stacked on top of a load unit $j \in \mathcal{C}_2$, even if $s_{ij} = 1$.

Theorem 3.8. *Problem $\mathcal{C}^{fix} = \emptyset$, fixed-grid($\ell$) $\mid \ell_i = \ell, s_{ij}^{to} \mid \#SI^{>1}$, where the stacking constraints s_{ij} define a total order, the stacks initially are empty and the arrival and departure times lead to two sets with non-overlapping time windows ($\mathcal{C}_1$ and $\mathcal{C}_2$), can be solved in $\mathcal{O}(n \log n)$ time.*

Proof. We use again the comparator $\overset{\circ}{=}$ that has been defined in the proof of Theorem 3.7. Consider first the load units of the set $\mathcal{C}_1$. Using the approach described in the proof of Theorem 3.7, create $\lfloor \frac{|\mathcal{C}_1|}{b} \rfloor$ stacks filled in full by the smallest load units and create one partly filled stack (if any) with $|\mathcal{C}_1| \bmod b$ largest load units. The problem with the remaining load units of the set $\mathcal{C}_2$ reduces then to the one considered in Theorem 3.7 with $\mathcal{C}^{fix} = \mathcal{C}_1$.

For the minimization of $\#SI^{>1}$, load units of both sets $\mathcal{C}_1$ and $\mathcal{C}_2$ can be moved to the ground level. The complexity of the algorithm is dominated by the sorting step, which is again $\mathcal{O}(n \log n)$. □

The fixed grid policy can be skipped without a change in the complexity as $\mathcal{C}^{fix} = \emptyset$ is assumed.

3.5 Problems with arbitrary load unit lengths

In this section we provide some complexity results for the storage planning problem with arbitrary load unit lengths. The section is divided into Section 3.5.1 considering easy and hard to solve problems without stacking and Section 3.5.2 where two easy problems with stacking are considered.

3.5.1 Problems without stacking ($b = 1$)

In this section a storage planning problem with a single train is shown to be polynomial and for the case of multiple trains NP-completeness is proven. Furthermore it is shown that fixed grids of a common length (which is part of the input) make the problem NP-complete.

We start with the situation of a single train.

Theorem 3.9. *The problem $m = 1, b = 1, \mathcal{C}^{fix} = \emptyset \mid q = 1 \mid TC(x, y)$ where a single train has to be unloaded in a storage area with one empty storage lane and without stacking, can be solved in $\mathcal{O}(n)$ time.*

Proof. As the load units delivered by a single train do not overlap on the train, do not exceed the storage borders and in the storage lane no restrictions beside the non-overlapping restriction for the load units of the set $\mathcal{C}$ have to be respected, the load units can be unloaded in parallel moves. For each load unit of the set $\mathcal{C}$ only the transport orthogonal to the tracks (which is in y-direction) accounts but no transport alongside the tracks, as the load units are stored at a location with x-position O_i. So there exists no other solution with less or even equal transportation costs.

As we just have to copy the origin load unit x-positions for all load units, the complexity of the parallel moves rule is $\mathcal{O}(n)$. □

We cannot generalize this problem to the case where q' trains have to be unloaded to q' storage lanes ($m = q', b = 1 \mid q = q' \mid TC(x, y)$), where the transportation costs to the q' storage lanes are different. This is the case, as the different transport costs orthogonal to the tracks may cause that it is cheaper to move some load units alongside the tracks to better utilize a lane that is close to the tracks. If we only consider transportation costs in x-direction $m = q', b = 1 \mid q = q' \mid TC(x)$, we can again apply parallel moves by assigning each of the q' trains to one of the q' storage lanes. The transportation cost for such a solution is zero which is obviously optimal.

Unfortunately, the problem becomes NP-complete if several trains have to be unloaded into a single storage lane while the transportation costs in x-direction have to be minimized.

Theorem 3.10. *Problem $m = 1, b = 1|q \geq 2 \mid TC(x)$ is NP-complete in the ordinary sense.*

Proof. We prove NP-completeness by a reduction from the NP-complete scheduling problem $1|d_i = d| \sum |L_i|$, see [54]. The scheduling problem $1|d_i = d| \sum |L_i|$ is a single machine scheduling problem where all jobs have a common due date d and the aim is to minimize the total absolute deviation of the completion times from the common due date.

For an instance of $1|d_i = d| \sum |L_i|$ we construct an instance of the storage planning problem where the number of load units is equal to the the number of jobs in the machine scheduling problem. For each load unit $i \in \mathcal{C}$ we set $\ell_i = p_i$, i.e. the length of the load unit is the corresponding job processing time and we set the origin x-position of the load unit to the common due date minus the processing time of the job $O_i = d - p_i$. Now we show that $1|d_i = d| \sum |L_i|$ has a solution with $\sum |L_i| \leq G$ iff $m = 1, b = 1|q \geq 2 \mid TC(x)$ has a solution with $TC(x) \leq G$

"$\Rightarrow$": Assume that $1|d_i = d| \sum |L_i|$ has a solution with $\sum |L_i| \leq G$. Then the start times of the jobs can be interpreted as x-positions of the storage problem. The solution is feasible since the single machine can only process one job per time and the load unit length and processing times of the corresponding jobs are the same. Furthermore, this solution has $TC(x) \leq G$ since the absolute lateness directly corresponds to $TC(x)$ due to the definition of the parameter O_i. So, the transport costs of the storage problem are $TC(x) = \sum |L_i| \leq G$.

"$\Leftarrow$": Assume that conversely a feasible solution to the stacking problem with transport costs $TC(x) \leq G$ exists. The solution describes a feasible solution for the machine scheduling problem by interpreting the assigned x-positions as start times of the corresponding jobs. Again due to the definition of the transport costs the objective of the scheduling problem is also the same $\sum |L_i| = TC(x) \leq G$. $\square$

In Section 3.2 we mentioned that Jaehn proved in [57] NP-completenes for two storage planning problems with arbitrary load unit length. Expressed in our notation Jaehn showed that $m = 1, b = 1, \mathcal{C}^{fix} \neq \emptyset \mid\mid -$ as well as $m = 1, \mathcal{C}^{fix} \neq \emptyset \mid\mid \#SI^{>1}$ are strongly NP-hard by a reduction from 3PART and that $m = 2, b = 1, \mathcal{C}^{fix} = \emptyset \mid\mid -$ as well as $m = 2, \mathcal{C}^{fix} = \emptyset \mid\mid \#SI^{>1}$ are NP-hard by a reduction from PART. In the following theorem we will show that for the first problem and fixed grids the assumption $\mathcal{C}^{fix} \neq \emptyset$ can be skipped.

Theorem 3.11. *The feasibility problem $m = 1, b = 1, fixed\text{-}grid(g) \mid\mid -$ with fixed grids is strongly NP-complete.*

Proof. We prove NP-completeness by a reduction from the strongly NP-complete problem 3-PARTITION (3-PART), see [47]. Recall from Theorem 3.4 that an instance of 3-PART is defined by a set $A = \{1, \ldots, 3k\}$ of $3k$ elements and a bound $B \in \mathbb{N}$. Associated with the elements $i \in A$ are numbers a_i with $\sum_{i=1}^{3k} a_i = kB$ and $B/4 < a_i < B/2$. The decision problem asks whether A contains a partition $S_1, \ldots, S_k$ such that $\sum_{i \in S_\lambda} a_i = B$ for $\lambda = 1, \ldots, k$. Note, that by the definition of a_i each set S_λ must contain exactly three elements.

For an instance of 3-PART we construct an instance of the storage planning problem with $|A| = 3k$ load units and length $\ell_i = a_i$ for $i = 1, \ldots, 3k$. We introduce a single storage lane with a length of $L = k \cdot B$ and a fixed with unit subarea length $g = B$. We show that 3-PART has a feasible solution iff a feasible solution for the stacking problem exists.

"$\Rightarrow$": Assume that 3-PART has a feasible solution. Then the $k = \frac{L}{B}$ subsets can be interpreted as load unit triples that are stored in the grid subareas. These triples have a total length B, so they fit in single subareas.

"$\Leftarrow$": Assume that conversely a feasible solution to the storage planning problem exists. The $\frac{L}{B}$ subareas have to be filled by three load units because the number of load units is $3k$. Since the total length of the load units per subarea is limited to B and the sum of the load unit lengths is $B \cdot k$, each subarea has to be filled with load units whose lengths sum up to B. Thus, the load units of the subareas can be interpreted as subsets with cardinality three for the partition problem, where for each subset the sum of the a_i-values is B. $\square$

Note, that the optimization problem $m = 1, \textit{fixed-grid}(g) \mid\mid \#SI^{>1}$ is also NP-complete. As a feasible solution for the problem $m = 1, b = 1, \textit{fixed-grid}(g) \mid\mid -$ exists, iff a solution with objective zero (i.e. without stacking) exists for the optimization problem $m = 1, \textit{fixed-grid}(g) \mid\mid \#SI^{>1}$.

3.5.2 Problems with stacking ($b \geq 2$)

In this section we show two polynomial cases which are transferred from Theorems 3.2 and 3.7 for the case of a fixed grid setting with uniform load unit lengths to cases with arbitrary load unit lengths.

First the idea for minimizing the number of unordered stackings of Theorem 3.2 can be used to minimize the space requirement in a setting with arbitrary load unit lengths.

Theorem 3.12. *Problem $m = 1, b = 2, \mathcal{C}^{fix} = \emptyset \mid s_{ij} \mid -$ with arbitrary stacking constraints s_{ij} can be solved as a minimum-weight perfect matching problem in $\mathcal{O}(n^3)$ time.*

Proof. At first we have to sort the load units by non-decreasing lengths. Now, we check how many of the smallest load units maximal fit to the ground level of the storage lane. This can be done by simply summing up the lengths of the sorted load units while the sum is not larger than L. We define this maximal number of ground level load units as a. Note that if all load units are stacked in the storage lane, it may not be possible to build a stacks due to the limited length (L) of the storage lane. But a is an upper bound for the number of stacks that can be built. If $a \geq n$, the instance has a trivial feasible solution with all load units stored at the ground level.

If $a < n$, we introduce $2a - n$ dummy load units in order to get $2 \cdot a$ load units in total. These dummy load units are used to open the possibility of storing load units at the ground level without another load unit stacked on top and to build dummy stacks if a lot of load units can be stacked. For all dummy load units $i = n + 1, \ldots, 2a$ we set $\ell_i = 0$.

Similar to the proof of Theorem 3.2 we introduce the undirected graph $G = (V_1 \cup V_2, E_1 \cup E_2)$ where V_1 and V_2 represent real ($i \in \mathcal{C}$) and dummy load units, respectively, $|V_1| = n$, $|V_2| = 2a - n$. Edges E_1 connect pairs of nodes corresponding to stackable real load units, edges $E_2 = \{\{i,j\} : i \in V_1 \cup V_2, j \in V_2\}$ contain all pairs involving at least one dummy load unit. For two incoming load units $i, j \in \mathcal{C}$ an edge $\{i,j\} \in E_1$ exists if i can be stacked on j or vice versa (i.e. if $s_{ij} + s_{ji} \geq 1$).

The costs c_{ij} for edges $\{i,j\} \in E_1 \cup E_2$ with $i, j \in \{1, \dots, 2a\}$ are set to the difference of the load unit lengths $c_{ij} := |\ell_i - \ell_j|$.

Consider the minimum-weight perfect matching problem defined for the graph G. Clearly, if a perfect matching does not exist, then also no feasible stacking solution exists since the number of stacks necessary is larger than the upper bound of the maximal possible stacks. Otherwise, a minimum-weight perfect matching will match all load units with the minimum possible total deviation of the lengths in stacks. To check whether the matching defines a feasible solution, we have to check whether the storage lane is long enough to hold the a stacks. For the a matched edges $\{i,j\}$ we sum up the maxima of the load unit lengths of the matched load units i and j. If this value is not greater than L, a feasible solution is found. In this case we define a solution to the storage loading problem based on the solution to the minimum-weight perfect matching problem. For edges $\{i,j\} \in E_1$ in the matching we define the order of i, j in the stack as follows. If $i, j \in \mathcal{C}$ are only stackable in one direction (i.e. $s_{ij} + s_{ji} = 1$), then they are stored in the unique possible way respecting the stacking constraints (i.e. if $s_{ij} = 1$, load unit i is stacked on load unit j). On the other hand, if load units $i, j \in \mathcal{C}$ are stackable in both directions (i.e. $s_{ij} = s_{ji} = 1$), we can select one direction. Edges $\{i,j\} \in E_2$ in the matching involving two dummy load units are ignored. For edges $\{i,j\} \in E_2$ with one dummy and one real load unit, we place the real load unit on the ground level (without any load unit on top).

To show that the proposed algorithm solves the storage loading problem, we have to show that there does not exist a solution of the storage problem if the sum of the maxima of the load unit lengths of matched load units (of the minimum-weight perfect matching problem) is greater than L. If for this case a feasible solution would exist, according to stacking limit $b = 2$ there would have to be at least one pair of load units that can be feasibly exchanged in the matching to reduce the space demand of the solution. But due to the definition of the costs c_{ij} also the costs for the minimum-weight perfect matching problem decrease, which is a contradiction since we started with an optimal solution.
Let us consider this proposition in more detail. Assume we have load units i_1 matched with i_2 in the matching as well as i_3 matched with i_4. Without loss of generality we can assume that $\ell_{i_1} \geq \ell_{i_2}$ and $\ell_{i_3} \geq \ell_{i_4}$ and that $\max(\ell_{i_1}, \ell_{i_2}) \geq \max(\ell_{i_3}, \ell_{i_4})$. Let us furthermore asume that $s_{i_1 i_3} + s_{i_3 i_1} \geq 1$ and $s_{i_2 i_4} + s_{i_4 i_2} \geq 1$ which means that the load units can also be matched in pairs i_1, i_3 and i_2, i_4. Let us assume that this load unit exchange causes a reduction in ground level usage. This could be the case if $\max(\ell_{i_1}, \ell_{i_2}) + \max(\ell_{i_3}, \ell_{i_4}) > \max(\ell_{i_1}, \ell_{i_3}) + \max(\ell_{i_2}, \ell_{i_4})$ which implies that $\ell_{i_3} > \ell_{i_2}$. But this contradicts $c_{i_1 i_2} + c_{i_3 i_4} \leq c_{i_1 i_3} + c_{i_2 i_4}$ which holds since we started with a minimum-weight perfect matching.

So, we showed that the storage planning problem has a feasible solution, iff the minimum-weight perfect matching has a feasible solution that corresponds to a storage solution with space demand less than L.

Since the number of nodes is $2 \cdot a$ for the case $a < n$, a minimum-weight perfect matching can be computed in $\mathcal{O}(n^3)$ time (cf. Gabow [45] and Lawler [67]). $\square$

For the problem where the stacking restrictions define a total order the idea of Theorem 3.7 can be transferred to the case without fixed grids and arbitrary load unit lengths.

Theorem 3.13. *The problem $\mathcal{C}^{fix} = \emptyset \mid s_{ij}^{to} \mid \#SI^{>1}$, where the stacking constraints s_{ij} define a total order, can be solved in $\mathcal{O}(n \log n)$ time.*

Proof. To solve the problem we use the algorithm proposed in the proof of Theorem 3.7 to solve the feasibility problem $\textit{fixed-grid}(\ell) \mid \ell_i = \ell, s_{ij} \mid -$. By sorting the load units non-increasingly by the comparator $\stackrel{\circ}{=}$ the load units are also sorted by non-increasing length. This is the case, since one of our basic (and practical) assumptions is that if two load units are stacked the length of the upper load unit is not allowed to exceed the length of the lower load unit. Let us define π_i as the position within the sorted sequence of load units, where the sequence is sorted non-increasingly according to the comparator $\stackrel{\circ}{=}$. Then $\ell_{\pi_1} \geq \ell_{\pi_2} \geq \ell_{\pi_3} \ldots \geq \ell_{\pi_n}$ holds and a solution of the algorithm of Theorem 3.7 has $\lceil \frac{n}{b} \rceil$ load unit stacks and the structure shown in Figure 3.13.

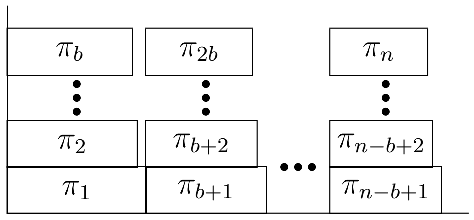


Figure 3.13: Structure of a solution for n mod $b = 0$

If n mod $b \neq 0$ the last stack does not contain b but n mod b load units. To validate if a given storage problem has a feasible solution we have to check whether the following constraint is satisfied:

$$L \geq \sum_{i \in \mathcal{C};\ \pi_i \bmod b = 1} \ell_i$$

So we have to check whether the length of all load unit stacks is smaller or equal to the length of the terminal. In the sum the lengths of load units π_1, π_{b+1}, π_{2b+1} and so on are added $(\ell_{\pi_1} + \ell_{\pi_{b+1}} + \ell_{\pi_{2b+1}} \ldots)$.

If a feasible solution is found, in order to minimize the number $\#SI^{>1}$ of load units stacked above the ground level, we position as many load units to the ground level as possible. This can be done by moving as many load units of the set $\mathcal{C}$ as possible from levels above the ground level to the ground proceeding the load units $i \in \mathcal{C}$ in non-decreasing order by the lengths ℓ_i.

Sorting the load units of $\mathcal{C}$ can be done in $\mathcal{O}(n \log n)$ time, while positioning load units to the ground level can be done in $\mathcal{O}(n)$ time. Thus, the problem can be solved in $\mathcal{O}(n \log n)$. $\square$

Note, that we can solve some of the stacking problems for the case of practical stacking restrictions $m = 1, \mathcal{C}^{fix} = \emptyset \mid s_{ij}(\hat{\ell}) \mid -$ where the stacking restrictions come from real world UIC load units (see Table 2.1 for a list and Figure 3.10 for the induced stacking graph). Recall from Section 3.1 in this case load units may be stacked, if they have the same corner casting distances and the overhangs of the upper load unit are not larger than those of the lower load unit.
As load units with different corner casting distances cannot be stacked, the problem for practical stacking conditions can be divided into three independent subproblems for the corner casting distances 20, 30 and 40 feet. If we have optimal solutions (minimum amount of used space) for all subproblems, we just have to check whether the terminal lane offers enough space for all the storage plans for the corner casting types. Note, that our algorithms for feasibility have the objective to minimize the amount of used space. So, these algorithms may be used for groups of the same corner casting, if they can solve the feasibility problem.
With Theorem 3.13 the case of load units with 20-feet corner casting distance can be solved. Furthermore, the case of load units with 30-feet corner castings can be reduced to the same problem. We just have to build stacks with the 31-feet load units and fill the last potentially only partial filled stack with load units of the type 30-feet up to b, if there are enough 30-feet load units. Note, that the 31-feet load units stacks can not be filled up, if there are not enough 30-feet load units. The remaining graph does not contain any load units of the 31-feet group. So, the stacking restrictions define a total order and the problem can be solved by the algorithm proposed in the proof of Theorem 3.13. Note, that this approach can be extended to handle out-trees and by a inverted approach (start filling up the storage with small load units at the maximum level) in-trees can be solved. Unfortunately the complexity of the case with 40-feet corner casting distance is an open problem, where we cannot provide an polynomial algorithm nor could prove NP-completeness.

With the storage planning problem $m = 1, \mathcal{C}^{fix} = \emptyset \mid s_{ij}(\hat{\ell}) \mid -$ where stacking restrictions come from the two overhangs also special temporal conditions for stacking restrictions can be modeled. This is the case, if $s_{ij} = 1$ iff $a_i \geq a_j$ and $d_i \leq d_j$ where a time point t_0 exists such that $a_i \leq t_0$ and $d_i \geq t_0$ for all $i \in \mathcal{C}$ and no reshuffling is allowed. So, until t_0 a loading phase takes places and after t_0 unloading is done. If no such t_0 exists locations can be used by different load units over time which is not the case in any storage planning problems we consider. For the case with t_0 we can transfer the situation motivated by a_i and d_i to the practical stacking conditions $\left(s_{ij}(\hat{\ell})\right)$ by defining the left overhang $L3_i := t_0 - a_i$ and $L4_i := d_i - t_0$ for all load units $i \in \mathcal{C}$. The overhangs $L3_i$ ensures that the upper load unit does not arrive earlier than the lower and $L4_i$ ensure that the upper load unit does not leave later than the lower load unit.

3.6 Summary of complexity results

In this section we give a short summary of our complexity results and the results from literature. Furthermore, we explain why our results can be helpful.

Problem	Reference	Complexity
$b = 2, \mathit{fixed\text{-}grid}(\ell) \mid \ell_i = \ell, s_{ij} \mid \#SI^{>1}$	Theorem 3.1	$\mathcal{O}\left((n + n^{fix})^{2.5}\right)$
$b = 2 \mid \ell_i = \ell, s_{ij} \mid \#US$	Theorem 3.2	$\mathcal{O}((n + n^{fix})^3)$
$b = 3, \mathcal{C}^{fix} = \emptyset, \mathit{fixed\text{-}grid}(\ell) \mid \ell_i = \ell, s_{ij}^{trans} \mid -$	Theorem 3.3	str. NP-com.
$b = 3, \mathcal{C}^{fix} = \emptyset, \mathit{fixed\text{-}grid}(\ell) \mid \ell_i = \ell, \mathit{weight\text{-}limit} \mid -$	Theorem 3.4	str. NP-com.
$b = 3, \mathcal{C}^{fix} = \emptyset, \mathit{fixed\text{-}grid}(\ell) \mid \ell_i = \ell, \mathit{height\text{-}limit} \mid -$	Theorem 3.4	str. NP-com.
$m = 1, \mathcal{C}^{fix} = \emptyset,\ \mathit{fixed\text{-}grid}(\ell) \mid \ell_i = \ell, s_{ij}, q = 1 \mid TC(x, y)$	Theorem 3.5	$\mathcal{O}(n)$
$\mathit{fixed\text{-}grid}(\ell) \mid \ell_i = \ell \mid w_1 TC(x, y) + w_2 \#SI^{>1}$	Theorem 3.6	$\mathcal{O}\left((n + S \cdot b)^3\right)$
$\mathit{fixed\text{-}grid}(\ell) \mid \ell_i = \ell, s_{ij}^{to} \mid \#SI^{>1}$	Theorem 3.7	$\mathcal{O}(n \log n + n^{fix} \log n^{fix})$
$\mathcal{C}^{fix} = \emptyset, \mathit{fixed\text{-}grid}(\ell) \mid \ell_i = \ell, s_{ij}^{to}$, with two sets of arriving load units $\mathcal{C}_1, \mathcal{C}_2 \mid \#SI^{>1}$	Theorem 3.8	$\mathcal{O}(n \log n)$
$m = 1, b = 1, \mathcal{C}^{fix} = \emptyset \mid q = 1 \mid TC(x, y)$	Theorem 3.9	$\mathcal{O}(n)$
$m = 1, b = 1 \mid q \geq 2 \mid TC(x)$	Theorem 3.10	NP-com.
$m = 1, b = 1, \mathit{fixed\text{-}grid}(g) \mid\mid -$	Theorem 3.11	str. NP-com.
$m = 1, b = 1, \mathcal{C}^{fix} \neq \emptyset \mid\mid -$	Jaehn [57]	str. NP-com.
$m = 2, b = 1, \mathcal{C}^{fix} = \emptyset \mid\mid -$	Jaehn [57]	NP-com.
$m = 1, b = 2, \mathcal{C}^{fix} = \emptyset \mid s_{ij} \mid -$	Theorem 3.12	$\mathcal{O}(n^3)$
$\mathcal{C}^{fix} = \emptyset \mid s_{ij}^{to} \mid \#SI^{>1}$	Theorem 3.13	$\mathcal{O}(n \log n)$

Table 3.1: Summary of complexity results

The known complexity results for storage planning problems are summarized in Table 3.1. For some special cases polynomial algorithms could be provided. This is often the case for problems with stacking height $b = 2$ or situations where the stacking restrictions s_{ij} define a total order. On the other hand, we could show that problems with stacking height $b = 3$ with transitive stacking conditions or additional stack limitations (weight or height limit) and problems with arbitrary load unit lengths even without stacking are NP-complete.

Note, that our results can also help to use appropriate methods for storage planning in practice. For terminals where the stacking height is limited to $b = 2$ or the utilization does not necessitate to build stacks of more than two load units, the polynomial algorithms can be used in practice. Unfortunately, this is only the case in a fixed grid setting with uniform load unit lengths or if the storage area is initially empty. Note, that we were not able to provide a polynomial algorithm for transportation costs, if stacking restrictions have to be considered. So, this is another limitation of the polynomial algorithms provided. As we in addition could show that with transportation costs even without stacking the problem with arbitrary load unit lengths is NP-complete, we think that for practical storage planning problems IP- or MIP-models as well as heuristics are the best choice for solution methods. Such solution methods are discussed in the following sections for the practical settings in German rail-road terminals (Sections 3.7 to 3.10). On the other hand, for NP-hard problems the complexity results may be helpful to identify polynomial solvable subproblems, if a decomposition approach for the storage planning problem shall be invented and implemented.

For the stacking height $b = 2$ where we could show that several settings can be solved in polynomial time we think that the problem $b = 2, fixed\text{-}grid(\ell) \mid \ell_i = \ell, s_{ij} \mid TC(x, y)$ involving transportation costs is of interest for further research.

It would also be interesting to study special situations of the stacking restrictions s_{ij} based on a partial order (i.e. s_{ij} is transitive, but not all load units are comparable). An example for such a situation is the setting with two overhangs (stacking restrictions according to $s_{ij}(\hat{\ell})$). A special case of this situation are the practical stacking restrictions for 40-feet corner casting distance (see Figure 3.10 for a graph representation). Recall from the end of Section 3.5.2, that also special temporal restrictions can be expressed in this overhang setting of the practical stacking restrictions $s_{ij}(\hat{\ell})$. So, this special situation of stacking restrictions is motivated by the practical load unit stacking restrictions and also by uniform containers or items with temporal restrictions.

3.7 Setting in German rail-road terminals

In this section we introduce the storage problem that occurs in German rail-road terminals. Here we consider the case $b = 3, \mathcal{C}^{fix} \neq \emptyset | s_{ij}(\hat{l}) | w_1 \cdot TC(x, y) + w_2 \cdot \#SI^{>1}$ which means that we consider a terminal where up to three load units can be stacked and where already load units are stored. This problem is NP-hard since $m = 1, b = 2, \mathcal{C}^{fix} = \emptyset \mid\mid \#SI^{>1}$ is a special case of the practical problem. Since for the proof of the decision problem $m = 1, b = 2, \mathcal{C}^{fix} = \emptyset \mid\mid -$ the question is whether all load units fit on the ground level, it does not matter if $b = 2$ or $b = 3$. We have to consider arbitrary load unit lengths and the stacking conditions according to corner casting distance but we only consider symmetric load units which are mostly used in the rail-road terminals. So, for each corner casting distance ($20, 30$ and 40 feet) a total order is defined by the stacking restrictions. Finally, the objective is to minimize a weighted sum of the transport costs and the number of load units that are not stored at the ground level.

The idea of the storage planning problem is that a number of trains have to be unloaded and we have to find storage positions for all load units. There are two situations that motivate this case: i) trains have to be parked on sidings as new trains enter the terminal and ii) train loading starts soon and all load units that arrived by trains and are still on the trains have to be unloaded.

As we are not using any grid policy we define a location in the storage as a triple of values: the discrete rows or also called lanes (y-axis), the distance from the begin of the terminal along the tracks to the left corner of the load unit (x-axis) and the discrete levels (z-axis). So we introduce three sets that describe load unit locations. $\mathcal{X} = \{1, \ldots, L\}$ is the set of decimeter positions in the terminal (alongside the tracks), $\mathcal{Y} = \{1, \ldots, m\}$ is the set of storage lanes and $\mathcal{Z} = \{1, \ldots, b\}$ are the different stacking levels.
The dimensions of typical storage areas in German terminals are about two to five storage lanes and a length of between 500 and 750 meters. In most terminals up to three stackable load units can be stacked.

We consider the set $\mathcal{M}$ of load units. These are divided in the load units that have to be positioned in the terminal $\mathcal{C}$ and the set $\mathcal{C}^{fix}$ of load units that are already stored in one of the terminal storage lanes with $\mathcal{M} = \mathcal{C}^{fix} \cup \mathcal{C}$. The load units that are stackable are contained in the set $\overline{\mathcal{M}}$ and the load units which are not stackable are in the set $\widehat{\mathcal{M}}$. Furthermore, we define $\overline{\mathcal{C}}$ as $\overline{\mathcal{C}} = \mathcal{C} \cap \overline{\mathcal{M}}$. Note that the intersections of $\mathcal{C}$ and $\mathcal{C}^{fix}$ as well as $\overline{\mathcal{M}}$ and $\widehat{\mathcal{M}}$ are all empty.

For all load units $i \in \mathcal{M}$ $\ell_i \in \mathbb{N}$ is the length of the load unit. As additional length parameters we introduce the minimum $\ell^- = \min_{i \in \mathcal{C}} \ell_i$ and the maximum load unit length $\ell^+ = \max_{i \in \mathcal{C}} \ell_i$. Beside the length for stacking the load unit type $\hat{\ell}_i \in \{0, 1, 2, 3\}$ (for all load units $i \in \mathcal{M}$) is important. The set $\{0, 1, 2, 3\}$ represents load units which are trailers (0) and such load units with 20 (1), 30 (2) and 40 feet corner castings (3). To represent the stacking conditions in addition the parameters $s_i \in \{0, 1\}$ are one if the load unit i is stackable or zero if the load unit cannot be stacked.
We assume that between each pair of load units a distance of 1 dm has to stay empty. Otherwise, it would not be possible to lift load units by crane without the risk of damaging the lifted or adjacent load units.

To measure the length of the loaded crane moves we introduce two distance parameters. For load units that have to be stored $d_{ik} \in \mathbb{N}$ for $i \in \mathcal{C}; k \in \mathcal{X}$ is the length of the x-direction move from the original position of load unit i to a location with x-value k (here the transport alongside the tracks is considered). The length of the moves in y-direction across the different terminal areas are $\overline{d_{il}} \in \mathbb{N}$ for $i \in \mathcal{C}; l \in \mathcal{Y}$ denoting the length from the position of load unit i to a location with y-value l. The original y-positions of load units are either the tracks of the trains or the parking lane (for load units that are delivered by truck).

For load units $j \in \mathcal{C}^{fix}$ which are already in the storage area, λ_j denotes the storage lane, α_j is the x-position in the storage and β_j is the storage level. In addition the triples $\mathcal{P}_j \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ for all $j \in \mathcal{C}^{fix}$ are the locations.

For the practical storage planning the following hard constraints have to be considered:

(H1) Each load unit is assigned to exactly one location.

(H2) Load unit locations in the same lane and level are not allowed to overlap. Furthermore, between all pairs of load units a distance of one decimeter has to be kept.

(H3) Only stackable load units are allowed to be stacked. This means load units that are not stackable are not allowed to be stacked on top of other load units nor other load units are allowed to be stacked on top of these load units. The locations of all load units of a stack need to have the same x-positions.

(H4) Only load units with the same distance in between the corner castings can be stacked. Furthermore, the upper load unit length is not allowed to exceed the length of the load unit it is stacked on.

Furthermore there are the following soft constraints:

(S1) The total length of the loaded crane moves in x- and y- direction for storing load units should be minimized.

(S2) Load units should be stored at the ground level.

In discussions with practitioners and administratives from German rail-road terminals we decided to choose a weighted sum of the maximum and the minimum of the length of the loaded crane moves in x- and y- direction for the crane moves. The maximum is important for the time the transport will take as the moves in both directions can be performed simultaneously with more or less the same speed. But from an energetic point of view also the minimum has to be minimized as all moves consume energy and the lengths of the moves also influence the maintenance intervals.
So, we decided to minimize a weighted sum of the maximum as well as the minimum total length of the loaded crane moves in x- and y- direction and the number of load units which are not stored at ground level.

3.8 MIP-models

In this section we will introduce four different MIP-models. The first one is based on pairwise load unit distances. The second one is based on ground coverage constraints while the other constraints are similar to the first model. The last two formulations are based on integer x-position variables which reduces the number of variables needed.

3.8.1 Distance model IP

In the following we introduce a first MIP-model which is based on constraints that ensure pairwise distances between load units to model the non overlapping constraint (H2).

We can restrict the potential locations ($\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$) for load units to store ($i \in \mathcal{C}$) according to different ideas. First of all we do not have to consider locations for load units of set $\mathcal{C}$ with x-positions that are covered by a load unit of set $\mathcal{C}^{fix}$ apart from the locations with the x-positions of stored load units. These locations are potential stacking locations in upper levels. Furthermore, we do not have to consider locations which would lead to an overlap with a stored load unit of the set $\mathcal{C}^{fix}$ for every load unit of set $\mathcal{C}$. Recall that the x-position is the left corner of the load unit.
We can introduce the locations that cannot be the storage locations of any load unit because they would lead to an overlap with load unit $j \in \mathcal{C}^{fix}$ as

$$\mathcal{A}_j^* = \left\{(k,l,e) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \text{ with } \lambda_j = l; \alpha_j - \ell^- \leq k \leq \alpha_j + \ell_j; (k \neq \alpha_j \vee e > \beta_j)\right\}$$

In the set $\mathcal{A}_j^*$ we consider the locations that are in the same lane as j and start no more than ℓ^- before and no more than the length of the stored load unit (ℓ_j) after the storage x-position of load unit j (α_j). But we do not consider the locations on top of j as these locations may be possible locations for stacked load units. We consider the locations with x-positions $\alpha_j - \ell^-$ and $\alpha_j + \ell_j$, as the load units have to be stored with a minimum distance of 1 dm in between. So, these locations are also blocked by load unit j.
By reducing the set of all locations we get

$$\mathcal{A} = (\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}) \setminus \bigcup_{j \in \mathcal{C}^{fix}} \mathcal{A}_j^*$$

which is the set of all locations that have to be considered for the storage planning. These are potential locations for load units of the set $\mathcal{C}$ (locations that are not blocked by any load unit of the set $\mathcal{C}^{fix}$) as well as the locations on top of stored load units of the set $\mathcal{C}^{fix}$.

In the calculations for the set $\mathcal{A}$ we only use the minimum length of load units ℓ^- to determine the storage locations that have to be considered for the storage planning. For a load unit i where $\ell_i > \ell^-$ additionally some of the ground level locations of $\mathcal{A}$ are not feasible as they would cause an overlap with an already stored load unit. So we introduce for each pair $(i,j) \in \mathcal{C} \times \mathcal{C}^{fix}$ the set $\mathcal{B}_{ij}^*$ as:

$$\mathcal{B}_{ij}^* = \left\{k \in \mathcal{X} \text{ with } \alpha_j - \ell_i \leq k \leq \alpha_j + \ell_j\right\}$$

which are infeasible x-positions for load unit i in lane λ_j at the ground level due to blocking by stored load unit j. Furthermore,

$$\mathcal{B}_{il} = \mathcal{X} \setminus \bigcup_{j \in \mathcal{C}^{fix} \text{ with } \lambda_j = l} \mathcal{B}_{ij}^*$$

are the sets of all feasible ground level x-positions in lane l for load unit i (with respect to restrictions concerning stored load units of set $\mathcal{C}^{fix}$). In contrast to set $\mathcal{A}$ we can consider

the actual load unit length (of load unit i).
For other constraints we need the feasible x-y-positions of load units. So we first define analogously to $\mathcal{B}^*_{ij}$ the set

$$\mathcal{D}^*_{ij} = \left\{(k,l) \in \mathcal{X} \times \mathcal{Y} \text{ with } l = \lambda_j; \alpha_j - \ell_i \leq k \leq \alpha_j + \ell_j\right\}$$

for $i \in \mathcal{C}$ and $j \in \mathcal{C}^{fix}$ which are infeasible x-y-positions for load unit i blocked by stored load unit j. Then all feasible positions can be obtained as

$$\mathcal{D}_i = (\mathcal{X} \times \mathcal{Y}) \setminus \bigcup_{j \in \mathcal{C}^{fix}} \mathcal{D}^*_{ij}$$

which are the sets of all feasible x-y-positions for load unit i (with respect to restrictions concerning stored load units).

Note, that for the construction of the sets $\mathcal{A}$, $\mathcal{B}_{ij}$ and $\mathcal{D}_i$ it is sufficient to consider only load units that are stored at the ground level instead of all stored load units of the set $\mathcal{C}^{fix}$. This is the case as load units that are not stored at ground level can never exceed the load units they are stacked onto. So, in the construction of the sets $\mathcal{A}$, $\mathcal{B}_{ij}$ and $\mathcal{D}_i$ we can replace $j \in \mathcal{C}^{fix}$ by $j \in \mathcal{C}^{fix}; \beta_j = 1$.

To formulate an IP we introduce the following binary decision variables.

- $x_{ikle} \in \{0,1\}$ for $i \in \mathcal{M}; (k,l,e) \in \mathcal{A}$ with

  $$x_{ikle} = \begin{cases} 1, & \text{iff load unit } i \text{ is stored at decimeter position } k \text{ in lane } l \text{ and level } e \\ 0, & \text{otherwise.} \end{cases}$$

- $v_{ij} \in \{0,1\}$ for $i,j \in \mathcal{C}; i < j$ with

  $$v_{ij} = \begin{cases} 1, & \text{if } i \text{ is stored at a smaller decimeter position than } j \text{ at the ground level,} \\ 0, & \text{otherwise.} \end{cases}$$

  If the two load units i and j are not stored in the same lane or at least one of them is stacked on top of any other load unit, the assignment of v_{ij} is not well defined and unimportant for the model.

The IP with pairwise load unit distance can then be stated as:

$$\min \sum_{i \in \mathcal{C}} \sum_{(k,l,e) \in \mathcal{A}} \left(w_1 \cdot \max(d_{ik}, \overline{d_{il}}) \cdot x_{ikle} + w_2 \cdot \min(d_{ik}, \overline{d_{il}}) \cdot x_{ikle}\right) \tag{3.2}$$

$$-w_3 \cdot \sum_{i \in \mathcal{C}} \sum_{(k,l,1) \in \mathcal{A}} x_{ikl1} \tag{3.3}$$

s.t.

$$\sum_{(k,l,e) \in \mathcal{A}} x_{ikle} = 1$$

$$(i \in \mathcal{C}) \tag{3.4}$$

$$\sum_{k \in \mathcal{B}_{jl}} k \cdot x_{jkl1} - \sum_{k' \in \mathcal{B}_{il}} (k' + \ell_i + 1) \cdot x_{ikl1} \geq -M_1(2 - v_{ij} - \sum_{k \in \mathcal{B}_{jl}} x_{jkl1})$$
$$(i, j \in \mathcal{C}; i < j; l \in \mathcal{Y}) \tag{3.5}$$

$$\sum_{k \in \mathcal{B}_{il}} k \cdot x_{ikl1} - \sum_{k' \in \mathcal{B}_{jl}} (k' + \ell_j + 1) \cdot x_{jkl1} \geq -M_1(v_{ij} + 1 - \sum_{k \in \mathcal{B}_{jl}} x_{jkl1})$$
$$(i, j \in \mathcal{C}; i < j; l \in \mathcal{Y}) \tag{3.6}$$

$$\sum_{i \in \mathcal{M}} s_i \cdot \ell_i \cdot x_{ikle} - \sum_{i \in \mathcal{M}} \ell_i \cdot x_{ikl,e+1} \geq 0$$
$$((k, l, e) \in \mathcal{A}; e \neq b) \tag{3.7}$$

$$\sum_{i \in \mathcal{M}} \hat{\ell}_i \cdot x_{ikle} - \sum_{i \in \mathcal{M}} \hat{\ell}_i \cdot x_{ikl,e+1} \leq M_2(1 - \sum_{i \in \mathcal{M}} x_{ikl,e+1})$$
$$((k, l, e) \in \mathcal{A}; e \neq b) \tag{3.8}$$

$$\sum_{i \in \mathcal{M}} \hat{\ell}_i \cdot x_{ikle} - \sum_{i \in \mathcal{M}} \hat{\ell}_i \cdot x_{ikl,e+1} \geq 0$$
$$((k, l, e) \in \mathcal{A}; e \neq b) \tag{3.9}$$

$$\sum_{(k,l,e) \in \mathcal{A}; k + \ell_i > L} x_{ikle} = 0$$
$$(i \in \mathcal{C}) \tag{3.10}$$

$$x_{ikle} = 1$$
$$(i \in \mathcal{C}^{fix}; (k, l, e) \in \mathcal{P}_i) \tag{3.11}$$

$$x_{ikle} = 0$$
$$(i \in \mathcal{C}^{fix}; (k, l, e) \in \mathcal{A} \setminus \mathcal{P}_i) \tag{3.12}$$

$$x_{ikle} = 0$$
$$(i \in \mathcal{C}, s_i = 0; (k, l, e) \in \mathcal{A}, e \neq 1) \tag{3.13}$$

$$x_{ikle} = 0$$
$$(i \in \mathcal{C}; (k, l, e) \in \mathcal{A} \setminus (\mathcal{D}_i \times \mathcal{Z})) \tag{3.14}$$

$$x_{ikle} \in \{0, 1\}$$
$$(i \in \mathcal{M}; (k, l, e) \in \mathcal{A}) \tag{3.15}$$

$$v_{ij} \in \{0, 1\}$$
$$(i, j \in \mathcal{C}; i < j) \tag{3.16}$$

The first part of the objective (3.2) aims at minimizing the total length of loaded crane moves (S1). The second part (3.3) aims at maximizing the number of load units that are stored at the ground level (S2) .
The constraints (3.4) ensure that each load unit from the set $\mathcal{C}$ is assigned to exactly one location in the storage (H1). Constraints (3.5) and (3.6) avoid overlapping of load units at the ground level for pairs (i, j) (H2). If load unit i is stored at a smaller x-position than j in the same lane l at the ground level, then v_{ij} as well as $\sum_{k \in \mathcal{X}} x_{jkl1}$ are one. So the right hand side of (3.5) is zero which activates this constraint for the according

parameters i, j. The x-position of i plus the length of i and the 1 dm distance between load units i and j must not be larger than the x-position of load unit j. If the load units i and j are stored in lane l at the ground level but load unit i is stored at a larger x-position than j, v_{ij} is zero and constraints (3.6) avoid overlapping. We do not have to introduce further constraints that restrict the values of v_{ij}. For two load units i and j stored in lane l at the ground level v_{ij} has to be chosen correctly because otherwise either constraints (3.5) or (3.6) would be violated. Assume that the x-position of i is smaller than the one of j and v_{ij} is zero. This is not part of any feasible solution because the left hand side of (3.6) is negative while the right hand side is zero, which is infeasible. If on the other hand, the position of i is larger than the one of j and v_{ij} would be one, the right hand side of (3.5) would be zero and the left hand side negative, which would also be infeasible. For the non-overlapping constraints we do not have to consider pairs of load units were $i, j \in \mathcal{C}^{fix}$ because we assume that all load units are initially stored in a feasible way. Constraints (3.7) ensure that load units can only be stacked onto stackable load units and furthermore that stacked load units do not exceed the length of the load unit they are stacked on (H3,H4). Constraints (3.8) and (3.9) omit that load units with different corner casting distances are stacked (H4). While due to (3.8) for each pair of locations with the same x-position, the same lane and adjacent levels the corner casting distance of the upper load unit (if there is an upper load unit) is not allowed to be smaller than the corner casting of the lower load unit, in (3.9) the corner casting of the lower load unit is not allowed to be smaller than the one of the upper load unit. So, together (3.8) and (3.9) force the corner casting distances to be equal in all load unit stacks. In (3.9) it is not necessary to deactivate the constraints, if the upper location is not occupied.

No load unit is allowed to exceed the end of the terminal in x-direction, this is modeled by constraints (3.10). For all load units $j \in \mathcal{C}^{fix}$ one can fix all x_{jkle} variables. This is done for the predefined locations in constraints (3.11). Furthermore, all other x_{ikle} are set to zero by constraints (3.12). Load units that are not stackable can only be stored at ground level. This is stated in (3.13) (H3). Some locations contained in $\mathcal{A}$ are not feasible for some load units i, because for the computation of $\mathcal{A}$ we considered the minimum load unit length ℓ^- instead of the actual load unit length. These load unit x-y-z-position combinations are fixed to zero by constraints (3.14) (H1). Finally, constraints (3.15) and (3.16) define the domains of the binary variables.

This formulation is based on two big-M parameters. M_1 is used for avoiding load unit overlapping and has to be at least larger than the maximum x-position:

$$M_1 > \max_{k \in \mathcal{X}} k$$

The second big-M parameter M_2 has to be larger than the greatest distance in between the corner castings respective the associated greatest parameter, so:

$$M_2 > \max_{i \in \mathcal{M}} \hat{\ell}_i$$

In the following we will refer to the IP-model introduced in this section as distance-IP.

3.8.2 Ground coverage model

In this section we introduce an IP-model that is based on the previous IP-model (see Section 3.8.1), but replace the constraints that ensure pairwise non-overlapping of load units by ground coverage constraints. These ground coverage constraints ensure for each dm x-position that it is covered by at most one load unit at the ground level. This ensures that no pair of load units exists that are stored at the ground level and overlap, as this overlapping would also cause that at least one dm x-position is covered twice. Again, the stacking constraints ensure non-overlapping for load units that are not stored at the ground level but stacked onto other load units.
We introduce a set of x-y-positions that have to be considered in the ground coverage constraints. In addition, we define a set for each position considered in the coverage constraint and each load unit $i \in \mathcal{C}$ that contains all storage positions that lead to a coverage of the coverage constraint position by load unit i. This enables us to check for all considered coverage positions if they are covered by more than one load unit.

We adapt the MIP-formulation of Section 3.8.1 by replacing constraints (3.5) and (3.6) by constraints that restrict the ground coverage. For this formulation we can skip the decision variables v_{ij}. To ensure, that each x-y-position at the ground level is not covered by more than one load unit, we do not have to consider all x-y-positions of the set $\mathcal{X} \times \mathcal{Y}$. In addition to the reduction of the location space for set $\mathcal{A}$ where locations to the left of stored load units within a distance not greater than ℓ^- can be ignored, we can leave out locations to the right of stored load units in most cases. So we introduce for stored load units $j \in \mathcal{C}^{fix}$:

$$\mathcal{A}_j^\circ = \{(k,l) \in \mathcal{X} \times \mathcal{Y} \text{ with } \lambda_j = l; \alpha_j - \ell^- \leq k \leq \alpha_j + \ell_j + \ell^-\}$$

which are the ground level locations that do not have to be considered for the coverage constraint caused by the stored load unit j. Unfortunately, we cannot ignore all positions within the range of ℓ^- to the right of stored load units. If the distance between stored load units i and j is not greater than $2 \cdot \ell^-$ in $\mathcal{A}_i^\circ \cup \mathcal{A}_j^\circ$, all decimeter positions between the load units are contained. As these sets are used to exclude x-y-positions this would cause that for no location in between i and j the ground coverage would be checked. So, we introduce additional sets for stored load units that only contain the x-y-position with distance of ℓ^- to the left of a stored load unit if the corresponding ground level location is contained in the set $\mathcal{A}$ of feasible load unit positions.

$$\mathcal{A}_i^\star = \{(k,l) \in \mathcal{X} \times \mathcal{Y} \text{ with } (k,l,1) \in \mathcal{A}; \lambda_i = l; k + \ell^- = \alpha_i\}$$

Now we can state the set of all ground level locations that have to be considered for the coverage constraint as

$$\hat{\mathcal{A}} = \left((\mathcal{X} \times \mathcal{Y}) \setminus \bigcup_{j \in \mathcal{C}^{fix}} \mathcal{A}_j^\circ \right) \cup \bigcup_{j \in \mathcal{C}^{fix}} \mathcal{A}_j^\star$$

To facilitate the MIP we introduce a further indexed set to access all ground level storage x-y-positions (k_1, l_1) for a load unit i that lead to a coverage of a considered x-y-position

$(k,l) \in \hat{\mathcal{A}}$ by load unit i:

$$\overline{\mathcal{A}}_{ikl} = \Big\{(k_1,l_1); (k_1,l_1,1) \in \mathcal{A}; l_1 = l; \max(0, k-\ell_i) \le k_1 \le k\Big\},$$

The resulting ground coverage IP reads:

$$\min \ (3.2) - (3.3)$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{C}} \sum_{(k_1,l_1) \in \overline{\mathcal{A}}_{ikl}} x_{ik_1l_11} \le 1 \qquad ((k,l) \in \hat{\mathcal{A}}) \tag{3.17}$$

$$(3.4), (3.7) - (3.15)$$

The new constraints (3.17) model that at the ground level each location is covered at most by one load unit. As the ground coverage constraint is the main difference to the distance-IP we will refer this model as coverage-IP.

3.8.3 A first integer x-position MIP

As the models presented so far are based on a spatial discretization, the number of variables is relatively large for storage areas with a large terminal length. An alternative is to determine the x-positions of the load units by positive integer variables S_i for all load units $i \in \mathcal{C}$. With this major change we adopt the distance model IP (see Section 3.8.1) and formulate a MIP-model without spatial discretization.

To complete the x-position information provided by the variables S_i to a whole location of a load unit, we furthermore introduce decision variables for storage at the ground level of the storage lane and stacking of load units.

Whether a load unit is stored at the ground level, is described by binary variables r_i with

$$r_i = \begin{cases} 1, & \text{iff load unit } i \text{ is stored at the ground level} \\ 0, & \text{otherwise.} \end{cases}$$

To determine the lane (y-position) of a load unit, we use the binary variables y_{il} for load units $i \in \mathcal{C}$ and lanes $l \in \mathcal{Y}$ with

$$y_{il} = \begin{cases} 1, & \text{iff load unit } i \text{ is stored in lane } l \\ 0, & \text{otherwise.} \end{cases}$$

Instead of introducing variables to determine the level of a load unit, we introduce binary variables z_{ij}, that describe if a load unit i is stacked onto a load unit j. The variables z_{ij} are only introduced for load units i and j, iff i can be stacked onto j. So, z_{ij} exists for load units i and j that are stackable, if the distance in between the corner castings is the same ($\hat{\ell}_i = \hat{\ell}_j$) and the length of load unit i is not larger than the length of load unit j ($\ell_i \le \ell_j$). By restricting the domains of the stacking variables z_{ij} the stacking restrictions of (H3) and (H4) are stated and do not have to be enforced in constraints. Note, that up to $b-1$ load unit may be stacked on top of a load unit but here we do not consider the stacking order. To access all load units that may be stacking partner above or below a stackable load unit we introduce two sets with load units. $\underline{\mathcal{M}}_i$ for all $i \in \overline{\mathcal{M}}$ contains all load units j where load unit i may be stacked onto and is defined as

$$\underline{\mathcal{M}}_i = \{j \in \overline{\mathcal{M}}; j \neq i; \hat{\ell}_i = \hat{\ell}_j; \ell_i \leq \ell_j\} \ .$$

In the set $\dot{\mathcal{M}}_j$ with $j \in \overline{\mathcal{M}}$ all load units i that can be stacked onto the stackable load unit j are contained. $\dot{\mathcal{M}}_j$ is defined as

$$\dot{\mathcal{M}}_j = \{i \in \overline{\mathcal{M}}; i \neq j; \hat{\ell}_i = \hat{\ell}_j; \ell_i \leq \ell_j)\} \ .$$

If more than one load unit is stacked onto another load unit, the load unit stack can be filled up with the stacked load units ordered by their lengths in non-decreasing order.
For different pairs of constraints we need to ensure that one of the constraints is respected. So we introduce switch-variables to activate or deactivate these constraints. These are the binary variables v_{ij} for all $i, j \in \mathcal{M}$ and c_i^{sw} for all $i \in \mathcal{C}$. While the variables v_{ij} are used to model pairwise non-overlapping of load units, c_i^{sw} is used to compute the minimum of the transport distances in x- and y-direction.
Additional auxiliary variables are needed for the objective function. To measure the transport distance in x-direction we introduce the auxiliary non-negative variables c_i^+ and c_i^- for every load unit $i \in \mathcal{C}$. To linearize the minimum and maximum expression in the objective we introduce the variables c_i^{min} and c_i^{max} for all load units $i \in \mathcal{C}$.

With the above defined decision variables and sets the integer x-position MIP can be stated as:

$$\min \sum_{i \in \mathcal{C}} \big(w_1 \cdot c_i^{max} + w_2 \cdot c_i^{min} \big) \tag{3.18}$$

$$+ w_3 \cdot \sum_{i \in \mathcal{C}} \sum_{j \in \mathcal{M}; \hat{\ell}_i = \hat{\ell}_j; \ell_i \leq \ell_j} z_{ij} \tag{3.19}$$

s.t.

$$S_i - \alpha_i - c_i^- + c_i^+ = 0 \qquad (i \in \mathcal{C}) \tag{3.20}$$

$$c_i^- + c_i^+ \leq c_i^{max} \qquad (i \in \mathcal{C}) \tag{3.21}$$

$$\sum_{l \in \mathcal{Y}} \overline{d_{il}} \cdot y_{il} \leq c_i^{max} \qquad (i \in \mathcal{C}) \tag{3.22}$$

$$c_i^- + c_i^+ \leq c_i^{min} + c_i^{sw} \cdot M_1 \qquad (i \in \mathcal{C}) \tag{3.23}$$

$$\sum_{l \in \mathcal{Y}} \overline{d_{il}} \cdot y_{il} \leq c_i^{min} + (1 - c_i^{sw}) \cdot M_1 \qquad (i \in \mathcal{C}) \tag{3.24}$$

$$r_i = 1 \qquad (i \in \widehat{\mathcal{C}}) \tag{3.25}$$

$$r_i + \sum_{j \in \underline{\mathcal{M}}_i} z_{ij} = 1 \qquad (i \in \overline{\mathcal{C}}) \tag{3.26}$$

$$\sum_{l\in\mathcal{Y}} y_{il} = 1 \qquad (i \in \mathcal{M}) \tag{3.27}$$

$$S_i + \ell_i + 1 - S_j \leq L \cdot (5 - r_i - r_j - v_{ji} - y_{il} - y_{jl}) \qquad (i,j \in \mathcal{M}; i \neq j; l \in \mathcal{Y}) \tag{3.28}$$

$$v_{ij} + v_{ji} = 1 \qquad (i,j \in \mathcal{M}; i \neq j) \tag{3.29}$$

$$(b-1) \cdot r_j - \sum_{i\in\dot{\mathcal{M}}_j} z_{ij} \geq 0 \qquad (j \in \overline{\mathcal{M}}) \tag{3.30}$$

$$S_i - S_j \leq L \cdot (1 - z_{ij}) \qquad (j \in \overline{\mathcal{M}}; i \in \dot{\mathcal{M}}_j) \tag{3.31}$$

$$S_i - S_j \geq -L \cdot (1 - z_{ij}) \qquad (j \in \overline{\mathcal{M}}; i \in \dot{\mathcal{M}}_j) \tag{3.32}$$

$$\sum_{l\in\mathcal{Y}} (l \cdot y_{il} - l \cdot y_{jl}) \leq m \cdot (1 - z_{ij}) \qquad (j \in \overline{\mathcal{M}}; i \in \dot{\mathcal{M}}_j) \tag{3.33}$$

$$\sum_{l\in\mathcal{Y}} (l \cdot y_{il} - l \cdot y_{jl}) \geq -m \cdot (1 - z_{ij}) \qquad (j \in \overline{\mathcal{M}}; i \in \dot{\mathcal{M}}_j) \tag{3.34}$$

$$S_i - \alpha_i = 0 \qquad (i \in \mathcal{C}^{fix}) \tag{3.35}$$

$$y_{i\lambda_i} = 1 \qquad (i \in \mathcal{C}^{fix}) \tag{3.36}$$

$$z_{ij} = 1 \qquad (i,j \in \mathcal{C}^{fix}; \beta_j = 1; \alpha_i = \alpha_j; \lambda_i = \lambda_j) \tag{3.37}$$

$$r_i = 1 \qquad (i \in \mathcal{C}^{fix}; \beta_i = 1) \tag{3.38}$$

$$1 \leq S_i \leq L - \ell_i \qquad (i \in \mathcal{M}) \tag{3.39}$$

$$y_{il} \in \{0,1\} \qquad (i \in \mathcal{M}; l \in \mathcal{Y}) \tag{3.40}$$

$$z_{ij} \in \{0,1\} \qquad (j \in \overline{\mathcal{M}}; i \in \dot{\mathcal{M}}_j) \tag{3.41}$$

$$v_{ij} \in \{0,1\} \qquad (i,j \in \mathcal{M}; i \neq j) \tag{3.42}$$

$$r_i \in \{0,1\} \qquad (i \in \mathcal{M}) \tag{3.43}$$

$$c_i^{sw} \in \{0,1\} \qquad (i \in \mathcal{C}) \tag{3.44}$$

$$S_i, c_i^-, c_i^+, c_i^{min}, c_i^{max} \geq 0 \qquad (i \in \mathcal{C}) \tag{3.45}$$

The objective is again to minimize a weighted sum of the maximum and minimum transport costs (3.18) and the number of load units that are not stored at the ground level (3.19). While in (3.19) the number of load units above the ground level can be easily calculated using the variables z_{ij}, some auxiliary variables are used to compute the maximum and minimum length of the loaded crane moves in (3.18). The transport costs in x-direction are calculated by the sum of the auxiliary variables c^+ and c^- which are forced to be set greater or equal to the transport distance in x-direction in constraints (3.20). If load unit i is transported to a larger x-position, c^+ accounts the transport distance in x-direction. If i is transported to a smaller x-position, c^- accounts the transport distance. Furthermore, due to constraints (3.21) to (3.24) the auxiliary variables c_i^{min} and c_i^{max} are set correctly. These variables are used in the objective function. While in (3.21) and (3.22) c_i^{max} is forced to be greater or equal to the transportation cost in x- and y-direction, the variables c_i^{min} only have to be greater than one of the transportation costs (see constraints (3.23) and (3.24)). The smaller one is selected by the variables c_i^{sw} which deactivate the x-direction if set to one or the y-direction if set to zero. Note, that the transportation cost in y-direction is determined by the assignment to a lane with the

variables y_{il} in (3.22) and (3.24). As for practical settings the length of the loaded crane move in x-direction is always greater than the length in y-direction, we can set $M_1 = L$. Constraints (3.25) force the ground level assignment variable r_i to be set to one for all load units $i \in \hat{\mathcal{C}}$ that are not stackable. For stackable load units $i \in \overline{\mathcal{C}}$ constraints (3.26) ensure that these load units are either stored at the ground level or stacked on top of one other load unit. For load units of the set $\mathcal{C}^{fix}$ these assignments are fixed and enforced later. For all load units $i \in \mathcal{M}$ exactly one lane has to be chosen due to (3.27).
Constraints (3.28) and (3.29) ensure non-overlapping for load units i, j that are both at the ground level in the same lane. The switch variable v_{ij} states that either i is stored to the left of j (case $v_{ij} = 1$) or j is stored to the left of i (case $v_{ji} = 1$). Due to (3.29) the sequence of the load units is defined in one direction. In (3.28) it is stated that load units do not overlap according to the chosen sequence if they are stored in the same lane at the ground level. If two load units i and j are not stored in the same lane or at least one of them is not stored at the ground level, the assignment of v_{ij} and v_{ji} is not of relevance for the model. In this case constraint (3.28) is switched off due to the right-hand side for the load units i and j.
Constraints (3.30) limit the stacking height for stackable load units and ensure that only onto load units at the ground level other load units may be stacked. Constraints (3.31) to (3.34) model that stacked load units have to be stored at the same x-position and in the same lane. While constraints (3.31) force $S_i - S_j$ to be smaller than or equal to zero if i is stacked onto j, (3.32) ensure that in the stacking case $S_i - S_j$ is greater than or equal to zero. So, together (3.31) to (3.32) force $S_i = S_j$ if i is stacked on top of j. For the lane equality is enforced in (3.31) and (3.34) analogously.
Constraints (3.35) to (3.38) do the fixation of the location for load units of the set $\mathcal{C}^{fix}$. The x-positions of load units $i \in \mathcal{C}^{fix}$ are fixed by constraints (3.35) while the storage lane of stored load units is fixed by constraints (3.36). If a load units is already stacked onto another one in the initial storage area setting, in constraints (3.37) it is enforced that this stacking is kept in the solution. Furthermore, for load units that are initially stored at the ground level, r_i is set to one in constraints (3.38). If this would not be done, it would be possible to store load units of the set $\mathcal{C}$ under load units of the set $\mathcal{C}^{fix}$ which is not feasible. In constraints (3.39) the values of the x-position variables S_i for all load units of the set $\mathcal{C}$ are restricted. To position a load unit without violating the lower end of the terminal S_i is not allowed to be smaller than one for all load units of the set $\mathcal{C}$. The upper end of the terminal limits the x-position variables to a maximum value of $L - \ell_i$ for all load units of the set $\mathcal{C}$.
Finally the domains of the variables are defined in (3.39) and (3.45). We will refer to this MIP formulation as x-pos I in the following.

3.8.4 An aggregated integer x-position MIP

In this section we will adapt the MIP formulation of Section 3.8.3 by aggregating the non-overlapping constraints for load units of the set $\mathcal{C}^{fix}$ to reduce the number of constraints and strengthen constraints (3.28) of the MIP formulation as x-pos I. After ilustrating the idea with a small example, we will present the adapted MIP formulation.

The positions that are blocked for a load unit $i \in \mathcal{C}$ by a load unit $j \in \mathcal{C}^{fix}$ are the

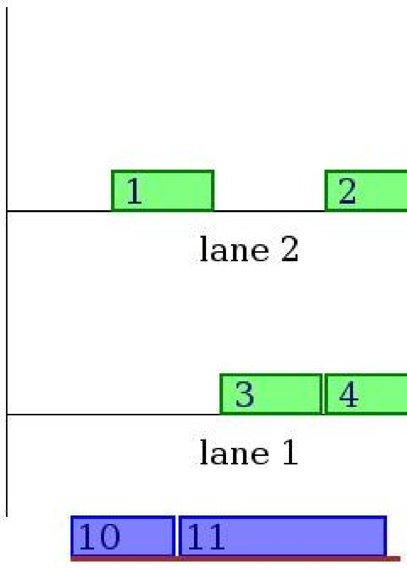


Figure 3.14: Example for load unit aggregation for integer x-position MIP.

positions in the interval $[S_j - \ell_i; S_j + \ell_j]$. The positions of these interval are excluded from the solution by constraints (3.28) and (3.29) in the x-pos I MIP of Section 3.8.3. If there is another load unit $j' \in \mathcal{C}^{fix}$ with $S_j + \ell_j + \ell_i \geq S_{j'}$ we can consider the consolidated interval $[S_j - \ell_i; S_{j'} + \ell_{j'}]$ which contains all blocked positions for load unit i caused by load units j and j'. We can build aggregates of load units of the set $\mathcal{C}^{fix}$ by load unit sequences π^i with maximum possible length f_i, if for consecutive members of π^i hold that $S_{\pi^i_j} + \ell_{\pi^i_j} + \ell_i \geq S_{\pi^i_{j+1}}$. For the aggregate π^i the interval $[S_{\pi^i_1} - \ell_i; S_{\pi^i_{f_i}} + \ell_{\pi^i_{f_i}}]$ is blocked for load unit i. The interval contains all x-positions starting with the smallest x-position blocked by the first load unit of the sequence and ending with the largest x-position blocked by the last load unit of the sequence π^i.

Example 3.5. To illustrate the idea of aggregating stored load units we consider an example with $m = 2$ storage lanes of length $L = 311$. We do not consider stacking in the example as it is not relevant for the aggregation. We consider five stored load units in the set $\mathcal{C}^{fix}$ which all have the length of $\ell_i = 61$ for $i = 1, \dots, 5$. The first two load units with the IDs 1 and 2 are positioned in lane two, the other three load units $(3, 4, 5)$ are stored in lane one. The starting and ending x-positions are listed in Table 3.2. We assume that the set $\mathcal{C}$ contains two load units $(10, 11)$ with length $\ell_{10} = 61$ and $\ell_{11} = 122$. We do not consider the objective, so starting position of the load units of the set $\mathcal{C}$ as well as transport costs are not of relevance. The setting of the example is visualized in Figure 3.14, where the two storage lanes with the load units of the set $\mathcal{C}^{fix}$ are shown in green and the load units of the set $\mathcal{C}$ are drawn in blue on a symbolized brown wagon.

In Table 3.3 for the load units 10 and 11 the unaggregated blocked intervals (used in (3.28) and (3.29)) and the aggregated intervals for valid sequences are presented. To model blocking due to already stored load units of the set $\mathcal{C}^{fix}$ 20 constraints are necessary. When aggregation is used, the same restrictions can be modeled with 5 constraints. These constraints often forbid larger intervals.

load unit	start position	end position	lane
1	63	124	1
2	188	249	1
3	126	187	2
4	188	249	2
5	250	311	2

Table 3.2: Start and end x-position of load units of the set $\mathcal{C}^{fix}$ in Example 3.5.

The aggregation (see Table 3.3) does not have an impact on the modeling of blocking for load unit 10 in lane two. But in lane one the three stored load units can be aggregated to one sequence. For load unit 11 for both lanes only one aggregate has to be considered.

	Load unit 10				Load unit 11			
	unaggregated		aggregated		unaggregated		aggregated	
load unit	first	last	first	last	first	last	first	last
1	2	124	2	124	-59	124	-59	249
2	127	249	126	249	66	249		
3	65	187	65	311	4	187	4	311
4	127	249			66	249		
5	189	311			128	311		

Table 3.3: First and last blocked x-position for load units and load unit aggregations in Example 3.5.

$\square$

As the storage area in practice is often high utilized, the aggregations can lead to a great reduction of blocking intervals.

We will now explain how we adopted the x-pos I MIP of Section 3.8.3 to integrate load unit aggregation. We restricted the domains of the constraints (3.28) and (3.29) and the herein used variable v_{ij}. Instead of considering all load unit pairs of the set $\mathcal{M}$ we just consider pairs of load units with $i, j \in \mathcal{C}$. We number the maximal sequences for load unit i from 1 to θ_i and name these sequences $\pi^{i\vartheta}$ for $\vartheta = 1, \dots, \theta_i$. Let us define the parameters $\sigma_{i\vartheta}$ and $\omega_{i\vartheta}$ such that $[\sigma_{i\vartheta}; \omega_{i\vartheta}]$ is the interval that is blocked for load unit i by aggregate ϑ and $\rho_{i\vartheta}$ is the lane of aggregation ϑ. The blockings caused by stored load units of the set $\mathcal{C}^{fix}$ are considered in new constraints which work analogously to constraints (3.28) and (3.29) but instead of single load units aggregates of load units are considered. So, these constraints exist for all load units $i \in \mathcal{C}$ times all θ_i the number of aggregates which are the maximal sequences for i. As we know the lane of aggregations and that they are at the ground level, we do not need to introduce switch variables for the lane and the ground level in the right-hand side of the new constraints. Analogously to v_{ij} we introduce the switch variables $u_{i\vartheta}$ for $i \in \mathcal{C}$ and $\vartheta = 1, \dots, \theta_i$. These variables indicate wether load unit i is stored to the left ($u_{i\vartheta}$=1) or the right ($u_{i\vartheta}$=0) of aggregation ϑ.

So the adapted MIP (which we call x-pos II MIP) reads as follows:

$$\min\ (3.18) - (3.19)$$

s.t.

$$
\begin{aligned}
S_i + \ell_i + 1 - S_j &\le L \cdot (5 - r_i - r_j - v_{ji} - y_{il} - y_{jl}) && (i,j \in \mathcal{C}; i \neq j; l \in \mathcal{Y}) && (3.46)\\
v_{ij} + v_{ji} &= 1 && (i,j \in \mathcal{C}; i \neq j) && (3.47)\\
S_i + 1 - \sigma_{i\vartheta} &\le L \cdot (3 - r_i - u_{i\vartheta} - y_{i\rho_{i\vartheta}}) && (i \in \mathcal{C}; \vartheta \in \{1,\dots,\theta_i\}) && (3.48)\\
S_i + 1 - \omega_{i\vartheta} &\ge L \cdot (2 - r_i + u_{i\vartheta} - y_{i\rho_{i\vartheta}}) && (i \in \mathcal{C}; \vartheta \in \{1,\dots,\theta_i\}) && (3.49)\\
v_{ij} &\in \{0,1\} && (i,j \in \mathcal{C}; i \neq j) && (3.50)\\
u_{i\vartheta} &\in \{0,1\} && (i \in \mathcal{C}; \vartheta \in \{1,\dots,\theta_i\}) && (3.51)\\
(3.20) - (3.27), (3.30) &- (3.41), (3.43) - (3.45)
\end{aligned}
$$

The objective is not changed. In constraints (3.46), (3.47) and (3.50) the domains of i and j are reduced from $i,j \in \mathcal{M}$ to $i,j \in \mathcal{C}$. The new constraints (3.48) and (3.49) together with variable $u_{i\vartheta}$ (see (3.51) for the definition) model that each load unit is to the left or the right of all corresponding aggregations, if it is stored in the lane of the aggregation at the ground level.

3.9 Scheduling heuristic

In this section we describe a proposed scheduling heuristic for the storage planning problem. In our case the scheduling is not related to temporal but spacial planning. It is based on a list scheduling procedure that plans the load units iteratively at the best possible empty location and updates the storage area which means to block the chosen location of the load unit that has just been considered in the list. To represent the lanes of the storage area we propose an array-based approach and a representation by trees, where the root node corresponds to a storage lane and stored or planned load units branch this node in two child nodes. For the scheduling list we analyze different initial sortings and propose a local search procedure to obtain improved solutions.

The idea of the list scheduler is to iteratively plan the load units of the list containing the elements of the set $\mathcal{C}$ at their best possible locations. To avoid gaps that cannot be used anymore, we decided to choose the *adjoin*(*mindist*) policy introduced in Section 3.1. So, load units are either stored next to another load unit in the same lane of the storage area or have a distance of at least *mindist* to the next load unit at the left and the right in the same lane.
If the best possible locations of two load units overlap, it depends on the order of these load units which of them gets the best possible location (if no other load unit that has been planned earlier – because it is in a smaller position in the list – partially occupies this best location).
Note, that due to the number and the properties of the load units in the sets $\mathcal{C}$ and $\mathcal{C}^{fix}$ or due to the scheduling list, it can happen that for a load unit of the set $\mathcal{C}$ no feasible position can be found in the storage area. E.g. for an instance with one lane that is just

capable to host one of the uniform load units and a stacking height of three, no instance with four load units has a feasible solution. On the other hand, it can happen that the scheduler does not find a feasible solution even so a feasible solution exists. This will be illustrated in Example 3.6.

Example 3.6. Consider a storage area with one lane and a length of $L = 200$ and three load units to be stored. Let the load units have the lengths $\ell_1 = \ell_2 = 61$ and $\ell_3 = 122$ (two 20 feet and one 40 feet container). If for the scheduling list $(1, 2, 3)$ the load units one and two are positioned at their best feasible locations at the ground level, there is no feasible location for load unit three. But if load units one and two build a stack at the beginning or the end of the lane, load unit three can be feasibly positioned at the ground level. So, this example shows that there are scheduling lists for which no feasible solution may be found even if a feasible solution for the problem exists. Note that he maximum stacking height b is not of relevance for this example, if it is not smaller than 2. If stacking is not feasible at all, the instance is infeasible. $\square$

For the implementation of the scheduler we chose two different approaches, an array-based approach and one based on trees, so-called area-trees. The array-based approach works with three arrays to represent a lane of the storage area. All these arrays have the length of the storage lane in decimeter. The first array is boolean and represents whether a x-position is free or occupied by a load unit. In the second array we store whether the occupying load unit is of the set $\mathcal{C}$ (load units to be planned) or $\mathcal{C}^{fix}$ (initial storage allocation) for occupied x-positions. This array is also boolean. The last array contains an id to the load unit that occupies the decimeter x-position, if the x-position is not free. With the first array it is easily possible to check whether a load unit can be positioned at a decimeter x-position in a storage lane. If all x-positions that are within the interval from the assignment x-position to the assignment x-position plus the length of the load unit to be assigned are free, the load unit can be assigned to the location with the assignment x-position, the considered lane and at the ground level. In addition we can check with the first array if either the load unit is stored adjoined to another load unit or if the distance of *mindist* stays empty to the right and the left. If the load unit shall be stored at that free location, we have to update the three arrays at the range x-positions. If the location is not free, the occupying load unit (accessible by the third array) is a possible candidate for stacking the load unit that has to be assigned. Here, it has to be checked if there is still an empty location on top of the occupying load unit and if the stacking restrictions are met. Whenever a feasible load unit location at the ground level or on top of another load unit is found, we calculate the partial objective value for assigning the load unit to the ground level or a stacking location and update the best location found so far if the objective is better than the best objective found so far.
For exploring the free locations in the storage area we successively treat the storage lanes. For each lane we start with the x-position that equals the load unit x-position on the train or the truck and continue simultaneously to the right and the left in decimeter steps unless a location is occupied. If a location is occupied, we calculate the smallest distance to reach a potential free x-position in the same lane at the ground level according to the occupying load unit. We have to do that for both directions and take the minimum of the two values to proceed. With this we can omit to check x-positions for which we already

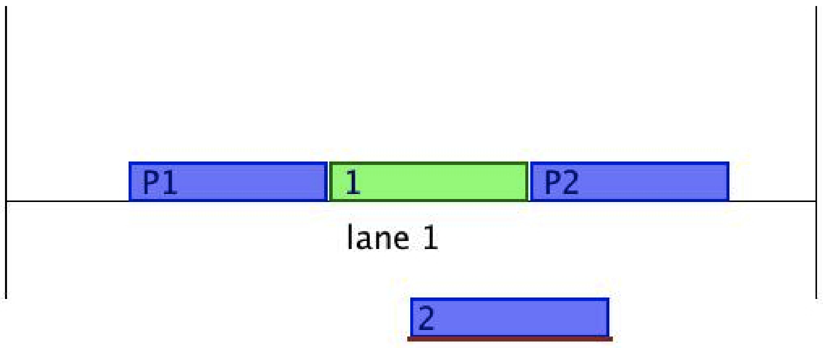


Figure 3.15: Example for location search in the scheduler.

know that they are not free. An example of this procedure is presented in Example 3.7. Furthermore, we can stop the search in a lane, when we check a x-position that causes a worse objective value than the best location found so far. As we increase the distance in x-direction, the objective cannot be better for a location with a greater x-distance.

Example 3.7. Let us consider a small example with one storage lane where load unit 1 with a length of 122 dm is stored at decimeter x-position 200 and load unit 2 with a length of 122 and x-position 250 on the train has to be assigned. This example setting is shown in Figure 3.15. The stored load unit 1 is visualized in green and load unit 2 is shown in blue on a symbolized wagon. Furthermore, the best location to the left of load unit 1 shown with the label P1 and the best location to the right with the label P2.

For x-position 250 we detect that the location is occupied. The minimum distance to the right is the x-position of the occupying load unit plus its length minus the x-position of the load unit (which has to be assigned) $200 + 122 - 250 = 72$ (this results in location P2 except the one dm safety distance). The minimum distance to the left is the x-position plus the length of the load unit that has to be assigned minus the x-position of the occupying load unit $250 + 122 - 200 = 172$ (this results in location P1 except the one dm safety distance). So we proceed with distance $\min(72, 172) + 1 = 73$ (the +1 corresponds to the min distance between load units of one decimeter) which means that we consider the x-positions $250 + 73 = 323 =$P2 and $250 - 73 = 177$. The x-position 323 is a free location and has to be considered. The x-position 177 is not feasible as the load unit would end at x-position 299 which would mean that the load units both occupy x-positions 200 to 299. Now, the minimum distances are 172 and 0. As the minimum is also 0, we take the last distance incremented by one as new distance which is 74. So, we consider the x-positions 324 and 176, where 324 is feasible but worse than 323 and 176 is again not feasible. At this state we can stop the search for a better location, as we will not find any better locations in this lane. Note, that for an instance with more than one lane, we have to continue with the next lane until a feasible ground level location is found or the partial objective value is higher than that of the best location found so far. □

After a best possible location is found for the list element to consider, we update the three arrays accordingly, which means to assign the load unit to the location and continue with

the next load unit in the list. After all load units have been assigned to feasible locations we can calculate the objective value as the sum of the partial objective values for assigning single load units.
To compute the storage plan for another list, we clear the storage area by deleting all the load units of the set $\mathcal{C}$. Therefore, the second array is used to check whether a x-position is occupied by a load unit of the set $\mathcal{C}$. If that is the case, the x-position is freed in the first array and the id of the load unit is deleted in the third list. Furthermore, load units that are stacked on top of other have to be deleted if they are element of $\mathcal{C}$. After this step the arrays for lanes represent the initial state of the storage area and the storage plane for another list can be calculated without the need of initializing the storage again.

The idea of an area tree is that it represents a whole lane or a subarea which is described by a start and end x-position. Such an area tree is completely empty if it does not have child nodes. If a load unit is stored in the area tree, the load unit is assigned to the area tree and the areas left and right of the load unit are represented by child area trees. An initial area tree offers mainly two methods – positioning a load unit in the area tree or calculating the best possible location for a load unit in the area tree. When a load unit is positioned at a specific location in the area tree, the load unit is assigned to the area tree if the area tree is a leaf node and represents the specific location which has to be feasible. If the area tree has already an assigned load unit, the load unit to be assigned is either stacked onto the assigned load unit (if this stacking location is intended) or forwarded to the child nodes and then handled in the child nodes. An area tree also offers to find a best possible solution in the area represented by the root node. Therefore, the stacking location on the assigned load unit is considered for area trees that are not leaf nodes and furthermore the request is forwarded to the leaf nodes. In leaf nodes first it is checked whether the represented area is large enough to store the requested load unit. If that is the case, the x-position of the load unit is considered if it is feasible which means that the load unit does not exceed the area and the load unit respects the minimum distance to the area borders. If this x-position is not feasible, further x-positions to be considered are the outmost locations of the area and the locations that exactly respect the minimum distance for storing a load unit without adjoining another load unit. For all these locations we have to check if the load unit does not exceed the area borders. So for area tree nodes that are not a leaf only one location is considered. For leaf nodes up to four locations have to be considered.
Also in the area tree implementation it is possible to clear the area tree to represent again the initial storage area state. For this purpose we start with the root area tree. In a node the assigned load unit and the child nodes are deleted (if they exist) if the assigned load unit is element of $\mathcal{C}$. Also stacked load units of the set $\mathcal{C}$ are deleted even if the assigned load unit is of the set $\mathcal{C}^{fix}$. If the child nodes are not deleted, the request of clearing is forwarded to them. This approach works, since we first place all load unit of the set $\mathcal{C}^{fix}$ into the storage area and afterwards the load units of $\mathcal{C}$. So, if in a node a load unit of the set $\mathcal{C}$ is assigned, there cannot be load units of the set $\mathcal{C}^{fix}$ in child nodes.

As a initial solution we test different ways of sorting the load units:

- ordering the load units descendingly by x-position (*pos*)

- ordering the load units descendingly by length (*length*)
- ordering the load units according to stackability, first non stackable then stackable load units (*stack*)
- ordering the load units hierarchically by length and stackability ($length, stack$)
- ordering the load units hierarchically by stackability and length ($stack, length$)

The option *pos* is just used as a neutral standard sorting. The length of the load units can have an impact on the solution quality as the number of available locations at the ground level is less for longer load units. If a lot of stackable load units are positioned at the ground level, there may not be enough locations for load units that are not stackable. So, also the property of stackability may have an impact on the solution quality for the initial list of the scheduling heuristic.

To obtain better solutions we propose a local search with the neighborhoods swap, forward shift and backward shift, where swap means that the positions of two load units are exchanged. The shift operators shift a load unit to the left or the right. Note, that with the shift also the load units that are passed by the shifted element change their position in the list by one. With these neighborhoods the list is changed and by the scheduler possibly a new, potentially better solution is generated. To avoid changes of the list that do not affect the generated schedule, we decided to consider only neighbors that effect load units with close origin x-positions. For the neighborhood swap we only consider swap operations of load units whose origin x-positions are within a distance of less than $maxdist$. So, for the load units i and j a swap is considered if $|O_i - O_j| < maxdist$. For the shift neighborhood we limit the shift operations to be considered with the same origin x-positions rule applied to the last load unit which is passed by the shifted load unit in the scheduling list. Let us denote this last passed load unit by j, then again the shift operation is considered if $|O_i - O_j| < maxdist$.

3.10 Computational experiments

In this section we describe results obtained with the above defined IP- and MIP-models and the heuristic with four real world data sets. After introducing the properties of the data sets we compare the results of the IP- and MIP-models with different runtimes for two small data sets in Section 3.10.1. The main focus of Section 3.10.2 is to calibrate the different parameters for the scheduling heuristic using one of the data sets. For the larger data sets results are compared for a MIP-model and the heuristic in Section 3.10.3. Finally, in Section 3.10.4 the results for the practical storage planning are discussed.

Our storage planning instances are composed of 4 data sets which differ in the number of lanes and the length of the terminal. We will first describe the properties of the storage area and afterwards specify the train properties.
Two groups of data sets consist of storage areas with two storage lanes and are 8 real world storage settings from the German DUSS terminals Großbeeren (close to Berlin) and Kornwestheim (close to Stuttgart). The two data sets differ in the length of the terminal.

One considers the case with storage lanes of 700 meters, which is the real length in the terminals. The other data set with two lanes considers storage area segments of 250 meters. Considering only partial storage areas can be motivated by fixed crane working areas, which restrict the x-positions a crane is allowed to work at. Furthermore, in real world instances load units that have a large distance in between them on the train, do not influence the decision of their storage locations. For a comparison of algorithms it is also helpful to consider instances that can be solved by exact approaches and it is usually easier (at least for models with spatial discretization) to solve instances if the length of the terminal is smaller.

The instances with five lanes are 4 real world storage area settings from the German DUSS terminals Hamburg Billwerder and Köln Eifeltor. As with the two lanes data sets we have one data set with a terminal length of 700 meters and one with a storage area subarea of 250 meters.
We always consider four trains that have to be unloaded. We have two different sets of trains. The first set consists of trains that are all loaded with stackable load units. For the second set about half of the load units are not stackable.
As we combine each storage area setting and each train set, this means that we have two data sets of 16 instances with 2 storage lanes and 2 data sets of 8 instances with 5 storage lanes. The properties of our data sets are summarized in Table 3.4. The number of lanes, the length of the terminal and the number of instances in the data set are listed. Note, that in data1 to data8 of data sets I and II as well as datakln1 to dataHAB2 of data sets III and IV only stackable load units occur. Contrarily, in data11 to data18 of data sets I and II as well as datakln3 to dataHAB4 of data sets III and IV also non-stackable load units occur. In the tables with results, we separate the instances with non-stackable and stackable load units by a horizontal line.

Note, that for instance data17 of the data set II no feasible solution exists due to the high utilization of the storage area and the high number of load units on the trains that are not stackable.

Data set	m	L	# instances	n
I	2	700	16	112 to 160
II	2	250	16	49 to 70
III	5	700	8	112 to 160
IV	5	250	8	49 to 70

Table 3.4: Storage planning data sets with properties.

All the computations where carried out on a Linux system with an intel i7 processor and 8 GB of RAM. For IPs and MIPs we used the modeling language Zimpl [62] to generate lp-files which where solved with CPLEX 12.4 [56]. The heuristics have been implemented in C++ and compiled with a gcc-compiler. All the runtimes are reported in seconds, if not declared different.

3.10.1 Comparison of MIP-formulations

In this section we report test results of the IP- and MIP-models for data set II. We analyze the quality of the best integer solution that could be found by CPLEX with the runtime limits 5 min and 3 h for the different IP- and MIP-models. Furthermore, we look at the lower bound provided by CPLEX. At the end of this section we also report results for the data set IV with only one train. Instances of data set IV are the largest instances for which we computed solutions with the spatial IP-models.

In Tables 3.5 and 3.6 the objective values and the gaps (in %) to lower bounds (reported by CPLEX) are listed for data set II for the distance- and the coverage-IP as well as for the MIP-models x-pos MIP I and x-pos MIP I with a time limit of 5 minutes and 3 hours, respectively.

Instance	distance-IP		coverage-IP		x-pos MIP I		x-pos MIP II	
	objective	gap	objective	gap	objective	gap	objective	gap
data1	13610.7	2.5	13610.7	2.5	19457.8	51.0	17082.0	43.6
data2	15255.7	0.1	15255.7	0.1	19423.0	45.6	16889.2	38.9
data3	14474.8	1.1	14474.8	1.1	20871.4	55.5	16057.3	42.4
data4	12718.8	1.5	12718.8	1.5	17532.0	46.0	13720.9	30.5
data5	16860.7	5.9	16860.7	5.9	23784.9	57.6	24615.7	53.9
data6	15679.7	2.6	15679.7	2.6	19956.0	52.3	20624.4	53.5
data7	22551.8	0.0	22551.8	0.0	23131.3	52.5	22572.4	37.6
data8	13891.8	2.6	13891.8	2.6	16294.6	44.3	21891.7	58.2
data11	68189.5	67.8	52705.4	58.4	no sol.	-	no sol.	-
data12	55570.9	52.8	55570.9	52.8	no sol.	-	no sol.	-
data13	57247.7	58.8	57247.7	58.8	no sol.	-	no sol.	-
data14	65673.6	67.6	65673.6	67.6	no sol.	-	no sol.	-
data15	no sol.	-	no sol.	-	no sol.	-	no sol.	-
data16	no sol.	-	no sol.	-	no sol.	-	no sol.	-
data17	inf	-	inf	-	inf	-	inf	-
data18	no sol.	-	no sol.	-	no sol.	-	no sol.	-

Table 3.5: Comparison of the results of the IP- and MIP-models with a runtime limit of 5min for data set II.

One clear observation we can make from the two Tables (3.5, 3.6) is that the IP-models with spatial discretization produce much better results than the MIP-models with integer x-positions. The objective values as well as the gaps are better and for more instances integer solutions could be obtained within the runtime limit. Within the 5 min time limit for 12 of the 15 instances where a feasible solution exists, an integer solution can be obtained with the IP-models. With the time limit of 3 hours for all feasible instances solutions are computed. The changes due to the extended time limit in the solution quality are quite small for the 8 instances with stackable load units on the trains (data1 to data8). Here, the maximum gap can be reduced from 5.9 to 1.5%. For the instances that contain non-stackable load units (data11 to data18) on the trains for more instances integer solutions can be provided and the reduction of the gaps is much larger (from a

Instance	distance-IP		coverage-IP		x-pos MIP I		x-pos MIP II	
	objective	gap	objective	gap	objective	gap	objective	gap
data1	13439.6	0.2	13439.6	0.2	14345.4	33.5	14098.6	31.4
data2	15255.7	0.0	15255.7	0.0	15378.5	30.8	14791.0	28.2
data3	14474.4	0.8	14474.4	0.8	15286.4	38.8	15636.9	40.8
data4	12642.5	0.4	12642.5	0.4	13177.1	26.8	13106.5	25.1
data5	16330.7	1.5	16330.7	1.5	18372.2	42.7	18047.0	36.2
data6	15526.2	0.1	15526.2	0.1	15984.6	40.2	15567.1	37.0
data7	22551.8	0.0	22551.8	0.0	22586.7	48.4	22296.3	34.2
data8	13722.3	0.2	13823.3	0.9	14173.3	35.9	14180.9	35.3
data11	26141.8	13.4	26141.8	13.4	no sol.	-	no sol.	-
data12	28995.9	2.9	28998.0	2.9	no sol.	-	no sol.	-
data13	25989.4	5.9	25954.3	5.8	no sol.	-	no sol.	-
data14	26138.9	16.0	26111.1	16.0	no sol.	-	no sol.	-
data15	36836.6	8.6	36836.6	8.6	no sol.	-	no sol.	-
data16	36939.9	11.2	36939.9	11.2	no sol.	-	no sol.	-
data17	inf	-	inf	-	inf	-	inf	-
data18	28208.5	6.3	28312.4	6.7	no sol.	-	no sol.	-

Table 3.6: Comparison of the results of IP- and MIP-models with a runtime limit of 3 hours for data set II.

maximum gap of about 68% (5 min) to 16% with the runtime of 3 hours). The infeasibility of instance data17 could be detected in less than 4 s with both IP-models. Furthermore, instance data2 could be solved to optimality in 274 s and data7 in less than 4 s. With the time limit of 5 min the coverage-IP performs slightly better than the distance-IP as for one instance a better solution can be obtained. With the time limit of 3 h for three instances the best solution can be obtained with the distance-IP and for two instances with the coverage-IP. So, overall there is no clear IP-model that outperforms the other one.
With the MIP-models even with the runtimes of 3 hours only for the instances with stackable load units on the trains integer solutions can be provided. For the integer x-pos MIPs the gaps can be reduced for the larger runtime. For the model x-pos MIP I the maximum gap can be reduced from 55.5 to 48.4% and the mean gap can be reduced from 50.6 to 37.1%. For the model x-pos MIP II the maximum gap can be reduced from 53.9 to 40.8% and the mean gap can be reduced from 44.8 to 33.5%. For the MIP-models no instance could be solved to optimality within the time limits. The detection of infeasibility for instance data17 took 170 s with both MIP-models. Comparing the solution quality of the MIP-models, we can say that for three instances x-pos MIP I could obtain the best objective and x-pos MIP II for 5 instances with the time limit of 5 min. For the larger time limit of 3 h x-pos MIP I could only find a best objective for one instance. For the other 7 instances with integer solutions x-pos MIP II outperformed x-pos MIP I according to the objective value. So, we see that the aggregation of load units leads to a better performance of the MIP-model.

In Table 3.7 we list the best lower bounds provided by CPLEX for the distance-IP, and the

Instance	distance-IP		x-pos MIP I		x-pos MIP II	
	5 min	3 h	5 min	3 h	5 min	3 h
data1	13269.8	13419.4	9523.8	9537.5	9633.9	9667.6
data2	15254.2	15255.7	10565.5	10637.2	10315.2	10618.4
data3	14313.5	14358.8	9296.8	9355.0	9243.6	9259.4
data4	12533.9	12589.4	9471.9	9648.2	9530.5	9819.0
data5	15867.0	16091.6	10091.6	10535.0	11342.6	11492.4
data6	15269.4	15517.9	9528.2	9563.0	9590.8	9812.0
data7	22551.7	22551.8	10981.0	11664.1	14089.2	14682.0
data8	13528.3	13698.1	9083.6	9084.2	9147.5	9181.3
data11	21947.1	22635.5	13297.6	13297.6	13388.8	13388.8
data12	26241.9	28148.3	14911.4	14911.4	14302.0	14312.4
data13	23576.1	24460.0	12752.1	12752.1	12788.7	12788.7
data14	21254.0	21946.4	13296.8	13296.9	13388.7	13423.3
data15	33266.7	33679.7	15146.4	15146.4	16983.1	16983.1
data16	32252.4	32789.1	13356.7	12544.2	13961.5	13961.5
data17	inf.	inf.	inf.	inf.	inf.	inf.
data18	25958.8	26421.4	12484.1	12484.1	12518.6	12518.6

Table 3.7: Comparison of the IP- and MIP bounds with a runtime limit of 5min and 3 hours for data set II.

models x-pos MIP I as well as x-pos MIP II. We did not list the bounds of the coverage-IP as the bounds did not differ from those of the distance-IP for the time limit of 5 min and differed only slightly for the three instances data11, data13 and data15 for the time limit of 3 h. For these instances the maximum deviation is 24.2 which corresponds to less than 0.1%. For the IPs the deviation of the bounds for the different time limits is in the same scale as the changes in the objective for the "stackable" instances data1 to data8. For the instances with load units that are not stackable the improvement of the bounds is quite small compared to the improvements in the best integer solution. For all instances the bounds of the IP-models are much better than those of the integer x-pos MIPs. The factor between the mean values is about 1.5.

In the following we will compare the results of the coverage-IP and the x-pos MIP II for a reduced version of data set IV with only one train. The main difference of data set IV is that the storage area contains 5 storage lanes. The results are listed in Table 3.8. The generated lp-files of the MIP-model x-pos MIP II have a size of about 100MB but the lp-files of the coverage-IP have a size of about 2GB. Generating these files with Zimpl took a few seconds for the MIP-model but up to 30 h for the IP-model. Even though the MIP-models need much more constraints according to load unit pairs, without the spatial discretization the lp-files are much smaller. On the other hand, Tables 3.7 and 3.8 (here the objectives equal the bounds since the instances are solved to optimality) show that the spatial discretization leads to considerable better bounds for data set II and the reduced data set IV. After the long time for generating the lp-files all the instances could be solved to optimality in less than 8 min for data set IV with the coverage-IP. With the model x-pos MIP II within the time limit of 3 h only for one instance the optimal solution could not be found. But due to the worse bounding only for 3 instances optimality could be proven. The mean gap reported by CPLEX is 11.6% but the mean gap to the optimal solution (known from the coverage-IP) is just 0.6%. The runtimes with CPLEX are much higher for the MIP-model as the execution is interrupted after the time limit for 5 of 8 instances. But for two of the instances the runtimes of the MIP-model outperform the runtime of the coverage-IP clearly (less then 4 s compared to more than 100 s).

Instance	coverage-IP			x-pos MIP II			
	objective	time	gap	objective	time	bound	gap
datakln1	5209.4	442	0.0	5350.8	10804	2846.7	46.8
datakln2	1560.7	140	0.0	1560.7	313	1560.7	0.0
dataHAB1	1240.2	111	0.0	1240.2	4	1240.2	0.0
dataHAB2	1040.3	104	0.0	1040.3	3	1040.2	0.0
datakln3	3915.2	290	0.0	3915.2	10804	3277.3	16.3
datakln4	4246.0	283	0.0	4246.0	10803	3321.1	21.8
dataHAB3	3497.1	348	0.0	3497.1	10803	3290.9	5.9
dataHAB4	3263.3	322	0.0	3263.3	10802	3194.3	2.1
mean	2996.5	255	0.0	3014.2	6791	2471.4	11.6

Table 3.8: Comparison of the results of the IP- and MIP-models for data set IV with 5 lanes but only one train.

Due to the long generation times with Zimpl we did neither test the coverage-IP nor the distance-IP (but the MIP-models) for instances with more load units (original data set IV) or a greater storage length (data sets I and III). Note, that hopefully the problem generation time for the IP-models can be reduced when CPLEX is directly called from C or JAVA. We did not implement this, but it might be timesaving if one wants to use IP-models and CPLEX for storage planning problems.

3.10.2 Calibration of the scheduling heuristic

In this section we analyze the results and runtimes of the scheduling heuristic using data set II. After looking at different sorting strategies for the load units (to be stored) to obtain an initial solution, we compare the different implementations of the scheduling heuristic and different options according to the scheduling heuristic like adjoin policies, first and best fit and the neighborhoods. We analyze the neighborhoods swap and forward as well as backward shift. We try to focus on "good" neighbors by limiting the *maxdist* parameter to force to consider neighbors that effect load units that are not too far from each other according to their origin x-positions. After these tests which are done on data set II we select parameters for the heuristic to be used for further computations and present results for data set II for these settings.

In Table 3.9 the mean objective ("obj.") and the number of instances for which no feasible solution could be found ("#inf") for the 16 instances of data set II are listed for different sorting strategies and heuristics. In the columns we tested to sort the load units by increasing origin x-positions O_i in "pos", by decreasing length in "length", by stackability

heuristic	pos		length		stack		length,stack		stack.length		time (s)
	obj.	#inf	obj.	#inf	obj.	#inf	obj.	#inf	obj.	#inf	
scheduler	27466.9	5	20467.5	8	23511.3	4	20467.5	8	22546.7	6	< 1
scheduler, reverse	23651.2	6	20038.1	9	18269.8	8	20038.1	9	20038.1	9	
LS, swap, first fit	19325.2	4	20812.5	3	20075.8	3	20634.5	3	21338.7	2	1
LS, swap, best fit	20370.7	3	20151.5	4	20084.5	3	21471.0	2	21342.9	2	6
reverse, LS, swap, first fit	20041.2	3	19109.6	5	20075.2	3	18819.0	5	18153.8	6	2
reverse, LS, swap, best fit	19637.6	4	18075.6	6	19249.1	4	18870.3	5	18254.5	6	8

Table 3.9: Computational results for different initial sequences and heuristic settings for data set II.

mindist	0		62		154		300		600		1000+		time (s)
heuristic	obj.	#	obj.	#	obj.	#	obj.	#	obj.	#	obj.	#	
LS, swap, first fit	21337.2	3	21416.4	0	21338.7	3	21298.1	1	21122.0	4	21096.0	5	1
LS, swap, best fit	21604.8	1	21387.2	1	21342.9	2	21364.2	3	21168.5	4	21158.4	4	5.5
LS, swap, shift, first fit	21191.1	1	21041.4	5	21267.3	1	21036.3	3	21078.5	1	21033.6	4	2
LS, swap, shift, best fit	21298.8	2	21235.9	2	21259.2	2	21204.9	0	21019.0	3	21008.9	4	17

Table 3.10: Computational results of the heuristic for data set II and different adjoin policies.

(first non stackable then stackable) in “stack” and by lexicographical combinations in “length,stack” and “stack,length”. We started with the following parameter setting. The minimum distance to other load units that allows to skip the adjoin policy is set to $mindist = 154$ dm. This value has been defined after discussions with practitioners. The motivation is that in a gap of 154 dm at least one of all load units fits. The parameter to restrict the size of the neighborhoods $maxdist$ is initially set to 3000 dm. We will test different values for these distance parameters below.
In the lines of Table 3.9 we listed the mean values for the scheduler without a further local search (“scheduler”) a local search with the neighborhood swap and first fit (“LS, swap, first fit”) and best fit (“LS, swap, best fit”). Furthermore, as an additional check we run all these settings with the reverse initial ordering of load units. This means we order the load units decreasingly by x-position or increasingly by length or first the stackable load units then the non stackable load units. In the lexicographical combinations also these reverse sortings are used. We used the area tree implementation of the list scheduler. The runtimes of the area tree implementation are listed for all sortings in the last column, as the mean runtimes over all instances did not deviate for the different sortings. The runtimes of the array and the area tree implementation are compared below (see Table 3.11).

By comparing the lines with the intended sorting and the “reverse” sorting one can see that the solution quality of the proposed sorting can almost always outperform the quality of the reverse sorting using the same type of heuristic. Only for the local search with first fit and sorting “pos” the reverse sorting is better. Here, we have to say that the sorting “pos” has no practical motivation and is just used as a neutral reference sorting. A better solution quality means, that either for more instances feasible solutions can be found or the objective values are less. The best results with only 2 instances without a feasible solution can be provided with the lexicographical sorting “stack,length”. One is the infeasible instance data17, the other is the instance data 15 which has a feasible solution. The difference between the local search strategies first and best fit are quite small for the mean values. Unfortunately, for instances that could also be solved with other sorting strategies different other sorting strategies could provide better objectives for single solutions (with a local search). For first fit and best fit in each case for 4 instances the best solution found by a heuristic could be found with the sorting “stack,length”. So, for 8 instances the best results are obtained with other sortings which do not provide the best mean results.

Analogously to the results with CPLEX, for the instances where load units have to be stored that are not stackable, it is harder to find a feasible solution with the heuristics. This is quite comprehensible, since load units that are not stackable reduce the number of load units that can be stored in a storage area, as they have to be stored at the ground level and the locations above them have to stay empty. To find feasible solutions for more instances with the list scheduler one could skip the cost for stacking load units or even introduce negative costs for stacked load units. With this adapted objective it is preferred to build stacks of load units which can help to reduce the utilization of the ground level to provide storage locations for more load units.
Note, that for the data sets $data1$ to $data8$ the sortings $length, stack$ and $stack, length$ lead to the same sequences since all load units are stackable. So, the differences in the

mean objective values are just caused by the instances with load units that are not stackable (data11 to data18).

Implementation	Pos	length	stack	length,stack	stack,length
arrays	574	680	662	649	505
area trees	6	6	6	6	6

Table 3.11: Runtimes of scheduler implementations for data set II.

In Table 3.11 we list the mean runtimes for a local search using the scheduler implementations with arrays and area trees with decision option best fit the only neighborhood swap and a maximum distance in between load units of 3000 dm for considering swapping the load units. The maximum runtimes were 2535 s for the array implementation and only 18 s for the area tree implementation. Also the mean runtimes in Table 3.11 differ strongly. While the mean runtimes for the array implementation are between 505 s and 680 s the same mean runtimes for the area tree implementation are around 6 s (plus or minus 0.5 s). So, the differences between the different initial sortings is relatively larger for the array implementation with the much larger runtimes. We think that the differences in the runtimes are caused by the less number of locations that have to be evaluated for the area tree implementation compared to the array implementation. While in the array implementation it has to be checked if a location is feasible and as a consequence a lot of infeasible locations are checked, in the area tree implementation just the feasible locations are considered. Furthermore, in the area tree implementation maximal four locations per empty area are considered. An additional cause may be that in the area tree implementation it can be checked with a single instruction if a location is feasible. For the array implementation to check whether a position for load unit i is feasible ℓ_i array values have to be checked. As the runtime difference is so broad we use the area tree implementation for all subsequent computations with the list scheduling heuristic.

In Table 3.10 we analyze the impact of different *mindist*-values for the adjoin policy. In the headline "*mindist*" the tested distances to the next load unit that has to be kept if a load unit is not stored with the adjoin policy are listed. For each distance we report the mean objective "obj." and the number of instances for which the best solution found by the heuristic could be found using this distance "#". In the last column the mean runtimes are listed. As these do not differ by more than one second for the different distances, we do not list the individual runtimes for the distances. We tested the adjoin distances $0, 62, 154, 300, 600, 1000, 2000, 300, 4000$ and 6000 dm. Here 0 has the meaning that the adjoin policy is switched off and values above 1000 dm indicate that all load units have to be stored adjoining another load unit. This is the case, as in none of the considered instances a lane exists with an empty space larger than 1000 dm. So, we listed the results for $0, 62, 154, 300$ and 600 dm and combined the results for greater distances in the column 1000+. We tested the different distances with a first and best fit local search and with two different neighborhoods. The first is just the swap neighborhood and the second is the neighborhood of all neighbors (swap, forward and backward shift). The objectives slightly differ for the different settings without a clear trend. The deviation between the maximum and minimum mean value is just 2.8%. For none of the settings for more than 5 instances the best solution provided by one of the settings can be found.

For some instances it is good to skip the adjoin policy completely. On the other hand, for many instances a hard adjoin policy (a great distance) leads to good results. As the runtimes are quite low, for the optimization in practice it may be worth to run a single instance with different distance values to obtain a good solution. Without clear trend, for further computations we use the distance suggested by the practitioners ($mindist = 154$ dm). In Table 3.10 we can also see the differences between the neighborhoods. The larger neighborhood with swap and shift operators leads to better results than only the swap neighborhood, but again the deviation is quite small. For the same *mindist*-value the maximum deviation is 1.7% for the best fit local search with distance 62 dm.

In the following, we study the influence of the *maxdist* which restricts the neighborhood to neighbors that effect load units that have a distance of less than *maxdist* in between their origin positions. In Table 3.12 the objective values for different heuristic settings and different values for *maxdist* are listed.

maxdist	500	1000	2000	3000	4000
LS, swap, first fit	21461.9	21350.5	21284.7	21338.7	21338.7
LS, swap, best fit	21570.4	21291.4	21342.9	21342.9	21342.9
LS, swap, shift, first fit	21231.4	21147.5	21272.0	21267.3	21267.3
LS, swap, shift, best fit	21336.5	21169.7	21259.2	21259.2	21259.2

Table 3.12: Objective values for different swap distances (in dm) of the heuristic with data set II.

The differences for the heuristics with different *maxdist*-values are quite small. The maximum deviation is 1.3% for the best fit local search with neighborhood swap. For this heuristic with $maxdist = 500$ dm the highest objective value occurs. The overall lowest objective value could be obtained for $maxdist = 1000$ dm with the local search with swap and shift neighborhood with the decision option first fit. For 3 of the 4 heuristics the best objective could be obtained with $maxdist = 1000$ dm and for one heuristic with $maxdist = 2000$ dm.

Note, that the deviation for different parameter settings is quite small. For further computations of the heuristic, if no other parameter settings are specified we used the initial sorting "stack,length", an adjoin distance of $mindist = 154$ dm and $maxdist = 1000$ dm. Due to the runtime we always use the area tree implementation. The influence of the *maxdist*-value is again tested for instances of data set I in the following section.

In Table 3.13 for the *mindist* and *maxdist* settings which performed best different neighborhoods and neighbor selection strategies are tested for data set II. The columns contain the obtained objectives in "obj." and the runtimes in seconds in "t." for different neighborhood settings. The best results could be obtained using the complete neighborhood (swap, forward- and backwardshift) and the neighbor decision strategy first fit. For a limited neighborhood where either only swapping or only shifting is considered, swapping performs better. For first fit the difference to the solution quality with complete neighborhood is 1.0% and 1.4% for best fit. For the shift neighborhood these values are slightly worse with 1.4% for first fit and 1.9% for best fit. For all neighborhoods first fit could slightly outperform best fit.

Instance	swap ,shift				swap				shift			
	first fit		best fit		first fit		best fit		first fit		best fit	
	obj.	t.	obj.	t.	obj.	t.	obj.	t.	obj.	t.	obj.	t.
data1	13873.5	1	13809.4	7	14095.8	0	13840.2	4	13889.7	1	14190.7	7
data2	15315.0	1	15341.6	8	15434.9	1	15841.0	2	15300.8	1	16160.5	4
data3	15013.7	1	14889.2	6	15098.4	1	14908.8	2	14972.1	1	14929.0	4
data4	13021.9	0	13329.0	8	12752.3	0	13187.4	4	13164.7	0	12942.4	6
data5	18388.3	1	18599.8	7	18720.8	1	18786.4	2	18991.3	1	18977.5	6
data6	15788.0	2	16207.3	10	15869.0	0	16055.0	3	16057.8	1	16788.1	3
data7	22683.4	1	22561.4	5	22601.0	0	22561.4	1	22798.0	1	22812.8	2
data8	14705.1	1	14846.3	8	14554.6	0	14706.3	3	14483.1	2	14841.8	6
data11	24066.5	4	24356.8	17	24437.0	1	24494.2	7	26050.7	2	25330.9	13
data12	29441.0	5	29291.2	24	29247.4	1	29323.8	9	29924.6	2	29642.2	16
data13	26727.7	3	25853.9	16	26720.3	1	25969.6	6	26860.9	2	26231.7	9
data14	23158.4	3	23109.3	32	24501.2	0	24774.0	7	23674.9	5	23707.1	18
data15	-	2	-	1	-	1	-	0	-	2	-	1
data16	34756.5	5	35198.4	21	34866.8	2	34756.7	7	35378.4	3	35998.1	17
data17	-	1	-	2	-	0	-	0	-	2	-	0
data18	29126.5	3	28982.0	27	30007.9	1	28875.2	11	28900.0	6	29391.8	17
mean	21147.5	2	21169.7	14	21350.5	0	21291.4	4	21460.5	2	21567.5	9

Table 3.13: Best fit and first fit results for different neighborhoods with the scheduling heuristic for data set II.

Also with the final heuristic setting for one instance less a feasible solution could be provided than with CPLEX. Compared to the bounds of the distance-IP reported by CPLEX after 3 h of runtime the gap of the instances that could be solved is 5.8% for first fit and all neighborhoods. Compared to the mean of the sum of the best objective values computed with CPLEX using the distance- or the coverage-IP with the time limit of 3 h the heuristic with the best setting could outperform CPLEX by 0.1% for the 14 instances for which a feasible solution could be provided with the heuristic. Compared to the same results with a 5 min runtime limit the heuristic can even provide feasible solutions for more instances. For the instances the coverage-IP could provide feasible solutions the best heuristic setting provides solutions with objectives that are about one third less than those of the integer solutions provided by CPLEX (with the coverage-IP). The runtimes of the heuristic are quite small with mean values of 2 s for first fit and 14 s for best fit, if all neighborhoods are used.

3.10.3 Comparison of the heuristic and the second MIP-model

In this section we compare results for the data sets I, III and IV computed with the x-pos MIP II and the heuristic with different neighborhoods. For data set I we also do an additional test of the performance with different values of *maxdist*. Here, we look at the objective and also on the runtime which can be less if a smaller neighborhood is considered due to a smaller *maxdist* value.

In Tables 3.14 and 3.15 computational results obtained with the MIP x-pos MIP II for the instances of data sets I, III and IV are presented. The tables show the best objective value, the best lower bound and the gap in % computed with CPLEX and a time limit of 3 h. The last line contains mean values. These mean values are in brackets, if not for all feasible instances of the data set feasible solutions could be found. Note, that we did not list instances data10 to data18 in Table 3.14. For none of these instances CPLEX could provide a feasible integer solution.

Instance	x-pos MIP II		
	objective	bound	gap
data1	38565.4	22310.8	42.2
data2	40103.2	23404.8	41.6
data3	40126.8	22588.1	43.7
data4	-	21866.7	-
data5	45154.1	23122.5	48.8
data6	43388.2	23245.2	46.4
data7	-	24917.9	-
data8	38604.2	21772.4	43.6
mean	(40990.0)	22903.6	(44.4)

Table 3.14: Computational results for data set I with the x-pos MIP II and time limit 3 h.

We do not report runtimes, as all computations were interrupted due to the time limit. So, unfortunately for none of the instances optimality could be proven and we do not have optimal solutions for the instances of these data sets. For 10 instances of the data set I with a terminal length of 700 m no feasible integer solution could be found by CPLEX. These are two instances with stackable load units and all 8 instances containing load units that are not stackable. Also, for data set III for two instances no feasible integer solution could be found by CPLEX.

Instance	data set III				data set IV		
	objective	bound	gap		objective	bound	gap
datakln1	-	11800.1	-		14472.2	6584.4	54.5
datakln2	64183.6	11428.7	82.2		15321.9	7822.0	49.0
dataHAB1	24996.6	10873.4	56.5		9989.1	6277.5	37.2
dataHAB2	28010.0	10574.0	62.3		10579.5	5607.2	47.0
datakln3	-	24147.2	-		25494.5	10270.0	59.7
datakln4	205483.0	18213.0	91.1		26703.3	11910.7	55.4
dataHAB3	4852.6	4420.8	8.9		17846.2	9689.2	45.7
dataHAB4	212845.0	17438.3	91.8		19971.9	8820.2	55.8
mean	(67546.3)	10593.5	(65.5)		17547.3	8372.6	50.5

Table 3.15: Computational results for data sets III and IV with the x-pos MIP II and time limit 3 h.

For the instances of data set I less instances can be solved than for data set II with the same number of storage lanes $m = 2$ but a shorter terminal length of $L = 250$ compared to $L = 700$ and also more load units for less instances integer solutions can be provided and the gaps are higher for instances with feasible solutions (44.4% compared to 33.5%). This is also the case for data sets III and IV with 5 storage lanes. Here, the instances of data set III have a terminal length of $L = 700$ and those of data set IV only have a terminal length of $L = 250$. Again, less feasible solutions can be provided and also the mean gaps are higher for the greater terminal length.

In Tables 3.16, 3.17 and 3.18 we report computational results for the heuristic with different settings for data sets I, III and IV. We report the obtained objectives in "obj." and the runtimes in seconds in "t.". In Table 3.16 again the influence of the maximum distance for load unit pairs to be considered for the neighborhoods is compared for two additional values of $maxdist$.

For the data sets I, III and IV the solution quality obtained with the scheduling heuristic could outperform the solution quality of CPLEX could obtain with the x-pos MIP II. For more instances feasible solutions can be provided and for the instances where CPLEX could compute a feasible integer solution, the solutions of the heuristic are better. Only for instance dataHAB3 of data set III the solution of CPLEX is slightly better than the solution obtained by the heuristics in the different settings. Let us look at the results for data set I obtained with the MIP-model x-pos MIP II (see Table 3.14) and with the scheduling heuristic using the shift and the swap neighbors as well as first fit for the decision of the neighbor to select (see second column in Table 3.16). While with CPLEX only for 6 instances feasible solution can be computed, the heuristic is able to compute feasible solutions for 17 of the 18 instances. For the 6 instances CPLEX provides a

Instance	swap ,shift				swap				shift			
	first fit		best fit		first fit		best fit		first fit		best fit	
	obj.	t.	obj.	t.	obj.	t.	obj.	t.	obj.	t.	obj.	t.
data1	34585.3	41	34608.1	236	34831.8	7	34404.5	76	34650.6	36	35003.1	160
data2	37009.1	56	37516.3	153	37266.7	6	37490.5	59	37121.7	48	37879.4	124
data3	37102.6	32	37110.5	137	37618.7	5	37166.3	53	36996.8	55	38239.1	74
data4	33837.6	37	34491.5	156	34617.2	7	34280.9	58	34362.7	32	35325.0	112
data5	42157.9	55	42252.1	144	42351.7	5	42531.3	53	42385.5	31	42571.4	107
data6	40904.6	35	40759.6	161	41424.0	7	40821.0	60	40748.7	53	40890.2	126
data7	54354.5	35	55683.1	112	54740.5	5	55901.8	37	54932.4	30	54930.4	116
data8	33723.4	35	33889.3	186	34191.7	6	34083.7	56	33625.0	75	34651.0	116
data11	58207.7	113	61383.4	523	59022.3	20	60465.5	213	58959.8	119	59784.0	309
data12	57940.1	141	58803.1	809	59012.3	23	60293.1	157	62216.6	102	59393.9	413
data13	56671.7	171	57049.9	734	57827.5	18	57908.4	207	58812.7	126	61000.5	323
data14	56644.4	82	57362.1	581	56830.8	15	58388.3	173	57076.3	106	57360.7	384
data15	84909.4	99	85094.6	477	82738.4	19	86887.4	176	84206.9	132	85353.4	392
data16	81359.3	96	79776.6	488	79912.9	15	79346.7	176	82335.3	167	79850.0	383
data17	-	39	-	29	-	5	-	17	-	34	-	19
data18	54855.4	172	55373.6	504	54847.5	18	55300.2	151	57352.7	128	55130.3	382
mean ($maxdist = 1000$)	50950.9	82	51410.3	362	51148.9	12	51684.6	114	51718.9	84	51824.1	236
mean $maxdist = 2000$	50990.7	84	51435.8	664	51019.3	23	51485.2	222	51718.9	84	51917.0	424
mean $maxdist = 3000$	51087.6	94	51418.3	883	50945.0	30	51254.3	309	51718.9	84	51762.8	586

Table 3.16: Best fit and first fit results for different neighborhoods with the scheduling heuristic for data set I with $maxdist = 1000$.

feasible solution the mean objective is 40990 for CPLEX and 37580.5 for the heuristic. So, the heuristic computes for 11 instances more feasible solutions and solutions that are 8.3% better for the 6 instances both methods obtained feasible solutions.

Concerning the decision on the next neighbor there is a huge difference between first fit and best fit. First fit is much faster. According to solution quality first fit is slightly better but the difference is not large. For data set I the best mean objective is obtained with the neighborhoods swap as well as the shift operators and first fit. For data set II the best results were obtained using the same neighborhood and also with first fit. Note, that the deviation for a reduced neighborhood or best fit are quite small. The results for all data sets show that the swap neighborhood leads to better results than the shift neighborhood alone. But as mentioned above with the combination of the two neighborhoods in most cases the best results could be obtained. Only for data set IV the best mean value could be computed using only the swap neighborhood.

According to the additional $maxdist$ values considered for instances of data set I the mean values in Table 3.16 show that with $maxdist$ values of 2000 and 3000 dm worse mean values are obtained than with $maxdist = 1000$ dm. Furthermore, the mean runtimes for instances of data set I show that the by smaller $maxdist$-values restricted neighborhoods lead to shorter runtimes. If, for example, the swap and shift neighborhood is considered with best fit, the runtime with $maxdist = 3000$ dm of 883 s can be reduced to less than the half (362 s) with $maxdist = 1000$ dm

At the end of Section 3.10.1 we present results for a reduced version of the instances of data set IV with only one train. We also tested the scheduling heuristic with these instances. For all settings (best fit or first fit combined with the neighborhoods "swap", "shift" and "swap,shift") 7 of the 8 instances the heuristic was able to find the optimal solution. The instance datakln1 could only be solved with a gap of 0.7% (by the heuristic in all settings). The runtime was for all settings and instances below one second.

3.10.4 Discussion of the results for the practical problem

For a practical application of our storage planning approaches we suggest to use our list scheduling heuristic with the neighborhoods swap and shift and the neighbor decision strategy first fit. With the heuristic in this setting we could obtain quite good results. Often the solutions of this heuristic are the best solutions obtained. Furthermore, the runtime is quite low. For the instances with a terminal length $L = 2500$ dm the maximum runtime is just 6 s and for $L = 7000$ dm the maximum computation time is below 3 min. If these runtimes are too large, the neighborhood can be reduced to swapping of load units. This leads to a slightly worse solution quality, but the runtimes can be reduced to maximum values of 4 s for $L = 2500$ dm and 28 s for $L = 7000$ dm.
Our aim was to provide solution methods for the storage planning in practice. For this purpose we think that the heuristic is a good optimization method. Solutions of high quality can be provided in a short runtime which is important as the planning is done during operations that change the input of the storage planning problem. For example, the setting in the storage is changed if a truck catches up or delivers a load unit.

Instance	swap ,shift				swap				shift			
	first fit		best fit		first fit		best fit		first fit		best fit	
	obj.	t.	obj.	t.	obj.	t.	obj.	t.	obj.	t.	obj.	t.
datakln1	33079.8	37	33168.6	183	33294	8	33274.6	59	33543.7	43	33608.6	103
datakln2	26335.4	49	26286.1	233	26171.4	8	26293.1	73	26357.7	43	26439.7	149
dataHAB1	16504.2	74	16533.2	283	16535.2	11	16500.3	98	16710.0	51	16806.6	148
dataHAB2	19448.0	50	19621.0	178	19448.3	6	19317.9	78	19643.8	42	19616.4	143
datakln3	53381.0	110	53256.8	683	52442.5	23	53809.3	207	55206.1	127	53515.5	421
datakln4	41315.8	149	41189.1	781	41012.5	28	41223.2	270	41594.7	162	41254.6	614
dataHAB3	4852.8	2	4852.7	5	4856.6	0	4852.8	2	4852.8	2	4852.8	3
dataHAB4	30176.3	159	30260.8	1200	30269.0	20	30255.3	334	30264.8	139	31171.4	585
mean	28136.6	78	28146	443	28003.7	13	28190.8	140	28521.7	76	28408.2	270

Table 3.17: Best fit and first fit results for different neighborhoods with the scheduling heuristic for data set III.

Instance	swap ,shift				swap				shift			
	first fit		best fit		first fit		best fit		first fit		best fit	
	obj.	t.	obj.	t.	obj.	t.	obj.	t.	obj.	t.	obj.	t.
datakln1	14241.3	1	14294.8	15	14423.8	1	14192.9	5	14271.4	2	14448.9	9
datakln2	15216.3	2	15534.4	13	15236.2	1	15534.4	4	15299.8	2	15699.0	7
dataHAB1	9577.1	3	9958.7	17	9636.5	1	10101.5	6	9837.8	2	9796.7	12
dataHAB2	10394.9	3	10265.9	14	10265.8	2	10265.9	4	10413.9	2	10266.0	12
datakln3	24019.7	8	24045.5	41	24185.7	2	24092.5	14	24233.0	5	23939.5	38
datakln4	24938.6	6	24863.6	53	24781.2	2	24921.3	21	25209.7	6	25544.6	32
dataHAB3	16841.0	8	16799.4	67	16524.4	4	16801.9	27	17413.5	6	16894.8	49
dataHAB4	18640.6	6	18353.1	48	18376.3	3	18353.1	17	18684.2	6	18511.3	39
mean	16733.7	4	16764.4	33	16678.7	2	16782.9	12	16920.4	3	16887.6	24

Table 3.18: Best fit and first fit results for different neighborhoods with the scheduling heuristic for data set IV.

Nevertheless, one may also aim to further improve the solution quality. As we could not find a parameter setting that clearly outperforms all other settings, we see three ways of using the proposed scheduling heuristic. The first idea is to try to find good parameter settings for each terminal or even for each module of a terminal as the load unit mix often differs in the modules. Note, that our practical instances were taken from different terminals and modules. So, we could not do this kind of calibration.
If the first idea does not work or is not preferred by terminal operators, one could run the heuristic with different settings and take the best solution computed. Typically, this can be done simultaneously on modern multiprocessor servers. Again it may be useful to select a set of parameter settings for each terminal.
So far our heuristic is just focused on improving a solution. The heuristic might provide better solutions, if we also integrate diversification phases in the heuristic. This could for example be done by meta-heuristics. The disadvantage of this approach is that also the runtime of the heuristic will increase.

Unfortunately, using zimpl to generate lp-files is not a promising approach for the storage planning problem as the generation of the lp-files takes a lot of time. For further tests it would be useful to call CPLEX directly from a C or JAVA program. Then the IP-models which performed better than the MIP-models might also be used for larger instances. Nevertheless, we think that our heuristic is a better alternative for the practical optimization as usual not perfect but good solutions are preferable for practitioners, if they can be computed very fast.

3.11 List of storage planning notations

common notations	
m	number of storage lanes
q	number of trains
L	length of storage lanes
b	maximum stacking height
n	number of load units
$\mathcal{C}$	set of load units that have to be stored
n^{fix}	number of load units that are already stored
$\mathcal{C}^{fix}$	set of load units that are already stored
c_{ip}	costs to transport load unit i to storage position p
ℓ_i	length of load unit i
$\hat{\ell}_i$	corner casting distance of load unit i
O_i	origin position of load unit i on the train
a_i	arrival time of load unit i
d_i	departure time of load unit i
s_{ij}	parameter for stacking restrictions where $s_{ij} = 1$, if i can be stacked on j
S	number of stacks in a fixed grid setting
$\mathcal{C}_k$	k-th set of arriving load units
$\mathcal{M}$	set of all load units ($\mathcal{C} \cup \mathcal{C}^{fix}$)
$\mathcal{X}$	set of x-positions

$\mathcal{Y}$	set of y-positions (lanes)
$\mathcal{Z}$	set of x-positions (levels)
d_{ik}	transport distance for load unit i to x-position k
$\overline{d_{il}}$	transport distance for load unit i to lane l
$\mathcal{P}_i$	location of load unit $i \in \mathcal{C}^{fix}$
$w_1, \dots, w_3$	weighting factors in the objective function
notations for the distance-IP	
$\mathcal{A}$	feasible locations for load units of the set $\mathcal{C}$
$\mathcal{B}_{il}$	feasible x-positions for load units $i \in \mathcal{C}$ in lane $l \in \mathcal{Y}$
$\mathcal{D}_i$	feasible x-y-positions for load units $i \in \mathcal{C}$
notations for the coverage-IP	
$\hat{\mathcal{A}}$	set of x-y-positions for the coverage constraint
$\overline{\mathcal{A}}_{ikl}$	set of x-y-positions that cover x-y-position (k, l) by load unit i
notations for the x-pos I MIP	
$\underline{\mathcal{M}}_i$	set of load units i may be stacked onto
$\dot{\mathcal{M}}_j$	set of load units that may be stacked onto j
notations for the x-pos II MIP	
θ_i	number of load unit aggregations blocking load unit i
$\pi^{i\vartheta}$	load unit aggregation ϑ blocking load unit i
$\sigma_{i\vartheta}$	begin of load unit aggregation ϑ for load unit i
$\omega_{i\vartheta}$	end of load unit aggregation ϑ for load unit i
$\rho_{i\vartheta}$	lane of load unit aggregation ϑ for load unit i
notations of the heuristic	
mindist	minimum distance to skip the adjoin policy
maxdist	maximum distance for load units to be considered in the neighborhoods

Table 3.19: Parameters of storage planning problems

4 Load planning

In this section the problem of load planning for trains in intermodal container terminals is studied. The objective is to assign load units to wagons of a train such that the utilization of the train is maximized, and setup and transportation costs in the terminal are minimized. In this section we assume that there is no uncertainty in any of the input parameters and face the deterministic setting of the load planning problem (see Section 5 for robust load planning). Most parts of this section can also be found in Bruns and Knust [27]. The additional contributions are mainly those on load planning models allowing weighted means of load unit weights in Section 4.3.3.

Contributions. After introducing the necessary technical details of the load planning problem we present different ways to model the length and weight restrictions in integer linear programming formulations. Contrary to previous approaches, additionally weight restrictions for the wagons are integrated into our models. For the weight restrictions we introduce two possibilities to model them: First, we show how to model the load patterns that are used in practice. Second, the underlying physical weight restrictions are modeled. For the first modeling approach that uses the load patterns of the practice we also model and analyze weighted means of the maximum load unit weights in different variants. Contrary to previous approaches, even non-commercial MIP-solvers can solve our models in a few seconds runtime.

Overview. This section is organized as follows. After introducing the load planning problem in Section 4.1 in more detail and giving a short literature review in Section 4.2, three integer linear programming formulations are stated in Section 4.3. Afterwards, some computational results are presented in Section 4.4. Finally, in Section 4.5 a list of notations can be found.

4.1 Problem definition

In this section we describe the load planning problem for trains in a rail-road-terminal in more detail. For the purpose of load planning also swap bodies are treated as containers because they differ from containers only in the possibility of stacking, which is not relevant here.

Given is an empty train consisting of m (approximately 20-30) wagons $j \in \mathcal{M} = \{1, \ldots, m\}$ where each wagon j belongs to a wagon type $c(j) \in \mathcal{W} = \{1, \ldots, p\}$. The train has a given weight limit G which may not be exceeded by the total weight of the assigned load units. The wagons can be used in different configurations which determine how many and which load units can be loaded onto slots of a wagon. For each wagon an initial configuration is given. If a wagon is changed to an other configuration, setup costs occur.

Furthermore, there are n (approximately 80) load units $i \in \mathcal{N} = \{1, \ldots, n\}$ with lengths ℓ_i and weights g_i. Further properties of the load units like widths and heights are not relevant and hence not considered here. We assume that all load units are placed in a storage area of the terminal. For each $i \in \mathcal{N}$ and each $j \in \mathcal{M}$ we denote by d_{ij}

the transportation costs for loading load unit i onto wagon j (mainly influenced by the distance between the storage place and the wagon).

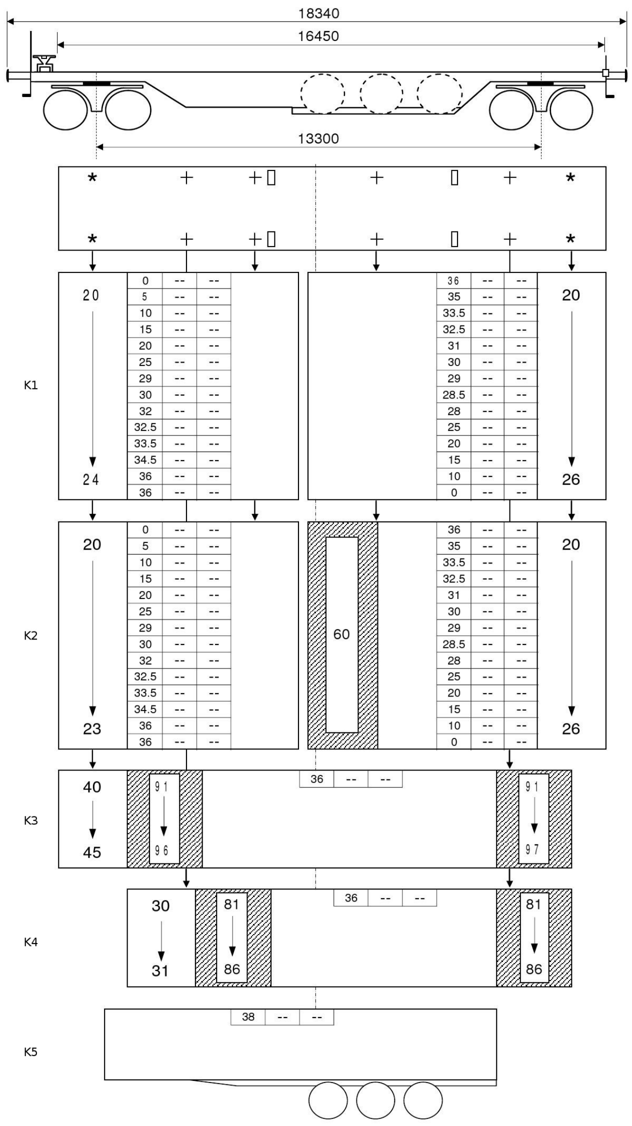


Figure 4.1: Examples for load patterns [79]

The loading of wagons is restricted with respect to the lengths and the weights of the load units. Figure 4.1 shows a visualization of feasible wagon loadings used by major rail companies like Schweizer Bundesbahnen (SBB) [79] or Deutsche Bahn (DB) [36]. At the top the side view of a wagon is shown, where the length of the wagon, the length of the loading space and the distance in between the bogie attachments are specified (a bogie is the turnable unit in which the axles are fixed). Dashed circles indicate the places where trailer wheels can be put. On the left side the loading space ends with a coupling plate used to fix trailers. The possible positions of the pins are shown in the top view (below the side view) with the symbols "+" and "*". One symbol ("+" or "*") stands for a pin at the front or rear side of the wagon. The boxes K1, K2, K3, K4 and K5 symbolize five different physical configurations for this wagon. A configuration defines the used pins (for containers) or wheel notches and a coupling plate (for trailers). The possible pin positions are also shown with vertical lines. If a configuration uses this pair of pins, an arrow ends at the corresponding configuration. A configuration provides a maximal number of slots (usually 1 to 4) on the wagon (for example, in Figure 4.1 configurations K1 and K2 provide two slots, K3, K4 and K5 only one). Each slot is symbolized by a box. The external areas of such a box contain the information which length-types of load units can be loaded up (cf. Table 2.1). While the external areas with white background contain symmetric load unit types, feasible asymmetric load unit types are listed in the shaded areas. All length-types within the given intervals are feasible (for example, in K1 length types 20-24 can be loaded onto the left slot, length types 20-26 onto the right slot).

Furthermore, in the slot boxes weight restrictions for the load units are encoded. In general the rows with weight distributions define maximum allowed payloads for the slots. For some weight distributions also minimum weights have to be respected (i.e. when a heavy container is assigned to one slot, for balancing reasons also some weight must be put on the other slot). It is allowed to build a weighted mean of weight distributions to create new ones. This means that it is allowed to create a new weight distributions as a convex combination of weight distributions. Different weight distributions can be combined and the sum of the weighting factors of these weight distributions has to add up to one. The maximum slot payloads of the new weight distribution are the sum of the products of the weighting factors and the maximum slot payloads of the original weight distributions. For example, for configuration "K1" and the weighting factor 0.5 for the first and the second weight distribution this results in a new weight distribution with maximum slot payloads of 2.5 tons for the first slot and 35.5 tons for the second slot.
The further columns with "- -" are not of interest for our studies. Nevertheless, we will explain the meaning of them and that of the three columns briefly. The first two columns correspond to the line categories "C" and "D" which restrict the maximum payload per axle to 20 and 22.5 tons. The third column defines the maximum slot payloads if the wagon is used in the "SS"-travel which permits a maximum travelling speed of up to 120 km/h. For the wagon in Figure 4.1 the last two columns are marked with "- -" as the wagon does not allow higher payloads for line category "D" and is not allowed to be used in the "SS"-travel mode at all. So, there are three different columns in the center of the boxes representing different weight restrictions caused by different line categories and travel modes (different maximum payloads per axle).

The three IP models we will present later on, differ in the modeling of the length and weight restrictions for the wagons. The first IP uses discrete weight distributions, furthermore exactly one load unit length-type fits onto a slot. The second IP uses the load unit fixation-type definition and also discrete weight distributions. With the fixation-type definition of load units slots might also be suitable for a group of load unit length types rather than for exactly one load unit length-type. Using the load unit fixation-types reduces the number of variables needed for the IP. We use discrete weight distributions in the first and second IP because nowadays they are commonly used in practice. The third IP uses a new continuous model of the weight restrictions which may offer more flexibility and leads to better computational results. According to the length restrictions the third IP is similar to the second IP model.

All our interpretations of the load restrictions according to the length of load units are based on the physical configurations of the wagons defining the use of pins for containers or the fixations for trailers. Let $\mathcal{K}_\tau$ be the set of all valid physical configurations for a wagon of type $\tau \in \mathcal{W}$. The first interpretation is based on so-called "type-weight lines". For each configuration $k \in \mathcal{K}_\tau$ a subset of possible type-weight lines $\mathcal{B}_k$ is given, where each element defines exactly one load unit length-type (see Section 2.2 for the definition) and a maximum payload for every slot. For example, configuration K1 in Figure 4.1 provides two slots with 5 different length-types (20-24) for the first slot and 7 length-types (20-26) for the second slot. Furthermore, there are 14 lines with weight distributions. Thus, in total we have $5 \cdot 7 \cdot 14 = 490$ type-weight lines for K1.

For the second interpretation we use the fixation-type definition of load units (see again Section 2.2 for the definition). In this case a configuration already defines a fixation-type for each slot and furthermore the valid overhangs above the pins and the corner castings. Therefore, only the decision on the weight distributions for the slots is open. For each configuration $k \in \mathcal{K}_\tau$ a subset of possible "weight distributions" $\mathcal{B}'_k$ is given, where each element defines maximum payloads for all slots. For example, for configuration K1 the 14 weight lines from above correspond to 14 feasible weight distributions.

In the third approach the configurations only take care of the length restrictions for the wagons. The weight restrictions are not explicitly given by the lines with weight distributions, but are taken into account by the underlying rules. Three weight conditions have to be respected for each wagon:

(W1) The payload per bogie is restricted according to the wagon bogies and the track the train runs on.

(W2) The payload on a bogie may not be larger than three times the payload on the other bogie (otherwise, the probability for wagons jumping the rails increases).

(W3) The payload for each slot on the wagon is limited due to stability reasons.

In order to check (W1) and (W2) for a specific situation where load units are loaded onto certain slots of a wagon, the payloads of the different bogies can easily be calculated by the lever principle (see Fischer et al. [43]). It can be assumed that the tare mass of the wagon distributes equally to the two bogies.

In Figure 4.2 a wagon with two bogies (A and B) and two load units (i_1 and i_2) is shown. The centers of mass of the load units are assumed to be in the middle of the load units (symbolized by arrows). In the figure three lengths are shown: the distance d in between the bogie attachments, and the levers e_1 and e_2 of the two load units in relation to bogie A. Let g_1 and g_2 be the weights of the two load units and W be the weight of the empty wagon. For this situation, the payload a of bogie A can be calculated as

$$a = \frac{d - e_1}{d} \cdot g_1 + \frac{d - e_2}{d} \cdot g_2 + \frac{W}{2}, \tag{4.1}$$

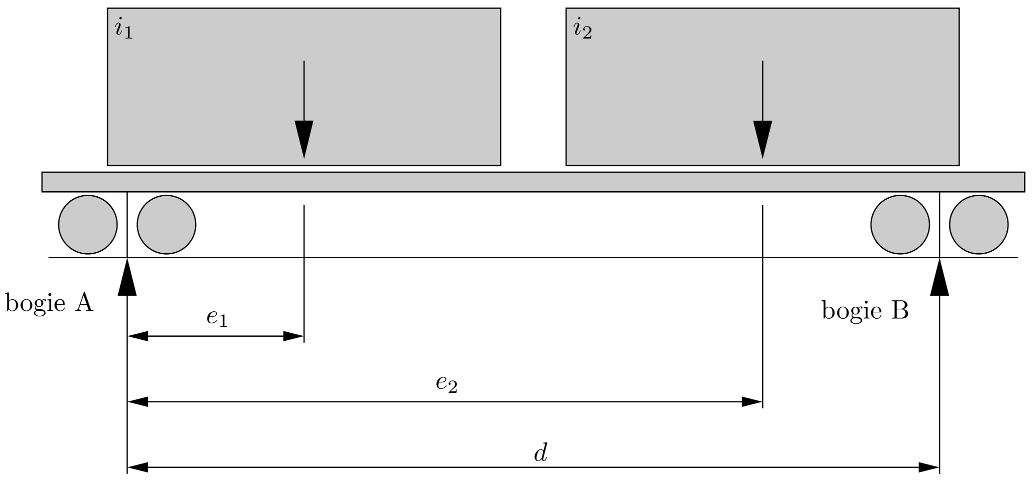


Figure 4.2: Lever principle for weight distributions of wagon bogies

symmetrically, for bogie B the resulting payload b is given by

$$b = \frac{e_1}{d} \cdot g_1 + \frac{e_2}{d} \cdot g_2 + \frac{W}{2}. \tag{4.2}$$

If more than two load units are loaded onto a wagon, for each additional load unit i an additive term $\frac{d-e_i}{d} \cdot g_i$ for bogie A and $\frac{e_i}{d} \cdot g_i$ for bogie B has to be taken into account.

When a and b are the calculated payloads for the bogies A and B of the wagon according to (4.1) and (4.2), and γ denotes the given maximum feasible payload for a bogie, the weight condition (W1) can be formulated as

$$a \leq \gamma \text{ and } b \leq \gamma. \tag{4.3}$$

The weight condition (W2) says that the payload of a bogie may not be larger than three times the payload of the other bogie. Thus, we must have

$$\frac{1}{3} \cdot a \leq b \leq 3 \cdot a. \tag{4.4}$$

Finally, we summarize all constraints for our problem and specify the objective function. A feasible solution is defined by the settings of all wagons (configurations or type-weight lines or weight distributions, depending on the used model) and an assignment of a subset of load units to the slots on the wagons such that

- for each wagon exactly one feasible setting is chosen,
- each load unit of the chosen subset is assigned to exactly one feasible slot on one wagon such that the length and weight restrictions for the slot are respected,
- at most one load unit is assigned to each slot,
- the sum of the weights of all assigned load units does not exceed the weight limit G for the total weight of the train.

In the objective function we consider three weighted elements:

- the utilization of the train (measured in total number, total length and total weight of assigned load units),
- the setup costs for changing configurations of wagons, and
- the transportation costs for the transport of load units within the terminal (from their storage place to the wagons).

While the utilization should be maximized, the two cost terms should be minimized.

Example 4.1. We consider a small instance with $n = 6$ load units belonging to 4 different length-types (and 3 fixation-types), $m = 3$ wagons belonging to $p = 2$ wagon types, and 2 slots for each wagon. We assume that the load units have length-types $(1, 1, 2, 3, 3, 4)$, fixation-types $(1, 1, 1, 2, 2, 3)$ and weights (in tons) are $(10, 24, 26, 36, 38, 35)$. The first and the second wagon are of type one, the third is of type two. Table 4.1 shows information on the possible load patterns where every row corresponds to a weight distribution $b \in \{1, \ldots, 7\}$. Each row contains the number b, the corresponding wagon type and the number of the associated configuration. Furthermore, for the two possible slots on each wagon the sets of feasible load unit length-types, feasible fixation-types, and the maximum payloads (in tons) are given.

b	wagon	configu-	slot 1			slot 2		
	type	ration	l.-types	f.-types	payloads	l.-types	f.-types	payloads
1	1	1	{1,2}	1	10	{1,2}	1	24
2	1	1	{1,2}	1	23	{1,2}	1	16
3	1	2	{3}	2	36	{}	-	0
4	1	3	{4}	3	38	{}	-	0
5	2	4	{1,2}	1	12	{1,2}	1	26
6	2	4	{1,2}	1	18	{1,2}	1	22
7	2	5	{3}	2	38	{}	-	0

Table 4.1: Information on the load patterns

For the type-weight line formulation each weight distribution b corresponds to different type-weight lines. For example, for the first row in Table 4.1 four type-weight lines exist modeling the different load unit length-type combinations for the two slots. Thus, we have four length-type combinations $(1, 1), (1, 2), (2, 1)$ and $(2, 2)$.

A feasible solution where the load units 1,2,4, and 5 are loaded, is presented in Table 4.2. For the three wagons the chosen weight distributions (and the associated configurations) as well as the assigned load units are listed. The solution respects all constraints for the load patterns: for each wagon a feasible weight distribution is chosen, the types of the assigned load units are compatible with it, and the weights of all load units are not larger than the allowed payloads for the corresponding slots.

Finally, we consider the weight restrictions (W1)-(W3) and show that the load of the first wagon is feasible with respect to these constraints. For the first configuration of a wagon

wagon	weight distribution	configuration	load units	
			slot 1	slot 2
1 (type 1)	1	1	1 (l.-type 1)	2 (l.-type 1)
2 (type 1)	3	2	4 (l.-type 3)	
3 (type 2)	7	5	5 (l.-type 3)	

Table 4.2: A feasible solution

of type one we assume $d = 11200$, $e_1 = 1500$, $e_2 = 8535$, $W = 16$ (recall Figure 4.2 for the meaning of these parameters). The allowed payload per bogie is assumed to be $\gamma = 40$ and the maximum payloads for the two slots are $\delta_1 = 26$ and $\delta_2 = 27$. For the first wagon of the solution shown in Table 4.2 the bogie loads are $a = \frac{d-e_1}{d} \cdot g_1 + \frac{d-e_2}{d} \cdot g_2 + \frac{W}{2} = 22.37$, and $b = \frac{e_1}{d} \cdot g_1 + \frac{e_2}{d} \cdot g_2 + \frac{W}{2} = 27.63$. The solution is feasible because

(W1) the bogie loads a, b are not greater than $\gamma = 40$,

(W2) $1/3 \cdot a = 7.46 \leq b = 27.63 \leq 3 \cdot a = 67.11$,

(W3) the weights $g_1 = 10, g_2 = 24$ of the assigned load units do not exceed the maximum allowed weights $\delta_1 = 26$ and $\delta_2 = 27$ for the slots.

□

4.2 Related literature

There are some papers dealing with simplified versions of the load planning problem without weight restrictions. Feo and Gonzáles-Velarde [42] treat a problem where trailers have to be assigned to wagons. Their models and solutions are based on the assumption that only two trailers together fit on one wagon, which limits adaptations of their methods to other situations. Powell and Carvalho [77] aim to optimize a tactical planning problem: the circulation of intermodal wagons. They determine the compilation of wagons for trains connecting terminals with different transport destinations which have certain demands. The types of the wagons satisfying a certain demand and load patterns for the wagons are determined, an assignment of load units to specific wagons is not calculated. The selection of load patterns is done heuristically, i.e., such a selection might not be optimal. The problem is modeled as a logistic queueing network. Corry and Kozan [34] optimize the load planning with respect to the handling time and the weight distribution within a train. Only one type of containers and no weight restrictions for the wagons are modeled, i.e., each container fits to each wagon. The problem is formulated as an integer linear program and is solved with CPLEX. In a subsequent paper Corry and Kozan [35] aimed at minimizing the train length and the total handling time. Furthermore, more container types are modeled. Load pattern restrictions are considered for the length of load units, but neither weight restrictions for the wagons nor for the whole train are integrated. The model is formulated (but not solved) as an integer linear program, real-world problem instances are tackled with local search.

For a north American setting different papers treat the load planning of trains with the objective of minimizing the energy consumption during the train run caused by gaps of empty wagons (see Lai and Barkan [64] or Lai et al. [65, 66]).

For rail interfaces of maritime terminals the load planning is done in Ambrosino et al.[6]. They present two MIP formulations, one with detailed practical constraints that could not be solved to optimality and another formulation with simplified constraints as well as an heuristic approach. For the assumed time limit of 10 minutes the heuristic worked best on randomly generated test data.

Two additional papers concerning operative planning in train-train terminals treat problems that are close to the load planning problem. In the work of Bostel and Dejax [22] the positioning of containers on incoming and outgoing trains is optimized with the aim of reducing transport distances of containers that have to be handled by cranes. The authors consider only one container type. Besides proposing different models, optimal and heuristic methods are presented. In Souffriau et al. [80] the operative planning in a train-hub is optimized with the objective of minimizing the makespan. The destinations for a group of trains are determined, containers are assigned to wagons and crane operating sequences are calculated. The assignment of containers to wagons is modeled with few constraints as IP and solved with CPLEX. Only three different container types as well as length restrictions for the wagons are considered.
For a terminal in Port Bou at the Spanish-French Border González et al. [52] model the transshipment planning. In this terminal all load units of a train have to be transferred to another train as Spain and France operate with different rail track gauges. The authors solve their MIP with AIMMS [1] but do not provide any computational data.

4.3 Integer linear programming models

In this section we present three integer linear programming formulations for the load planning problem with weight restrictions. These formulations use the three different interpretations of load patterns introduced before. So, the formulations differ in the modeling of the length and weight restrictions of the wagons.

4.3.1 First integer linear program

The first integer linear programming formulation is based on the length-type container categorization (i.e. we have 23 different container types and one type for trailers as described in Section 2.2) and type-weight lines. In this model, the configurations of the wagons are determined implicitly because each type-weight line belongs to exactly one configuration. The configurations are mainly used to model the setup costs for the changes of wagons, but do not exactly determine if a load unit fits onto a certain slot. Therefore, type and weight restrictions have to be taken into account separately by additional constraints.

At first we introduce some additional parameters:

- $\mathcal{S}_j$ denotes the set of all possible slots on wagon j.
- The binary coefficients $\kappa^0_{jk} \in \{0,1\}$ for wagons $j \in \mathcal{M}$ and all configurations $k \in \mathcal{K}_{c(j)}$ for the corresponding wagon type $c(j)$ are equal to 1, if wagon j is initially used in configuration k.
- The set of different load unit length-types is denoted by $\mathcal{L}$, all load units of the same type are grouped into sets $\mathcal{N}_t$ $(t \in \mathcal{L})$.
- For each wagon j the binary coefficients λ_{tbs} encode if length-type $t \in \mathcal{L}$ fits onto slot $s \in \mathcal{S}_j$ when the type-weight line $b \in \bigcup\limits_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k$ is used for j:

$$\lambda_{tbs} = \begin{cases} 1, & \text{if in type-weight line } b \text{ length-type } t \text{ fits onto slot } s \\ 0, & \text{otherwise.} \end{cases}$$

- The coefficient $\gamma_{bs} \in \mathbb{R}$ defines the maximum payload for slot $s \in \mathcal{S}_j$ of wagon j used in type-weight line $b \in \bigcup\limits_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k$.

We introduce the following decision variables in order to choose type-weight lines for all wagons and to assign load units to slots on the wagons:

- $y_{jb} \in \{0,1\}$ for $j \in \mathcal{M};\ b \in \bigcup\limits_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k$ with

$$y_{jb} = \begin{cases} 1, & \text{if type-weight line } b \text{ is chosen for wagon } j \\ 0, & \text{otherwise.} \end{cases}$$

- $x_{ijs} \in \{0,1\}$ for $i \in \mathcal{N};\ j \in \mathcal{M};\ s \in \mathcal{S}_j$ with

$$x_{ijs} = \begin{cases} 1, & \text{if load unit } i \text{ is assigned to slot } s \text{ on wagon } j \\ 0, & \text{otherwise.} \end{cases}$$

Note, that by choosing a type-weight line implicitly a configuration is determined as each type-weight line belongs to exactly one configuration. With these decision variables the first integer linear programming model reads

$$\max \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} (w_1 + w_2 \cdot \ell_i + w_3 \cdot g_i) \cdot x_{ijs} \tag{4.5}$$

$$- w_4 \, (m - \sum_{j \in \mathcal{M}} \sum_{k \in \mathcal{K}_{c(j)}} \kappa^0_{jk} \sum_{b \in \mathcal{B}_k} y_{jb}) \tag{4.6}$$

$$- w_5 \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} d_{ij} \cdot x_{ijs} \tag{4.7}$$

s.t.

$$\sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} x_{ijs} \leq 1 \qquad (i \in \mathcal{N}) \tag{4.8}$$

$$\sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k} y_{jb} = 1 \qquad (j \in \mathcal{M}) \tag{4.9}$$

$$\sum_{i \in \mathcal{N}_t} x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k} \lambda_{bts} \cdot y_{jb} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j; t \in \mathcal{L}) \tag{4.10}$$

$$\sum_{i \in \mathcal{N}} g_i \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k} \gamma_{bs} \cdot y_{jb} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j) \tag{4.11}$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} g_i \cdot x_{ijs} \leq G \tag{4.12}$$

$$x_{ijs} \in \{0,1\} \qquad (i \in \mathcal{N}; j \in \mathcal{M}; s \in \mathcal{S}_j) \tag{4.13}$$

$$y_{jb} \in \{0,1\} \qquad (j \in \mathcal{M}; b \in \bigcup_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k) \tag{4.14}$$

The parameters $w_1, \ldots, w_5 \geq 0$ in the objective function are coefficients weighting the different components. While w_1, w_2, w_3 weight the total number, total length and total weight of assigned load units in (4.5), w_4 and w_5 are factors for the setup and transportation costs in (4.6) and (4.7), respectively. The setup costs for changing a configuration are assumed to be constant, i.e. in (4.6) the total number of changed configurations is minimized.

Constraints (4.8) ensure that each load unit is assigned to at most one slot, (4.9) guarantees that for each wagon exactly one associated type-weight line is chosen. Constraints (4.10) and (4.11) take into account that the length-type and the weight of a load unit fit to the assigned slot. Finally, the total weight limit of the train is modeled by (4.12).

4.3.2 Second integer linear program

In the following we describe a second integer linear programming formulation for the load planning problem, which uses the fixation-types categorization of load units and the weight distributions. This formulation has the advantage that the numbers of variables and constraints are much smaller. To describe the attributes of a load unit, in addition to the fixation-type the overhangs to both sides must be known. With this definition of load unit types a configuration determines a feasible load unit fixation-type for each slot and maximum allowed overhangs to both sides. Since cranes used in typical European train terminals can easily turn load units, we do not treat the orientation of load units. Therefore, we consider the smaller and the larger overhang instead of the right and the left. Note that overhangs are irrelevant for trailers.

To model the restrictions of the configurations and the weight distributions we introduce the following parameters:

- The set of different load unit fixation-types is denoted by $\mathcal{T}$, all load units of the same type are grouped into sets $\mathcal{N}_t$ $(t \in \mathcal{T})$.

- For a wagon j in configuration $k \in \mathcal{K}_{c(j)}$ the binary coefficients α_{tks} encode if fixation-type $t \in \mathcal{T}$ fits onto slot $s \in \mathcal{S}_j$:

$$\alpha_{tks} = \begin{cases} 1, & \text{if in configuration } k \text{ fixation-type } t \text{ fits onto slot } s \\ 0, & \text{otherwise.} \end{cases}$$

- For each load unit $i \in \mathcal{N}$ the coefficients $u_i^+, u_i^- \in \mathbb{R}$ denote the larger and the smaller overhang.

- For a wagon j in configuration $k \in \mathcal{K}_{c(j)}$ the coefficients $\beta_{ks}^+, \beta_{ks}^- \in \mathbb{R}$ denote the larger and smaller maximum allowed overhang of a load unit on slot $s \in \mathcal{S}_j$.

- For each wagon j the coefficients $\gamma_{bs} \in \mathbb{R}$ encode the maximum feasible payload on slot $s \in \mathcal{S}_j$ for weight distribution $b \in \bigcup\limits_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k'$.

Using the decision variables $x_{ijs} \in \{0,1\}$ as before and variables

$$y_{jb} = \begin{cases} 1, & \text{if weight distribution } b \text{ is chosen for wagon } j \\ 0, & \text{otherwise} \end{cases}$$

for $j \in \mathcal{M}$; $b \in \bigcup\limits_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k'$, we get

$$\max \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} (w_1 + w_2 \cdot \ell_i + w_3 \cdot g_i) \cdot x_{ijs} \tag{4.15}$$

$$- w_4 \, (m - \sum_{j \in \mathcal{M}} \sum_{k \in \mathcal{K}_{c(j)}} \kappa_{jk}^0 \sum_{b \in \mathcal{B}_k'} y_{jb}) \tag{4.16}$$

$$- w_5 \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} d_{ij} \cdot x_{ijs} \tag{4.17}$$

s.t.

$$\sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} x_{ijs} \leq 1 \qquad (i \in \mathcal{N}) \tag{4.18}$$

$$\sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k'} y_{jb} = 1 \qquad (j \in \mathcal{M}) \tag{4.19}$$

$$\sum_{i \in \mathcal{N}_t} x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k'} \alpha_{tks} \cdot y_{jb} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j; t \in \mathcal{T}) \tag{4.20}$$

$$\sum_{i \in \mathcal{N}} u_i^+ \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k'} \beta_{ks}^+ \cdot y_{jb} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j) \tag{4.21}$$

$$\sum_{i \in \mathcal{N}} u_i^- \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k'} \beta_{ks}^- \cdot y_{jb} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j) \tag{4.22}$$

$$\sum_{i \in \mathcal{N}} g_i \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k'} \gamma_{bs} \cdot y_{jb} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j) \tag{4.23}$$

$$\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\sum_{s\in\mathcal{S}_j} g_i \cdot x_{ijs} \le G \tag{4.24}$$

$$x_{ijs} \in \{0,1\} \qquad (i\in\mathcal{N}; j\in\mathcal{M}; s\in\mathcal{S}_j) \tag{4.25}$$

$$y_{jb} \in \{0,1\} \qquad (j\in\mathcal{M}; b\in \bigcup_{k\in\mathcal{K}_{c(j)}} \mathcal{B}'_k) \tag{4.26}$$

The objective function (4.15)-(4.17) remains the same as before. Constraints (4.18), (4.19) and (4.24) are similar to constraints (4.8), (4.9) and (4.12) of the first integer linear program. In constraints (4.23) the weight distributions play the role of the type-weight lines from constraints (4.11). The new constraints model the restrictions of the configurations: (4.20) ensures that the fixation-types of all assigned load units are feasible, while constraints (4.21) and (4.22) represent the limitations of the overhangs.

The first and the second integer linear program have the same set of feasible solutions and hence also the same optimal solutions. But, as mentioned before, the numbers of type-weight lines and weight distributions differ, i.e., the numbers of variables and constraints in the two IPs are different. For typical load planning instances in practice the number of type-weight lines is about 10-30 times the number of weight distributions.

The smaller number of wagon settings (weight distributions) leads to less decision variables in the second IP compared to the first IP using the type-weight lines. Furthermore, the usage of the 4 fixation-types causes less constraints in comparison to the 24 length-types. Thus, also the number of constraints in the second IP is much smaller.

4.3.3 Second integer linear program allowing weighted means

In this section we will introduce two variants of the second integer linear program, that allow weighted means of weight distributions (which is feasible in practice as described in Section 4.1). The first variant directly models the weighted means of two adjacent weight distributions. The second approach allows to build the weighted means of all weight distributions of one configuration.

The idea of the first variant of the second integer linear program with weighted means is the following: It is allowed to chose two adjacent weight distributions and decide in a second step which weighted mean of these distributions is selected. We will call it weighted means of adjacent weight distributions ($\text{WM}_{\text{adjacent}}$).

The decision variables y_{jb} are still binary and now define two weight distributions that can be combined (as convex combination). If y_{jb} is one, this means that the weight distributions b and $b+1$ are used. To ensure that the weight distribution $b+1$ of the same configuration exists, one needs another set $\mathcal{B}^-_k$ that contains all weight distributions belonging to a configuration with more than one weight distribution except the last weight distribution of each configuration. The set $\mathcal{B}^1_k$ contains all weight distributions of configurations with exactly one weight distribution. Note that $\mathcal{B}'_k$ is not equal to $\mathcal{B}^-_k \cup \mathcal{B}^1_k$ as in the latter set the last weight distribution of each configuration with more than one weight distribution is not contained. y_{jb} is defined as follows:

$$y_{jb} = \begin{cases} 1, & \text{if } b \in \bigcup\limits_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k^- \text{ and weight distributions } b \text{ and } b+1 \text{ are chosen for} \\ & \text{wagon } j, \\ & \text{or if } b \in \bigcup\limits_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k^1 \text{ and weight distributions } b \text{ is chosen for wagon } j \\ 0, & \text{otherwise} \end{cases}$$

for $j \in \mathcal{M};\ b \in \bigcup\limits_{k \in \mathcal{K}_{c(j)}} \left(\mathcal{B}_k^- \cup \mathcal{B}_k^1\right)$. The new decision variables $z_{jb} \in [0,1]$ define the ratio between the weight distributions b and $b+1$ if a configuration with more than one weight distribution is implicitly chosen by y_{jb} (which is the case if $\sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k^-} y_{jb} = 1$): z_{jb} are the weight factors for weight distribution b for $j \in \mathcal{M};\ b \in \bigcup\limits_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k'$. Each weight factor z_{jb} is between 0 and 1. The relevant factors z_{jb} and $z_{j,b+1}$ for a wagon j where $y_{jb} = 1$ have to add up to exactly one, because otherwise the resulting weight distribution is not a convex combination. If a configuration with just one weight distribution is again implicitly chosen, z_{jb} is one for the chosen weight distribution. The length restrictions are modeled using the configuration of the weight distribution defined by y_{jb} (see conditions (4.35), (4.36) and (4.37)). The actual weight limit is calculated using the z_{jb}-variables (see (4.38)).

The resulting MIP is the following:

$$\max \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} (w_1 + w_2 \cdot \ell_i + w_3 \cdot g_i) \cdot x_{ijs} \tag{4.27}$$

$$- w_4 \,(m - \sum_{j \in \mathcal{M}} \sum_{k \in \mathcal{K}_{c(j)}} \kappa_{jk}^0 \sum_{b \in \mathcal{B}_k'} y_{jb}) \tag{4.28}$$

$$- w_5 \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} d_{ij} \cdot x_{ijs} \tag{4.29}$$

s.t.

$$\sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} x_{ijs} \leq 1 \qquad (i \in \mathcal{N}) \tag{4.30}$$

$$\sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k^-} y_{jb} = 1 \qquad (j \in \mathcal{M}) \tag{4.31}$$

$$\sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k'} z_{jb} = 1 \qquad (j \in \mathcal{M}) \tag{4.32}$$

$$z_{jb} + z_{j,b+1} \geq y_{jb} \qquad (j \in \mathcal{M}; b \in \bigcup_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k^-) \tag{4.33}$$

$$z_{jb} = y_{jb} \qquad (j \in \mathcal{M}; b \in \bigcup_{k \in \mathcal{K}_{c(j)}} \mathcal{B}_k^1) \tag{4.34}$$

$$\sum_{i \in \mathcal{N}_t} x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}_k'} \alpha_{tks} \cdot y_{jb} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j; t \in \mathcal{T}) \tag{4.35}$$

$$\sum_{i\in\mathcal{N}} u_i^+ \cdot x_{ijs} - \sum_{k\in\mathcal{K}_{c(j)}} \sum_{b\in\mathcal{B}'_k} \beta_{ks}^+ \cdot y_{jb} \leq 0 \qquad (j\in\mathcal{M}; s\in\mathcal{S}_j) \tag{4.36}$$

$$\sum_{i\in\mathcal{N}} u_i^- \cdot x_{ijs} - \sum_{k\in\mathcal{K}_{c(j)}} \sum_{b\in\mathcal{B}'_k} \beta_{ks}^- \cdot y_{jb} \leq 0 \qquad (j\in\mathcal{M}; s\in\mathcal{S}_j) \tag{4.37}$$

$$\sum_{i\in\mathcal{N}} g_i \cdot x_{ijs} - \sum_{k\in\mathcal{K}_{c(j)}} \sum_{b\in\mathcal{B}'_k} \gamma_{bs} \cdot z_{jb} \leq 0 \qquad (j\in\mathcal{M}; s\in\mathcal{S}_j) \tag{4.38}$$

$$\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{S}_j} g_i \cdot x_{ijs} \leq G \tag{4.39}$$

$$x_{ijs} \in \{0,1\} \qquad (i\in\mathcal{N}; j\in\mathcal{M}; s\in\mathcal{S}_j) \tag{4.40}$$

$$y_{jb} \in \{0,1\} \qquad (j\in\mathcal{M}; b\in \bigcup_{k\in\mathcal{K}_{c(j)}} (\mathcal{B}_k^- \cup \mathcal{B}_k^1)) \tag{4.41}$$

$$z_{jb} \in [0,1] \qquad (j\in\mathcal{M}; b\in \bigcup_{k\in\mathcal{K}_{c(j)}} \mathcal{B}'_k) \tag{4.42}$$

The objective function (4.27)-(4.29) and constraints (4.30), (4.35), (4.36), (4.37) and (4.39) remain the same as before. In constraints (4.31) $\mathcal{B}'_k$ is replaced by $\mathcal{B}_k^-$ in the sum. The new constraints (4.32) ensure that for each wagon j the z_{jb}-variables sum up to one. Constraints (4.33) are also new and force z_{jb} and $z_{j,b+1}$ to add up to at least one if y_{jb} is one. So, constraints (4.32) and (4.33) together ensure that z_{jb} and $z_{j,b+1}$ add up to one iff y_{jb} is chosen to one. As the sum of z_{jb} and $z_{j,b+1}$ does not have to be zero if y_{jb} is zero, it is not possible to force equality in constraints (4.33). This is the case as z_{jb} could differ from zero, if $y_{j,b-1}$ is one. The same holds for $z_{j,b+1}$ as it could differ from zero, if $y_{j,b+1}$ is one. Assume a wagon j with one configuration with 3 weight distributions $(1, 2, 3)$ and that we want to choose a weighted mean of the weight distributions 1 and 2. In that case we would have to handle with the decision variables y_{j1}, y_{j2}, z_{j1}, z_{j2} and z_{j3}. For the y-variables we have to choose $y_{j1} = 1$ and $y_{j2} = 0$ as we want to build a weighted mean of the first and the second weight distribution but not of the second and the third. The latter decision implies (for forced equality in constraints (4.33)) that z_{j2} is zero, but this makes it impossible to choose a weighted mean of weight distributions 1 and 2 as the weighting factor for weight distribution 2 is zero. Analogously to constraints (4.33) constraints (4.34) ensure the correspondence of y_{jb} and z_{jb} for weight distributions of configurations with only one weight distribution. In constraints (4.38) the factor for γ_{bs} changes from y_{jb} to z_{jb}. Note that now up to two elements of the sum can be larger than zero. The domains of the variables are defined in (4.40) to (4.42).

The new MIP (WM$_{\text{adjacent}}$) defines a feasible region of the solution space that contains the feasible solutions of the second IP as each weight distribution of the second IP can be expressed as a weighted mean in the WM$_{\text{adjacent}}$-MIP and all other constraints stay the same. We have to consider three cases of weight distributions $b \in \bigcup_{j\in\mathcal{M};k\in\mathcal{K}_{c(j)}} \mathcal{B}'_k$: the weight distributions of configurations with just one weight distribution given by $b \in \bigcup_{j\in\mathcal{M};k\in\mathcal{K}_{c(j)}} \mathcal{B}_k^1$, the weight distributions of configurations with several weight distributions expect the last of those weight distributions given by $b \in \bigcup_{j\in\mathcal{M};k\in\mathcal{K}_{c(j)}} \mathcal{B}_k^-$ and the last weight distributions of configurations with several weight distributions given by $b \in \bigcup_{j\in\mathcal{M};k\in\mathcal{K}_{c(j)}} \left(\mathcal{B}'_k \setminus (\mathcal{B}_k^1 \cup \mathcal{B}_k^-)\right)$.

- Weight distributions $b \in \mathcal{B}^1_k$ can be expressed in the $\mathrm{WM}_{\mathrm{adjacent}}$-MIP by choosing y_{jb} to one which implies that also z_{jb} has to be chosen to one for a specific wagon j.
- All weight distributions $b \in \mathcal{B}^-_k$ can be expressed by choosing y_{jb} and z_{jb} to one and $z_{j,b+1}$ to zero (for a considered wagon j).
- For those weight distributions $b \in \bigcup_{j\in\mathcal{M};k\in\mathcal{K}_{c(j)}} \left(\mathcal{B}'_k \setminus (\mathcal{B}^1_k \cup \mathcal{B}^-_k)\right)$ that are the last of a configuration k with several weight distributions one can choose y_{jb} to one for the b that represents the last element of $\mathcal{B}^-_k$ (again for a wagon j). In addition the relevant z-variables have to be chosen to $z_{jb} = 0$ and $z_{j,b+1} = 1$.

As we showed that all three cases of weight distributions of the second IP can be modeled in the $\mathrm{WM}_{\mathrm{adjacent}}$-MIP, we showed that all feasible solutions for the second IP are also feasible for the $\mathrm{WM}_{\mathrm{adjacent}}$-MIP. The set of feasible solutions may also contain additional solutions that are infeasible for the second IP but feasible when a weighted mean of weight distributions is used.

The $\mathrm{WM}_{\mathrm{adjacent}}$-MIP has more variables and constraints than the second IP but as the weight constraints of the wagons are modeled in a "semi"-continuous way (binary decision on two weight distributions, continuous decision on the weighting factors), it is not clear whether the runtimes will be better or worse than those of the second IP.

To model the weighted means, another variant is to allow the weighted means of all weight distributions of a configuration (short $\mathrm{WM}_{\mathrm{all}}$). This is less restrictive than $\mathrm{WM}_{\mathrm{adjacent}}$ and can be modeled easier as MIP.

For the wagon settings we use the following decision variables:

- $y_{jk} \in \{0,1\}$ for $j \in \mathcal{M};\ k \in \mathcal{K}_{c(j)}$ with

$$y_{jk} = \begin{cases} 1, & \text{if configuration } k \text{ is chosen for wagon } j \\ 0, & \text{otherwise} \end{cases}$$

- $z_{jb} \in [0,1]$ for $j \in \mathcal{M};\ b \in \bigcup\limits_{k\in\mathcal{K}_{c(j)}} \mathcal{B}'_k$ is the weight factor for weight distribution b.

The length restrictions are modeled using the configuration defined by y_{jk} (see conditions (4.49), (4.50) and (4.51)). The actual weight limit is calculated using the z_{jb}-variables (see (4.52)).

The resulting MIP is the following:

$$\max \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{S}_j} (w_1 + w_2 \cdot \ell_i + w_3 \cdot g_i) \cdot x_{ijs} \tag{4.43}$$

$$- w_4 \, (m - \sum_{j\in\mathcal{M}} \sum_{k\in\mathcal{K}_{c(j)}} \kappa^0_{jk} y_{jk}) \tag{4.44}$$

$$- w_5 \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} d_{ij} \cdot x_{ijs} \tag{4.45}$$

s.t.

$$\sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} x_{ijs} \leq 1 \qquad (i \in \mathcal{N}) \tag{4.46}$$

$$\sum_{k \in \mathcal{K}_{c(j)}} y_{jk} = 1 \qquad (j \in \mathcal{M}) \tag{4.47}$$

$$\sum_{b \in \mathcal{B}'_k} z_{jb} = y_{jk} \qquad (j \in \mathcal{M}; k \in \mathcal{K}_{c(j)}) \tag{4.48}$$

$$\sum_{i \in \mathcal{N}_t} x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \alpha_{tks} \cdot y_{jk} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j; t \in \mathcal{T}) \tag{4.49}$$

$$\sum_{i \in \mathcal{N}} u_i^+ \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks}^+ \cdot y_{jk} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j) \tag{4.50}$$

$$\sum_{i \in \mathcal{N}} u_i^- \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks}^- \cdot y_{jk} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j) \tag{4.51}$$

$$\sum_{i \in \mathcal{N}} g_i \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \sum_{b \in \mathcal{B}'_k} \gamma_{bs} \cdot z_{jb} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{S}_j) \tag{4.52}$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{S}_j} g_i \cdot x_{ijs} \leq G \tag{4.53}$$

$$x_{ijs} \in \{0,1\} \qquad (i \in \mathcal{N}; j \in \mathcal{M}; s \in \mathcal{S}_j) \tag{4.54}$$

$$y_{jk} \in \{0,1\} \qquad (j \in \mathcal{M}; k \in \mathcal{K}_{c(j)}) \tag{4.55}$$

$$z_{jb} \in [0,1] \qquad (j \in \mathcal{M}; b \in \bigcup_{k \in \mathcal{K}_{c(j)}} \mathcal{B}'_k) \tag{4.56}$$

The objective function (4.43)-(4.45) and constraints (4.46), and (4.53) remain the same as before. In constraints (4.49), (4.50) and (4.51) that model the length restrictions, y_{jb} is replaced by y_{jk}. This is also the case for constraints (4.47). Constraints (4.48) are new and ensure that weight factors z_{jb} for the weight distributions belonging to the configuration chosen by y_{jk} will add up to one. Furthermore, constraints (4.48) ensure that the weight factors z_{jb} for a weight distribution b belonging to a configuration that is not chosen will be zero. In constraints (4.52) the factor before γ_{bs} is again z_{jb}. The domains of the variables are defined in (4.54) to (4.56).

The feasible region of WM$_{\text{all}}$ may even contain more solutions than the one of WM$_{\text{adjacent}}$. Especially, all feasible solutions of the second IP are still feasible. WM$_{\text{adjacent}}$ uses the same set of weight distributions $\mathcal{B}'_k$ as the second IP and as weighted mean especially each individual weight distribution can be chosen. Furthermore, also weighted means of adjacent weight distributions can be expressed in the model WM$_{\text{all}}$. So all feasible solutions of the second IP and the WM$_{\text{adjacent}}$-MIP are feasible for the WM$_{\text{all}}$-MIP.

4.3.4 Third integer linear program

In the following we describe a third IP formulation which models the weight conditions by formulas instead of using a fixed finite number of feasible weight distributions for the wagons as in the previous two models. In this model configurations and fixation-types are used to represent length restrictions for load units on the wagons.

As described in Section 4.1, there are three weight restrictions (W1),(W2),(W3) for wagons. Since for the weight restrictions (W1) and (W2) the bogie payloads have to be calculated, the exact positions of the slots on each wagon have to be known to estimate the levers of the slots (and the assigned load units, respectively). For the first and the second IP in the set $\mathcal{S}_j$ all possible slots for a wagon j are numbered independently of the configurations and $|\mathcal{S}_j|$ is equal to the maximal number of possible slots among all configurations for wagon j (for example, in Figure 4.1 we have $\mathcal{S}_j = \{1, 2\}$ since at most 2 slots are provided by a configuration). For a specific configuration only a subset of these slots may be active (e.g., for the configuration K1 all 2 slots are active, while the configuration K3 only provides 1 slot). Furthermore, the first slot of the wagon may be a slot for a 20, a 24 or a 45 feet load unit (which is determined by the chosen configuration). Since for these three possibilities the first slot has different positions and levers, for the third IP the slots must be treated differently. For this reason, we number the slots consecutively for each wagon over all feasible configurations. For example, for the wagon in Figure 4.1 we have $2 + 2 + 1 + 1 + 1 = 7$ possible slots.

In the IP-formulation the following parameters are used:

- $\mathcal{P}_\tau$ denotes the set of all possible slots on wagons of type τ, numbered consecutively over all possible configurations.
- For a wagon of type τ the coefficient $\gamma_\tau \in \mathbb{R}$ denotes the maximum payload for its bogies, the parameter $d_\tau \in \mathbb{R}$ denotes the distance in between the bogie attachments, and $t_\tau \in \mathbb{R}$ denotes the tare mass of the wagon.
- For a wagon j of type τ and a slot $s \in \mathcal{P}_{c(j)}$ the coefficient $\delta_{\tau s} \in \mathbb{R}$ denotes the maximum payload for slot s and the value $e_{\tau s} \in \mathbb{R}$ is the lever for a load unit on slot s relating to the first bogie.

The decision variable y_{jb} is replaced by z_{jk} with $k \in \mathcal{K}_{c(j)}$ defining the configuration of wagon j. Furthermore, we introduce auxiliary variables $a_j, b_j \in \mathbb{R}$ measuring the bogie payloads of wagon j. While a_j denotes the payload for the front bogie of wagon j, b_j is the payload of the rear bogie.

Then the linear program reads:

$$\max \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{P}_{c(j)}} (w_1 + w_2 \cdot \ell_i + w_3 \cdot g_i) \cdot x_{ijs} \tag{4.57}$$

$$- w_4 \,(m - \sum_{j\in\mathcal{M}} \sum_{k\in\mathcal{K}_{c(j)}} \kappa^0_{jk} z_{jk}) \tag{4.58}$$

$$- w_5 \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{P}_{c(j)}} d_{ij} \cdot x_{ijs} \tag{4.59}$$

s.t.

$$\sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{P}_{c(j)}} x_{ijs} \leq 1 \qquad (i \in \mathcal{N}) \tag{4.60}$$

$$\sum_{k \in \mathcal{K}_{c(j)}} z_{jk} = 1 \qquad (j \in \mathcal{M}) \tag{4.61}$$

$$\sum_{i \in \mathcal{N}_t} x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \alpha_{tks} \cdot z_{jk} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{P}_{c(j)}; t \in \mathcal{T}) \tag{4.62}$$

$$\sum_{i \in \mathcal{N}} u_i^+ \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks}^+ \cdot z_{jk} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{P}_{c(j)}) \tag{4.63}$$

$$\sum_{i \in \mathcal{N}} u_i^- \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks}^- \cdot z_{jk} \leq 0 \qquad (j \in \mathcal{M}; s \in \mathcal{P}_{c(j)}) \tag{4.64}$$

$$a_j - \sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} g_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} = \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{4.65}$$

$$b_j - \sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} g_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} = \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{4.66}$$

$$a_j \leq \gamma_{c(j)} \qquad (j \in \mathcal{M}) \tag{4.67}$$

$$b_j \leq \gamma_{c(j)} \qquad (j \in \mathcal{M}) \tag{4.68}$$

$$a_j - 3 \cdot b_j \leq 0 \qquad (j \in \mathcal{M}) \tag{4.69}$$

$$b_j - 3 \cdot a_j \leq 0 \qquad (j \in \mathcal{M}) \tag{4.70}$$

$$\sum_{i \in \mathcal{N}} g_i \cdot x_{ijs} \leq \delta_{c(j),s} \qquad (j \in \mathcal{M}; s \in \mathcal{P}_{c(j)}) \tag{4.71}$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{P}_{c(j)}} g_i \cdot x_{ijs} \leq G \tag{4.72}$$

$$x_{ijs} \in \{0,1\} \qquad (i \in \mathcal{N}; j \in \mathcal{M}; s \in \mathcal{P}_{c(j)}) \tag{4.73}$$

$$z_{jk} \in \{0,1\} \qquad (j \in \mathcal{M}; k \in \mathcal{K}_{c(j)}) \tag{4.74}$$

$$a_j, b_j \in \mathbb{R} \qquad (j \in \mathcal{M}) \tag{4.75}$$

Constraints (4.60) and (4.72) are equivalent to constraints (4.18) and (4.24). Furthermore, (4.61) ensures that for each wagon exactly one configuration is chosen. Constraints (4.62), (4.63) and (4.64) model the length restrictions for load units on slots like (4.20), (4.21) and (4.22) in the second linear program. Due to (4.65) and (4.66) the auxiliary variables a_j, b_j are set to the correct values. According to formula (4.3) for the weight restriction (W1) the payload of all bogies is restricted due to (4.67) and (4.68). The conditions from (4.4) for (W2) are modeled by (4.69) and (4.70). Finally, due to (4.71) the maximum payload for all slots is restricted according to (W3).

The third IP differs from the first and second IP with respect to feasibility and (as a consequence) also the optimal objective value. This is caused by the different modeling of the weight restrictions for the wagons. For the third IP the weight restrictions are not discretized in a finite number of options (like the type-weight lines or weight distributions in the first and second IP). As a result, all weight distributions that respect the weight conditions (W2) are feasible (especially, also all weighted means between weight distributions are allowed), i.e. the set of feasible solutions is larger. On the other hand, the first and second IP cannot ensure that the payload of each bogie is not greater than three times the payload of the other bogie. Hence, taking into account constraints (W2) leads to a reduction of the set of feasible solutions for the third IP compared to the solution sets of the first and second IP. Thus, the third IP has a different solution space: on the one hand, more solutions are feasible due to the continuous weight restrictions, on the other hand, less solutions are feasible due to the weight constraints (W2).

An advantage of the third IP is that parameter changes are easier to adopt. For example, if the maximum bogie payload is reduced due to the track the train will run on, for the first and second IP completely new feasible weight distributions for all wagons are necessary. In contrast, for the third IP simply the parameters γ_τ have to be adopted.

4.4 Computational experiments

In this section we describe some computational experiments with respect to the three integer linear programs. In order to solve the IPs we used three different solvers: the non-commercial solvers CBC 2.3.1 [72] and SCIP 1.2.0 [2] (with SoPlex 1.4.2 as LP-Solver) and the commercial product CPLEX 11.0 [56]. All calculations were done on a PC with operating system Linux, an Intel Core 2 Duo E8400 processor with 3 GHz and 2 GB main storage and a time limit of 20 minutes.

We used the 24 load unit types of the UIC definition, i.e. we have symmetric and asymmetric containers, swap bodies, and trailers. We modeled ten different wagon types used by the Swiss company HUPAC [79], the shortest one with a length of 40 feet, the longest with 104 feet. Five of the wagons are container flatcars that can just be loaded with containers and swap bodies, the other five are pocket wagons that can also haul trailers.

To test the three IP models we used 20 load planning examples. These examples are based on real-world instances obtained from the terminals Hamburg Billwerder and Dörpen in Germany. Since the positions of the load units in the storage area were not provided, we added them afterwards. For each of the 10 problem instances, we generated two variants resulting in a total of 20 problem instances for our numerical experiments. For the first variant the load unit positions in the storage area were chosen close to the actual loading positions of the units on the wagons. For the second variant we chose random storage positions for the load units.

In Table 4.3 some characteristics of the instances are listed. For each instance the number n of load units, the number $|\mathcal{L}|$ of load unit length-types, the number $|\mathcal{T}|$ of load unit fixation-types, the number m of wagons, and the number p of wagon types are shown. Additionally, we indicate whether all load units together fit onto the train or not (determined by solving the first IP to optimality). Finally, we list the total number of configurations for all wagons as well as the lengths of the trains. For 6 of the 20 load planning examples it is possible to load all units onto the train; for the other 14 examples only a subset of the units can be assigned.

Instance	n	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{T}\rvert$	m	p	do all units fit?	number conf.	train length (m)
1	49	7	3	33	7	yes	61	625
2	58	7	3	33	7	no	61	625
3	50	5	3	24	3	no	30	416
4	55	5	3	24	3	no	30	416
5	75	5	2	38	3	yes	30	612
6	85	5	2	38	3	no	30	612
7	50	4	3	30	6	no	40	513
8	50	24	4	30	6	no	40	513
9	37	6	3	36	7	yes	48	652
10	74	6	3	36	7	no	48	652

Table 4.3: Characteristics of the load planning examples

In the following we describe how the parameters encoding feasible wagon loadings were obtained. For the first integer linear program the parameters λ_{tbs} and γ_{bs} were extracted from figures like in Figure 4.1. The coefficient λ_{tbs} corresponds to the set of valid load units and γ_{bs} to the weight lines. For the second IP λ_{tbs} is replaced by α_{tks}, β^{+}_{ks} and β^{-}_{ks}. While α_{tks} can be determined from the fixation-type of one feasible load unit, for β^{+}_{ks} and β^{-}_{ks} the larger and smaller maximum overhangs of the feasible load units have to be calculated. For the third IP $\delta_{\tau s}$ can be obtained like α_{tks}. The levers $e_{\tau s}$ of the different slot positions s are calculated using the sets of feasible load units. The load unit with the maximum length gives a "length" of a slot. For each slot the lever $e_{\tau s}$ (e.g. e_1 or e_2 in Figure 4.2) is the sum of the lengths of the previous slots plus half of the length of the current slot. The lengths d_τ in between the bogie attachments and the tare mass t_τ of the wagon types were provided by train companies like HUPAC [79].

After some computational experiments and discussions with practitioners the weighting factors $w_1 = 10$, $w_2 = 1$, $w_3 = 1$, $w_4 = 1$ and $w_5 = 0.9$ were chosen. The number of load units that fit onto a train is the most important part of the objective. The other elements of the objective are more or less equally weighted. For other settings of the weighting factors the computation times did not change significantly.

As mentioned before, the three IP-formulations have different numbers of variables and constraints. This is caused by the different numbers of wagon settings (according to the usage of type-weight lines, weight distributions or configurations), the different numbers of length- and fixation-types, and by the different dimensions of the slot sets $\mathcal{S}_j$ and $\mathcal{P}_{c(j)}$. For example, for instance 6 the first IP has 87025 variables, the second 14023, and the third 45438. Furthermore, we have 806, 694 and 2936 constraints, respectively.

In Table 4.4 computational results for the three integer linear programming formulations obtained with the solvers CPLEX, SCIP and CBC are shown. All instances could be solved to optimality with CPLEX, detailed information on the runtimes and objective

model	CPLEX		SCIP			CBC		
	objective	time	objective	time	error	objective	time	error
first	1837.10	40.7	1827.01	614.5	0.55	1831.76	648.6	0.29
second	1837.10	31.5	1831.84	627.7	0.29	1827.44	855.4	0.53
third	1844.51	0.6	1844.51	13.9	0.00	1843.23	12.1	0.07

Table 4.4: Computational results

values for all problem instances can be found in Table 4.5. The instances with "r" at the end are those where the load unit positions are chosen randomly. SCIP and CBC could only solve all instances for the third IP (CBC sometimes produced different objective values, probably caused by numerical problems). For each solver we report the mean objective value for the 20 instances, the mean runtime (in seconds) and the mean relative error (in %) in comparison to the exact solutions obtained by CPLEX. The maximum runtimes for the first, second and third IP were 424, 250 and 2 seconds for CPLEX. On the other hand, for the first and second IP the non-commercial solvers reached the 20 minute time limit for several instances. For the third IP the maximum runtimes of SCIP and CBC were 30 and 37 seconds, respectively.

Instance	first IP		second IP		third IP	
	objective	time	objective	time	objective	time
1	1607.02	0.6	1607.02	0.4	1609.82	0.4
1r	1592.12	12.0	1592.12	1.1	1590.32	0.9
2	1875.11	31.2	1875.11	7.7	1880.51	1.0
2r	1859.91	21.6	1859.91	6.1	1860.81	1.3
3	1650.93	0.9	1650.93	0.8	1650.93	0.2
3r	1584.91	18.0	1584.91	18.0	1604.91	0.4
4	1694.65	11.0	1694.65	32.7	1715.81	0.5
4r	1651.43	192.7	1651.43	250.0	1676.71	0.8
5	2171.67	1.0	2171.67	1.0	2178.86	0.4
5r	2129.27	21.0	2129.27	3.1	2132.47	1.0
6	2321.81	16.3	2321.81	80.7	2337.55	1.0
6r	2236.48	47.6	2236.48	7.2	2241.28	1.2
7	1752.25	0.2	1752.25	0.3	1754.85	0.2
7r	1799.40	0.1	1799.40	0.1	1799.40	0.2
8	1764.54	2.1	1764.54	7.1	1779.20	0.2
8r	1808.36	6.4	1808.36	7.8	1824.86	0.5
9	1430.86	0.2	1430.86	0.1	1430.86	0.2
9r	1408.16	0.4	1408.16	0.2	1409.06	0.2
10	2202.76	6.9	2202.76	3.5	2204.56	0.6
10r	2200.47	424.3	2200.47	201.4	2207.48	0.8

Table 4.5: Detailed computational results obtained with CPLEX.

Due to the fact that the first IP has more decision variables and constraints than the second IP, we expected that the second IP could be solved faster. However, in fact the runtimes were quite similar. The mean runtime with CPLEX for the first and second IP differed just about a factor of 1.3. With the non-commercial solvers the computational results are slightly better for the second IP with SCIP, but slightly worse for CBC.

Compared to the runtimes of the first and the second IP those of the third IP are much smaller. The mean runtime with CPLEX is reduced to 0.6 seconds. Even the non-commercial solvers SCIP and CBC solved all problems within less than a minute.

The difference of the mean objective values for the first and second compared to the third IP can be explained by the fact that the solution space of the third IP is different. The weight restrictions for the first and second IP derived from load patterns like in Figure 4.1 are usually stricter than the conditions (W1) and (W3) used in the third IP. This is motivated by the fact that infeasible wagon loadings have to be definitely avoided in practice. Especially, the maximum possible weight limits (due to weight conditions (W1) and (W3)) are often not reached for the used weight distributions. Hence, the continuous weight restrictions in the third IP lead to better objective values.

model	first IP				second IP			
	original		weighted means		original		weighted means	
instance	b	time	b	time	b'	time	b'	time
1	1925	0.6	7123	3.5	265	0.4	922	0.7
1r	1925	12.0	7123	12.4	265	1.1	922	2.8
2	1925	31.2	7123	195.1	265	7.7	922	20.5
2r	1925	21.6	7123	183.6	265	6.1	922	24.8
3	364	0.9	1309	40.5	113	0.8	392	7.8
3r	364	18.0	1309	152.4	113	18.0	392	19.6
4	364	11.0	1309	1200.0	113	32.7	392	1200.0
4r	364	192.7	1309	679.9	113	250.0	392	481.9
5	478	1.0	1732	8.0	113	1.0	392	1.4
5r	478	21.0	1732	91.9	113	3.1	392	9.1
6	478	16.3	1732	132.8	113	80.7	392	28.1
6r	478	47.6	1732	302.9	113	7.2	392	12.1
7	128	0.2	431	0.3	166	0.3	574	0.4
7r	128	0.1	431	0.1	166	0.1	574	0.2
8	2045	2.1	7442	78.5	166	7.1	574	4.9
8r	2045	6.4	7442	254.0	166	7.8	574	24.5
9	1369	0.2	5112	2.1	229	0.1	787	0.2
9r	1369	0.4	5112	4.6	229	0.2	787	0.5
10	1369	6.9	5112	53.5	229	3.5	787	8.2
10r	1369	424.3	5112	1200.0	229	201.4	787	332.5

Table 4.6: Computational results for first and second IP with additional weighted means in the input data

In order to see whether a larger number of feasible weight distributions leads to better objective values, we tested the first and second IP also with additional weight distributions (type-weight lines, respectively) build by a weighted mean of adjacent weight distributions. For each pair of adjacent weight distributions we introduced three equidistant new weight distributions. The resulting IPs have more variables and constraints than the second IP (without weighted means weight distributions). We denote by $b := \sum_{c=1}^{p} \sum_{k \in \mathcal{K}_c} |\mathcal{B}_k|$ and $b' := \sum_{c=1}^{p} \sum_{k \in \mathcal{K}_c} |\mathcal{B}'_k|$ the total number (summed over all wagon types and configurations) of type-weight lines and weight distributions, respectively. In Table 4.6 we report the runtimes (in seconds, obtained by CPLEX) and the numbers b, b' for all problem instances. The mean runtime of CPLEX for these IPs is considerably larger (230 seconds for the first IP and 109 for the second compared to 41 and 32 seconds for the original version without weighted means weight distributions). Hence, with more type-weight lines or weight distributions the different numbers of variables and constraints of the first and second IP have a considerable impact on the runtimes. With the first IP two instances and for the second IP one instance could not be solved to optimality within the time limit of 20 minutes. The objective values were slightly better than the ones without additional weighted mean weight distributions (1837.75 for the first and second IP, compared to the original value of 1837.10). Thus, using weighted means of weight distributions in the data gives only a small benefit, but induces much larger runtimes.

Instance	second IP		$\mathrm{WM}_{\mathrm{all}}$		$\mathrm{WM}_{\mathrm{adjacent}}$	
	objective	time	objective	time	objective	time
1	1607.02	0.4	1608.82	0.3	1608.82	0.5
1r	1592.12	1.1	1592.12	1.5	1592.12	2.7
2	1875.11	7.7	1879.61	17.3	1876.01	290.9
2r	1859.91	6.1	1859.91	10.1	1859.91	11.7
3	1650.93	0.8	1650.93	4.6	1650.93	13.0
3r	1584.91	18.0	1591.21	76.4	1591.21	493.9
4	1694.65	32.7	1695.66	385.2	1695.66	1200.0
4r	1651.43	250.0	1661.93	53.6	1660.13	1200.0
5	2171.67	1.0	2171.67	1.7	2171.67	2.8
5r	2129.27	3.1	2131.36	4.4	2131.37	13.7
6	2321.81	80.7	2326.56	15.1	2324.50	219.8
6r	2236.48	7.2	2236.48	8.2	2236.48	19.5
7	1752.25	0.3	1752.25	0.2	1752.25	0.4
7r	1799.40	0.1	1799.40	0.1	1799.40	0.1
8	1764.54	7.1	1764.54	7.7	1764.54	12.4
8r	1808.36	7.8	1808.36	13.9	1808.36	25.6
9	1430.86	0.1	1430.86	0.1	1430.86	0.1
9r	1408.16	0.2	1408.16	0.2	1408.16	0.2
10	2202.76	3.5	2202.76	4.9	2202.76	15.6
10r	2200.47	201.4	2201.18	144.4	2201.18	1200.0

Table 4.7: Computational results for the second IP and the second IP with weighted means over all weight distributions of a configuration ($\mathrm{WM}_{\mathrm{all}}$) and the second IP with weighted means of adjacent weight distributions ($\mathrm{WM}_{\mathrm{adjacent}}$).

We will now present some results of the second IP with modeled weighted means of weight distributions. For the second IP with weighted means of all weight distributions of a configuration (WM_{all}) the mean runtime with CPLEX stays nearly the same as for the second IP (31.5s for the second IP and 33.9s for WM_{all}). For the second IP with weighted means of adjacent weight distributions ($\text{WM}_{\text{adjacent}}$) the runtime is larger than the runtime of the second IP. The mean runtime is 236.2s compared to 31.5s for the second IP. For three instances the gap could not be reduced to 0% within the runtime of 20 minutes (but all gaps were below 0.5%). Actually, an optimal solution has been found for all instances, but CPLEX could not prove optimality for three instance within the time limit (optimality could be proven after about 2 hours). The mean objective slightly improves to 1838.69 for WM_{all} and 1838.32 for $\text{WM}_{\text{adjacent}}$ (from 1837.10 for the second IP). The results for $\text{WM}_{\text{adjacent}}$ and WM_{all} are also slightly better than those of the second IP with a fixed number of additional weight distributions (see previous paragraph) where the objective is 1837.75. In Table 4.7 the runtimes and objective values of all instances are compared for the second IP and the variants with weighted means $\text{WM}_{\text{adjacent}}$ and WM_{all}. For three instances the improvement with WM_{all} is higher than with $\text{WM}_{\text{adjacent}}$. For the second IP with WM_{all} or $\text{WM}_{\text{adjacent}}$ the objective improves for 8 of the 20 instances. The runtime of WM_{all} stays often more or less the same. For 3 instances it is considerably smaller than with the second IP but for two instances it is larger (up to a factor of over 10). The runtime of $\text{WM}_{\text{adjacent}}$ is sometimes equal but often larger than the runtime of the second IP (with a factor of up to over 20).

The second IP with WM_{all} shows that the differences in the objective values between the second and third IP are not mainly caused by the continuous modeling. Unfortunately this indicates that the main differences in the objective are caused by errors of the parameters for the third IP that are needed for the calculations by the lever principle (e.g. the levers $e_{\tau s}$).

Finally, we analyzed the computational results of CPLEX to find out which instances are easy or hard to solve. For all three IPs the instances with random load unit positions were more difficult than instances with chosen positions. The runtime of the problems with random positions were about 10 (first IP), 4 (second IP) and 1.4 (third IP) times larger. For all IPs the instances where only a subset of load units could be assigned to the train needed larger runtimes than instances where all load units together fit onto the train.

The computational results show that the load planning problem enlarged by weight restrictions can be solved very fast (even with non-commercial IP-solvers) for real-world instances. The results of the first and second IP imply that it is possible to use the well-established load patterns currently used by companies to formulate the weight restrictions. The model extension WM_{all} of the second IP furthermore shows that weighted means of weight lines are possible. On the other hand, modeling the weight restrictions directly by the underlying formulas seems to be advantageous. The third IP could be solved in a few seconds. Furthermore, this formulation is more flexible and no pre-calculation of a set of feasible load patterns is necessary.

4.5 List of load planning notations

general notations	
m	number of wagons
$\mathcal{M}$	set of all wagons
p	number of wagon types
$\mathcal{W}$	set of all wagon types
$c(j)$	type of wagon j
G	total weight limit of the train
n	number of load units
$\mathcal{N}$	set of all load units
ℓ_i	length of load unit i
g_i	weight of load unit i
d_{ij}	transportation cost for load unit i to wagon j
$\mathcal{K}_\tau$	set of physical configurations for wagons of type τ
κ^0_{jk}	is equal to 1 if k is the initial configuration of wagon j
$w_1, \dots, w_5$	weighting factors in the objective function
notations for the first IP	
$\mathcal{L}$	set of load unit length-types
$\mathcal{N}_t$	set of load units belonging to length-type t
$\mathcal{B}_k$	set of type-weight lines for configuration k
λ_{tbs}	is equal to 1 if load unit length type t is feasible for slot s in type-weight line b
$\mathcal{S}_j$	set of slots on wagon j (also second IP)
γ_{bs}	maximum payload for slot s in type-weight line or weight distribution b (also second IP)
notations for the second IP	
$\mathcal{T}$	set of load unit fixation-types
$\mathcal{N}_t$	set of load units belonging to fixation-type t
$\mathcal{B}'_k$	set of weight distributions for configuration k
α_{tks}	is equal to 1 if load unit fixation-type t if feasible for slot s in configuration k (also third IP)
u^+_i, u^-_i	overhangs of load unit i (also third IP)
$\beta^+_{ks}, \beta^-_{ks}$	feasible overhangs for slot s in configuration k (also third IP)
notations for the third IP	
$\mathcal{P}_\tau$	set of all slot positions on wagon of type τ
γ_τ	maximum bogie payloads for wagons of type τ
d_τ	distance in between the bogie attachments for wagons of type τ
t_τ	tare mass for wagons of type τ
$\delta_{\tau s}$	maximum slot payload for wagons of type τ on slot s
$e_{\tau s}$	lever for slot s for wagons of type τ
a_j, b_j	bogie payloads for wagon j

Table 4.8: List of load planning notations

5 Robust load planning

In this section the problem of robust load planning for trains in intermodal container terminals is studied. The deterministic problem of load planning is introduced and described in Section 4. The goal of the load planning problem is to choose wagon settings and assign load units to the wagons of a train. It is important that weight restrictions and other restrictions have to be respected in order to guarantee that the train is not overloaded and correctly balanced. The objective of load planning is to maximize the utilization of the train and to minimize the total costs in the terminal. This section is based on the paper Bruns et al. [26].

In real-world applications many of the parameters needed for the model are not known exactly. For example, the weights and lengths of load units may differ from the announced values or a wagon may have a bug and it may not be allowed to load this wagon. Uncertainty in optimization problems may be tackled by stochastic or robust optimization. In general, stochastic optimization is appropriate if the goal is to find a solution which performs well on average. On the other hand, in classic robust optimization, the worst-case is considered, hence a robust solution guarantees feasibility for *all* possible scenarios. In the load planning problem, it is crucial to guarantee a feasible loading of the train, no matter how wrong the information in the preloading phase was. Hence, a robust approach is needed. We applied robust approaches to the third integer linear program (see Section 4.3.4) as this models all real world constraints and is the model of Section 4 that could be solved in the least amount of time.

Treating uncertainties in optimization by robustness concepts has also been successfully applied to a wide range of other real-world problems. Examples for other applications include sawmill planning by Alvarez and Vera [5], timetabling and timetable information (see Goerigk and Schöbel [50] or Goerigk et al. [49]), container repositioning for maritime terminals by Erera et al. [40], line planning by Bessas et al. [17] and many more.

Contributions. We study three different sources of uncertainties: interval-based uncertainties in overhangs, interval-based uncertainties in weights and failure sets for bugs of wagons. We begin with *strict* robustness (for the concept of strict robustness, see Ben-Tal and Nemirovski [13] or Ben-Tal et al. [11] and references therein) and require that the configurations and the assignments need to be feasible for all possible scenarios which may be revealed later on. In order to relax this rather conservative approach we follow two approaches: we firstly reduce the uncertainty set to more likely scenarios as done in Bertsimas and Sim [16] and secondly allow to adjust the assignment in a second phase when the scenario is known, which is an application of the concept of *adjustable* robustness as introduced in Ben-Tal et al. [12]. For most of the robust counterparts we obtained formulations as mixed-integer linear programs with a finite number of constraints. The adjustable robust counterpart is harder to solve because of an infinite number of constraints. Therefore, we provide a heuristic approach.

Extended computational experiments on instances motivated by real-world settings in German terminals show that most of our formulations can be solved within a few minutes of runtime, which is good enough for real-world applications. We furthermore discuss the resulting robust solutions as well as their usefulness for our application. To this end, we introduce a new way to measure the robustness of a solution which is based on the intuition that a solution is more robust if it is feasible for more realizations of the uncertain parameters.

Our experimental results show that relatively large gains in robustness can sometimes be achieved at only small costs in terms of nominal quality. Thus, a robustness analysis should be an essential tool for practitioners. Furthermore, adjustable robust solutions, though requiring an increased computational effort, outperform nominal and strictly robust solutions with respect to nominal quality when given sufficient time to modify the load plan accordingly.

Overview. The remainder of this section is organized as follows. In Section 5.1 we introduce the considered robustness concepts. In Section 5.2 we recall a MIP formulation for the deterministic problem, subsequently we give a reformulation to eliminate the auxiliary variables (Section 5.3) and specify the different considered uncertainty sets in Section 5.4. While in Section 5.5 we discuss the concept of strict robustness, in Section 5.6 we deal with adjustable robustness. After presenting computational experiments in Section 5.7 we conclude this section with some remarks in Section 5.8.

5.1 Robustness concepts

In this section we introduce the considered robustness concepts. First, we motivate robust optimization and introduce a definition of a robust optimization problem. Afterwards, we describe the concepts of strict robustness (Section 5.1.1), controlled robustness (Section 5.1.2) and adjustable robustness (Section 5.1.3).

When we look at practical optimization problems, we are often facing uncertainties in the input parameters. Uncertainties in parameters may e.g. be caused by measurement or forecast errors. If we want to find a feasible solution for an optimization problem under uncertainties, we have to evaluate or if this is not possible, to define the uncertainty we want to hedge against. This is e.g. possible by declaring a range for all uncertain parameters or by declaring scenarios that may appear. Furthermore, there are different ways how feasibility is handled in robustness concepts. E.g. the policy can be that the solution is feasible for all possible realizations of the uncertain parameters or only for a restricted set of realizations. Another possible policy is that the solution has not to be feasible, but almost feasible. Additionally, a robust objective is necessary to define a robustness concept. Often the worst case over all scenarios is considered. If the application allows to postpone parts of the decision on the solution until the uncertainty reveals, a two step approach may be chosen. So in the first step parts of the solution are chosen under uncertainty and completed when all parameters reveal.

We now introduce a general robust optimization problem to explain the robustness concepts we apply. To define a robust optimization problem let us consider a general maximization problem

$$(P) \qquad \max\{f(x) : F(x) \leq 0, x \in \mathbb{R}^n\}$$

with $f : \mathbb{R}^n \to \mathbb{R}$ as objective function and $F : \mathbb{R}^n \to \mathbb{R}^m$ representing the constraints, so x is feasible if $F(x) \leq 0$. The objective function $f(x)$ and the function $F(x)$ of such optimization problems usually depend on parameters which are in most practical applications not exactly known. For this purpose we introduce an *uncertainty set* (or *scenario set*) $\mathcal{U} \subseteq \mathbb{R}^M$ that contains all possible realizations of the M unknown parameters. As the functions $f(x)$ and $F(x)$ depend on the scenario parameters, we redefine them as

$$f : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R} \text{ and } F : \mathbb{R}^n \times \mathcal{U} \to \mathbb{R}^m.$$

So, we can rewrite the problem as

$$\begin{array}{llll} P(\xi) & \max & f(x,\xi) & \\ & \text{s.t.} & F(x,\xi) \leq 0 & \xi \in \mathcal{U} \\ & & x \in \mathbb{R}^n & \end{array}$$

indicating that both the objective function and the constraints depend on some (unknown) parameters $\xi \in \mathbb{R}^M$ and we have to consider all parameters (or scenarios) of the uncertainty set $\mathcal{U}$.

The corresponding *uncertain optimization problem* is given as

$$P(\xi), \xi \in \mathcal{U}. \tag{5.1}$$

Often it is assumed that there is some *nominal scenario* $\hat{\xi} \in \mathcal{U}$ which is not accurate or which may be disturbed. For example, $\mathcal{U}$ could be given as

$$\mathcal{U} = \{\xi \in \mathbb{R}^M : \|\xi - \hat{\xi}\| \leq r\}$$

for some norm $\|\cdot\|$ and a given radius $r \in \mathbb{R}_+$. The nominal scenario $\hat{\xi}$ may refer to the one which would be used in the non-robust problem (P), i.e. the scenario in which the parameters are equal to their "expected" values. In the following we will assume that $P(\hat{\xi})$ is feasible and that an optimal solution to $P(\hat{\xi})$ with finite objective value

$$z^* = \max\{f(x,\hat{\xi}) : F(x,\hat{\xi}) \leq 0, x \in \mathbb{R}^n\}$$

exists.

Example 5.1. As an example for a robust optimization problem we consider a train travel routing and scheduling problem. Assume we want to optimize a travel from a starting station A to a destination station B and there are two tracks we could use with intermediate stations C for the first track and D for the second track. Furthermore, we consider different connections which may contain a change of train in the intermediate stations. We assume that we know the maximum delay that may occur between two stations for each train. If delays occur, one may not reach the connecting train at the intermediate station. As objective function we want to reach the destination station as early as possible. So, we consider the train travel routing and scheduling problem as a minimization problem. □

In the following we discuss some possibilities how to deal with such an uncertain optimization problem and show how the concepts could be applied to the above example.

5.1.1 Strict robustness

Strict robustness might be considered as the oldest and most conservative approach to uncertainty. It was introduced by Soyster [81] and considerably extended by Ben-Tal, Ghaoui and Nemirovski, see [13, 11] and references therein.

A solution x to the uncertain problem (5.1) is called *strictly robust* if it is feasible for any of the scenarios in $\mathcal{U}$. This means that $x \in \mathbb{R}^n$ is *strictly robust* for $P(\xi), \xi \in \mathcal{U}$, if

$$F(x, \xi) \leq 0 \text{ for all } \xi \in \mathcal{U}.$$

In many applications (e.g. building a bridge, flying an airplane or doing a space travel) security is the most important goal such that only strictly robust solutions can be taken into consideration. The *robust counterpart* (RC) of (5.1) due to Ben-Tal and Nemirovski [13] is given as

$$\begin{array}{lrl} \text{(RC)} & \max & \inf\limits_{\xi \in \mathcal{U}} f(x, \xi) \\ & \text{s.t.} & F(x, \xi) \leq 0 \text{ for all } \xi \in \mathcal{U} \\ & & x \in \mathbb{R}^n \end{array}$$

Its objective follows a pessimistic view: The goal is to maximize the worst-case over all scenarios.

Example 5.2. We can apply the concept of strict robustness to the travel routing and scheduling problem (Example 5.1) by assuming that we have to exactly use the trains we decided in the beginning. So the initial train assignment has to be feasible even if the maximum delays for the selected trains occur. This implies, that we may not take a possible connecting train at an intermediate station, because with a delay we could have missed this train. So, we decide to wait for the train, that has been chosen in the initial plan. The worst case objective is defined by the scenario where the maximum delays occur on all of the used tracks. One implication of this concept may be that we reach the station C on time, but we do not take the next train as we initially decided to take a later train (because we assumed the worst case delay). $\square$

5.1.2 Controlled robustness model of Bertsimas and Sim

As it may be unlikely that all parameters take their worst case or extreme values simultaneously, Bertsimas and Sim [16] introduced an approach where the number of uncertain coefficients is limited. This limit controls the uncertainty set and is often called Γ.

Based on the idea of Bertsimas and Sim we restrict the uncertainty set and especially the scenarios that have to be considered in the following way:

$$\mathcal{U}^\Gamma = \{\xi \in \mathbb{R}^M : \exists I \subseteq \{1, \ldots, M\}, |I| \leq \Gamma,$$

$$\xi_i = \tilde{\xi}_i \text{ for } i \notin I,\ |\xi_i - \hat{\xi}_i| \leq r\} \text{ for } i \in I\}.$$

This definition states that Γ elements of ξ are within a given radius r from the corresponding entry of the nominal scenario $\tilde{\xi}$. All other elements of ξ are not disturbed at all.

A solution x is feasible if it is feasible for all scenarios of the controlled uncertainty set $\mathcal{U}^\Gamma$. The *controlled robust counterpart* (cRC) reads similar to the one of strict robustness:

$$\begin{array}{llll}
\text{(cRC)} & \max \inf\limits_{\xi \in \mathcal{U}} & f(x,\xi) \\
& \text{s.t.} & F(x,\xi) \leq 0 \text{ for all } \xi \in \mathcal{U}^\Gamma \\
& & x \in \mathbb{R}^n
\end{array}$$

In contrast to strict robustness the approach of Bertsimas and Sim is less conservative as it considers less scenarios. Accordingly, for less scenarios feasibility has to be ensured, so the approach of Bertsimas and Sim may allow better objective values. Depending on the size of Γ the worst case scenario of $\mathcal{U}$ may not be element of $\mathcal{U}^\Gamma$. In this situation the controlled robustness approach of Bertsimas and Sim allows better objectives than strict robustness. If the worst case scenario of $\mathcal{U}$ is also element of $\mathcal{U}^\Gamma$ for a given Γ, strict and controlled robustness have the same optimal solutions and objective values.

Example 5.3. While in Example 5.1 we considered the maximum delays for all used train-track combinations for strict robustness, applying controlled robustness we limit the number of train-track combinations where a delay may occur. For example we could choose $\Gamma = 1$ which means that we consider the case that only on one of the used tracks a delay occur. So, we assume that either between A and C or between C and B a delay occurs (for the other route either between A and D or between D and B) but not at both tracks at the same travel. The approach of controlled robustness is even more interesting when we consider routes with more than one intermediate station and possibly change of trains. With more intermediate stations it is also interesting to look at larger Γ-values.
$\square$

5.1.3 Adjustable robustness

Another concept of reducing the level of conservatism uses the observation that usually not all of the variables of an uncertain optimization problem have to be fixed before the realization of the scenario $\xi \in \mathcal{U}$ is known. In the context of robust optimization this has been realized in Ben-Tal et al. [12]. Here the variables $x \in \mathbb{R}^n$ are separated as

$$x = (u, v) \quad u \in \mathbb{R}^{n_u}, v \in \mathbb{R}^{n_v}$$

with $n_u + n_v = n$. The variables u have to be fixed before the scenario ξ is known. They are not adjustable and are called “here and now”-variables. On the other hand, the variables v can be adjusted after the occurring scenario ξ becomes known. They are referred to as *adjustable* or “wait and see”-variables. We rewrite the optimization problem $P(\xi)$ equivalently as

$$\begin{array}{llll} P(\xi) & \max & f(u,v,\xi) & \\ & \text{s.t.} & F(u,v,\xi) \leq 0 & \xi \in \mathcal{U} \\ & & u \in \mathbb{R}^{n_u} & \\ & & v \in \mathbb{R}^{n_v} & \end{array}$$

and call an assignment of the variables $u \in \mathbb{R}^{n_u}$ *adjustable robust* for the uncertain optimization problem $P(\xi), \xi \in \mathcal{U}$, if for any scenario $\xi \in \mathcal{U}$ there exists an assignment of the “wait and see”-variables $v \in \mathbb{R}^{n_v}$ such that all constrains are feasible. We denote the set of all adjustable robust solutions by aR $\subseteq \mathbb{R}^{n_u}$ which is defined as:

$$\text{aR} = \{u \in \mathbb{R}^{n_u} : \forall \xi \in \mathcal{U}\ \exists\ v \in \mathbb{R}^{n_v}; F(u,v,\xi) \leq 0\}.$$

The objective of such a partial solution defined by the “here and now”-variables u is:

$$z^{aR}(u) = \inf_{\xi \in \mathcal{U}} \sup_{v:(u,v)\in\mathcal{F}(\xi)} f(u,v,\xi),$$

where the set $\mathcal{F}(\xi)$ for the scenario $\xi \in \mathcal{U}$ contains $(u,v) \in \mathbb{R}^{n_u} \times \mathbb{R}^{n_v}$ if $F(u,v,\xi) \leq 0$, i.e. if there are variables u such that the solution (u,v) is feasible.

We hence obtain as the *adjustable robust counterpart* (aRC):

$$\begin{array}{lll} (\text{aRC}) & \max & z^{aR}(u) \\ & \text{s.t.} & u \in \text{aR} \end{array}$$

We have to choose a feasible assignment of the “here and now”-variables u with maximal value $z^{aR}(u)$, which is equal to the infimum of the objective over all scenarios for the best possible assignment of the “wait and see”-variables v.

Now assume that $u \in$ aR has been chosen and that the scenario $\overline{\xi} \in \mathcal{U}$ is the realization of the uncertainty set. Then the adjustable variables v are chosen as good as possible, i.e. the chosen variables $\overline{v}$ have to fullfill

$$f(\overline{v}, y, \overline{\xi}) = \sup_{v:(u,v)\in\mathcal{F}(\overline{\xi})} f(u,v,\overline{\xi}).$$

Example 5.4. The concept of adjustable robustness can be applied to the travel routing and scheduling problem from Example 5.1 by defining a travel route in the first stage ("here and now") and choosing the concrete trains depending on the actual delays in the second stage ("wait and see"). This means that we consider in the first stage the worst case for each travel route and take the route with the best worst case objective. When we start travelling we decide on connecting trains according to the time we reach intermediate stations which is influenced by the actual delays that occur during the travel with the used trains. So, this is quite similar to the approach railway companies use. If a connection train is missed, they announce an alternative train for the same route or even for different routes. The fact that rail companies even choose new routes for passengers, makes their approach an online optimization. □

Note, that the three introduced robustness concepts can not be compared directly as they consider different planning or uncertainty situations.

For a "fair" comparison we can not directly compare the solutions of the one stage approaches (strict and controlled robustness) with the two stage approach of adjustable robustness. The approaches are not comparable because adjustable robustness allows individual solutions for the "wait and see"-variables for each scenario while strict and controlled robustness work with one solution for all scenarios. To compare we only take the "here and now"-variables as fixed for the solutions of strict and controlled robustness. For the revealed scenario we complete the solution by calculating optimized "wait and see"-variables. So we adapt the one stage concepts of strict and controlled robustness to a two stage approach. Nevertheless, in the first stage the optimization acts like an one stage approach.

Note that the robustness concepts used here are not the only robustness concepts considered in the literature. Other concepts often allow solutions that are not feasible but almost feasible, like light robustness (cf. Fischetti and Monaci [44]) or soft robustness (see Ben-Tal et al. [10]). Another concept is recovery robustness (see e.g. Stiller [83] where a solution is provided together with a group of recovery algorithms so that for each scenario at least one algorithm of the group can repair the initially provided solution (to a feasible solution). In contrast in regret robustness the aim is to restrict the deviation in the objective function caused by uncertain parameters in the objective function. Feasibility is not considered. So, solutions may not be feasible for some scenarios.
All approaches that do not ensure feasibility in all cases are not applicable to the load planning problem as feasibility has to be ensured. The approach of recovery robustness may be the focus of future research.

5.2 A MIP-formulation for the deterministic load planning

In this section we assume that all information needed to model the load planning problem are known and state a variant of the third IP of Section 4.3.4 that will be used for our robustness approaches.

As we assume that the overhangs to both sides are similar, for the robustness approach we have to adapt some parameters: For a wagon j in configuration $k \in \mathcal{K}_{c(j)}$ the coefficients $\beta_{ks} \in \mathbb{R}$ denote the maximum allowed overhang of a load unit on slot $s \in \mathcal{P}_{c(j)}$. Furthermore, the overhang over the corner castings to both sides is u_i for load unit $i \in \mathcal{N}$.

As in Section 4.3.4 we have the binary decision variables y_{jk} with $k \in \mathcal{K}_{c(j)}$ defining the configuration of wagon $j \in \mathcal{M}$ and x_{ijs} determining the slot assignment for load unit $i \in \mathcal{N}$ to wagon $j \in \mathcal{M}$ and slot $s \in \mathcal{P}_{c(j)}$. Furthermore, we again use auxiliary variables $a_j, b_j \in \mathbb{R}$ measuring the front and rear bogie payloads of wagon j.

Then the mixed-integer linear program reads:

$$\textbf{(LP)} \quad \max \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{P}_{c(j)}} (w_1 + w_2 \cdot \ell_i + w_3 \cdot g_i) \cdot x_{ijs} \tag{5.2}$$

$$- w_4 \, (m - \sum_{j\in\mathcal{M}} \sum_{k\in\mathcal{K}_{c(j)}} \kappa^0_{jk} y_{jk}) \tag{5.3}$$

$$- w_5 \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{P}_{c(j)}} d_{ij} \cdot x_{ijs} \tag{5.4}$$

s.t.

$$\sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{P}_{c(j)}} x_{ijs} \le 1 \qquad (i \in \mathcal{N}) \tag{5.5}$$

$$\sum_{k\in\mathcal{K}_{c(j)}} y_{jk} = 1 \qquad (j \in \mathcal{M}) \tag{5.6}$$

$$\sum_{i\in\mathcal{N}_t} x_{ijs} - \sum_{k\in\mathcal{K}_{c(j)}} \alpha_{tks} \cdot y_{jk} \le 0 \qquad (j \in \mathcal{M}, s \in \mathcal{P}_{c(j)}, t \in \mathcal{T}) \tag{5.7}$$

$$\sum_{i\in\mathcal{N}} u_i \cdot x_{ijs} - \sum_{k\in\mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} \le 0 \qquad (j \in \mathcal{M}, s \in \mathcal{P}_{c(j)}) \tag{5.8}$$

$$a_j - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{P}_{c(j)}} g_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} = \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{5.9}$$

$$b_j - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{P}_{c(j)}} g_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} = \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{5.10}$$

$$a_j \le \gamma_{c(j)} \qquad (j \in \mathcal{M}) \tag{5.11}$$

$$b_j \le \gamma_{c(j)} \qquad (j \in \mathcal{M}) \tag{5.12}$$

$$a_j - 3 \cdot b_j \le 0 \qquad (j \in \mathcal{M}) \tag{5.13}$$

$$b_j - 3 \cdot a_j \le 0 \qquad (j \in \mathcal{M}) \tag{5.14}$$

$$\sum_{i\in\mathcal{N}} g_i \cdot x_{ijs} \le \delta_{c(j),s} \qquad (j \in \mathcal{M}, s \in \mathcal{P}_{c(j)}) \tag{5.15}$$

$$\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{P}_{c(j)}} g_i \cdot x_{ijs} \le G \tag{5.16}$$

$$x_{ijs} \in \{0, 1\} \qquad (i \in \mathcal{N}, j \in \mathcal{M}, s \in \mathcal{P}_{c(j)}) \tag{5.17}$$

$$y_{jk} \in \{0,1\} \qquad (j \in \mathcal{M}, k \in \mathcal{K}_{c(j)}) \tag{5.18}$$
$$a_j, b_j \in \mathbb{R} \qquad (j \in \mathcal{M}) \tag{5.19}$$

The parameters $w_1, \ldots, w_5 \geq 0$ in the objective function are coefficients weighting the different components. While w_1, w_2, w_3 weight the total number, total length and total weight of assigned load units in (5.2), w_4 is the factor for the setup costs in (5.3) and w_5 is the factor for the transportation costs in (5.4). The setup costs for changing a configuration are assumed to be constant, i.e. in (5.3) the number of changed configurations is minimized. We will write $obj(x,y)$ for the objective function, when the coefficients are clear from the context.

Constraints (5.5) ensure that each load unit is assigned to at most one slot, (5.6) guarantees that for each wagon one associated configuration is chosen. Constraint (5.7) ensures that the types of all assigned load units are feasible, while constraints (5.8) represent the limits of the overhangs. Due to (5.9) and (5.10) the auxiliary variables a_j, b_j are set to the correct values. The payload of all bogies is restricted due to (5.11) and (5.12) (W1). Due to (5.13) and (5.14) the payload on a bogie may not be larger than three times the payload on the other bogie (W2). Constraint (5.15) limits the maximum slot payloads (W3). Finally, the total weight limit of the train is modeled by (5.16). The domains of the variables are defined in (5.17) to (5.19).

The above stated MIP differs from the third integer program (see 4.3.4) manly by one constraint that replaces two of (4.3.4). The two replaced constraints are constraints (4.63) and (4.64) which limit the maximum feasible greater and smaller overhang for slots. These constraints are replaced by constraints (5.8) which limits the maximum feasible overhang as we only consider symmetric load units for the robust load planning.

5.3 Problem reformulation

The presented problem formulation has a drawback from the point of view of robust optimization: It should be possible to determine the values of auxiliary variables (like a_j and b_j) when scenario information becomes known, as they do not represent a choice of the decision maker, but are introduced only to contain information within the MIP. Furthermore, the auxiliary variables come with the equality constraints (5.9) and (5.10) which circumvent the possibility to find a solution that is feasible for every possible scenario especially for uncertainty sets that influence the load unit weights, as changes in load unit weights change also the bogie payloads. Therefore, we give an equivalent reformulation in which equality constraints and auxiliary variables are removed. Note that the problem of the equality constraints does not occur with equality constraints 5.6 as they are not used to calculate auxiliary variables.

By combining equations (5.9) and (5.10) with the inequalities (5.13) and (5.14) (which eliminates the auxiliary variables a_j and b_j), for all $j \in \mathcal{M}$ we get

$$\left(\sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} g_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \frac{t_{c(j)}}{2} \right) - 3 \left(\sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} g_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \frac{t_{c(j)}}{2} \right) \leq 0$$

$$\left(\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{P}_{c(j)}} g_i\cdot\frac{e_{c(j),s}}{d_{c(j)}}\cdot x_{ijs}+\frac{t_{c(j)}}{2}\right)-3\left(\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{P}_{c(j)}} g_i\cdot\frac{d_{c(j)}-e_{c(j),s}}{d_{c(j)}}\cdot x_{ijs}+\frac{t_{c(j)}}{2}\right)\leq 0$$

Reformulation yields

$$\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{P}_{c(j)}} g_i\cdot\frac{d_{c(j)}-4e_{c(j),s}}{d_{c(j)}}\cdot x_{ijs}\leq t_{c(j)} \qquad (j\in\mathcal{M}) \tag{5.20}$$

$$\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{P}_{c(j)}} g_i\cdot\frac{4e_{c(j),s}-3d_{c(j)}}{d_{c(j)}}\cdot x_{ijs}\leq t_{c(j)} \qquad (j\in\mathcal{M}) \tag{5.21}$$

and, for inequalities (5.11)-(5.12):

$$\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{P}_{c(j)}} g_i\cdot\frac{d_{c(j)}-e_{c(j),s}}{d_{c(j)}}\cdot x_{ijs}\leq \gamma_{c(j)}-\frac{t_{c(j)}}{2} \qquad (j\in\mathcal{M}) \tag{5.22}$$

$$\sum_{i\in\mathcal{N}}\sum_{s\in\mathcal{P}_{c(j)}} g_i\cdot\frac{e_{c(j),s}}{d_{c(j)}}\cdot x_{ijs}\leq \gamma_{c(j)}-\frac{t_{c(j)}}{2} \qquad (j\in\mathcal{M}) \tag{5.23}$$

We hence obtain:

Lemma 5.1. *Constraints (5.9)-(5.14) are equivalent to constraints (5.20)-(5.23).*

Finally, to handle the problem constraints easier, we introduce a more compact problem formulation. We split the constraints into three classes: Those containing only x-variables, constraints containing only y-variables, and constraints containing both. We obtain

$$Ay \qquad = p \tag{5.24}$$

$$Bx \leq q \tag{5.25}$$

$$Cy + Dx \leq r \tag{5.26}$$

$$x, y \text{ binary} \tag{5.27}$$

where (5.24) represents constraint (5.6), while (5.25) represents constraints (5.5), (5.15), (5.16), (5.20)-(5.23). The coupling inequality (5.26) covers constraints (5.7) and (5.8).

5.4 Specifying the uncertainty

The quality of the input data of the load planning problem differs depending on the time of planning. We consider two different planning phases: pre-loading planning and planning-while-loading. The pre-loading planning phase takes place before train loading begins. In this phase, *configurations* defining how many and which types of load units should be placed on a wagon are predefined. To this end, information on the wagons of the train and the load units that should be placed onto the train are given to the terminal operator by booking systems of the railway companies. However, both information are uncertain: Sometimes wagons of a train have a bug and can not be loaded at all; on the other hand, the information about weights or overhangs of the load units is often wrong. After the train has arrived, has been unloaded and the configurations have been set as planned, the loading phase begins. During this planning-while-loading process all information on load units and the train are revealed such that there is no uncertainty left. The configurations cannot be changed any more. This would take too much time because a worker would have to walk around the entire train which may have a length of up to 700 meters. If the assignment of load units to wagons is not feasible (e.g., weight restrictions or balancing constraints are not met), the assignment of load units to the train may be (slightly) updated, or some load units may be left back and not be loaded on the train. So we consider the pre-loading planning with uncertainty and the freedom to change configurations as well as the planning-while-loading phase without uncertainty but also without the possibility to change wagon configurations.

As described above, almost inevitably the parameters used during problem planning will differ from those that come up during operation. Since feasibility of the resulting load distribution is the most crucial constraint and has to be satisfied no matter what happens, we follow a robust optimization approach. In order to set up a robust optimization approach for load planning we first have to identify which parameters are uncertain and which values they may take. This is done in the following by specifying *uncertainty sets.* Note that in practice nearly all parameters may be uncertain, but the most relevant ones with the highest possibility of deviations are the following:

1. The lengths of the overhangs of load units differ sometimes from the ones announced by the booking system. Hence, we assume that the parameter u_i for each load unit i varies in an interval $[u_i^{min}, u_i^{max}]$, which results in the following interval scenario set:
$$\mathcal{U}_1 = \{u \in \mathbb{R}^n : u^{min} \leq u \leq u^{max}\}.$$

2. Often the weights g_i of the load units differ, which may be modeled again by an interval scenario set given as
$$\mathcal{U}_2 = \{g \in \mathbb{R}^n : g^{min} \leq g \leq g^{max}\}.$$
Since it is very unlikely that the weights of all load units vary we also consider the situation were only a limited number $\Gamma \in \{0, \ldots, n\}$ of load unit weights may differ from the expected weights (see Section 5.1.2 for controlled robustness). If we denote by $\tilde{g}$ the expected (*nominal*) weights, the uncertainty set
$$\begin{aligned}\mathcal{U}_2^\Gamma = \{g \in \mathbb{R}^n : \exists I \subseteq \mathcal{N}, |I| \leq \Gamma,\\ g_i = \tilde{g}_i \text{ for } i \notin I,\ g_i^{min} \leq g_i \leq g_i^{max} \text{ for } i \in I\}\end{aligned}$$
says that the weights of at most Γ load units differ in their given intervals. Note that for $\Gamma = n$ the set $\mathcal{U}_2^\Gamma$ equals $\mathcal{U}_2$.

   For the computational results we will analyze two variants of $\mathcal{U}_2^\Gamma$. Normally we assume that the weights g_i of the load units are distributed around the nominal values. In practice, sometimes also the extreme case occurs that a load unit which originally has been booked as empty, is loaded. If $\overline{g}_i$ denotes the maximum feasible payload of load unit i, we model this situation as a special case of $\mathcal{U}_2^\Gamma$ with $g_i^{min} := \tilde{g}_i$ (a load unit does not become lighter) and $g_i^{max} := \overline{g}_i$:

$$\overline{\mathcal{U}}_2^\Gamma := \{g \in \mathbb{R}^n : \exists I \subseteq \mathcal{N}, |I| \leq \Gamma, \\ g_i = \tilde{g}_i \text{ for } i \notin I,\ \tilde{g}_i \leq g_i \leq \overline{g}_i \text{ for } i \in I\}.$$

Note that $\overline{g}_i$ does not depend on the nominal weight $\tilde{g}_i$ since it is mainly influenced by the size of the load unit. In principle such deviations might happen to any load unit.

3. Sometimes a wagon has some kind of bug and as a consequence no load unit can be assigned to it. This uncertainty is not reflected in the problem's parameters used in (5.5)-(5.19). We hence introduce a *bug parameter* $f_j \in \{0,1\}$ which indicates that wagon j has a bug when f_j is zero and change constraint (5.7) to

$$\sum_{i \in \mathcal{N}_t} x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \alpha_{tks} \cdot f_j \cdot y_{jk} \leq 0 \qquad (j \in \mathcal{M}, s \in \mathcal{P}_{c(j)}, t \in \mathcal{T}). \tag{5.28}$$

This ensures that no load units are assigned to wagon j if it has a bug. Similar to the uncertainty $\mathcal{U}_2$, it might be unrealistic to assume that all wagons have a bug at the same time. Hence, we define the uncertainty set as

$$\mathcal{U}_3^\Gamma = \{f \in \{0,1\}^m : \sum_{j \in \mathcal{M}} f_j \geq m - \Gamma\}$$

for a parameter $\Gamma \in \{0, \dots, m\}$ saying that at most Γ wagons may have a bug. Such an uncertainty set is also known as *failure set.*

Note that for uncertainties $\mathcal{U}_2^\Gamma$ and $\mathcal{U}_3^\Gamma$ the case $\Gamma = 0$ corresponds to the situation where no parameter is uncertain. These cases are equivalent to the nominal problem where robustness is not considered. We do not study an uncertainty set $\mathcal{U}_1^\Gamma$ because the uncertainty in parameter u only affects constraints concerning individual load unit assignments, but no groups of load unit assignments. So for $\Gamma > 0$ we have $\mathcal{U}_1^\Gamma = \mathcal{U}_1$. For the uncertainty $\mathcal{U}_3^\Gamma$ the case $\Gamma = m$ corresponds to the situation that all wagons may have a bug (according to our previous definitions this could also be notated as $\mathcal{U}_3$). This set is not meaningful, since it contains the worst-case scenario $f_j = 0$ for all $j \in \mathcal{M}$, (i.e. all wagons have a bug) which does not allow any assignment of load units.

Note that all uncertainties above only affect the *constraints* of the load planning problem, not the *objective.* This reflects the fact that in every scenario feasibility is the main priority for the decision maker here; however, maximizing the worst-case performance of a solution would be a natural extension and a promising line of further research. In that case, we could write the uncertain objective as an additional constraint and extend the following discussion of worst-cases.

That we do not consider uncertainty in the objective means for e.g. strict robustness that we consider an objective of the form $\max f(x, \tilde{\xi})$ which is the objective of the nominal scenario.

Example 5.5. To illustrate the uncertainty sets we apply some of them to the load planning problem of Example 4.1.
Considering uncertainty set $\mathcal{U}_1$, if the overhangs of load unit one and two are of the interval $[100, 200]$ and the overhangs for length-types 1 and 2 are 100 and 200 respectively, the solution described in Table 4.2 is feasible for all scenarios. If we consider $\mathcal{U}_2$ and weights for load unit two from the interval $[23, 25]$, the solution of Table 4.2 is no longer feasible for all scenarios (as the maximum slot payload is 24). □

Having specified different types of uncertainties for the load planning problem, we will in the following write $LP(\xi), \xi \in \mathcal{U}$ to indicate that the problem depends on the unknown parameters $\xi \in \mathcal{U}$. As an example, $LP(g), g \in \mathcal{U}_2$ indicates that the weights g are uncertain and stem from the uncertainty set $\mathcal{U}_2$. In our case the nominal scenario $\tilde{u} \in \mathcal{U}_1$ contains the overhangs announced in the booking system, and the nominal scenario for $\mathcal{U}_2$ as well as for $\mathcal{U}_2^\Gamma$ is given by the nominal weight values $\tilde{g}_i$. The nominal scenario for $\mathcal{U}_3^\Gamma$ assumes that no wagon has a bug, i.e. $\tilde{f}_j = 1$ for all $j \in \mathcal{M}$. Solving the nominal problem hence means to use the nominal parameters in the formulation (LP). The result is called *nominal solution.*

The uncertainty $\mathcal{U}_1$ influences the coefficients of the matrix D in (5.26). We denote this by writing $D(u), u \in \mathcal{U}_1$, to make clear that D is actually a function that maps scenarios to coefficients. Analogously, we write $B(g)$ for $g \in \mathcal{U}_2$ and $C(f)$ for $f \in \mathcal{U}_3^\Gamma$.

5.5 Strictly robust load planning

In strictly robust load planning (see Section 5.1.1 for strict robustness), the goal is to hedge against *all* possible scenarios by requiring that a solution (x, y) is feasible for $LP(\xi)$ for all $\xi \in \mathcal{U}$. The resulting robust problems depend on the type of uncertainty we consider. We denote the strictly robust problem for uncertainty set $\mathcal{U}_i$ by (SR_i) and the set of strictly robust solutions by F_{SR_i}.

Since a configuration once implemented cannot be changed any more during the loading phase and there might even not be enough time to communicate changes in the load unit assignment to the human crane operators, the once determined variables have to be feasible for all scenarios in the uncertainty set. Hence, in this situation the load planning problem is a typical application for strictly robust optimization.

5.5.1 Uncertainty $\mathcal{U}_1$

If we apply the first type of uncertainty to the problem, only the matrix D in constraints (5.26) is affected. Hence, we have to find (x, y) maximizing $obj(x, y)$ such that

$$Ay \qquad = p \tag{5.29}$$

$$Bx \leq q \tag{5.30}$$

$$Cy + D(u)x \leq r \quad \forall u \in \mathcal{U}_1 \tag{5.31}$$

$$x, y \text{ binary.} \tag{5.32}$$

As all variables x_{ijs} are non-negative, the worst-case for the inequality (5.8) can be directly calculated by using the respective upper bounds of the uncertainty set $\mathcal{U}_1$.

Lemma 5.2. *The strictly robust load planning problem with uncertainty set $\mathcal{U}_1$ is equivalent to the following MIP:*

$$\begin{aligned}(SR_1) \quad & \max \mathit{obj}(x,y) \\ & s.t. \quad \sum_{i \in \mathcal{N}} u_i^{max} \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} \leq 0 \qquad (j \in \mathcal{M}, s \in \mathcal{P}_{c(j)}) \\ & \qquad (5.5) - (5.7), (5.9) - (5.19)\end{aligned}$$

Proof. Let F be the set of feasible solutions for (SR_1), i.e.,

$$\begin{aligned}F = \{(x,y) : \; & \textstyle\sum_{i \in \mathcal{N}} u_i^{max} \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} \leq 0 \; \forall j \in \mathcal{M}, s \in \mathcal{P}_{c(j)}, \\ & (5.5) - (5.7), (5.9) - (5.19)\}.\end{aligned}$$

We show that F is equal to the set F_{SR_1} of strictly robust solutions w.r.t. $\mathcal{U}_1$. Due to $u^{max} \in \mathcal{U}_1$ we have $F \subseteq F_{SR_1}$. Now let $(x,y) \in F_{SR_1}$ and assume there are $\hat{u} \in \mathcal{U}_1$ and j, s such that $\sum_{i \in \mathcal{N}} \hat{u}_i \cdot x_{ijs} - \sum_{k \in \mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} > 0$. As $x \geq 0$, we have $\sum_{i \in \mathcal{N}} \hat{u}_i \cdot x_{ijs} \leq \sum_{i \in \mathcal{N}} u_i^{max} \cdot x_{ijs}$, which is a contradiction. Therefore, $F_{SR_1} \subseteq F$ and hence $F = F_{SR_1}$. □

Lemma 5.2 shows that (SR_1) is again a load planning problem, where the overhangs are replaced by their respective worst-case values. Therefore, this problem can be solved as efficiently as the nominal problem.

5.5.2 Uncertainty $\mathcal{U}_2$

As the uncertainty set $\mathcal{U}_2$ affects constraints (5.25), the problem we consider here consists of finding a solution (x, y) maximizing $\mathit{obj}(x, y)$ such that

$$Ay \qquad = p \tag{5.33}$$

$$B(g)x \leq q \quad \forall g \in \mathcal{U}_2 \tag{5.34}$$

$$Cy + \quad Dx \leq r \tag{5.35}$$

$$x, y \text{ binary.} \tag{5.36}$$

For $\mathcal{U}_1$ it was straightforward to see that the worst-case occurs if u_i takes the value u_i^{max}. Now we look at $\mathcal{U}_2$: For constraints (5.15) and (5.16) we see that $g_i = g_i^{max}$ is the worst-case. However, this is not so clear when looking at constraints (5.20)-(5.23).

Since all variables x_{ijs} and the parameters d_τ are non-negative, for the worst-case scenario we have to distinguish summands where $d_{c(j)} - 4e_{c(j),s}$ is positive and negative, respectively. As described above, if $d_{c(j)} - 4e_{c(j),s}$ is negative, g_i has to be replaced by g_i^{min};

otherwise g_i has to be replaced by g_i^{max}. For an IP-formulation we define new parameters $\hat{g}^l_{i\tau s} \in \mathbb{R}$ for $i \in \mathcal{N}, \tau \in \mathcal{W}, s \in \mathcal{P}_\tau$ to replace g_i in constraints (5.20) where

$$\hat{g}^l_{i\tau s} = \begin{cases} g_i^{min}, & \text{if } d_\tau - 4e_{\tau s} < 0 \\ g_i^{max}, & \text{otherwise.} \end{cases}$$

In constraints (5.21) for the worst-case scenario we replace g_i by $\hat{g}^r_{i\tau s} \in \mathbb{R}$ for $i \in \mathcal{N}, \tau \in \mathcal{W}, s \in \mathcal{P}_\tau$ with

$$\hat{g}^r_{i\tau s} = \begin{cases} g_i^{min}, & \text{if } 4e_{\tau s} - 3d_\tau < 0 \\ g_i^{max}, & \text{otherwise.} \end{cases}$$

The inequalities used to determine $\hat{g}^l_{i\tau s}$ and $\hat{g}^r_{i\tau s}$ can also be interpreted geometrically: If we want to maximize the left-hand side of constraints (5.20), for all wagons j and slots s where the distance from the attachment of bogie A (see Figure 4.2) to the center of mass of the load unit is less than a quarter of the distance in between the bogie attachments ($e_{c(j),s} < \frac{d_{c(j)}}{4}$), the weights g_i of the assigned load units i have to be chosen as g_i^{max}. All other load unit weights have to be chosen as g_i^{min}.

Also, if we want to maximize the violation of constraints (5.21), for all wagons j and slots s with $e_{c(j),s} \geq \frac{3 \cdot d_{c(j)}}{4}$ the weights g_i have to be chosen as g_i^{max}. These are the load units where the distance from the attachment of bogie A to the center of mass is larger than $\frac{3 \cdot d_{c(j)}}{4}$ (which is equivalent to the fact that the distance from the attachment of bogie B to the center of mass is smaller than $\frac{d_{c(j)}}{4}$). Again all other load unit weights have to be chosen as g_i^{min}.

As for all wagons we know that $e_{c(j),s}$ as well as $d_{c(j)} - e_{c(j),s}$ are positive, for constraints (5.22) and (5.23) again the maximum weights g_i^{max} are the worst-case.

By using constraints (5.20) and (5.21) with the above described replacement of g_i we can now formulate an integer linear program for the strictly robust problem.

Lemma 5.3. *The strictly robust load planning problem with uncertainty set $\mathcal{U}_2$ is equivalent to the following IP:*

$$(SR_2) \quad \max obj(x, y)$$

s.t.

$$\sum_{i \in \mathcal{N}} g_i^{max} \cdot x_{ijs} \leq \delta_{c(j),s} \qquad (j \in \mathcal{M}, s \in \mathcal{P}_{c(j)}) \tag{5.37}$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{P}_{c(j)}} g_i^{max} \cdot x_{ijs} \leq G \tag{5.38}$$

$$\sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} \hat{g}^l_{i,c(j),s} \cdot \frac{d_{c(j)} - 4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \leq t_{c(j)} \qquad (j \in \mathcal{M}) \tag{5.39}$$

$$\sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} \hat{g}^r_{i,c(j),s} \cdot \frac{4e_{c(j),s} - 3d_{c(j)}}{d_{c(j)}} \cdot x_{ijs} \leq t_{c(j)} \qquad (j \in \mathcal{M}) \tag{5.40}$$

$$\sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} g_i^{max} \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \leq \gamma_{c(j)} - \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{5.41}$$

$$\sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} g_i^{max} \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \leq \gamma_{c(j)} - \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{5.42}$$

$$(5.5) - (5.8), (5.17) - (5.18)$$

Proof. Let F be the set of feasible solutions for (SR_2). We show that F is equal to the set F_{SR_2} of strictly robust solutions w.r.t. $\mathcal{U}_2$. To show that $F \subseteq F_{SR_2}$ holds, we give a scenario of $\mathcal{U}_2$ for inequalities (5.37) to (5.42). Due to constraints (5.5) for each $i \in \mathcal{N}$ at most one of the values g_i^{min} and g_i^{max} is used for the calculation of constraints (5.39) and (5.40). So the used values for g equal a scenario of $\mathcal{U}_2$. The same holds for inequalities (5.37), (5.38), (5.41) and (5.42) where the values g_i^{max} are used for all $i \in \mathcal{N}$.

Conversely, let $(x, y) \in F_{SR_2}$ and assume there are $\hat{g} \in \mathcal{U}_2$ and j, s such that one of the inequalities (5.37), (5.38), (5.41) or (5.42) is violated. As $x \geq 0$, this would imply that $\sum_{i \in \mathcal{N}} \hat{g}_i \cdot x_{ijs} \leq \sum_{i \in \mathcal{N}} g_i^{max} \cdot x_{ijs}$, which is a contradiction. For constraints (5.39) and (5.40) we cannot choose a $\hat{g} \in \mathcal{U}_2$ that enlarges the left-hand side because the values $\hat{g}^l_{i,c(j),s}$ and $\hat{g}^r_{i,c(j),s}$ are chosen in a way to maximize the left-hand sides (since they include maximum values for positive summands and minimum values for negative ones). Therefore, $F_{SR_2} \subseteq F$ and hence $F = F_{SR_2}$. $\square$

5.5.3 Uncertainty $\mathcal{U}_2^\Gamma$

We now consider the problem of finding (x, y) maximizing $obj(x, y)$ such that

$$\begin{aligned} Ay \qquad &= p & (5.43)\\ B(g)x &\leq q \quad \forall g \in \mathcal{U}_2^\Gamma & (5.44)\\ Cy + \quad Dx &\leq r & (5.45)\\ x, y &\text{ binary.} & (5.46) \end{aligned}$$

First, we consider a single uncertain constraint of the type

$$\sum_{j=1}^{n} B_{ij}(g) x_j \leq q_i \quad \forall g \in \mathcal{U}_2^\Gamma \tag{5.47}$$

for an arbitrary but fixed row i and assume that we have variables x_j for $j = 1, \dots, n$. Similar to Bertsimas and Sim [16], we determine the robust counterpart of this constraint. Let x be fixed. Recall that $\tilde{g}$ is the nominal value of g and let g^{wc} be the worst-case value of g that maximizes the left-hand side of (5.47). Each component g_j^{wc} is either equal to g_j^{min} or g_j^{max} depending on the sign of the corresponding B_{ij}-value. We denote by

$$\beta'(x, \Gamma) = \max \left\{ \sum_{j=1}^{n} (B_{ij}(g^{wc}) - B_{ij}(\tilde{g})) x_j \alpha_j : \sum_{j=1}^{n} \alpha_j \leq \Gamma, \ \alpha \in \{0,1\}^n \right\}$$

the *worst-case* for the left-hand side of constraint (5.47). $\beta'(x,\Gamma)$ delivers the worst possible deviation for a constraint, compared to the nominal parameter when x is fixed. If we relax the integrality constraint for the variables α_j, we get

$$\beta(x,\Gamma) = \max\left\{\sum_{j=1}^{n}(B_{ij}(g^{wc}) - B_{ij}(\tilde{g}))x_j\alpha_j : \sum_{j=1}^{n}\alpha_j \leq \Gamma,\ 0 \leq \alpha \leq 1\right\}. \tag{5.48}$$

It is easy to see that $\beta'(x,\Gamma) = \beta(x,\Gamma)$.

Using formulation (5.48), we conclude that

$$\sum_{j=1}^{n} B_{ij}(\tilde{g})x_j + \beta(x,\Gamma) \leq q_i \ \Leftrightarrow\ \sum_{j=1}^{n} B_{ij}(g)x_j \leq q_i \text{ for all } g \in \mathcal{U}_2^{\Gamma}. \tag{5.49}$$

We now consider the dual problem of (5.48):

$$\begin{aligned}\beta^D(x,\Gamma) \quad = \quad & \min\Big\{\Gamma z + \sum_{j=1}^{n}\pi_j : z + \pi_j \geq (B_{ij}(g^{wc}) - B_{ij}(\tilde{g}))x_j \ \forall j = 1,\ldots,n,\\ & z, \pi_j \geq 0 \ \forall j = 1,\ldots,n\Big\},\end{aligned}$$

where z is the dual variable for constraint $\sum_{j=1}^{n}\alpha_j \leq \Gamma$ and π_j is dual to $\alpha_j \leq 1$. Using strong duality, we conclude:

Lemma 5.4. *Constraint (5.47) for fixed i and a given $x \in \{0,1\}^n$ holds if and only if there are real numbers z, $\pi_1,\ldots,\pi_n$ such that*

$$\begin{aligned}\sum_{j=1}^{n} B_{ij}(\tilde{g})x_j + \Gamma z + \sum_{j=1}^{n}\pi_j &\leq q_i\\ z + \pi_j &\geq (B_{ij}(g^{wc}) - B_{ij}(\tilde{g}))x_j \qquad && (j = 1,\ldots,n)\\ \pi_j &\geq 0 && (j = 1,\ldots,n)\\ z &\geq 0\end{aligned}$$

We are now in the position to reformulate the problem:

Theorem 5.5. *For $\Gamma \geq 1$, the strictly robust load planning problem with uncertainty set $\mathcal{U}_2^{\Gamma}$ is equivalent to*

$$(SR_2^{\Gamma}) \quad \max\ obj(x,y)$$

s.t.

$$\sum_{i\in\mathcal{N}} g_i^{max}\cdot x_{ijs} \leq \delta_{c(j),s} \qquad (j \in \mathcal{M}, s \in \mathcal{P}_{c(j)}) \tag{5.50}$$

$$\sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{M}}\sum_{s\in\mathcal{P}_{c(j)}} \tilde{g}_i\cdot x_{ijs} + \Gamma z^{(5.16)} + \sum_{i\in\mathcal{N}}\pi_i^{(5.16)} \leq G \tag{5.51}$$

$$z^{(5.16)} + \pi_i^{(5.16)} - (g_i^{max} - \tilde{g}_i) \sum_{j \in \mathcal{M}} \sum_{s \in \mathcal{P}_{c(j)}} x_{ijs} \geq 0 \qquad (i \in \mathcal{N}) \tag{5.52}$$

$$\sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} \tilde{g}_i \cdot \frac{d_{c(j)} - 4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \Gamma z_j^{(5.20)} + \sum_{i \in \mathcal{N}} \pi_{ij}^{(5.20)} \leq t_{c(j)} \qquad (j \in \mathcal{M}) \tag{5.53}$$

$$z^{(5.20)} + \pi_{ij}^{(5.20)} - (\hat{g}_{i\tau s}^{l} - \tilde{g}_i) \sum_{s \in \mathcal{P}_{c(j)}} \frac{d_{c(j)} - 4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \geq 0 \qquad (i \in \mathcal{N}, j \in \mathcal{M}) \tag{5.54}$$

$$\sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} \tilde{g}_i \cdot \frac{4e_{c(j),s} - 3d_{c(j)}}{d_{c(j)}} \cdot x_{ijs} + \Gamma z_j^{(5.21)} + \sum_{i \in \mathcal{N}} \pi_{ij}^{(5.21)} \leq t_{c(j)} \qquad (j \in \mathcal{M}) \tag{5.55}$$

$$z^{(5.21)} + \pi_{ij}^{(5.21)} - (\hat{g}_{i\tau s}^{r} - \tilde{g}_i) \sum_{s \in \mathcal{P}_{c(j)}} \frac{4e_{c(j),s} - 3d_{c(j)}}{d_{c(j)}} \cdot x_{ijs} \geq 0 \qquad (i \in \mathcal{N}, j \in \mathcal{M}) \tag{5.56}$$

$$\sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} \tilde{g}_i \cdot \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \Gamma z_j^{(5.22)} + \sum_{i \in \mathcal{N}} \pi_{ij}^{(5.22)} \leq \gamma_{c(j)} - \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{5.57}$$

$$z^{(5.22)} + \pi_{ij}^{(5.22)} - (g_i^{max} - \tilde{g}_i) \sum_{s \in \mathcal{P}_{c(j)}} \frac{d_{c(j)} - e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \geq 0 \qquad (i \in \mathcal{N}, j \in \mathcal{M}) \tag{5.58}$$

$$\sum_{i \in \mathcal{N}} \sum_{s \in \mathcal{P}_{c(j)}} \tilde{g}_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} + \Gamma z_j^{(5.23)} + \sum_{i \in \mathcal{N}} \pi_{ij}^{(5.23)} \leq \gamma_{c(j)} - \frac{t_{c(j)}}{2} \qquad (j \in \mathcal{M}) \tag{5.59}$$

$$z^{(5.23)} + \pi_{ij}^{(5.23)} - (g_i^{max} - \tilde{g}_i) \sum_{s \in \mathcal{P}_{c(j)}} \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs} \geq 0 \qquad (i \in \mathcal{N}, j \in \mathcal{M}) \tag{5.60}$$

$$(5.5) - (5.8), (5.17) - (5.18) \tag{5.61}$$

$$z^{(5.16)}, z^{(5.20)}, z^{(5.21)}, z^{(5.22)}, z^{(5.23)} \geq 0 \tag{5.62}$$

$$\pi^{(5.16)}, \pi^{(5.20)}, \pi^{(5.21)}, \pi^{(5.22)}, \pi^{(5.23)} \geq 0 \tag{5.63}$$

Proof. By applying Lemma 5.4 to the constraints (5.16) and (5.20)-(5.23), the formulation follows. Doing so is not necessary for constraints (5.15), since for each j and s at most one of the variables x_{ijs} may be equal to 1. Therefore, in (5.50) we may simply use the worst-case g^{max} for this constraint. □

5.5.4 Uncertainty $\mathcal{U}_3^\Gamma$

We now consider the problem of finding a solution (x, y) maximizing $obj(x, y)$ such that

$$Ay = p \tag{5.64}$$

$$Bx \leq q \tag{5.65}$$

$$C(f)y + Dx \leq r \quad \forall f \in \mathcal{U}_3^\Gamma \tag{5.66}$$

$$x, y \text{ binary.} \tag{5.67}$$

As it turns out, this setting is too restrictive to assign any load unit at all.

Lemma 5.6. *The strictly robust load planning problem with respect to the uncertainty set $\mathcal{U}_3^\Gamma$ for $\Gamma \geq 1$ is equivalent to the MIP*

$$\begin{aligned} (SR_3^\Gamma) \quad & \max\ obj(x,y) \\ & s.t. \quad \sum_{i\in\mathcal{N}_t} x_{ijs} \leq 0 \qquad\qquad (j\in\mathcal{M}, s\in\mathcal{P}_{c(j)}, t\in\mathcal{T}) \qquad (5.68) \\ & \qquad\quad (5.5),(5.6),(5.8)-(5.19). \end{aligned}$$

Proof. Let F be the set of feasible solutions for (SR_3^Γ). We show that F is equal to the set $F_{SR_3^\Gamma}$ of strictly robust solutions w.r.t. $\mathcal{U}_3^\Gamma$. Obviously, we have $F \subseteq F_{SR_3^\Gamma}$. On the other hand, let (x,y) be in $F_{SR_3^\Gamma}$. Assume that for a wagon j and a slot $s\in\mathcal{P}_{c(j)}$ there is a type t and a load unit $i\in\mathcal{N}_t$ such that $x_{ijs}=1$ holds. Then (x,y) is not feasible for the scenario $f_j=0$, $f_h=1$ for $h\neq j$, which is in $\mathcal{U}_3^\Gamma$. Therefore, $x=0$ for $(x,y)\in F_{SR_3^\Gamma}$ which implies $F_{SR_3^\Gamma}\subseteq F$ and hence $F=F_{SR_3^\Gamma}$. □

Lemma 5.7. *In every feasible solution to (SR_3^Γ) we must have $x=0$, meaning that no load unit is assigned at all.*

Proof. Let (x,y) be a feasible solution to (SR_3^Γ). From (5.68) we see that for all load units $i\in\mathcal{N}$ we have $x_{ijs}=0$ for all $j\in\mathcal{M}, s\in\mathcal{P}_{c(j)}$. □

5.6 Adjustable robust load planning

In the context of *adjustable robust load planning*, we search for a solution y, i.e. a configuration of wagons such that for every scenario ξ from the uncertainty set $\mathcal{U}$ we can find feasible values for x, i.e., the assignment of load units to wagon slots, such that (x,y) is feasible for $LP(\xi)$. Such a solution is called *adjustable robust.* The aim of the adjustable robust counterpart (aR) is to find an adjustable robust solution that performs best in its worst-case.

For load planning, this approach is motivated by the two planning phases of pre-loading and planning-while-loading as described in Section 5.4. As it is time-consuming to change the configuration of a wagon (basically, a worker has to go around the whole train), but easily accomplished to put a load unit onto another wagon, we assume that the assignment of load units to slots can be done when the realized scenario is revealed.

In general, adjustable robust counterparts are computationally intractable in the sense that they consist of infinitely many constraints and variables, and there are no generally applicable methods available to reduce these counterparts to finite problems. In Bertsimas et al. [15], circumstances are identified when the strictly robust solution is still an approximation to the adjustable robust problem. In this section, we give a finite counterpart in the case of $\mathcal{U}_1$ and propose a scenario sampling approach for the other uncertainty sets.

5.6.1 Uncertainty $\mathcal{U}_1$

Assuming that a wagon configuration y is fixed and satisfies $Ay = p$, the set of possible load unit assignments in scenario $u \in \mathcal{U}_1$ is given as

$$\mathcal{F}(y,u) := \{x \in \{0,1\}^n : Bx \le q, Cy + D(u)x \le r, \} .$$

The adjustable robust load planning problem with respect to the uncertainty $\mathcal{U}_1$ is then stated as follows:

$$\begin{aligned} (aR_1) \quad & \max_{y:Ay=p} \min_{u \in \mathcal{U}_1} \max_{x \in \mathcal{F}(y,u)} obj(x,y) \\ & \text{s.t. } \forall u \in \mathcal{U}_1 \; \exists x \; : \; x \in \mathcal{F}(y,u) \end{aligned}$$

Solving the adjustable robust counterpart is in general harder than solving the strictly robust counterpart (see Ben-Tal et al. [12]). The direct formulation of the adjustable robust counterpart (aR_1) is a quantified integer program (solving this kind of problems is in general PSPACE-complete, see Subramani [84]). Even the problem of checking if a given solution y has an objective value of at least K leads to figuring out if

$$\{x \in \{0,1\}^n : obj(x,y) \ge K \text{ and } Bx \le q \text{ and } D(u)x \le r - Cy \text{ for all } u \in \mathcal{U}\}$$

is non-empty, and checking if a polyhedron contains an integer point is in general NP-complete (see Garey and Johnson [47]). However, for the case of uncertainties in load unit overhangs, we show that the problem can be reformulated as an integer linear program again.

To analyze (aR_1), we introduce the following notation:

Definition 5.8. *A scenario $u^{wc} \in \mathcal{U}_1$ fulfilling*

$$\mathcal{F}(y, u^{wc}) \subseteq \mathcal{F}(y,u) \quad \forall u \in \mathcal{U}_1, \; \forall y, \tag{5.69}$$

is called a worst-case scenario.

A worst-case scenario can be considered as a single parameter setting that dominates all other possible scenarios from the given uncertainty set. In Goerigk and Schöbel [51] it is shown that the existence of a worst-case scenario yields the following implications:

Theorem 5.9. *Let $u^{wc} \in \mathcal{U}_1$ be a worst-case scenario. Then, the set of feasible solutions to (aR_1) equals the projection of feasible solutions to (SR_1) onto the y-variables. Furthermore, the optimal objective values of (aR_1) and (SR_1) coincide.*

As has already been demonstrated in the proof of Lemma 5.2, this is indeed the case for the considered problem:

Lemma 5.10. *u^{max} is a worst-case scenario to (aR_1).*

We can therefore conclude that instead of considering all infinitely many scenarios $u \in \mathcal{U}_1$ for (aR_1), we only need to consider u^{max}. Therefore, solving (aR_1) simply amounts to solving (SR_1).

Corollary 5.11. *Let (x^*, y^*) be an optimal solution to (SR_1). Then y^* is an optimal solution to (aR_1).*

5.6.2 Uncertainties $\mathcal{U}_2$, $\mathcal{U}_2^\Gamma$ and $\mathcal{U}_3^\Gamma$

Analogously to the setting in Section 5.6.1, for uncertainty set $\mathcal{U}_2$ the problem we consider here is the following:

$$\begin{aligned}(aR_2) \quad & \max_{y} \min_{g\in\mathcal{U}_2} \max_{x\in\mathcal{F}(y,B(g))} obj(x,y)\\ & \text{s.t. } \forall g \in \mathcal{U}_2\ \exists x^{B(g)}\ :\ x^{B(g)} \in \mathcal{F}(y, B(g)),\end{aligned}$$

where

$$\mathcal{F}(y, B(g)) := \{x : Ay = p, B(g)x \le q, Cy + Dx \le r\}\,.$$

Unfortunately, as has been discussed in Section 5.5.2, there is no single worst-case scenario as in $\mathcal{U}_1$. From an applied point of view, there is no set of load unit weights that maximally disturbs the balance constraints of all wagon assignments at the same time; from a theoretical point of view, each of the constraints (5.20) and (5.21) has a different worst-case.

Instead of reducing the infinite scenario set $\mathcal{U}_2$ to a single scenario, we therefore consider the following heuristic approach: We generate a finite subset $\mathcal{G} = \{g^1, \ldots, g^N\}$ of scenarios from $\mathcal{U}_2$ and solve

$$\begin{aligned}(aR_2)^{\mathcal{G}} \quad & \max z & \\ \text{s.t.} \quad & obj(x^i, y) \ge z & (i = 1, \ldots, N)\\ & Ay = p & \\ & B(g^i)x^i \le q & (i = 1, \ldots, N)\\ & Cy + Dx^i \le r & (i = 1, \ldots, N)\\ & x^i, y \text{ binary} & (i = 1, \ldots, N)\end{aligned}$$

i.e., we find solutions (y, x^i) to every scenario $g^i \in \mathcal{G}$.

As the number of restrictions in $(aR_2)^{\mathcal{G}}$ grows for larger sets $\mathcal{G}$, the objective value decreases. If $\mathcal{G}^1 \subseteq \mathcal{G}^2 \subseteq \mathcal{U}_2$ are finite sets, then the optimal objective value of $(aR_2)^{\mathcal{G}^1}$ is not smaller than the optimal objective value of $(aR_2)^{\mathcal{G}^2}$. In fact, as $\mathcal{U}_2$ is the largest set of scenarios that can be considered, the objective value of (aR_2) is overestimated by this heuristic approach, i.e., for any finite set $\mathcal{G} \subseteq \mathcal{U}_2$, the optimal objective value for $(aR_2)^{\mathcal{G}}$ is not smaller than the optimal objective value for (aR_2). On the other hand, the solution gets more robust if more scenarios are considered. Scenarios may be drawn uniformly from the uncertainty set, or systematically in order to represent a broad scope of possible scenarios. If properties about the scenarios maximizing the worst case of a solution are known, this information may be used. In our experiments we randomly chose extreme points of the uncertainty set.

Similarly, the intractability of (aR_2^Γ) and (aR_3^Γ) cannot be resolved by using a single, dominating scenario. Instead, the heuristic approach of using a finite subset of scenarios is applicable and the above results hold analogously.

5.7 Computational experiments

In this section we report computational results for the robust load planning problem. We evaluate the different concepts according to their nominal quality (i.e. the objective value in case that all parameters keep the values that were already assumed for planning) and according to their robustness. To this end, we introduce a measure for robustness in Section 5.7.1. Then we analyze the concept of strict robustness by comparing the two uncertainty sets $\mathcal{U}_1$ and $\mathcal{U}_2$ in Section 5.7.2 and investigating the effects the parameter Γ has when it is introduced in the uncertainty sets $\mathcal{U}_2^\Gamma$ and $\overline{\mathcal{U}}_2^\Gamma$ in Section 5.7.3. Finally, we discuss the concept of adjustable robustness in Section 5.7.4.

Note that all values are mean values for 20 load planning instances with n between 37 and 85 load units, m from 24 to 38 wagons and an overall number of configurations from 30 to 61 per instance. The test instances are motivated by real-world settings and are described in more detail in Section 4.4. The runtimes were restricted to 20 minutes for each instance. We used CPLEX 12.2 on a computer with an Intel Core 2 Duo, two cores with 2.2 GHz, 4 GB RAM and Mac OS X 10.7.2.

5.7.1 Measuring the robustness of a solution

In this section we specify how the interval borders of the uncertainty sets are chosen in our experiments. Furthermore, we introduce a robustness measure of a solution and show how it can be computed for the different uncertainty sets.

Different ways to measure the degree of robustness of a solution have been introduced in recent literature, e.g., by considering the degree of constraint violation (cf. Ben-Tal and Nemirovski [14] and Fischetti and Monaci [44]), or the effort to update a solution, see Liebchen et al. [70]. However, these methods assume an uncertainty set of fixed size, but even the size of the uncertainty set is usually not known exactly in practice. A decision maker would rather prefer to know how much robustness he will gain, if he increases the size of the uncertainty he hedges against in the robust counterpart. To this end, we propose a new method that yields the amount of "implicit" robustness a solution contains. A similar approach has been applied to robust facility location in Carrizosa and Nickel [32].

We define the robustness of a given solution (x, y) for a given instance as the size of the largest uncertainty set for which it is still strictly robust. To this end, we have to specify the size of our uncertainty sets $\mathcal{U}$. This can be done in different ways. For uncertainty sets $\mathcal{U}_1, \mathcal{U}_2$ and $\mathcal{U}_2^\Gamma$ we assume a percental deviation of σ around the nominal values which specifies the size of the uncertainty sets. For example, for the uncertainty set $\mathcal{U}_1$ and a deviation of σ this means

$$\mathcal{U}_1 = \mathcal{U}_1(\sigma) = \{u \in \mathbb{R}^n : \max\{0, (1-\sigma)\cdot\tilde{u}_i\} \le u_i \le (1+\sigma)\cdot\tilde{u}_i\}\,.$$

Due to the maximum-expression for the left border we assure that neither the length nor the weight parameters are negative (this does not make any sense in practice). Analogously, we may write $\mathcal{U}_2(\sigma)$ and $\mathcal{U}_2^\Gamma(\sigma)$ in order to emphasize the dependence of $\mathcal{U}$ on σ.

Similarly, we define the size of the uncertainty sets $\mathcal{U}_2^\Gamma$ and $\mathcal{U}_3^\Gamma$ as Γ, i.e., as the number of elements which may deviate.

In order to determine *how robust* a given solution for a given instance of the load planning problem is we use the following definition which relies on the size of the uncertainty sets measured by a scalar parameter θ. Note that θ may be σ or Γ.

Definition 5.12. *Let a solution (x, y) to an uncertain load planning instance $I = (LP(\xi), \xi \in \mathcal{U})$ be given and assume that the uncertainty set $\mathcal{U} = \mathcal{U}(\theta)$ is dependent on some $\theta \in \mathbb{R}$. Then the robustness of (x, y) is defined as*

$$rob^\theta(I, x, y) = \max\{\theta : (x, y) \textit{ is feasible for all } \xi \in \mathcal{U}(\theta)\},$$

i.e. $rob^\theta(I, x, y)$ is the maximal size of the uncertainty set for which (x, y) is still strictly robust for instance I.

As the considered instance I is usually clear from the context, in the following we will use the simplified notation $rob^\theta(x, y) := rob^\theta(I, x, y)$. This definition can be applied to the uncertainty sets $\mathcal{U}_1 = \mathcal{U}_1(\sigma)$, $\mathcal{U}_2 = \mathcal{U}_2(\sigma)$ and $\mathcal{U}_2^\Gamma = \mathcal{U}_2^\Gamma(\sigma)$ resulting in a robustness function $rob^\sigma(x, y)$. Analogously we may apply the definition to $\mathcal{U}_2^\Gamma$ and $\mathcal{U}_3^\Gamma$ and obtain the robustness function $rob^\Gamma(x, y)$.

In the following we will show how to calculate the robustness of a given solution with respect to $\mathcal{U}_1$ and $\mathcal{U}_2$ according to Definition 5.12. For a fixed solution vector $x = (x_1, \ldots, x_n)$ we consider constraints of the type

$$\sum_{j=1}^{n} B_{ij}(\tilde{h}) x_j \leq q_i \tag{5.70}$$

for an arbitrary row i where $\tilde{h}$ represents either the nominal overhangs $\tilde{u}$ or the nominal weights $\tilde{g}$. To determine the worst-case that maximizes the left-hand side of (5.70) we introduce

$$\varrho_{ij} := \begin{cases} -1, & \text{if } B_{ij} < 0 \\ 1, & \text{otherwise.} \end{cases}$$

For constraint i let r_i be the maximum percentage by which the values may deviate from the nominal values $\tilde{h}$ such that the constraint is still satisfied:

$$\sum_{j=1}^{n} (1 + \varrho_{ij} \cdot r_i) B_{ij}(\tilde{h}) x_j \leq q_i. \tag{5.71}$$

By resolving inequality (5.71) for r_i we get:

$$r_i = \frac{q_i - \sum_{j=1}^{n} B_{ij}(\tilde{h}) x_j}{\sum_{j=1}^{n} \varrho_{ij} \cdot B_{ij}(\tilde{h}) x_j} \tag{5.72}$$

where the numerator is the slack of the constraint for the nominal values $\tilde{h}$ and the denominator is the sum of the contributions to the left-hand side of the vector $\tilde{h}$.

By taking the minimum of the r_i-values for all constraints i, we obtain the overall robustness of a given solution. Note that we assume a deviation of all elements at the same time. Now we will show how we can apply the general robustness equation (5.72) to uncertainty sets $\mathcal{U}_1$ and $\mathcal{U}_2$.

For the uncertainty set $\mathcal{U}_1$ we introduce $r_{js}^{(5.8)}(x,y)$ for all pairs $(j,s) \in \mathcal{M} \times \mathcal{P}_{c(j)}$ of slots s on wagon j with $\sum_{k\in\mathcal{K}_{c(j)}} \beta_{ks} > 0$ (i.e. in principle a load unit can be assigned to this slot):

$$r_{js}^{(5.8)}(x,y) = \begin{cases} \infty, & \text{if } \sum_{k\in\mathcal{K}_{c(j)}} \beta_{ks} \cdot y_{jk} \geq \sum_{i\in\mathcal{N}} x_{ijs} \cdot \overline{u}_i \\ \frac{\sum_{k\in\mathcal{K}_{c(j)}} \beta_{ks}\cdot y_{jk} - \sum_{i\in\mathcal{N}} x_{ijs}\cdot \tilde{u}_i}{\sum_{i\in\mathcal{N}} x_{ijs}\cdot\tilde{u}_i}, & \text{otherwise} \end{cases}$$

where for every load unit i the value $\overline{u}_i$ denotes the maximum real-world overhang of the corresponding load unit type. The motivation of the ∞-definition is that we assume that load units always belong to a real-world class. So if a load unit is booked as one with 20 ft corner casting distance, we assume that at most a 26 ft load unit will be delivered because no load units with greater overhang exist in the European setting we analyzed. We assume that if a load unit with 20 ft corner casting distance is assigned to a slot that allows to put a 26 ft load unit on top, there are no problems with feasibility for any scenario. Note that this assumption is not affected by the minimal overhang of the load unit. We do not assume any deviation for trailers because they do not have an overhang in the same sense as containers or swap bodies. So assignments of trailers are assumed to be feasible for any scenario. Note that the value $r_{js}^{(5.8)}(x,y)$ is also ∞ if no load unit is assigned to slot s on wagon j.

Lemma 5.13. *For the uncertainty set $\mathcal{U}_1$ with parameter σ we have*

$$rob^{\sigma}(x,y) = \min_{(j,s)\in\mathcal{M}\times\mathcal{P}_{c(j)}} r_{js}^{(5.8)}(x,y).$$

For the uncertainty set $\mathcal{U}_2$ we proceed as follows. For constraints (5.37) we introduce for all pairs $(j,s) \in \mathcal{M} \times \mathcal{P}_{c(j)}$ of slots s on wagon j the values

$$r_{js}^{(5.37)}(x,y) = \frac{\delta_{c(j),s} - \sum_{i\in\mathcal{N}} \tilde{g}_i \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \tilde{g}_i \cdot x_{ijs}}.$$

For constraints (5.38) we introduce

$$r^{(5.38)}(x,y) = \frac{G - \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{P}_{c(j)}} \tilde{g}_i \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{M}} \sum_{s\in\mathcal{P}_{c(j)}} \tilde{g}_i \cdot x_{ijs}}.$$

For constraints (5.39) let

$$\varrho_{i\tau s}^{(5.39)} := \begin{cases} -1, & \text{if } d_\tau - 4e_{\tau s} < 0 \\ 1, & \text{otherwise.} \end{cases}$$

To calculate the robustness w.r.t. constraints (5.39), for every wagon $j \in \mathcal{M}$ we determine the value

$$r_j^{(5.39)}(x,y) = \frac{t_{c(j)} - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{P}_{c(j)}} \tilde{g}_i \cdot \frac{d_{c(j)}-4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{P}_{c(j)}} \varrho^{(5.39)}_{i,c(j),s} \cdot \tilde{g}_i \cdot \frac{d_{c(j)}-4e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}.$$

Analogously for constraints (5.40) let

$$\varrho^{(5.40)}_{i\tau s} := \begin{cases} -1, & \text{if } 4e_{\tau s} - 3d_\tau < 0 \\ 1, & \text{otherwise.} \end{cases}$$

For every wagon $j \in \mathcal{M}$ we define

$$r_j^{(5.40)}(x,y) = \frac{t_{c(j)} - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{P}_{c(j)}} \tilde{g}_i \cdot \frac{4e_{\tau s}-3d_\tau}{d_{c(j)}} \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{P}_{c(j)}} \varrho^{(5.40)}_{i,c(j),s} \cdot \tilde{g}_i \cdot \frac{4e_{\tau s}-3d_\tau}{d_{c(j)}} \cdot x_{ijs}}.$$

For constraints (5.41)-(5.42) we do not have to introduce ϱ-values since all $e_{c(j),s}$ as well as $d_{c(j)} - e_{c(j),s}$ are positive. For every wagon $j \in \mathcal{M}$ let

$$r_j^{(5.41)}(x,y) = \frac{\gamma_{c(j)} - \frac{t_{c(j)}}{2} - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{P}_{c(j)}} \tilde{g}_i \cdot \frac{d_{c(j)}-e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{P}_{c(j)}} \tilde{g}_i \cdot \frac{d_{c(j)}-e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}$$

and

$$r_j^{(5.42)}(x,y) = \frac{\gamma_{c(j)} - \frac{t_{c(j)}}{2} - \sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{P}_{c(j)}} \tilde{g}_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}{\sum_{i\in\mathcal{N}} \sum_{s\in\mathcal{P}_{c(j)}} \tilde{g}_i \cdot \frac{e_{c(j),s}}{d_{c(j)}} \cdot x_{ijs}}.$$

Lemma 5.14. *For the uncertainty set $\mathcal{U}_2$ with parameter σ we have*

$$rob^\sigma(x,y) = \min_{(j,s)\in\mathcal{M}\times\mathcal{S}_{c(j)}} \Big\{ r_{js}^{(5.37)}(x,y), r^{(5.38)}(x,y), r_j^{(5.39)}(x,y), r_j^{(5.40)}(x,y), \\ r_j^{(5.41)}(x,y), r_j^{(5.42)}(x,y) \Big\}.$$

For uncertainty set $\mathcal{U}_2^\Gamma$ and its variant $\overline{\mathcal{U}}_2^\Gamma$ we determine the robustness of a solution as follows. For $\mathcal{U}_2^\Gamma$ we keep the parameter σ fixed and compute the largest Γ for which a given solution is still feasible. To do this, again we have to consider the robustness according to the different constraints as for uncertainty set $\mathcal{U}_2$ (see above). For each constraint of the form

$$\sum_{j=1}^{n} B_{ij}(g)x_j \le q_i \quad \forall g \in \mathcal{U}_2^\Gamma \cup \overline{\mathcal{U}}_2^\Gamma \tag{5.73}$$

we choose the vector g^{wc} which is the worst-case in the considered constraint for the uncertainty set $\mathcal{U}_2^\Gamma$ or $\overline{\mathcal{U}}_2^\Gamma$. According to Lemma 5.4 we have to solve the following MIP to calculate the maximum robustness Γ:

$$\max \Gamma$$

$$\begin{aligned} \text{s.t.} \ \sum_{j=1}^{n} B_{ij}(\tilde{g})x_j + \Gamma z + \sum_{j=1}^{n} \pi_j &\leq q_i \\ z + \pi_j - (B_{ij}(g^{wc}) - B_{ij}(\tilde{g}))x_j &\geq 0 && (j = 1, \dots, n) \\ \pi_j &\geq 0 && (j = 1, \dots, n) \\ z &\geq 0 \\ \Gamma &\in \mathbb{N} \end{aligned}$$

To determine Γ we do not have to solve the stated MIP, but can determine Γ in a direct way: For each constraint we calculate its left-hand side for the nominal scenario where no element deviates. Afterwards, we compute the change of the left-hand side caused by choosing each load unit individually to be uncertain (which means that the load unit is in the set I). By ordering these changes of the left-hand side in a decreasing order it is easy to compute the robustness rob_i^Γ with respect to the analyzed constraint i. We have to add the ordered changes to the left-hand side as long as the constraint stays feasible. The number of values that could be added without violating the constraint i is the robustness rob_i^Γ. If we can sum up all changes, rob_i^Γ is not limited according to constraint i. The overall robustness is equal to the minimum of the rob_i^Γ-values over all constraints i.

Conversely, for $\mathcal{U}_2^\Gamma$ we can fix the number Γ of deviating elements and can compute the robustness measure $rob^\sigma(x, y)$. To do this, for each constraint of the form (5.73) we compute the robustness as

$$\begin{aligned} &\max \sigma \\ \text{s.t.} \ \sum_{j=1}^{n} B_{ij}(\tilde{g})x_j + \sigma(\Gamma z + \sum_{j=1}^{n} \pi_j) &\leq q_i \\ z + \pi_j - |B_{ij}(\tilde{g}))x_j| &\geq 0 && (j = 1, \dots, n) \\ \pi_j &\geq 0 && (j = 1, \dots, n) \\ z, \sigma &\geq 0 \end{aligned}$$

Again, to determine σ we do not have to solve the stated linear program. By computing the Γ largest values of $|B_{ij}(\tilde{g}))x_j|$ for each constraint, we can compute the ratio between the slack of the constraint and the sum of these largest values.

Since it is not possible to calculate meaningful strict robust solutions for $\mathcal{U}_3^\Gamma$ (because only $x = 0$ is feasible), the robustness of solutions is trivial for uncertainty set $\mathcal{U}_3^\Gamma$. For $x = 0$ all load units may deviate (i.e. $rob^\Gamma(0, y) = n$), for all other solutions $rob^\Gamma(x, y) = 0$.

5.7.2 Strict robustness for the uncertainty sets $\mathcal{U}_1$ and $\mathcal{U}_2$

Setup. For each of the 20 test instances (described in Section 4.4) we calculated a strict robust solution for varying values of σ using the uncertainties $\mathcal{U}_1$ and $\mathcal{U}_2$. Motivated from real-world settings, we varied σ from 0% to 1025% for $\mathcal{U}_1$, while the range for $\mathcal{U}_2$ is 0% up to 50%. For solutions to (SR_1), we calculate the average *price of robustness* (cf. Bertsimas and Sim [16]), that is the additional costs in terms of objective value of a robust solution, as well as the minimal, average, and maximal robustness rob^σ over all solutions that have bounded robustness and report for how many load unit assignments this is the case. For (SR_2), we report the average objective value as well as the minimal, average and maximal robustness rob^σ.

Discussion of results. In Table 5.1 we report computational results w.r.t. $\mathcal{U}_1$. The columns contain the considered deviation σ (in percent), the mean objective value, the minimum, mean and maximum value of rob^σ over all instances (only considered for occupied slots with limited robustness) as well as the percentage of slots for which rob^σ is limited which means that the ∞-case does not apply. These are slots that are neither dedicated to trailers nor allow a maximum overhang which is not smaller than the maximum real-world overhang $\overline{u}_i$ of the assigned load unit. Note that we took steps of 5% for σ but left out lines if the robustness ($\min rob^\sigma$) for a smaller value of σ exceeds the potential σ-value (e.g. $\sigma = 30\%$ is not listed as $\min rob^\sigma = 32.72$ for $\sigma = 25\%$).

σ(%)	obj.	$\min rob^\sigma$	mean rob^σ	$\max rob^\sigma$	$<\infty$ (%)
0	1845.27	0.00	1.16	23.17	66.85
5	1762.28	6.39	23.42	51.66	47.01
10	1761.59	10.63	23.85	51.66	46.81
15	1752.70	16.78	57.16	532.68	46.29
20	1752.08	21.70	58.43	532.68	45.77
25	1694.96	32.72	74.95	532.68	39.39
35	1694.65	36.05	75.12	532.68	39.32
40	1694.65	43.83	75.55	532.68	39.32
45	1693.38	51.66	78.30	532.68	38.44
55	1642.93	61.52	252.86	769.57	32.52
65	1642.74	71.75	253.89	769.57	32.32
75	1641.57	77.10	256.08	769.57	32.32
80	1634.61	93.66	276.23	769.57	29.89
95	1634.03	118.71	308.67	769.57	29.19
120	1634.03	132.69	310.06	769.57	29.19
135	1632.82	144.93	313.95	769.57	29.19
145	1630.38	198.07	369.88	769.57	28.33
200	1621.21	226.70	620.50	859.51	23.92
230	1620.52	249.52	636.94	859.51	23.72
250	1620.52	271.85	638.05	859.51	23.72
275	1620.16	532.68	662.94	859.51	23.72
535	1615.95	679.02	742.37	859.51	22.94
680	1608.53	769.57	795.89	859.51	22.81
770	1608.53	825.12	832.00	859.51	22.81
830	1554.93	859.51	887.61	1020.49	22.29
860	1470.19	769.57	976.81	1020.49	16.79
885	1457.68	1020.49	1020.49	1020.49	12.10
1025	1439.40	-	-	-	0.00

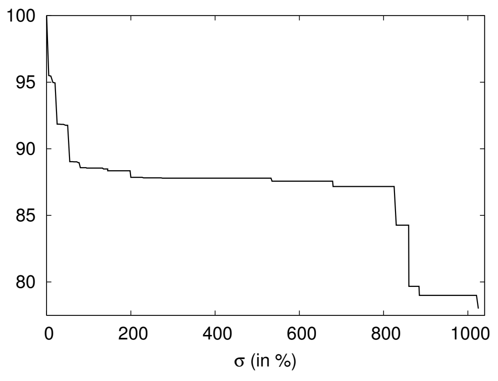
Table 5.1: Computational results for strict robustness w.r.t. $\mathcal{U}_1$.

There are different greater steps of the objective values e.g. between 0% and 5%, 20% and 25% as well as between 45% and 55%. These greater steps can also be seen in the change of the percentage of load units that have a limited robustness rob^σ. The runtimes are all about 1 second for each instance.

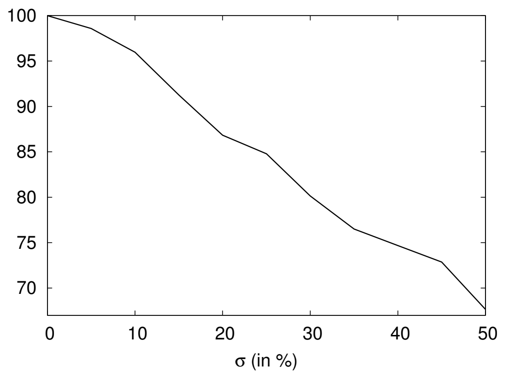
$\sigma(\%)$	obj.	$\min rob^\sigma$	mean rob^σ	max rob^σ
0	1845.27	0.00	1.60	6.29
5	1819.19	5.01	6.22	11.77
10	1770.92	10.01	11.15	12.00
15	1684.12	15.05	15.48	17.86
20	1602.32	20.00	22.02	26.67
25	1564.62	25.00	26.15	31.03
30	1478.77	30.12	31.55	35.00
35	1411.70	35.00	36.22	38.18
40	1378.12	40.00	41.74	46.15
45	1344.56	45.00	45.68	46.15
50	1248.34	50.00	50.91	60.14

Table 5.2: Computational results for strict robustness w.r.t. $\mathcal{U}_2$.

The results of strict robustness w.r.t. $\mathcal{U}_2$ are shown in Table 5.2. We report σ, the mean objective values as well as again the minimum, mean and maximum rob^σ. Runtimes are again in the order of a few seconds for each instance.



(a) $\mathcal{U}_1$

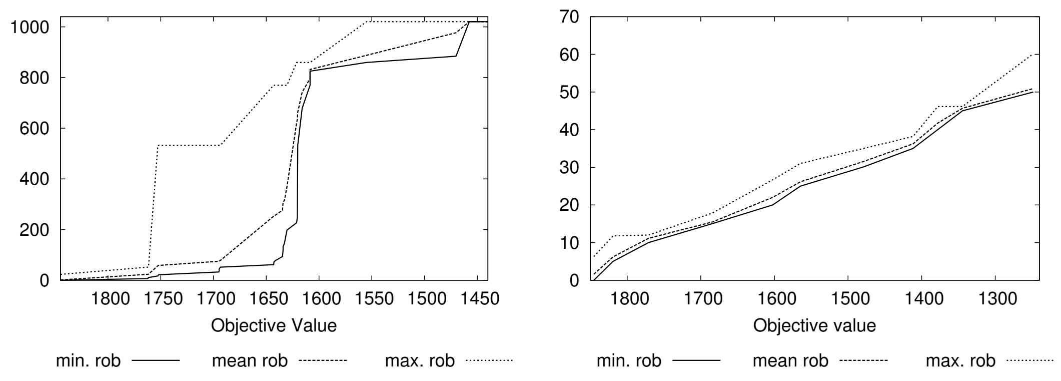


(b) $\mathcal{U}_2$

Figure 5.1: Solution quality in percent of nominal objective for different σ-values

The results w.r.t. $\mathcal{U}_1$ and $\mathcal{U}_2$ are compared in Figures 5.1, 5.2 and 5.3. In Figure 5.1, the uncertainty parameter σ is plotted against the quotient of the objective value of the resulting strictly robust solution and the nominal objective value (in percent) showing the price of robustness. We can note a qualitatively different behavior between the solutions to the uncertainty sets $\mathcal{U}_1$ and $\mathcal{U}_2$. While for the latter one (which affects the *weight* of containers) the objective value is smoothly reduced for increasing values of σ, the former (which affects the *overhang* of containers) shows the presence of plateaus in the plot. This is due to the structure of the real-world instances used: Within the considered setting of the European container market, there is only a small set of possible container sizes, while this is not the case for the naturally continuous weights.

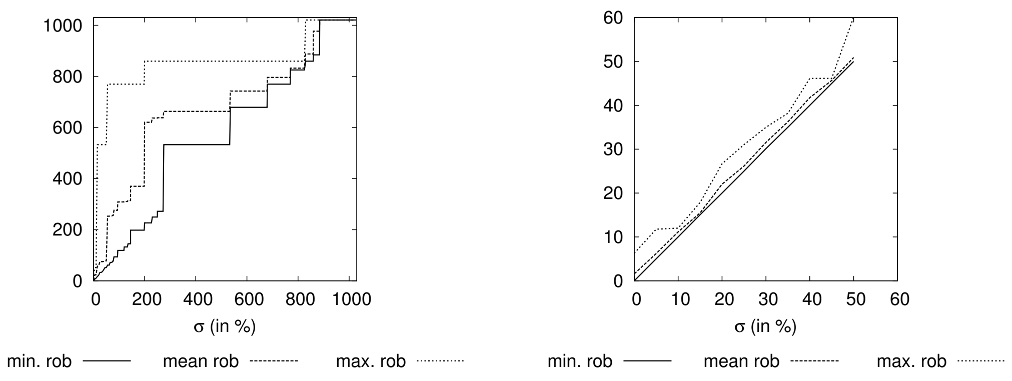
A similar behavior can be noted in Figure 5.2, where for each σ the mean objective value of the solutions (for the 20 example instances) is plotted against its minimum, mean and maximum robustness rob^σ. While in the case of $\mathcal{U}_1$, the robustness of a solution might



(a) $\mathcal{U}_1$ (b) $\mathcal{U}_2$

Figure 5.2: Robustness rob^σ (in %) for different objective values

increase drastically if the objective value is decreased only slightly, this correlation is rather linear for $\mathcal{U}_2$.



(a) $\mathcal{U}_1$ (b) $\mathcal{U}_2$

Figure 5.3: Robustness rob^σ (in %) for different σ-values

Finally, Figure 5.3 shows the robustness of solutions for different values of σ. Note that all values must lie above the diagonal, i.e. have a robustness larger than σ, due to the concept of strict robustness and our definition of robustness. However, the more the robustness of a solution lies above the bisector, the more "implicit" robustness there is. As the plots show, the lines tend to be closer to the bisector in the case of $\mathcal{U}_2$ than for $\mathcal{U}_1$.

We also applied the uncertainty sets $\mathcal{U}_1$ and $\mathcal{U}_2$ simultaneously. This is possible as $\mathcal{U}_1$ and $\mathcal{U}_2$ affect different constraints. The objective values for different sizes of the uncertainty sets showed a mainly independent behavior for the uncertainty sets. If the size of uncertainty set $\mathcal{U}_1$ is fixed while the size of $\mathcal{U}_2$ is varied, then the behavior is similar to the case where only $\mathcal{U}_2$ is considered (of course, the objective values are slightly worse). This is also the case if the roles of $\mathcal{U}_1$ and $\mathcal{U}_2$ are exchanged.

We conclude from our experiments that studying robustness in detail is more important for uncertainty of type $\mathcal{U}_1$ than for $\mathcal{U}_2$. While for the latter, the correlation between objective value and robustness is rather linear, a large robustness gain at only small costs is possible for the former. Therefore, for $\mathcal{U}_2$ it generally suffices to add a security buffer to the solution, while for $\mathcal{U}_1$ the available data should be carefully considered.

5.7.3 The effects of Γ when restricting the uncertainty set

Setup. In order to analyze the impact of the parameter Γ on strict robust solutions, we solved (SR_2^Γ) on the considered benchmark set using $\mathcal{U}_2^\Gamma$ with the values $\sigma \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and $\Gamma \in \{0, 1, \dots, 9, 10, 20, \dots, 80, 85\}$ (85 is the greatest number of load units in one of our example instances). We measured the computation time and determined the respective average values for the objective and rob^Γ. Furthermore, we calculated solutions using $\overline{\mathcal{U}}_2^\Gamma$ for values $\Gamma \in \{0, 1, \dots, 9, 10, 20, \dots, 80, 85\}$ - note that σ plays no role in this case. We chose $\overline{g}_i$ as the minimum value that is not smaller than g_i and not smaller than 36 tons for load units with a corner casting distance greater than 30 feet and trailers, or 30 tons for load units with 20 feet corner casting distance.

Discussion of results. The results for strict robustness w.r.t. $\mathcal{U}_2^\Gamma$ are reported in Table 5.3 for different values of σ. Depending on the size of σ for a sufficiently large number Γ the solutions of $\mathcal{U}_2^\Gamma$ equal those of $\mathcal{U}_2$. E.g., for $\sigma = 40\%$ and $\sigma = 50\%$ the value $\Gamma \geq 3$ is large enough. In this situation rob^Γ is unbounded (marked by -) because the solution is feasible even if the weights of all load units are uncertain within the intervals defined by σ.

$\sigma(\%)$	10		20		30		40		50	
Γ	rob^Γ	obj.	rob^Γ	obj.	rob^Γ	obj.	rob^Γ	obj.	rob^Γ	obj.
1	1	1775.47	1	1611.03	1	1484.16	1	1379.85	1	1250.55
2	10	1774.74	2	1610.02	23	1479.86	2	1378.30	2	1248.10
3	10	1774.74	10	1610.02	23	1479.86	-	1378.12	-	1248.34
10	10	1774.74	10	1610.02	23	1479.86	-	1378.12	-	1248.34
20	20	1773.36	20	1608.16	23	1479.86	-	1378.12	-	1248.34
30	30	1772.13	30	1605.04	30	1479.51	-	1378.12	-	1248.34
40	41	1771.08	42	1602.76	-	1478.77	-	1378.12	-	1248.34
50	-	1770.92	-	1602.32	-	1478.77	-	1378.12	-	1248.34

Table 5.3: Robustness of strictly robust solutions w.r.t. $\mathcal{U}_2^\Gamma$.

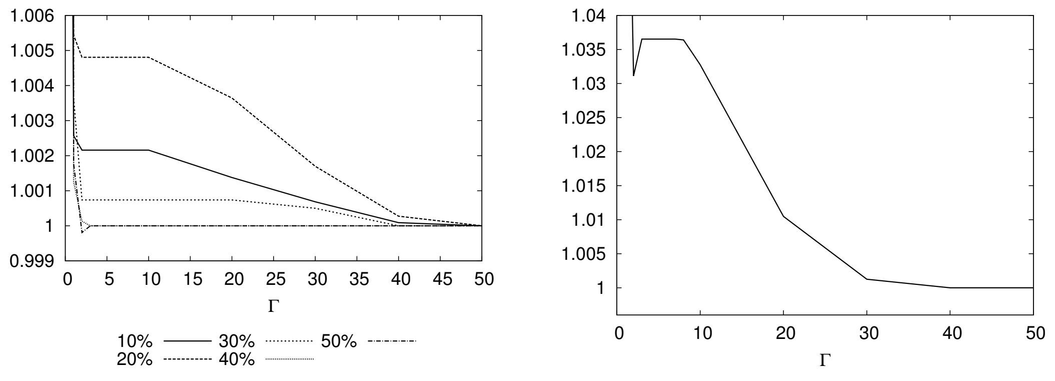
The results for $\overline{\mathcal{U}}_2^\Gamma$ are presented in Table 5.4. The instances marked by ' after the number Γ are calculated by the MIP-formulation (SR_2^Γ). For all other instances (for $\Gamma \geq 3$) the introduced reformulation is only used for constraints (4.12) which limit the total weight of the train. For constraints (5.20)-(5.23) we do not have to use the reformulation because in these constraints at most three positive x_{ijs} can occur for the example instances used. So there is no difference in using the worst-case formulations stated in constraints (5.20)-(5.23). If for at least one instance the calculation is stopped after the time limit of 20 minutes, we mark the runtime by "*". The runtimes for $\Gamma \leq 3$ and the MIP-formulation

Γ	obj.	rob^Γ	time (sec)
0’	1551.96	0	0.85
1’	1531.76	1	192.07 *
2’	1435.63	7	1200.00 *
3’	1437.66	7	1200.00 *
3	1443.15	7	8.59
10	1437.91	10	243.04 *
20	1406.90	20	245.15 *
30	1394.06	31	545.56 *
40	1392.31	-	423.05 *

Table 5.4: Robustness of strictly robust solutions w.r.t. $\overline{\mathcal{U}}_2^\Gamma$.

(SR_2^Γ) are all below 130 seconds with an average value of 60 seconds. On the other hand, if we use the reformulation only for constraints (5.16), for $\Gamma \geq 3$ the runtimes are in order of a few seconds.

Figures 5.4 and 5.5 illustrate these values. In Figure 5.4, the ratio of the objective value of a solution w.r.t. (SR_2^Γ) to the objective value of the corresponding (SR_2)-solution (which corresponds to $\Gamma = n$) is presented. For increasing values of Γ, both objective values become equal, while the solutions to (SR_2^Γ) are better for small values of Γ. As can be seen in the case of $\mathcal{U}_2^\Gamma$, the objective values are only slightly better than for $\mathcal{U}_2$. This means that the restriction that not all weights deviate from their nominal value at the same time does not considerably increase the nominal quality of a solution. This is different when considering $\overline{\mathcal{U}}_2^\Gamma$, where the increase in the objective value becomes more apparent.

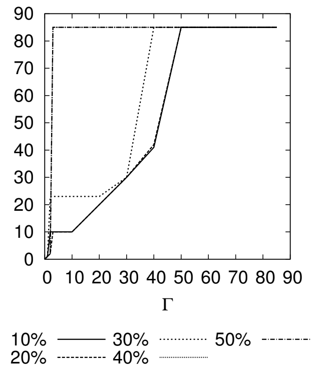


(a) $\mathcal{U}_2^\Gamma$ for different σ-values

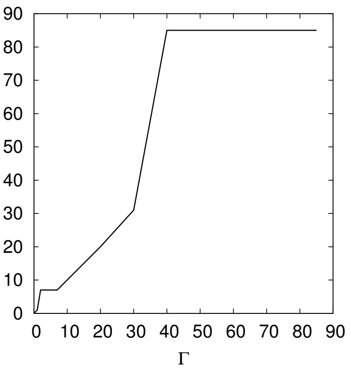
(b) $\overline{\mathcal{U}}_2^\Gamma$

Figure 5.4: Ratio between objective for limited and unlimited number of deviating elements for different Γ-values

The robustness rob^Γ is presented in Figure 5.5. “Implicit” robustness appears whenever rob^Γ is greater than Γ. For both $\mathcal{U}_2^\Gamma$ and $\overline{\mathcal{U}}_2^\Gamma$, the qualitative behavior is roughly the same. For values of Γ larger than 40, the robustness of a solution can be increased at nearly no additional costs. The reason therefore is that for Γ larger than 40 and uncertainty set



(a) $\mathcal{U}_2^\Gamma$ for different σ-values



(b) $\overline{\mathcal{U}}_2^\Gamma$

Figure 5.5: Robustness rob^Γ for different Γ-values

$\overline{\mathcal{U}}_2^\Gamma$ for our instances only between 27 and 47 (out of the potential 37 to 85) load units can be loaded on the train. Furthermore, the uncertainty set does not affect load units with a nominal weight which is not smaller than $\overline{g}_i$. For $\mathcal{U}_2^\Gamma$ all load unit weights increase if they get uncertain but for small weights the change is very small and sometimes will not affect feasibility of assignments at all. This causes that there is no difference if Γ is increased above 40.

The effects of limiting the number of deviating elements (Γ) showed that it might be worse for practitioners to restrict the number of deviating elements if the deviation is independent of the nominal values (which is the case for $\overline{\mathcal{U}}_2^\Gamma$) and the number of deviating elements is quite small. For $\mathcal{U}_2^\Gamma$ with a deviation around the nominal values there is an increase in the objective values but below 1%.

For the solutions of $\mathcal{U}_2^\Gamma$ we also computed the robustness rob^σ. We do not list these robustness values since rob^σ of $\mathcal{U}_2^\Gamma$ is again quite linear (as the one of $\mathcal{U}_2$). The "implicit" robustness is for all calculated combinations of Γ and σ (see Setup) below 1% of the absolute σ-value.

5.7.4 Discussion of adjustable robust solutions

For the discussion of adjustable robust solutions, there are plenty possible experiments to consider. In this section, we present some of the most interesting results. In some cases, as for the uncertainty set $\mathcal{U}_2$, we found that there is a worst-case scenario in all conducted experiments and therefore discuss this only briefly in the following.

Setup. In this experiment, we compare adjustable solutions to their strict robust counterpart for $\mathcal{U}_2^\Gamma$ and $\overline{\mathcal{U}}_2^\Gamma$ and analyze adjustable solutions for $\mathcal{U}_3^\Gamma$.

We chose $\sigma \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and $\Gamma \in \{0, 1, \ldots, 9, 10, 20, 30, 40\}$. For each combination of the two parameters for $\mathcal{U}_2^\Gamma$ and $\overline{\mathcal{U}}_2^\Gamma$, we generated 10 initial scenarios using the maximum or the minimum load unit weights (with the same probability) for all load units individually and used the heuristic approach as described in Section 5.6.2 to calculate adjustable solutions. We measured their objective value as the heuristic approximation of the objective function of (aR_2^Γ) and additionally by the following simulation: We generated 10 independent test-scenarios, completed the adjustable solutions to a solution of each of these scenarios and measured the objective value of the completed solutions. We compared the results of this simulation with the simulation results of the strict robust solutions.

For $\mathcal{U}_3^\Gamma$, we used again a set of 10 initial scenarios to calculate heuristic solutions for $\Gamma \in \{0, \ldots, 9\}$ and additional 10 independent test-scenarios to evaluate the objective value after completion to the best possible feasible solution in the respective scenario. Here we compared the simulation with the solutions of the basic model as there do not exist strict robust solutions for $\mathcal{U}_3^\Gamma$.

Discussion of results. As already noted at the beginning of this section, for all our benchmark instances we found the existence of a worst-case scenario (see Definition 5.8) in the case of $\mathcal{U}_2$. Though this does not need to be the case from a theoretical point of view, the scenario g^{max} dominated all other load unit weights. Therefore, adjustable and strictly robust solutions are the same.

$\sigma(\%)$	10		20		30		40		50	
Γ	obj.	time	obj.	time	obj.	time	obj.	time	obj.	time
1	1836.75	115	1829.80	135	1820.85	242 *	1816.75	195 *	1811.29	147
2	1833.26	147	1820.91	119	1804.33	214 *	1800.99	228 *	1803.07	262 *
10	1821.89	141	1776.27	240 *	1742.40	300 *	1723.56	433 *	1692.73	326 *
20	1806.47	260 *	1732.59	348 *	1676.92	396 *	1630.98	402 *	1580.07	574 *
30	1799.21	283 *	1687.70	307 *	1599.04	317 *	1536.12	219 *	1450.87	83 *
40	1783.86	234 *	1633.64	270 *	1541.52	384 *	1453.24	253 *	1334.99	120 *
50	1748.46	259 *	1606.87	369 *	1496.17	335 *	1391.83	126	1265.75	116

Table 5.5: Computational results for adjustable robustness w.r.t. $\mathcal{U}_2^\Gamma$.

Table 5.5 shows the mean heuristic objective values of (aR_2^Γ) and the mean computation times for $\mathcal{U}_2^\Gamma$. In Table 5.6 computational results for the variant $\overline{\mathcal{U}}_2^\Gamma$ are presented. The simulated objective values using the 10 test scenarios are shown in Figure 5.6.

In Table 5.7 we state computational results w.r.t. $\mathcal{U}_3^\Gamma$. The mean runtimes (with a time limit of 20 minutes) for $\mathcal{U}_3^\Gamma$ and the adjustable heuristic are between 400 and 600 seconds. For all computed Γ for at least one example instance the time limit was reached. The simulated objective values of the adjustable solutions (simA) and the nominal solutions (simN) are given. Furthermore, we computed the mean objective values of the adjustable solutions for the nominal scenarios (nomA). In line 1* for the situation where one wagon has a bug, all possible scenarios are considered as initial as well as test scenarios. So the number of scenarios corresponds to the number of wagons. The results show a slightly worse objective as for the 10 initial scenarios but there is nearly no difference in the objective for the simulation, i.e. the number of considered initial scenarios is sufficient (at least for $\Gamma = 1$).

	(SR_2^Γ)		(aR_2^Γ)		simulation	
Γ	obj.	time	obj.	time	(SR_2^Γ)	(aR_2^Γ)
0’	1551.96	0.85	-	-	-	-
1’	1531.76	192.07 *	1842.33	142.92	1717.75	1840.54
2’	1435.63	1200.00 *	1840.37	173.23	1683.34	1836.90
3’	1437.66	1200.00 *	1840.37	173.23	1681.60	1835.47
3	1443.15	8.59	1840.37	173.23	1680.30	1835.47
7	1443.15	7.47	1828.63	520.40	1675.67	1822.81
8	1442.99	4.78	1825.51	485.26	1680.69	1822.49
9	1440.46	242.72 *	1824.06	571.27	1683.50	1817.90
10	1437.91	243.04 *	1818.44	538.23	1680.16	1808.73
20	1406.90	245.15 *	1760.19	554.52	1668.80	1761.58
30	1394.06	545.56 *	1702.21	565.14	1621.53	1689.49
40	1392.31	423.05 *	1638.34	846.25	1562.16	1598.12
all	1392.31	303.51 *	-	-	-	-

Table 5.6: Computational results for strict and adjustable robustness w.r.t. $\overline{\mathcal{U}}_2^\Gamma$.

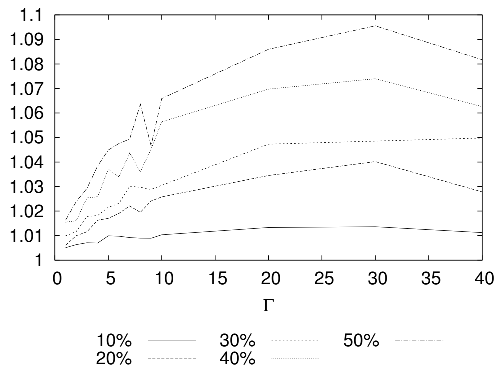
Γ	obj.	simA(min)	simA(mean)	simN(min)	simN(mean)	nomA
0	1845.27	-	-	-	-	1845.27
1*	1783.40	1790.53	1806.06	1783.92	1805.59	-
1	1798.85	1789.95	1804.11	1790.61	1805.12	1825.55
2	1758.44	1744.34	1765.16	1740.76	1764.99	1818.56
3	1720.13	1702.58	1724.76	1691.23	1719.75	1811.69
4	1676.80	1658.17	1683.16	1644.94	1674.58	1800.42
5	1636.00	1608.38	1639.34	1600.13	1630.57	1795.36
6	1589.55	1556.37	1594.29	1547.87	1582.48	1789.57
7	1537.91	1510.26	1544.64	1486.57	1532.11	1771.55
8	1495.36	1461.2	1500.27	1441.52	1482.52	1768.66
9	1444.63	1406.11	1445.87	1376.41	1429.66	1751.82

Table 5.7: Computational results for adjustable robustness w.r.t $\mathcal{U}_3^\Gamma$.

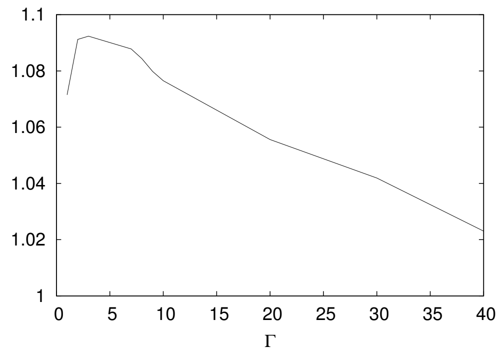
The results of the simulation show that the later decision on the “wait and see”-variables enables much better objective values than the offline optimization that is used with strict robustness. Furthermore, the fixed y_{jk}-values computed with the heuristic for adjustable robustness (see simulation (SR_2^Γ) in Table 5.6 or simA(min) and simA(mean) in Table 5.7) permit slightly better objectives than the ones obtained by strict robustness (see simulation (aR_2^Γ) in Table 5.6) or the nominal problem for $\mathcal{U}_3^\Gamma$ (see simN(min) and simN(mean) in Table 5.7)).

The objective values for adjustable robustness w.r.t. $\mathcal{U}_2^\Gamma$ (reported in Table 5.5) are better than those of strict robustness (reported in Table 5.3), but also the simulated values are, as Figure 5.6 shows. The same holds for $\overline{\mathcal{U}}_2^\Gamma$, as shown in Table 5.6.

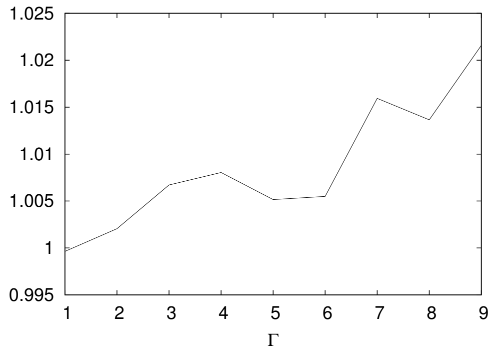
Interestingly, the objective value compared to the strict robust counterpart is different, depending on the uncertainty set used. For $\mathcal{U}_2^\Gamma$ and $\mathcal{U}_3^\Gamma$, a larger Γ seems to indicate a better (larger) ratio for adjustable robust solutions, while this is the opposite for $\overline{\mathcal{U}}_2^\Gamma$.



(a) $\mathcal{U}_2^\Gamma$ for different σ-values



(b) $\overline{\mathcal{U}}_2^\Gamma$



(c) $\mathcal{U}_3^\Gamma$

Figure 5.6: Objective ratio between (aR_2^Γ) and (SR_2^Γ) for different Γ-values

This can be explained by the considered weights and the flexibility the "here and now"-variables offer. For larger Γ and $\mathcal{U}_2^\Gamma$ there are still load units of different weights which could be assigned to the same slot. So the assignment of load units offers still some flexibility. This is not the case for $\overline{\mathcal{U}}_2^\Gamma$ where for larger Γ nearly all load units have their maximum weights. So the flexibility of changing load unit slot assignments does not enable much better solutions because load unit weights are nearly the same for all load units that fit on a certain slot.

To conclude, using adjustable instead of strictly robust solutions for uncertainty sets $\mathcal{U}_2^\Gamma$ and $\overline{\mathcal{U}}_2^\Gamma$ can have a significant impact on the solution quality if in practice the time is available to rearrange the assigned load units. If a practitioner is able to do so, he may increase his gain (in terms of the presented objective function) by up to 10% in our setting. For uncertainty set $\mathcal{U}_3^\Gamma$, where a limited number of wagons can not be loaded at all, it is not so easy to achieve solutions that are much better than the solutions based on the fixed configurations of the nominal solutions. As there is a worse objective for the nominal solution if no wagon has a bug, for uncertainty set $\mathcal{U}_3^\Gamma$ it may not be sensible to use adjustable solutions if the probability of at least one wagon having a bug is not close to one.

5.8 Discussion of the results

In this section we introduced the problem of determining load plans in intermodal transportation that take different kinds of uncertainties into consideration. We presented two approaches to include this uncertainty: Strict robust load plans, in which it is assumed that the solution cannot be changed once it is implemented, and adjustable robust load plans, that give the planner the possibility to react once the true problem parameters become known.

For different robustness models and uncertainty sets, we derived mixed-integer linear programs for their robust counterparts. Their performance was analyzed in comprehensive experiments on real-world instances that demonstrated that already at small costs for the planner, the reliability of the solution can considerably be increased. Our work shows that robustness is an important aspect in real-world problems and can be taken into consideration even for mixed integer linear programs with a bundle of different constraints.

Further research will focus on the application of robustness models that give the planner a larger choice on how to react to parameter changes, as in the concept of *recoverable robustness* (cf. Liebchen et al. [70]). Moreover, it is interesting to also include uncertainties in the objective function.

5.9 List of robust load planning notations

The notations of the robust load planning model are the same as those of the third IP in Section 4 (see Section 4.5). Two new parameters and the notations concerning the robustness concepts are listed in Table 5.8.

additional notations for load planning model	
u_i	overhang of load unit i
β_{ks}	feasible overhang for slot s in configuration k
notations for the robustness approaches	
ξ	a scenario
$\tilde{\xi}$	the nominal scenario
$\mathcal{U}_1$	uncertainty set with uncertain load unit lengths
$\mathcal{U}_2$	uncertainty set with uncertain load unit weights
$\mathcal{U}_2^\Gamma$	uncertainty set with uncertain load unit weights (for controlled robustness)
$\overline{\mathcal{U}}_2^\Gamma$	uncertainty set with uncertain load unit weights (for controlled robustness and with weights independent of the nominal values)
$\mathcal{U}_3^\Gamma$	uncertainty set with uncertain wagon failures (for controlled robustness)
SR_i	strictly robust problem for uncertainty set $\mathcal{U}_i$
F_{SR_i}	the set of strictly robust solutions for uncertainty set $\mathcal{U}_i$
aR_i	adjustable robust problem for uncertainty set $\mathcal{U}_i$
$\hat{g}^l_{i\tau s}$	replacement parameter for $\mathcal{U}_2$ to maximally disturb bogie payload balance constraints (5.20) for load unit i, a wagon of type τ and slot s
$\hat{g}^r_{i\tau s}$	replacement parameter for $\mathcal{U}_2$ to maximally disturb bogie payload balance constraints (5.21) for load unit i, a wagon of type τ and slot s

Table 5.8: Notations of the robust load planning problem

6 Crane planning

In this section we consider the crane planning problem where a set of transport jobs is scheduled on a set of cranes. This means that each transport job is assigned to a crane and a start time for the transport job is determined. The start times of the transport jobs of one crane define a processing sequence of transport jobs. We consider the crane planning setting in rail-road terminals where up to three cranes operate on the same special tracks. This implies that for the cranes interference or so-called non-crossing constraints have to be considered. The main aim of crane planning is to minimize the total length of the empty crane moves. These are the crane moves without load units which are the moves from the sink position of a load unit to the source position of the load unit that is transported next.

The basis of the data we use to test our algorithms are the real-world storage planning instances presented in Section 3.10. The output of the storage planning heuristic (see Section 3.9) is used as input for the crane planning. This section and the modeling of crane planning within it are motivated by a research project in cooperation with Deutsche Bahn, Deutsche Umschlaggesellschaft Schiene-Straße (DUSS) and Berghof Automation GmbH [55]. Especially, we use the same hard and soft constraints as in the project.

Contributions. We provide a crane planning MIP that models the non-crossing constraints for transport jobs with non-equal source and sink positions. We do that in a detailed way, such that cranes can perform parallel moves if the minimum distance in between them is respected. As most crane planning in literature concerns maritime terminals, we are not aware of a publication that considers jobs with different source and sink positions and model the interference constraints as detailed as we do. In the literature it is assumed that only the trolley is moved to perform the loaded move.
Furthermore, we provide a heuristic based on the same model as the MIP that is able to find solutions in a short amount of runtime.

Overview. This section is organized as follows. After a more detailed problem definition in Section 6.1 we analyze literature about crane planning in Section 6.2. We present three optimization approaches for the crane planning problem – a MIP model (Section 6.3), a list scheduling heuristic combined with the meta-heuristics Simulated Annealing and Tabu-search (Section 6.4). In the last section we compare some results of the different optimization approaches (Section 6.5).

6.1 Problem definition

In this section we will first explain the properties and restrictions of rail mounted gantry cranes (RMG). Afterwards, properties of transport jobs, the detailed handling of a transport job by a crane and crane policies are introduced. This section ends with a crane planning example.

We consider a rail-road terminal where m RMGs of the set $\mathcal{K} = \{1, \dots, m\}$ are used to lift and transport load units inside the terminal. We call the transport of a load unit from a fixed source to a fixed sink position a transport job.

In rail-road terminals nowadays up to three RMGs operate on one pair of special tracks which causes crane interferences or so called non-crossing constraints. Non-crossing constraints state the following: For each point in time the positions of the cranes have to respect the physical ordering of the cranes. Especially, cranes may not pass each other. So, for all planning time points t and all cranes $k \in \mathcal{K}$ with a fixed numbering $k = 1, \ldots, m$ the crane x-positions $\hat{x}_{kt}$ have to fulfill the following inequality: $\hat{x}_{1t} < \hat{x}_{2t} < \ldots < \hat{x}_{mt}$. It is not important whether a crane is performing an empty move, transports a load unit or is idle, crossings are not allowed. Beside the hard non-crossing constraints the maximum velocity of two cranes that undercut a certain safety distance is reduced to the half due to safety regulations. This safety distance is denoted by δ. The violation of the soft crane safety distance leads to time delays in crane processing and is therefore undesirable.

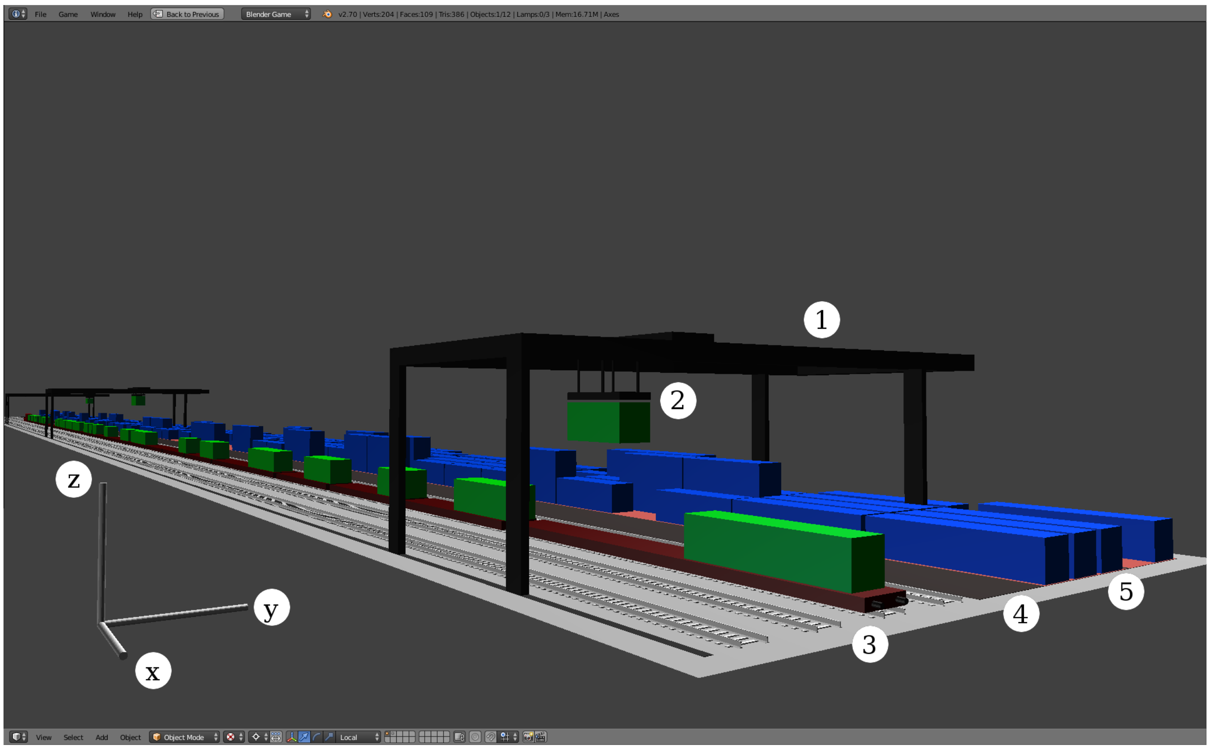


Figure 6.1: Blender 3D visualization of a rail-road terminal with axes and the different elements of the terminal: a crane (1) that lifts a load unit with the spreader (2). Under the crane are the different areas of the terminal – tracks with trains (3), driving and parking lanes for trucks (4) and the storage area with stored load units (5). Implementation of the 3D visualization was done by Aßbrock [8]

The cranes can move along three axes which are shown in Figure 6.1 (provided by Aßbrock [8]). The most important movement is alongside the tracks (x-axis). These moves are performed by driving of the crane on the special crane tracks. Such a move is necessary to travel between different wagons or different stacks inside the storage. As x-position of a crane we take the middle of the crane.
The movement across the different areas (rail, road and storage indicated with (3) to (5) in Figure 6.1) is done by the trolley (y-axis). For the y-axis we consider each track, the

parking and the driving lane as well as all storage lanes as discrete y-positions. So, a transport job from the storage to a wagon or a truck includes a movement within the y-axis.
Furthermore, we consider discrete levels as z-axis which describes the elevation of the spreader. The crane spreaders, which are used to fix load units while they are lifted, transported or lowered again, can be used in four different settings. Three settings offer to fix 20, 30 and 40 ft containers at the upper corner castings, respectively. For swap bodies and trailers the fourth setting with grapplers is used to pick these ones at the base of the load units. Note, that only on the storage area load units can be stacked and levels above the ground level can be used. A load units needs to be lifted up to the highest possible level to be transported in the x- or y-axis. Nearly all transport jobs consider a movement in all three axes. Only very few jobs can be handled without a movement in the x- or y-direction.

The transport jobs that have to be planned are given in the set Ω. A transport job consists of exactly one load unit to be transfered. For the transport jobs fixed source and sink locations (3D) are given which means that the crane move under load is predefined. The sink locations are determined by the storage and load planning (cf. Sections 3 and 4).

The source x- and y-positions are denoted by s_i and $\tilde{s_i} \in \mathbb{N}$ while g_i and $\tilde{g_i} \in \mathbb{N}$ are the sink or goal positions of transport job i. Each transport job i has a priority $\varphi_i \in \mathbb{N}$, one of the four possible spreader settings σ_i, a release date $r_i \in \mathbb{N}$ and a due date $d_i \in \mathbb{N}$. Furthermore, all transport jobs that are delivered or caught up by truck are contained in set $\mathcal{L} \subseteq \Omega$. While the release date is the earliest feasible start time of a job, the due date defines a time point that should not be exceeded by the processing of the job. Some transport jobs have predecessors which have to be processed before them. These precedence constraints may be caused by a position conflict or a necessary reshuffle. A position conflict means that the load unit of transport job i shall be transported to the source location of the load unit of transport job j. Obviously, transport job j has to be performed before job i. Reshuffles describe that a load unit has to be moved away to access a load unit in a lower level of the same stack. In the case of reshuffles the upper load unit has to be transported before the lower one. We use the standard notation for precedence constraints where $i \rightarrow j$ means that job i has to be completed before job j starts.

The execution of a transport job by a RMG consists of different operations. At the beginning the crane has to move from its current position to the source location of the job (which is the current location of the load unit to be transported). Then the spreader needs to be lowered to adjust the load unit. Afterwards the spreader with the load unit needs to be lifted up. The crane movement in x- and y-direction to the sink location of the job is done subsequently. The sink location of the job is the new load unit location. At the sink x-y-position the load unit is placed on a wagon, on a truck or on the ground (inside the storage area). After the empty spreader is lifted up again, the crane can start the next job by moving to the source location of the next job.

Each crane k has an initial (triple-)position and an initial spreader setting. We assume uniform cranes, so the velocity of the cranes as well as processing times p_i for jobs are the

i	s_i	$\tilde{s}_i$	g_i	$\tilde{g}_i$	d_i^{load}
1	0	0	150	50	150
2	200	150	1000	0	800
3	1250	200	1250	50	150
4	800	0	200	200	600

Table 6.1: Example instance, transport job x- and y-positions and the length of the loaded crane moves d_i^{load}

i	1	2	3	4
1	-	100	1100	650
2	1000	-	250	200
3	1250	1050	-	450
4	200	50	1050	-

Table 6.2: Length of the empty crane moves d_{ij}

same for all cranes. Also the width ω of the cranes is identical. Furthermore, we assume that only the time for moving in x- and y- direction differs for different transport jobs. The time needed for pick up and stabling of load units is assumed to be identical for all transport jobs and is denoted by b. The time span b is the mean time a crane needs for lowering the spreader, fixing a load unit, lifting a load unit, lowering and stabling a load unit as well as lifting up the spreader again. Note, that these operations are preempted by the loaded crane move so the half of the operations are performed at the source location of the load unit and the other half at the sink location. During the time span b the crane spreader moves in the z-direction but the crane does not move in x-direction and the trolley stays at the y-position. Once the spreader is in the upper position (with or without a load unit), a crane can move simultaneously in x- and y- direction. So, as distances for a move in x- and y- direction we take the maximum of the distances in x- and y-direction because we assume overlaid moves. The length of the empty move between transport jobs i and j is denoted by d_{ij}. This is the move from the sink location of job i (x-position g_i and y-position $\tilde{g}_i$) to the source location of j (x-position s_j and y-position $\tilde{s}_j$). So, d_{ij} is defined as $d_{ij} = \max(|g_i - s_j|, |\tilde{g}_i - \tilde{s}_j|)$.
Note, that we are facing an asymmetric transportation problem such that the values d_{ij} and d_{ji} differ in most cases for transport jobs $i, j \in \Omega$ with $i \neq j$. To illustrate this, we present a small crane planning instance with distances d_{ij} in Example 6.1.

Example 6.1. In Table 6.1 the source and sink x- and y-positions (s_i, $\tilde{s}_i$, g_i and $\tilde{g}_i$) and the length of the loaded moves (d_i^{load}) are shown for four example transport jobs. In Table 6.2 the resulting distances d_{ij} for empty moves are listed. For some job pairs d_{ij} is equal to the distance in x-direction which dominates the distance in y-direction (e.g. for d_{23}) and for some it is the other way round (e.g. d_{12}). For the considered transport jobs no two jobs i and j exist such that $d_{ij} = d_{ji}$. So, for all pairs of jobs it makes a difference in which order the jobs are processed. □

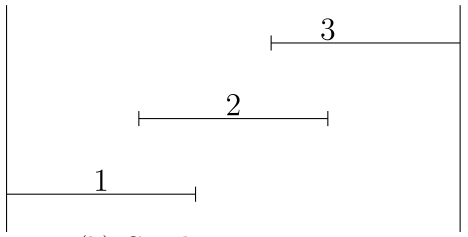
The processing time $p_i \in \mathbb{Q}$ of transport job i is the time span for pick up and stabling of a load unit (b) plus the travel time for the maximum distance of x- and y-distances for the loaded crane move (with the length d_i^{load}, see Table 6.1). The travel time t_{ij} between job i and j is the travel time for the distance d_{ij}. Note, that as generally d_{ij} is not equal to d_{ji} also t_{ij} and t_{ji} may differ.

In the terminal practice as well as in the literature (see e.g. Alicke [3]) different crane policies are discussed: Fixed overlapping crane areas and free crane movement.

The first option is to choose **fixed overlapping crane areas** as proposed by Souffirau et al. [80]. An example of fixed crane areas of a terminal with a length of 6000 dm and three cranes is shown in Table 6.2. In this example two overlapping areas exist: From 1750 dm to 2500 dm in x-direction cranes one and two both operate and from 3500 dm to 4250 dm in x-direction cranes two and three operate together. Furthermore, each crane has an exclusive working area. In the example this is for crane 1 from 0 dm to 175 dm in x-direction. In these exclusive working areas no crane interferences occur.
For some jobs these fixed crane areas cause that the job need to be split in up to m jobs. For the example in Table 6.2 this is the case for a job that implies a transport from 2400 dm to 4500 dm in x-direction. For the execution of this transport job at least two cranes are necessary. Crane 2 can pick up the load unit of the transport job at x-position 2400 dm and transport it to the upper end of its working area (x-position 2450 dm). At this position crane 3 has to pick up the load unit to deliver the load unit to the sink position of the transport job at x-position 4500 dm. On the other hand, fixed overlapping crane areas have the advantage that for many jobs the crane assignment is predefined by the crane areas which reduces the problem complexity. If a job has to be transported from 550 dm to 2400 dm in x-direction, only crane 1 can handle such a job, if the example working areas of Table 6.2 are considered.

k	min x-value	max x-value
1	0	2500
2	1750	4250
3	3500	6000

(a) Table representation



(b) Graphic representation

Figure 6.2: Example for fixed overlapping crane areas

Fixed overlapping crane areas can be defined by determining the length of the overlapping areas and by dividing the terminal in equidistant crane areas. An other approach is to calculate fixed crane areas with the aim of having balanced crane workloads (see Boysen et al. [23]).

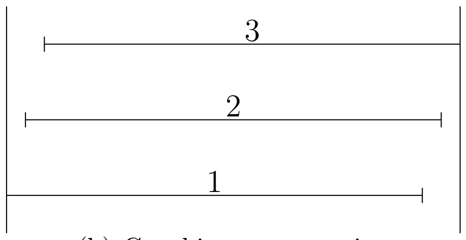
The alternative **free crane movement** limits the crane areas only by physical restrictions. This means each crane has to stay inside the terminal, so the first crane is not able to reach the upper end of the terminal. A crane k can reach a maximum x-position $\rho_k^{max} = l - (m - k) \cdot \omega$ and a minimum x-position $\rho_k^{min} = 0 + k \cdot \omega$ where l is the length of the terminal in x-direction and ω is the width of a crane. An example for resulting physical crane areas is shown in Table 6.3.

Even without operational limitations of crane areas (beside physical restrictions) it might be necessary to split up jobs to be handled by two or more cranes. But most jobs can potentially be handled by all cranes. This leads to a greater solution space compared to fixed overlapping crane areas and as a consequence to a more complex planning problem.

During the crane planning each transport job is assigned to a crane, for all cranes the assigned transport jobs are sequenced and start times for all jobs are defined.

k	min x-value	max x-value
1	0	5500
2	250	5750
3	500	6000

(a) Table representation



(b) Graphic representation

Figure 6.3: Example for physical restrictions of crane areas when a crane has a width of 250 dm

A crane plan in a terminal of our cooperation partner DUSS has to respect the following hard constraints:

(H1) Two cranes cannot undercut a distance of one dm in between of them. As a consequence they can not pass each other (non-crossing constraint).

(H2) The release dates (r_i) of the transport jobs i have to be respected.

(H3) If a transport job j is handled after job i by the same crane, j cannot start before i has been completed and the crane has traveled empty to the source location of job j.

(H4) Given precedence relationships between transport jobs have to be met.

(H5) The moves of all cranes are restricted to the associated crane working areas (which may be motivated physically or operationally).

Soft constraints consists of the following elements:

(S1) Minimize the total length of the empty crane moves.

(S2) Minimize the number of violated job due dates.

(S3) Minimize the total tardiness of jobs with violated due dates.

(S4) Minimize the number of jobs that have less than the minimum crane safety distance δ in between them and that are handled simultaneously (by different cranes).

(S5) Minimize the waiting times of load units that are fetched or delivered by truck.

(S6) Minimize the waiting times of high-priority load units. As waiting time we account the time span between the release date and the start time of a job.

(S7) Minimize the number of turnarounds of cranes. The turnarounds consider the movement in x-direction. So if a crane first moves from a smaller to a larger x-position and afterwards moves again to a smaller x-position, a turnaround is accounted.

(S8) Minimize the changes of spreader settings. The spreader has to be changed when the load units of two consecutive transport jobs assigned to the same crane have different spreader settings (settings are 20, 30 and 40 ft as well as grapplers).

The aim of (S1) is to minimize energy consumption and overall handling time. The due dates (S2) are mainly caused by the deadlines of closing for cargo of trains. If a due date is violated, either the train departure delays or the load unit is left in the terminal, while the assigned slot remains empty. The length of the delays (the total tardiness) should also be minimized (S3) which is very important for load units that are fetched by truck because the truck has to wait until the load unit is placed on the trailer. (S4) aims to keep a minimum soft safety distance between adjacent cranes to avoid reduced crane velocities. Attaining a high satisfaction level of customers by quickly performing high priority transport jobs is the aim of (S5) for truckage companies and (S6) for railway companies. High priority jobs are for example jobs with load units that are delivered or fetched by truck. For such transport jobs (S5) aims to minimize the time of truck circulation. The soft constraints (S5) and (S6) have the same mathematical structure but model different terminal situations. While during most time of the day transport jobs evolving trucks are the most important jobs, if the closing for train loading is close, transport jobs with sink or source positions on the trains are most important. These different terminal situations are modeled in different constraints. The objective components (S7) and (S8) shall increase the acceptance of crane plans by human crane operators. Furthermore, (S8) aims to minimize overall handling time by reducing changes of spreader settings which takes some time.

In the following we illustrate the crane planning problem in Example 6.2.

Example 6.2. In this example we consider a problem with three cranes ($\mathcal{K} = \{1,2,3\}$) and ten transport jobs ($\Omega = \{1,\ldots,10\}$). Main properties of the transport jobs are listed in Table 6.3. Columns are the job number (i), the source- and goal-x- (s_i, g_i) and y-positions ($\tilde{s_i}, \tilde{g_i}$) of the jobs, the release and due dates (r_i and d_i). The column "truck" contains a 1, if $i \in \mathcal{L}$ (so if load unit of transport job i is delivered or fetched by truck). Further columns are the spreader setting (1, 2 and 3 are 20, 30 and 40 feet corner casting fixing for containers and 4 is the grappler setting for swap bodies and trailers), the transport job priority φ_i and jobs minimum and maximum x-positions e_i^{max} and e_i^{min} (can be calculated by $e_i^{max} := \max\{s_i, g_i\}$ and $e_i^{min} := \min\{s_i, g_i\}$). In this example all positions are measured in dm and the times r_i and d_i are in minutes.

Furthermore, the following precedence relationships have to be considered: $5 \rightarrow 1, 4 \rightarrow 2, 6 \rightarrow 4, 3 \rightarrow 6, 6 \rightarrow 7$. The crane properties like initial crane x-position, minimum and maximum feasible crane positions are listed in Table 6.4.

The terminal length is assumed to be 7000 dm and the maximum crane velocity in x- and y-direction is set to $1800\frac{\text{dm}}{\text{min}}$. To process a transport job takes the time for moving the crane (simultaneously) in x- and y-direction plus a fixed time span b of one minute which takes into account lowering and lifting the spreader as well as fixing and unlocking the load unit. The width of a crane ω is assumed to be 250 dm and the (soft) minimum crane safety distance is $\delta = 500$ dm.
For example, the process time of job 1 is ≈ 1.93 min which is the sum of the fixed job

i	s_i	g_i	$\tilde{s}_i$	$\tilde{g}_i$	r_i	d_i	truck	spreader setting	φ_i	e_i^{max}	e_i^{min}
1	4840	160	3170	480	7	11	0	2	1	3170	4840
2	3959	720	2000	80	280	16	0	2	1	2000	3959
3	4688	640	4331	80	7	20	1	3	1	4331	4688
4	1726	0	1909	0	1	15	1	2	3	1726	1909
5	6035	400	5669	560	2	15	1	2	2	5669	6035
6	7000	560	6835	480	4	13	1	1	3	6835	7000
7	5365	0	6211	80	1	14	0	4	3	5365	6211
8	1438	560	3133	320	3	6	0	4	1	1438	3133
9	5777	720	3699	640	3	10	0	1	1	3699	5777
10	2293	80	832	160	4	14	0	2	2	832	2293

Table 6.3: Properties of transport jobs of Example 6.2

k	initial crane		minimum	maximum	initial
	x-position	y-position	crane x-position	crane x-position	spreader setting
1	200	8	0	5500	2
2	3500	10	750	6250	3
3	6000	12	1500	7000	2

Table 6.4: Crane properties of Example 6.2

duration (1 min) and the time for moving the crane from the source to the sink location. The time for moving the crane from the source to the sink x-y-position can be calculated by

$$\max(|s_i - g_i|, |\tilde{s}_i - \tilde{g}_i|)/v,$$

where v is the crane velocity. For job 1 the time of the loaded crane move is $\max(|4840 - 3170|, |160 - 480|)/1800 \approx 0.93$ min.

Figure 6.4 shows a feasible schedule for the crane planning example. Horizontally the time in minutes is shown, while vertically the crane travel in x-direction in dm is plotted. The crane path of the cranes is plotted in blue, magenta and red. The beginning of a load unit transport by a crane is denoted by a filled square, the loaded crane move is printed bold and the end of a transport job is denoted by a circle. The numbers of the transport jobs are printed beside the bold line of the loaded crane move. Note, that crane 1 works at the smallest x-positions, crane 2 is operating in the middle of the terminal and crane 3 handles transport jobs at the upper end of the terminal.
During the processing of job 1 by crane 3, crane 2 has to move to x-position 2920 to enable the processing of job 1 without crane crossings. We call such a move a "give way" move. When transport job 1 is finished by crane 3, cranes 2 and 3 move simultaneously towards greater x-positions always keeping the minimum hard distance of 250 dm.
In Table 6.5 the start times of the transport jobs are listed.

For the solution presented in Figure 6.4 and Table 6.5 the objective elements are the following:

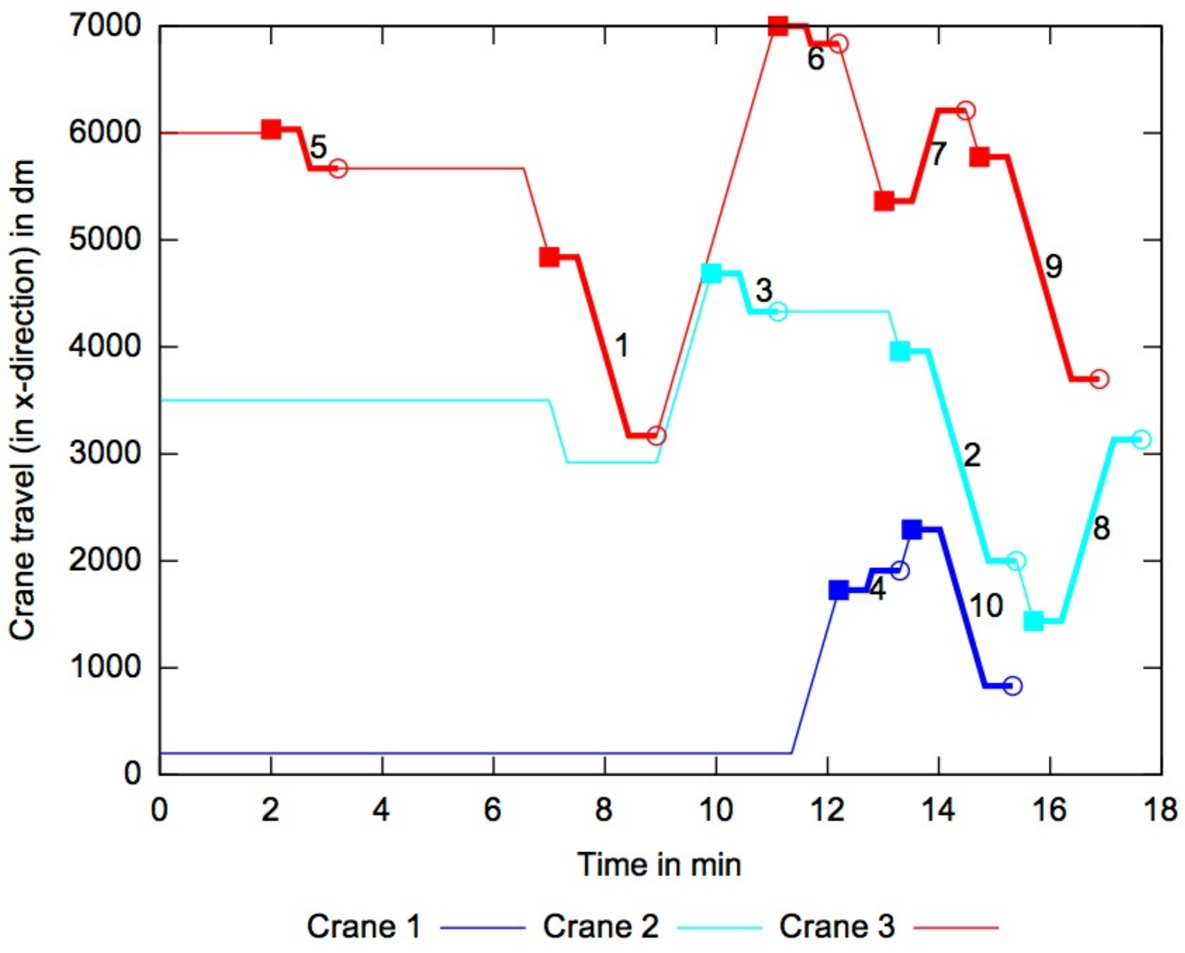


Figure 6.4: Feasible crane planning solution for Example 6.2

(S1) empty crane moves with a total length of 11790 dm

(S2) due dates of four transport jobs (7,8,9,10) are violated

(S3) the total delay is 20.3 min

(S4) two spatial crane conflicts occur (cranes come closer than 750 dm). These conflicts are between the cranes during handling of jobs 1 and 3 as well as between those handling 8 and 9

(S5) the total waiting time of transport jobs that are delivered or fetched by truck is 21.2 min

(S6) the total weighted (by the priority) waiting time of high priority jobs is 143.6 min

(S7) cranes do have to turnaround eight times (at the end of jobs 1 and 7, at the beginning of jobs 6, 7, 3, 8 and 10 as well as before job 3 due to give way move of crane 3)

i	1	2	3	4	5	6	7	8	9	10
S_i	7	13,30	9,91	12,2	2	11,11	13,02	15,71	14,73	13,52
crane	3	2	2	1	3	3	3	2	3	1

Table 6.5: Crane planning example start times S_i of transport jobs

(S8) the crane spreaders have to be changed five times (between transport jobs 1 and 6, 6 and 7, 7 and 9, 3 and 2 as well as 2 and 8)

□

Before we will introduce a MIP-model and heuristics for the crane planning, we will have a look at related literature in the next section.

6.2 Related literature

We will first analyze the operational setting and the scientific work carried out in the field of maritime terminals and afterwards have a view on crane planning in rail-rail or hub terminals.

In maritime terminals mainly three different crane planning situations occur. Planning of berth/quay cranes, yard cranes, and planning of RMGs in train interfaces. In the following we will illustrate similarities and differences of these crane planning situations to the crane planning in rail-road terminals.

Berth/quay cranes do not have to move alongside the bay (in x-direction) while loading or unloading a ship. They transfer containers to the nearest point on the quay where some transport vehicles (e.g. AGVs) take them to bring them to a yard area. In contrast to rail-road terminals the crane move under load does not include movement in x-direction which causes a difference in modeling non-crossing constraints (see below).

In yard areas one has to distinguish two cases. Viewed from top, yards can be seen as a rectangle with a short and a long side, where the cranes drive along the long side and the trolley moves along the short side. We will only have a look at yards where two or more cranes operate in a block because otherwise no crane interferences or non-crossing constraints have to be considered.
The first possibility are yards of the Asian layout with a handover area along one long side often operated by rubber tiered gantry cranes (RTGs) (described e.g. in Ng [75]). In such a situation the vehicles that deliver or collect containers stop on the exchange area close to the current or planned storage x-position of their container. So the container handling with the crane reduces to trolley movement in y-direction. The crane will not move with the container along the long block side in x-direction. So again the characteristics of transport jobs are quite different to those of jobs inrail-road terminals because there is no container transportation in x-direction. So the difference is again the modeling of non-crossing constraints.
On the other hand, there are European layout yards with handover-points at the short

sides (twin-RMG as considered in Park et al. [76]). In such a situation the seaside handover point is reserved for vehicles (e.g. AGVs) that drive to the quay and the landside handover point is reserved for external trucks or vehicles that transport container sto a train interface. Such yards are usually operated by two cranes. So the seaside crane has to handle all jobs that are delivered to or collected from the seaside handover point and the same for the landside crane and handover point. An alternative are cranes of different sizes that can pass each other (DRMG or triple cranes as described in Dorndorf and Schneider [39]). If cranes cannot pass each other and more than two cranes operate in the terminal, the cranes in the middle can only execute reshuffle jobs because it is not possible to reach any vehicles at the handover points for the middle crane. This is the cause why only two yard cranes or cranes of different sizes are deployed in such yards.
In crane settings that are an extension of twin-RMGs the same non-crossing modeling as in rail-road terminals is used at least for some cranes. The huge crane in a triple crane setting has different crane interferences to the two small cranes. But the optimization problem is still quite different. With twin-RMGs only some reshuffling job can be assigned to both cranes (both small cranes in triple crane setting). Furthermore, for all jobs that start or end at a handover point the crane assignment is fixed for twin-RMGs.

The objective of related work in the area of maritime terminals is to minimize the makespan or a weighted sum of the crane finish times. We first present some work that is cited as quay crane planning by the literature review of Stahlbock and Voß [82]. Some publications that concern multiple cranes and non-crossing constraints use MIP formulations that are based on the one of Kim and Park [60] (in brackets we list the proposed solution methods): Moccia et al. [74] (MIP, branch-and-cut), Jung and Kim [58] (GA/SA). Kim and Park [60] minimize a weighted sum of the makespan and the crane finish times. They concern travelling times of empty movements and precedence constraints, but no time windows for the transport jobs. Kim and Park's [60] MIP formulation is a VRP formulation with additional non-crossing constraints. Kim and Park assume that for each job i exactly one quay position l_i is given. The crane that handles job i has to stay at position l_i during the execution of job i. The non-crossing constraints used by Kim and Park [60] forbid, that some tasks i and j are performed simultaneously if $l_i < l_j$ and i should be performed by a crane which position is greater than the position of the crane that should perform job j. So for the non-crossing constraints only loaded crane moves are considered. Crossings between cranes where at least one of the cranes performs an unloaded move or stays idle can not be omitted. This is not a problem as long as cranes do not move in x-direction while they are loaded and the objective is to minimize the overall handling time. If a crane is not executing a job, it eventually has to give way to other cranes that are working. Moccia et al. [74] pointed out that it might be necessary to introduce idle times for the cranes that are working, if another crane has to move away. These idle times consider the time needed by the other cranes to move away till a security distance is kept.
If the cranes move while they are loaded, the "give way" moves can have a great influence on the objective function. If the movement in x-direction is small, the "give way" moves do not cause a huge change of the objective function. But for longer job moves in x-direction the influence of the unproductive "give way" moves is very high. Therefore the solution classified as optimal may no longer be optimal for the problem with "give

way" moves. So if one wants to use a LP model based on the one of Kim and Park [60] for a terminal with crane transportation in x-direction like rail-road terminals, one need to overcome these problems with "give way" moves.

Moccia et al. [74] introduce stronger constraints for the Kim and Park formulation. These constraints overcome some failures of Kim and Park and try to improve the feasibility of the MIP. But even the improved formulation of Moccia et al. has runtimes that are not acceptable for practical applications.

Lee et al. [68] objective is to minimize the makespan. They do neither concern travel-times, precedence constraints nor time windows for transport jobs (MIP, GA). Lim et al. [71] have the same objective and constraints as Lee et al. (MIP, LS, SA, backtracking).

A resent survey of quay crane planning has been provided by Bierwirth and Meisel [19]. The same authors also provided a fast heuristic for the quay crane scheduling [18] and one of the authors introduced a further improved MIP which concerns time windows [73]. In the latter work, a tree-search-based heuristic is proposed as solution method.

Ng [75] presents a MIP where crane interferences between yard cranes are modeled. Therefore for each crane the positions over time are determined by decision variables. The objective is to minimize the total completion time, travel time of cranes is accounted and jobs have release dates but no due dates nor are precedence constraints between jobs considered. Due to the spatial modeling of cranes the MIP does not seem to be promising as solution approach. Ng presents a dynamic programming approach for partitioning of crane working areas and a heuristic that subsequently swaps jobs between cranes.

For the crane planning in rail-rail or hub terminals only very few publications are available. Alicke [3] presents a solution approach for a hub terminal where load units are mainly transfered between different trains. He assumes predefined overlapping crane areas. Using a constraint programing approach start times for transport jobs and cranes for jobs in the overlapping area of two cranes are determined.

Souffirau et al. [80] present an optimization approach for a rail-rail terminal at the Spanish-French border where they consider various optimization problems. The crane planning (without time windows) is modeled as a RCPSP with the aim of minimizing the makespan. The two cranes have fixed crane areas which are 60 % of the terminal area. However, 20 % of the terminal are considered as overlapping area and modeled as a resource with unit capacity, so if a crane is working in the overlapping area, it is blocked for the other crane. To solve the problem, Souffirau et al. [80] suggest a scheduling heuristic based on a priority list and local search, where the priority list is changed by the neighborhoods adjacent swap, forward and backward shift. The scheduler transforms the priority list into a crane plan. If a job is in the overlapping area, this job is assigned greedily to a crane with the shorter distance to the source of the job.
The local search is an iterative improvement with best fit steps within the neighborhoods ordered increasingly by their complexity. So shift operations are only considered if no improvement with any adjacent swap is possible. For diversification beginning from a priority list position the following m jobs are removed from the list and planned at their earliest feasible position.

6.3 A MIP formulation

We present a MIP formulation for the crane planning problem based on a flow formulation of a VRPTW with some additional constraints (e.g. non-crossing and precedence constraints). After introducing the necessary parameters and the basic idea of the MIP formulation we present the MIP itself and discuss some of its properties.

Our formulation is an adaptation of the model of Kim and Park [60]. We use some of the improvements for Kim and Park's model presented by Moccia et al. [74] in their second formulation.

To formulate the MIP-model we need some additional parameters:

- 0_k and $*_k$ for $k \in \mathcal{K}$ are dummy nodes that symbolize the first and the last job of crane k.
- e_i^{min} and e_i^{max} are the minimum and the maximum x-positions of the source and sink locations of transport job i.
- $\Phi := \{(i,j) \in \Omega \times \Omega; e_i^{min} < e_j^{max} + \omega\}$ is the set of job pairs that have to be considered to avoid crane crossings.
- $t_{ij}^+, t_{ij}^- \in \mathbb{R}$ for $(i,j) \in \Phi$ are the minimum necessary time differences between the end time of job i and the start time of job j to avoid crossings when job j is performed by a crane with a greater ordering number (t_{ij}^+) or a smaller ordering number (t_{ij}^-) than the crane that handles job i. Note that t_{ij}^+ and t_{ij}^- can have negative values which means that job j is allowed to start before job i ends or even before i has been started.
- $\Psi := \{(i,j) \in \Omega \times \Omega; e_i^{min} < e_j^{max} + \omega + \delta\}$ is the set of job pairs that have to be considered to count the undercuts of the crane safety distance δ between two cranes.
- $\tau_{ij}^+, \tau_{ij}^- \in \mathbb{R}$ for $(i,j) \in \Psi$ are the minimum necessary time differences between the end time of job i and the start time of job j to avoid that the minimum crane safety distance is undercut when job j is performed by a crane with a greater ordering number (τ_{ij}^+) or a smaller ordering number (τ_{ij}^-) than the crane that handles job i. Note, that τ_{ij}^+ and τ_{ij}^- (like t_{ij}^+ and t_{ij}^-) can have negative values which means that job j is allowed to start before job i ends or even before i has been started.
- $c_{ij} \in \{0,1\}$ for $i \in \Omega \cup \{0_k; k \in \mathcal{K}\}; j \in \Omega$ is one if job i (or the crane k initially for $i = 0_k$) has a different spreader setting than job j. So if c_{ij} is one, the spreader has to be changed between the processing of job i and j or the initial spreader setting of crane k needs to be changed to process job j (for $i = 0_k$).
- $\sigma_{ij} \in \{0,1,2\}$ for $i \in \Omega \cup \{0_k; k \in \mathcal{K}\}; j \in \Omega$ is the number of crane turnarounds when job j is handled directly after job i. Here, the necessary turnaround during the empty move between i and j as well as the turnarounds during processing j are counted. Again for $i = 0_k$ σ_{ij} is only the number of crane turnarounds during

processing of job i as we assume that the cranes do not have an initial moving direction.

- a_k^{min} and a_k^{max} are the borders of the (physically or operationally) limited working areas in x-direction of crane k. Note that e.g. the first of three cranes cannot reach the upper end of the terminal because then it would have to pass the two other cranes.
- $\mathcal{P} \subset \Omega \times \Omega$ is the set of precedences relations. If (i, j) is contained in $\mathcal{P}$ job i has to be finished before job j starts.
- $w_\tau \in \mathbb{R}$ for $\tau \in \{1, \ldots, 8\}$ are weighting factors for the soft constraints (S1) bis (S8)

To model the crane scheduling problem we consider the graph $G = (V, A)$ with $V = \Omega \cup \{0_k, *_k; k \in \mathcal{K}\}$ with 0_k and $*_k$ being dummy start and end jobs for crane k. Note that $A \subseteq V \times V$ only contains arc (i, j) if it is feasible to visit job j after job i by the same crane so if j is not a predecessor of job i. The parameters $d_{0_k i}$ and $t_{0_k i}$ for $i \in \Omega; k \in \mathcal{K}$ are the distances and travel times for crane k from its starting position to the source x-y-position of job i. The distances and travel times d_{i*_k} and t_{i*_k} for all jobs $i \in \Omega; k \in \mathcal{K}$ are set to zero because the end position of cranes is not predefined.

Before we introduce the decision variables and the MIP itself we discuss the properties of the parameters t_{ij}^+ and t_{ij}^- and how to calculate these parameters.

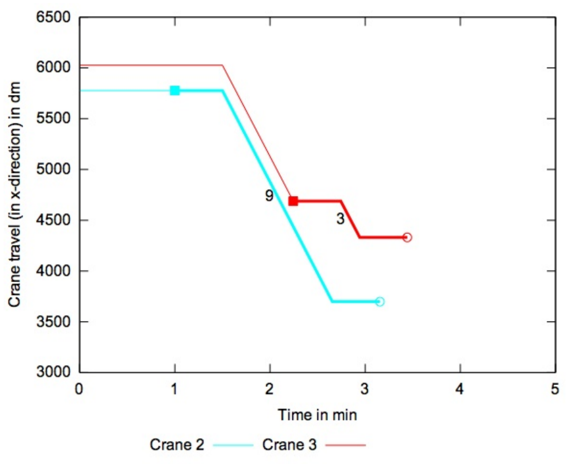
In Figure 6.5 we illustrate the need of different parameters for a greater (t_{ij}^+) and a smaller (t_{ij}^-) crane handling transport job j. For this purpose in Figure 6.5(a) the crane sequence of transport job 9 and 3 (cf. Example 6.2) is shown where job 3 is executed by a greater crane and Figure 6.5(a) shows the same sequence when job 3 is executed by a smaller crane. If job 3 is executed by a greater crane, job 3 can start during the processing of job 9. If on the other hand, job 3 is executed by a smaller crane, job 3 even cannot start immediately after completion of job 9.

As t_{ij}^+ and t_{ij}^- are the time-lags between the completion of job i and the start time of job j, in Figure 6.5 it is easy to see, that t_{93}^+ (in Figure 6.5(a)) is smaller than zero and t_{93}^- (in Figure 6.5(b)) is greater than zero. In the following we show how t_{93}^+ and t_{93}^- can be calculated:

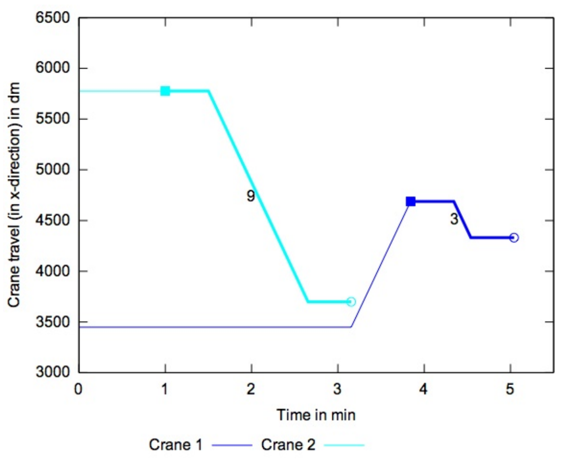
$$t_{93}^+ = -p_9 + \frac{b}{2} + \frac{s_9 + \delta - s_3}{v} = -2.15 + 0.5 + \frac{6002 - 4688}{1800} \approx -0.92$$

The first part of the sum is the negative of p_9 (which shifts the completion of job 9 to the start time of job 9). The second part of the sum is the half of the basic job duration (for lowering and lifting the spreader), while the third part is the time a crane needs to travel from sink x-position 6002 to x-position 4688. 6002 is the nearest greater x-position for crane 3 that allows (crane 2) to perform job 9 without a spatial conflict.

$$t_{93}^- = \frac{s_3 - s_9 - \delta}{v} = \frac{4688 - 3449}{1800} \approx 0.69$$



(a) t^+_{93}



(b) t^-_{93}

Figure 6.5: Different transport job time-lags according to crane ordering

For t^-_{93} the calculation is easier because only the travel time from the x-position without spatial conflict to the source position of job 3 has to be accounted.

For all $(i, j) \in \Phi$ we need to calculate the parameters t^+_{ij} and t^-_{ij}. To explain this calculation we introduce two definitions:

- $\theta(\varsigma)$ is defined as the crane travel time for a distance ς.
- $\eta_i := s_i - g_i$ is defined as source x-position minus sink x-position (of transport job i). If η_i is negative, the travel direction of transport job i is upwards and downwards otherwise.

By default t^+_{ij} and t^-_{ij} are set to $t_{ij} + \theta(\omega)$, which is the time a crane needs to travel from the sink x-position of transport job i to the source x-position of job j (t_{ij} plus the time needed to travel the minimum (hard) crane distance ω which is the width of a crane). Note, that this calculation is e.g. used for the parameter t^-_{93} in Example 6.2 (cf. Figure 6.5(b)). Furthermore, one has to distinguish three cases for transport jobs i with $\eta_i \geq 0$ where the direction of job j is downwards:

- $g_i - \omega < s_j < s_i + \omega$
  When the direction of job i is downwards and the source x-position of job j lies between the source and sink x-position of job i (extended by the crane width), the time-lag for a greater crane can be reduced to:

$$t^+_{ij} := -p_i + \theta(s_i - s_j + \omega) + \frac{b}{2}$$

  This situation and the calculation of t^+_{ij} is analog to the example t^+_{93} presented in Figure 6.5(a). The first part of the sum $(-p_i)$ shifts the reference moment from the completion of job i to the start time of job i. The second summand is the time needed for a greater crane to travel from the smallest feasible x-position during the start of job i (at s_i) to the source x-position of job j. That the greater crane has to wait till the load unit of job i is lifted up and the crane of transport job i starts to move in x-direction, causes the summand $\frac{b}{2}$. As one can see in Figure 6.5(b) in the considered situation, it is not feasible to reduce the default t^-_{ij}.

- $s_j - \omega < g_i < g_j + \omega$
  When the direction of job i is downwards and the sink x-position of job i lies between the x-source and sink position of job j (which then has to go upwards, cf Figure 6.6(a)), the time-lag for a smaller crane can be reduced to:

  $$t^-_{ij} := -\theta(g_i - s_j - \omega) - \frac{b}{2}$$

  The load unit of transport job j can be picked up ($-\frac{b}{2}$) and transported to the greatest feasible position of a smaller crane while i is finished at x-position g_i (first summand) before completion of job i.

- $g_j - \omega < s_i < s_j + \omega$
  When the direction of job i is downwards and the source x-position of job i lies between the x-source and sink position of job j (which is also downwards, cf. Figure 6.6(b)), we need to introduce a further parameter: $\alpha := \max(0, g_j - s_i)$, which is needed for situations where job j ends at a greater x-position than the source x-position of job i but within the distance of ω (hard crane distance) to the source x-position of job i. The time-lag for a greater crane can be reduced to:

  $$t^+_{ij} := -\theta(s_j - s_i - \omega + \alpha) - p_i$$

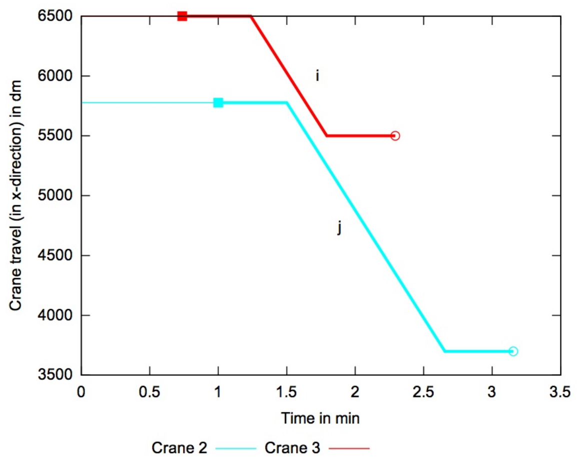
  Until the spreader of transport job i is lifted with the load unit, load unit of transport job j can be lifted up with the spreader and also be transported to the minimum feasible crane position while job i is started at s_i for a greater crane.

For the case of transport jobs i which are upwards, three cases have to be considered. These cases are symmetric to the cases for jobs which are downwards.

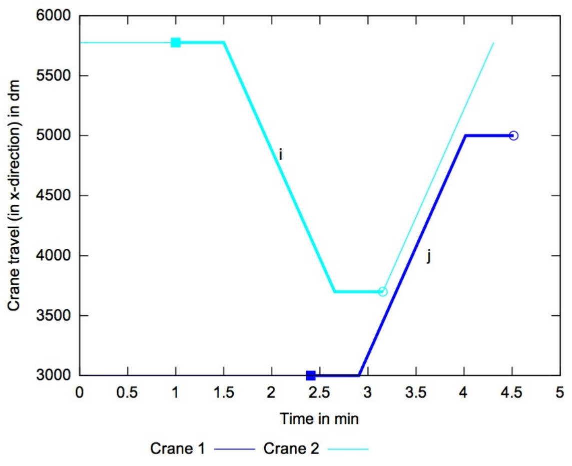
- $\eta_i < 0$ and $s_i - \omega < s_j < g_i + \omega$
  When the direction of job i is upwards and the source x-position of job j lies between the x-source and sink position of job i (see Figure 6.7(a) for an example), the time-lag for a smaller crane can be reduced to:

  $$t^-_{ij} := \theta(s_j - s_i + \omega) + \frac{b}{2} - p_i$$

  Transport job j can be started after the load unit of job i is lifted with the spreader ($\frac{b}{2}$) and the smaller crane can travel from the greatest feasible x-position to the source position of transport job j ($\theta(s_j - s_i + \omega)$). Again the summand $-p_i$ shifts the reference time to the start time of job i (from completion time).



(a) t_{ij}^{-} with $s_j - \omega < g_i < g_j + \omega$



(b) t_{ij}^{+} with $g_j - \omega < s_i < s_j + \omega$

Figure 6.6: Cases for special transport job time-lags

- $\eta_i < 0$ and $s_j - \omega < s_i < g_j + \omega$
  When the direction of job i is upwards and the source x-position of job i lies between the x-source and sink position of job j (see Figure 6.7(b) for an example), we again need to introduce a further parameter: $\beta := \max(0, s_i - s_j)$, which is needed for situations where job j ends at a greater x-position than the source x-position of job i but within the distance of ω (hard crane distance) to the source x-position of job i. The time-lag for a greater crane can be reduced to:

  $$t_{ij}^{+} := -\theta(s_i - s_j + \omega - \beta) - p_i$$

  Until the spreader of transport job i is lifted with the load unit, load unit of transport job j can be lifted up with the spreader also and be transported to the maximum feasible crane position for a smaller crane.

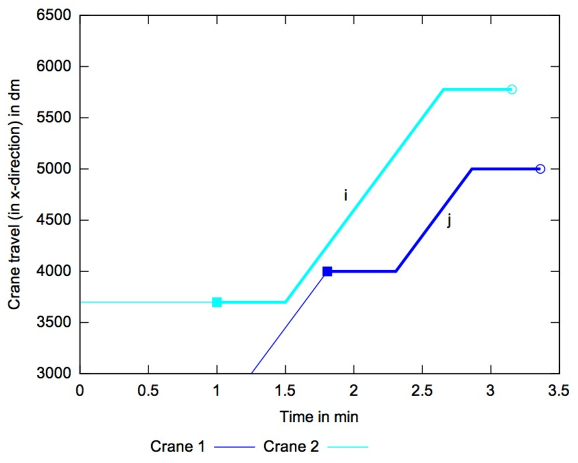
- $\eta_i < 0$ and $g_j - \omega < g_i < s_j + \omega$
  When the direction of job i is upwards and the sink x-position of job i lies between the x-source and sink position of job j (see Figure 6.7(c) for an example), the time-lag for a greater crane can be reduced to:

  $$t_{ij}^{+} := -\theta(s_j - g_i - \omega) - \frac{b}{2}$$

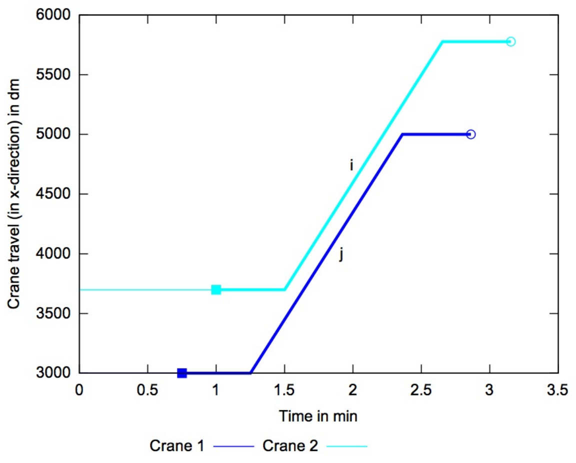
  The load unit of transport job j can be picked up $(-\frac{b}{2})$ and transported to the smallest feasible position of a greater crane (first summand) until completion of job i.

The calculation of the parameters τ_{ij}^{+} and τ_{ij}^{-} is analog to those of t_{ij}^{+} and t_{ij}^{-} but in all conditions, formulas and calculations ω has to be changed to $\omega + \delta$.

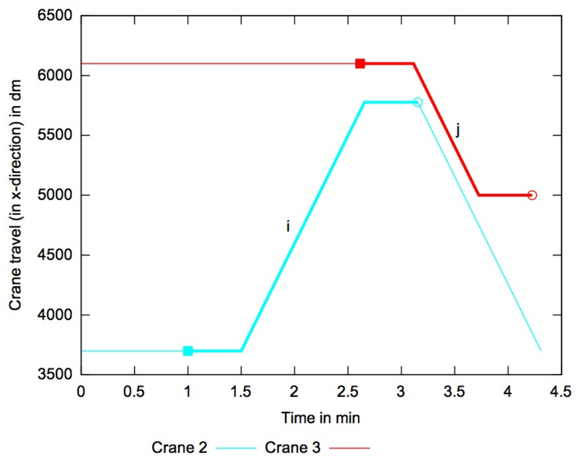
We use the following decision variables:



(a) t^-_{ij} with $s_i - \omega < s_j < g_i + \omega$



(b) t^+_{ij} with $s_j - \omega < s_i < g_j + \omega$



(c) t^+_{ij} with $g_j - \omega < g_i < s_j + \omega$

Figure 6.7: Further cases for special transport job time-lags

- $x^k_{ij} \in \{0, 1\} \forall i, j \in \Omega; k \in \mathcal{K}$ with

$$x^k_{ij} = \begin{cases} 1, & \text{if job } i \text{ is handled directly before } j \text{ by crane } k \\ 0, & \text{else.} \end{cases}$$

- S_i for all $i \in V$ is the start time of job i
- $z^+_{ij}, z^-_{ij} \in \{0, 1\}$ for $(i, j) \in \Phi$ with

$$z^+_{ij} = \begin{cases} 1, & \text{if job } i \text{ is processed before job } j \text{ and the ordering number of the} \\ & \text{assigned crane of job } j \text{ is greater than the one of job } i \\ 0, & \text{else.} \end{cases}$$

$$z^-_{ij} = \begin{cases} 1, & \text{if job } i \text{ is processed before job } j \text{ and the ordering number of the} \\ & \text{assigned crane of job } j \text{ is smaller than the one of job } i \\ 0, & \text{else.} \end{cases}$$

z_{ij}^+ and z_{ij}^- are used to avoid crossings of cranes (non-crossing-constraints).

- $\tilde{z}_{ij}^+, \tilde{z}_{ij}^- \in \{0,1\}$ for $(i,j) \in \Psi$ are defined analog to z_{ij}^+ and z_{ij}^- but are used to count and avoid the undercut of the soft crane safety distance δ.
- $y_i \in \{0,1\}$ for $i \in \Omega$ with

$$y_i = \begin{cases} 1, & \text{if the due date } d_i \text{ of job } i \text{ is violated} \\ 0, & \text{else.} \end{cases}$$

- $\tilde{y}_i \in \mathbb{R}^+$ for $i \in \Omega$ is the tardiness of job i or 0 if job i is not delayed
- $v_{ij} \in \{0,1\}$ for $(i,j) \in \Psi$ with

$$v_{ij} = \begin{cases} 1, & \text{if during the processing of jobs } i \text{ and } j \text{ the safety distance between two cranes} \\ & \text{is undercut} \\ 0, & \text{else.} \end{cases}$$

$$\min w_1 \cdot \sum_{k\in\mathcal{K}} \sum_{(i,j)\in A} d_{ij} \cdot x_{ij}^k \tag{6.1}$$

$$+w_2 \cdot \sum_{i\in\Omega} y_i \tag{6.2}$$

$$+w_3 \cdot \sum_{i\in\Omega} \tilde{y}_i \tag{6.3}$$

$$+w_4 \cdot \sum_{(i,j)\in\Psi} v_{ij} \tag{6.4}$$

$$+w_5 \cdot \sum_{i\in\mathcal{L}} (S_i - r_i) \tag{6.5}$$

$$+w_6 \cdot \sum_{i\in\Omega} \varphi_i \cdot (S_i - r_i) \tag{6.6}$$

$$+w_7 \cdot \sum_{k\in\mathcal{K}} \sum_{i\in\Omega\cup 0_k} \sum_{j\in\Omega} \sigma_{ij} \cdot x_{ij}^k \tag{6.7}$$

$$+w_8 \cdot \sum_{k\in\mathcal{K}} \sum_{(i,j)\in A} c_{ij} \cdot x_{ij}^k \tag{6.8}$$

s.t.

$$\sum_{j\in\Omega} x_{0_k j}^k = 1 \qquad (\forall k \in \mathcal{K}) \tag{6.9}$$

$$\sum_{i\in\Omega} x_{i*_k}^k = 1 \qquad (\forall k \in \mathcal{K}) \tag{6.10}$$

$$\sum_{k\in\mathcal{K}} \sum_{i\in\Omega\cup 0_k} x_{ij}^k = 1 \qquad (\forall j \in \Omega) \tag{6.11}$$

$$\sum_{j\in(\Omega\cup *_k)} x_{ij}^k - \sum_{j\in(\Omega\cup 0_k)} x_{ji}^k = 0 \qquad (\forall i\in\Omega; k\in\mathcal{K}) \tag{6.12}$$

$$S_i + p_i + t_{ij} - S_j - M(1 - x_{ij}^k) \le 0 \qquad (\forall i\in\Omega\cup 0_k; j\in\Omega; k\in\mathcal{K}) \tag{6.13}$$

$$S_i + p_i - S_j \le 0 \qquad (\forall (i,j)\in\mathcal{P}) \tag{6.14}$$

$$\sum_{h=1}^{k}\sum_{l\in(\Omega\cup 0_h)} x_{lj}^h + \sum_{h=k+1}^{m}\sum_{l\in(\Omega\cup 0_h)} x_{li}^h - z_{ij}^+ - z_{ij}^- \le 1 \qquad (\forall (i,j)\in\Phi; k\in\mathcal{K}) \tag{6.15}$$

$$S_i + p_i + t_{ij}^- - S_j - M(1 - z_{ij}^-) \le 0 \qquad (\forall (i,j)\in\Phi) \tag{6.16}$$

$$S_j + p_j + t_{ij}^+ - S_i - M(1 - z_{ij}^+) \le 0 \qquad (\forall (i,j)\in\Phi) \tag{6.17}$$

$$\sum_{h=1}^{k}\sum_{l\in(\Omega\cup 0_h)} x_{lj}^h + \sum_{h=k+1}^{m}\sum_{l\in(\Omega\cup 0_h)} x_{li}^h - \tilde{z}_{ij}^+ - \tilde{z}_{ij}^- \le 1 \qquad (\forall (i,j)\in\Psi; k\in\mathcal{K}) \tag{6.18}$$

$$S_i + p_i + \tau_{ij}^- - S_j - M(1 - \tilde{z}_{ij}^-) \le M\cdot v_{ij} \qquad (\forall (i,j)\in\Psi) \tag{6.19}$$

$$S_j + p_j + \tau_{ij}^+ - S_i - M(1 - \tilde{z}_{ij}^+) \le M\cdot v_{ij} \qquad (\forall (i,j)\in\Psi) \tag{6.20}$$

$$S_i \ge r_i \qquad (\forall i\in\Omega) \tag{6.21}$$

$$S_i + p_i - M\cdot y_i \le d_i \qquad (\forall i\in\Omega) \tag{6.22}$$

$$S_i + p_i - \tilde{y}_i \le d_i \qquad (\forall i\in\Omega) \tag{6.23}$$

$$\sum_{j\in\Omega\cup 0_k} x_{ji}^k = 0 \qquad (\forall i\in\Omega; k\in\mathcal{K}; e_i^{min} < a_k^{min}) \tag{6.24}$$

$$\sum_{j\in\Omega\cup 0_k} x_{ji}^k = 0 \qquad (\forall i\in\Omega; k\in\mathcal{K}; e_i^{max} > a_k^{max}) \tag{6.25}$$

$$S_i \in \mathbb{R}^+ \qquad (\forall i\in V) \tag{6.26}$$

$$x_{ij}^k \in \{0,1\} \qquad (\forall (i,j)\in A; k\in\mathcal{K}) \tag{6.27}$$

$$z_{ij}^+, z_{ij}^- \in \{0,1\} \qquad (\forall (i,j)\in\Phi) \tag{6.28}$$

$$\tilde{z}_{ij}^+, \tilde{z}_{ij}^- \in \{0,1\} \qquad (\forall (i,j)\in\Psi) \tag{6.29}$$

$$y_i \in \{0,1\} \qquad (\forall i\in\Omega) \tag{6.30}$$

$$\tilde{y}_i \in \mathbb{R}^+ \qquad (\forall i\in\Omega) \tag{6.31}$$

$$v_{ij} \in \{0,1\} \qquad (\forall (i,j)\in\Psi) \tag{6.32}$$

The objective function is a weighted sum of (6.1) the total length of the empty crane moves (S1), (6.2) the number of violated due dates (S2), (6.3) the total transport job delay (S3), (6.4) the number of undercuts of the minimum crane safety distance (S4), (6.5) the waiting times of truck (S5), (6.6) the waiting times of high priority jobs (S6), (6.7) the number of crane turnarounds (S7) and (6.8) the cost of crane spreader setting changes (S8).

The hard constraints begin with the flow constraints. Constraints (6.9) and (6.10) ensure that each crane starts with the corresponding initial job and ends with the dummy end job. This is important for the initial crane positions and the spreader setting costs for the

assignment of the first crane jobs. Furthermore, each job has to be planned exactly once (6.11). So each job is assigned to one position of a crane sequence. The next constraints (6.12) are the flow conservation constraints. Together with (6.9) and (6.10) they ensure that for each crane there is a path from the initial to the dummy end job.
Constraints (6.13) represent the time restrictions for jobs that are handled by the same crane. If two jobs are processed directly after each other by the same crane, the later job is not allowed to start before the earlier job is finished and the crane moved from the sink x-y-position of the earlier job to the source x-y-position of the later job (H3). If job i is a predecessor of job j, constraints (6.14) ensure that job i is finished before job j starts (H4).
Constraints (6.15), (6.16) and (6.17) are the non-crossing constraints (H1). In constraint (6.15) $\sum_{h=1}^{k}\sum_{l\in(\Omega\cup 0_h)} x_{lj}^h + \sum_{h=k+1}^{m}\sum_{l\in(\Omega\cup 0_h)} x_{li}^h$ is two if a job j is assigned to a lower crane than a job i with $(i,j)\in\Phi$ which implies that $e_i^{min} < e_j^{max} + \omega$. This means that job i takes place at the same or at a lower place than job j, so the assignment of job i to a higher crane causes a crossing conflict. In a situation of a crossing conflict either z_{ij}^- or z_{ij}^+ has to be chosen to one. This implies together with constraints (6.16) and (6.17) that the crossing is avoided due to the time-lag between the starting times S_i and S_j. The necessary time-lags are given by t_{ij}^+ and t_{ij}^-. So with (6.15) the variables z_{ij}^- or z_{ij}^+ are set to one for jobs i and j with spatial conflicts due to their crane assignment and (6.16) and (6.17) ensure a temporal distance for jobs i and j where either z_{ij}^- or z_{ij}^+ is one which is equivalent to a spatial conflict of jobs i and j.
The number of crane safety distance conflicts is counted by constraints (6.18), (6.19) and (6.20). The principles are analog to those of the non-crossing constraints ((6.15), (6.16) and (6.17)) but constraints (6.19) and (6.20) are not hard and violation is counted by v_{ij}. Due to constraint (6.21) release dates have to be respected (H2). The number of violations of due dates is measured in constraint (6.22) so these constraints (together with the objective element (6.2)) define the variables y_i while the transport jobs tardiness is measured in (6.23) by variable $\tilde{y}_i$. Constraints (6.24) and (6.25) restrict the crane working areas by excluding jobs from the crane sequences that cannot be handled by the crane (H5). Finally in (6.26) to (6.32) the domains of all variables are defined.

In our MIP-formulation a big M occurs which is needed for temporal constraints. So M has to be greater than the latest transport job completion time, which has to be estimated due to the number ($|\Omega|$), duration (p_i) and release dates (r_i) of jobs as well as due to precedence relationships ($\mathcal{P}$).

In the described MIP-formulation "give way" crane moves to enable other cranes to process a transport job are not modeled. An example for this problem is shown in Figure 6.8 which is the best solution of the presented MIP-formulation for the crane planning example in 6.2 computed by CPLEX in about one minute. After processing of transport job 9 crane 3 has to move to the source position of job 6. Otherwise, there is a spatial conflict with crane 2, which moves to the source position of transport job 3. This "give way" move does not change the objective value because crane 3 anyway has to move to the source x-position of job 6.
But in other situations "give way" moves do have a contribution to the objective value. Consider the solution presented in Example 6.2 in Figure 4.1. Our crane planning MIP-model does not model the "give way" moves of crane 2 that enables crane 3 to perform

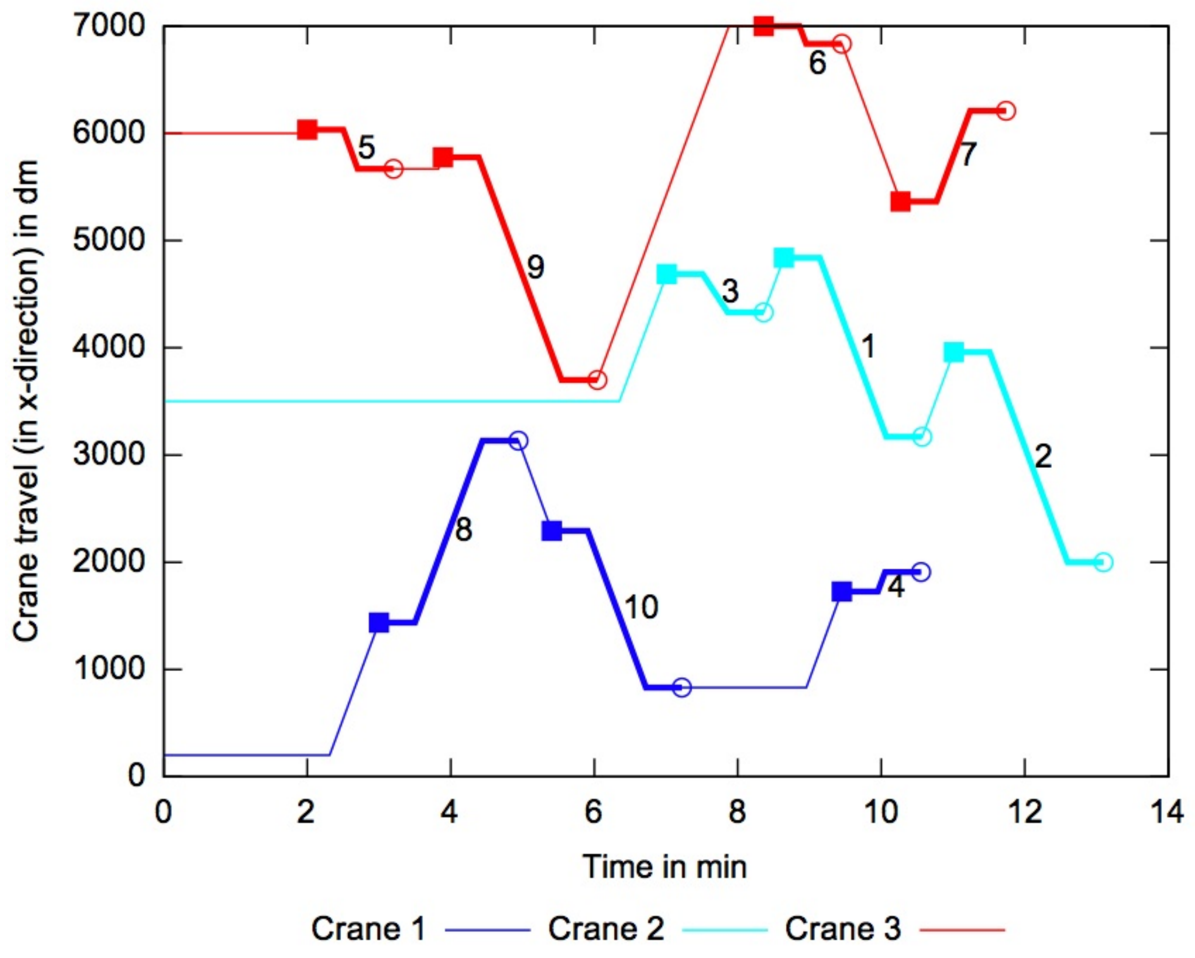


Figure 6.8: Best solution provided by CPLEX using the MIP-formulation for the instance of Example 6.2

transport job 1. So the costs for the length of the empty moves (S1) would not be measured correctly.

A further problem is that the crane ordering might get disturbed or the distance of one dm between cranes might be undercut at the end of the planning situation. Consider again the solution plotted in Figure 6.8. If crane 1 stays at its last x-position, which is the sink x-position of job four (1909 dm), crane 2 is not able to reach the sink x-position of job two (2000 dm) because of the crane width of 250 dm. So to obtain a feasible crane schedule, crane 1 has to perform a "give way" move after the processing of job four which changes the objective value.

We think that it is necessary to model the crane positions at any time points, if all "give way" moves shall be modelled. Such a spatial crane MIP-model is introduced by Ng [75] but no computational results are reported for the model. We reimplemented the model and tested it. Unfortunately, it was not possible to solve instances with more than 4 transport jobs. As we additionally never found "give way" moves with additional costs in solutions provided by CPLEX for our practical instances, we decided not to use the spatial crane MIP presented by Ng nor another MIP-model which determines crane positions.

6.4 List scheduling heuristic

In this section we introduce our heuristic approaches. We first present a scheduler which is afterwards used in a local search list scheduling to generate solutions from a transport job list. To further improve the solution quality we use the metaheuristics Tabu Search and Simulated Annealing.

We propose a list scheduling heuristic for the crane planning. A sequence of the transport jobs for all cranes is processed to a crane plan by a greedy heuristic. A local search is used to change and improve this sequence. We only consider sequences that fulfill the precedence constraints.

To create an initial feasible sequence one can add all jobs to the sequence iteratively. When a job is added to the sequence, it is recursively checked whether the predecessors of the job are already contained in the sequence. If not, the predecessors are added to the sequence first. Other options are sorted or random sequencing of jobs (however the precedence constraints have to be checked). To sort the transport jobs initially source or sink x-positions (s_i and g_i), the job priorities (ϕ_i), or release or due dates may be used.

Algorithm 6.1 List scheduling.

```
1:  set crane properties to initial state
2:  objective := 0;
3:  for $i$ in SEQUENCE do
4:     for $k$ in $\mathcal{K}$ do
5:        if crane $k$ can handle job $i$ then
6:           calculate EST;
7:           calculate objectiveChange[$k$];
8:        end if
9:     end for
10:    Choose best crane $k_{best}$ and update its properties;
11:    objective += objectiveChange[$k_{best}$];
12: end for
13: return objective;
```

A pseudo code of the scheduler is shown in Algorithm 6.1. The scheduler plans the transport jobs according to the global scheduling sequence iteratively (3). For all cranes an earliest start time (EST) for the transport job is calculated (6), if the crane is able to handle the job. The release date is the earliest possible start time, but it can be delayed due to the travel from the sink x-y-position of the last job crane k transported. Delays may also be caused by spatial conflicts with other cranes or precedence constraints. According to the EST for crane k the change of the objective value caused by assigning the next transport job to crane k is evaluated (7). The costs for job waiting times and due date violation as well as the crane moves, set up costs and violations of the soft crane distance are accounted. The transport job is assigned to a crane with the minimum objective change. The attributes of the selected crane like ready time, settings and x-y-position are updated afterwards. For the beginning initial ready times and positions are part of the input data. Finally, the scheduler returns the objective value which is the sum of the objective changes of the assigned cranes for every transport job.

To improve the objective value the sequence is usually varied by a local search. A local search is a search in the solution space. From a current solution neighbor solutions are explored and under special conditions chosen as the new current solution. Neighborhoods are usually defined by operators or moves that modify the current solution partially. In the case of a list scheduling the scheduling list is modified. The modified list may lead to a different solution, but it may also be the case that the modification of the list does not change the solution produced by the scheduler. For the crane scheduling the neighborhoods for this local search (on the scheduling list) are forward and backward shift of a transport job and the swap of two transport jobs. We restrict the search to feasible solutions so neighbors can only be chosen, if the resulting sequence respects the precedence constraints.
For a swap of jobs at positions i and j in the sequence one has to check if none of the elements from positions i to $j-1$ is a predecessor of element j and that i is not a predecessor of any sequence element at positions $i+1$ to j. This is analog for the shift operators.

If one uses the simplest local search strategy (often called iterative improvement or hill climbing, which only accepts better neighbor solutions) the risk of ending in a local optimum is high. Furthermore such local optima cannot left behind easily by accepting solutions that are worse than the current solution because often cycles occur. A widely used alternative is to choose metaheuristic approaches to obtain better solutions. In the following we describe the metaheuristics tabu search and Simulated Annealing for the list scheduling.

The idea of tabu search is to accept better and worse solutions and avoid cycles by setting already visited solutions tabu and forbid to choose such solutions. A solution which is tabu may only be chosen, if it satisfies an aspiration criterion (e.g. the solution has a better objective than the best solution found so far, which is the solution s^*). Which solutions are tabu is saved in a tabu list. Such a tabu list typically does not contain complete solutions but attributes that describe a solution. Furthermore, a solution is not set tabu forever. After some time the tabu status of attributes elapses. This is necessary because with attributes not only visited solutions are set tabu, but also solutions that have never been explored.
A tabu search procedure is shown in Algorithm 6.2. For step 7 different options like first fit (accept first improving solution) or best fit (explore all neighbor solutions and choose the best one) exist. Note, that for each evaluation of a neighbor solution which is given by a special sequence, a run of the scheduler is necessary.

For the crane scheduling problem we choose a best fit tabu search. To avoid cycles during the tabu search, we use forward and backward tabu lists for the sequence positions suggested by Glover and Laguna [48]. When a sequence position is forward tabu, it is

Algorithm 6.2 Tabu-search pseudo code. Adapted from Brucker and Knust [25].

```
 1: generate an initial solution $s \in S$ ;
 2: $best$ := c($s$);
 3: $s^* := s$;
 4: $TL := \emptyset$
 5: repeat
 6:     $Cand(s) := \{s' \in \mathcal{N}(s) |$ the move from $s$ to $s'$ is not tabu or $s'$ satisfies an aspiration criterion$\}$;
 7:     Choose a solution $s' \in Cand(s)$;
 8:     Update the tabu list $TL$;
 9:     $s := s'$;
10:     if c($s'$) $> best$ then
11:         $s^* := s'$;
12:         $best$ := c($s'$);
13:     end if
14: until a stopping condition is satisfied;
```

not allowed to move the element at a position forward and vice versa for the backward tabu list. The swap of two elements is for the tabu lists interpreted as a simultaneous forward shift of one and a backward shift of the other element. Furthermore, we decided to consider a tabu list with variable length. The initial length is set to $tabu_{ini}$. Each time the solution can be improved, the length of the list is reduced by one. On the other hand, the length of the tabu list is increased by one, if a worse solution is chosen. The length of the tabu list is further limited to the interval $[tabu_{min}; tabu_{max}]$ (with $tabu_{min} < tabu_{max}$). Our Tabu Search terminates after a fixed number of iterations (which also can be called steps).

While tabu search aims to avoid to return to local optima by forbidden moves, Simulated Annealing uses stochastic elements for the same aim. As shown in Algorithm 6.3 during Simulated Annealing random solutions are generated. These solutions are accepted with a probability of $\min\{1, e^{\frac{c(s')-c(s)}{t_i}}\}$ calculated using a sequence t_i with $\lim_{i\to\infty} t_i = 0$. t_i is interpreted as the temperature of a cooling process. When the temperature is high, worse solutions are accepted with a higher probability. Furthermore, the probability for a worse solution to be chosen is higher, the smaller the difference between the objective of the new solution and the objective of the current solution is. Note that solutions with a better objective are always accepted.

For the crane scheduling we define a sequence t_i as follows. The initial temperature $t_1 := t_{ini}$ is set to an input value (where different values will be tested). Furthermore, we introduce a cooling factor $c < 1$ which is close to one. We define a calculation step where t_i is calculated using t_{i-1} (for $i \geq 2$):

$$t_i = t_{i-1} \cdot c.$$

As stopping criterion we chose $t_i < \epsilon$ and tested different values for ϵ.

Algorithm 6.3 Simulated Annealing pseudo code (Brucker and Knust [25]).

```
1: $i$: $= 1$;
2: generate a initial solution $s \in S$ ;
3: $best$ := c($s$);
4: $s^*$ := $s$;
5: **repeat**
6:    Generate randomly a solution $s' \in \mathcal{N}(s)$;
7:    **if** Rand(0,1) $< \min\{1, e^{\frac{c(s')-c(s)}{t_i}}\}$ **then**
8:       $s := s'$;
9:       **if** c($s'$) $>$ $best$ **then**
10:         $s^* := s'$;
11:         $best$ := c($s'$);
12:      **end if**
13:   **end if**
14:   $i := i + 1$;
15: **until** a stopping condition is satisfied ;
```

6.5 Computational results

In this section we report results of our algorithms for the crane planning for instances which are the output of the storage planning heuristic, applied on real world data from German rail-road terminals. In addition, we consider also small generated instances for which the MIP-model can be solved. In the generated instances time windows and high priority jobs play a more important role than in the practical data sets.
Firstly, we will report some tests on initial sorting sequences and parameter settings of the Tabu Search and the Simulated Annealing applied to the practical instances. In this context we will also compare the results of the metaheuristics with a simple Hill Climber local search (with settings first fit and best fit).
Subsequently, we test the metaheuristics on all 4 data sets of practical instances which are the output of the storage planning. Furthermore, beside operational crane working areas we also tested our heuristics only considering physical crane limitations.

One group of tests of our crane planning algorithms is done using the output of the storage planning heuristic (cf. Section 3). So, we consider a situation where the load units of several trains have to be unloaded to the storage area and optimized storage positions are selected to minimize the total length of the loaded crane moves. We think that with this approach quite realistic properties of the transport jobs are considered. The total length of the loaded crane moves is minimized which implies that if possible the cranes will travel only short distances in x-direction, while the cranes are loaded. But also long distances occur, if no feasible position with short length of the loaded move is available. Note, that for some instances the storage planning heuristic could not find a feasible solution. For these storage planning instances, we generated a crane planning instance where just the load units have to be transfered which could be positioned feasibly. This slightly reduces the number of transport jobs compared to the number of load units on the trains in the storage planning problem.

In the following, we describe how we generated instances for additional tests of our crane planning algorithms. The basic input for the generate routine are the number of transport jobs $|\Omega|$ and the length of the terminal L as well as a maximum length of loaded moves in x-direction. We generate the y-positions of the transport jobs randomly and independent. The source x-positions (s_i) are also generated randomly. To determine the sink x-positions we randomly determine the direction of each transport job and furthermore a distance under load smaller than the maximum value. If the x-position is within the terminal (non-negative and not larger than L), it is directly chosen. Otherwise, for negative values 0 is selected and for too large values the upper end of the terminal ($g_i = L$). Furthermore, random release dates (not greater than 10 min) and due dates greater than the release dates (at least 3 min and up to 15 min later) as well as a priority out of $\{0, 1, 2, 3\}$ and a corner casting type out of $\{1, 2, 3, 4\}$ are generated. Finally, it is generated, whether each transport job is delivered or fetched by truck which is also relevant for the objective value.

In the input data for the crane planning derived from the storage planning the transport jobs are ordered hierarchically by trains and increasing source x-positions (s_i) due to the input of the storage planning heuristic. With these input sequences we generated feasible scheduling lists, by first considering predecessors recursively, if there are predecessors. As alternative basis for the initial sequences we tested sortings of the transport jobs by source (s_i) and sink x-positions (g_i) as well as by the sum of the values of source and sink x-positions ($s_i + g_i$).

In Table 6.6 we report the objective values and runtimes for different heuristics and initial sortings. In the columns the different sortings are presented. The lines contain different heuristic settings. As sortings we tested the output of the storage planning (train, s_i ($\leq$)). This sorting means that the transport jobs are sorted hirarchically by the trains the load units come from and increasing by the load units x-positions, within the transport jobs of a single train. Further columns contain sortings by x-positions described by s_i for source, g_i for sink position and by the sum of both s_i+g_i. Here, ($\leq$) means increasing sorting and ($\geq$) means decreasing sorting. For each of the sortings, the objective and the runtime in seconds are listed. Accountable differences for the different sortings can only be observed for the scheduler (without local search) and for the Tabu Search with only 10 steps. Here the relative deviations between the best and the worst sorting are 7.2% (scheduler) and 3.3% (Tabu Search). For all other settings of heuristics and metaheuristics the differences are below 0.5%. So, for further computations we will not consider different sortings and take the default sorting for all heuristics. Note, that we let run Simulated Annealing always for five times to build a mean value of the objective to handle the probability which is part of the calculation in the Simulated Annealing algorithm.
For Simulated Annealing we also tested smaller or larger values of t_{ini}. For $t_{ini} = 10$ and $t_{ini} = 50$ the objectives were slightly worse and the mean runtimes were the same. For larger values up to $t_{ini} = 100.000$ no improvement of the mean solution quality could be achieved, but the runtime was about the double for $t_{ini} = 100.000$ compared to $t_{ini} = 100$. For Tabu Search we also tested other values of $tabu_{min}, tabu_{ini}$ and $tabu_{max}$ but within a runtime of less than 10 min we could not found greater or smaller values for the parameters that lead to a better solution quality. With a larger number of iterations the runtime increases, but also with different values for $tabu_{min}, tabu_{ini}$ and $tabu_{max}$ no improvement

heuristic	train, s_i ($\leq$)		s_i ($\leq$)		s_i ($\geq$)		g_i ($\leq$)		g_i ($\geq$)		$s_i + g_i$ ($\leq$)		$s_i + g_i$ ($\geq$)	
	obj.	t	obj.	t	obj.	t	obj.	t	obj.	t	obj.	t	obj.	t
scheduler	1336.8	0	1251.2	0	1305.8	0	1310.5	0	1247.2	0	1270.5	0	1266.5	0
fist fit Hill Climber	1167.0	5	1169.3	5	1167.8	7	1170.0	6	1165.3	4	1169.7	5	1168.8	5
best fit Hill Climber	1163.3	11	1166.8	9	1166.0	10	1167.1	9	1165.7	9	1167.6	9	1164.6	9
TABU Search ($tabu_{min} = 20$ $tabu_{ini} = 80$ $tabu_{max} = 200$														
10 steps	1236.9	2	1201.7	1	1210.2	1	1211.9	1	1197.0	1	1200.9	1	1201.3	1
50 steps	1165.9	9	1167.5	9	1168.6	9	1168.5	8	1167.3	9	1168.2	9	1165.5	9
200 steps	1163.1	35	1165.5	35	1165.8	35	1166.6	35	1165.1	35	1167.1	35	1163.8	35
Simulated Annealing ($t_{ini} = 100$)														
$\epsilon = 0.001$, $c = 0.999$	1173.1	1	1170.3	1	1172.1	1	1173.8	1	1172.9	1	1171.9	1	1172.1	1
$\epsilon = 0.0001$, $c = 0.999$	1171.8	1	1171.5	1	1171.3	1	1170.6	1	1171.6	1	1168.0	1	1170.3	1
$\epsilon = 0.0001$, $c = 0.9999$	1160.8	9	1159.0	9	1160.1	9	1159.7	9	1160.7	9	1160.3	9	1161.6	9

Table 6.6: Computational results of the crane planning for different initial sequences and heuristic settings for storage planning data set II

of the solution quality could be obtained. Compared to a best fit Hill Climber heuristic, Tabu Search can only slightly improve the solution quality, if we do not consider much larger runtimes.

With Simulated Annealing for the in Table 6.6 considered instances with the setting $\epsilon = 0.0001$, $c = 0.9999$ the best objectives compared to all other heuristics could be obtained in a short runtime of below 10 s. The data set consists of two groups of instances: the first 8 instances contain $|\Omega| = 49$ transport jobs and the last 8 instances contain $|\Omega| = 70$ transport jobs. Note, that considering these instances with different number of load units is motivated for the storage planning by the equal size of the terminal (number of lanes and terminal length L). Tabu Search with 200 steps took 17 s (± 1 s) for the instances with $|\Omega| = 49$ transport jobs and 54 s (± 1 s) for the instances with $|\Omega| = 70$ transport jobs. For Simulated Annealing the differences of the runtimes are smaller with 7 s (± 1 s) and 10 s (± 1 s) for the values $\epsilon = 0.0001$ and $c = 0.9999$. So, for further tests, we decided to re-group the practical instances such that a data set contains instances with the same number of transport jobs. Note, that this can be done since the size – especially the number of storage lanes – of the terminal is not of relevance for the crane planning. We built four groups of data sets with 12 instances containing 49, 70, 112, and 160 transport jobs, respectively. We call these data sets A, B, C and D (ordered by increasing number of transport jobs). Note, that for data sets A and B the source and sink x-positions are all smaller than 2501 dm ($0 \leq s_i \leq 2500$, $0 \leq g_i \leq 2500$), but for the data sets C and D x-positions are all over the terminal ($0 \leq s_i \leq 7000$, $0 \leq g_i \leq 7000$).

heuristic	first fit HC		best fit HC		Tabu Search		Simulated Annealing	
	obj.	t	obj.	t	obj.	t	obj.	t
$m = 1$	673.6	1	671.3	0	671.1	17	669.9	6
$m = 2$, operational	210.8	1	209.6	4	209.6	27	209.5	7
$m = 2$, physical	214.6	3	211.3	7	211.2	45	211.2	36

Table 6.7: Computational results for data set A

In the following we show some results for the new data set A. We tested first fit and best fit local search as well as Tabu Search and Simulated Annealing. Furthermore, we varied the number of cranes and the crane working areas. The first tests (cf. first line Table 6.7) are with one crane, a further test is with two cranes with operational working areas (0 to 1500 and 1000 to 2500) as well as a setting with two cranes and physical working areas (0 to 2250 and 250 to 2500). The results are presented in Table 6.7. The huge reduction of the objective value for two cranes is mainly caused by less total tardiness and a smaller number of delayed jobs. Unfortunately, in the short amount of runtime with the Tabu Search only for very few instances the solution quality of the best fit Hill Climber could be outperformed. Simulated Annealing could outperform or at least reach the solution quality of the best fit Hill Climber, but the computation times are a bit longer. To check whether, if the greedy crane assignment in the scheduler has negative effects, we also tested a scheduler where not only a list but also a crane assignment is processed to a crane schedule. For this purpose, we extended the neighborhood by an operator which changes the crane assignment of one job. Using this modified scheduler and the additional neighborhood the results for two cranes ($m = 2$) and physical crane working areas could only improve the solution quality by less than 0.5% for Tabu Search and Simulated Annealing (in the same settings). The runtime is reduced to about the half. This is the case, as in the scheduler only for the assigned crane a start time and objective has to be calculated. For the greedy scheduler and physical crane working areas for almost all jobs the impact of assigning the job to all cranes has to be evaluated.

The evaluations of data sets B are done analogously to data set A and are presented in Table 6.8. Here, again due to temporal constraints the objective is much smaller for two cranes. For two cranes with both crane area policies Simulated Annealing could not reach the solution quality of the best fit Hill Climber or the Tabu Search.

heuristic	first fit HC		best fit HC		Tabu Search		Simulated Annealing	
	obj.	t	obj.	t	obj.	t	obj.	t
$m = 1$	1611.6	9	1609.0	18	1608.8	54	1602.4	10
$m = 2$, operational	652.2	14	608.0	25	607.5	94	625.9	14
$m = 2$, physical	619.4	23	615.0	45	614.0	163	618.3	85

Table 6.8: Computational results for data set B

The computational results for data set C are presented in Table 6.9. As we expected larger runtimes due to the higher number of transport jobs, we reduced the number of steps for Tabu Search to 150 and the parameter c for Simulated Annealing to $c = 0.999$. The trend of differences in the solution quality of the different approaches between data sets A and B continues. For data set C Simulated Annealing is even not able to outperform the solution quality of the first fit Hill Climber.

heuristic	first fit HC		best fit HC		Tabu Search		Simulated Annealing	
	obj.	t	obj.	t	obj.	t	obj.	t
$m = 3$, operational	988.4	57	983.4	167	983.7	423	1010.0	39
$m = 3$, physical	1012.8	277	997.1	589	997.5	1631	1028.0	1303

Table 6.9: Computational results for data set C

For the largest instance set D we only computed a first and a best fit Hill Climber heuristic. The first fit Hill Climber took about 800 s runtime and the best fit Hill Climber even took about 1050 s for operational crane working areas. According the mean solution quality best fit was better, but for some instances first fit could outperform best fit. For physical working areas again the computation took much longer runtime. The mean solution quality is with 2863.8 worse for physical crane working areas, if the scheduler and neighborhood with fixed crane assignment are used. On the other hand, the runtime can be reduced to a mean value of 3643 s to 1706 s.

The MIP presented in Section 6.3 could not be solved to optimality nor provide feasible solutions for the smallest data set A. So, we generated 20 small instances consisting of 12 transport jobs with our instance generator and solved them using Gurobi 4.0. The transport job have source and sink x-positions which can be anywhere in the terminal ($0 \leq s_i \leq 7000$, $0 \leq g_i \leq 7000$) and the maximum length of the loaded crane moves is limited to 3000 dm. Furthermore, the instances are planned for three cranes. Table

heuristic	first fit HC		best fit HC	
	obj.	t	obj.	t
$m = 3$, operational	2508.4	784	2356.6	1048
$m = 3$, physical	2497.5	1937	2378.7	3643

Table 6.10: Computational results for data set D

6.11 shows mean objective values and mean runtimes over the 12 instance. Again for each instance 5 runs of the Simulated Annealing are performed. The mean as well as the minimum and maximum objectives for Simulated Annealing are reported. The instances with operational crane areas (as defined in Table 6.2) can be solved faster than the instances with physical crane working areas (as defined in Table 6.3) with the MIP and Simulated Annealing. Simulated Annealing is able to compute solutions of high quality (gap of less than 2% to optimality) in a few seconds. The quality of the Tabu Search is worse and the computation takes longer (about three times).

	MIP		Simulated Annealing				Tabu Search	
	objective	t	mean (gap)	min.	max.	t	objective	t
operational	72.3	143.2	73.6 (1.9 %)	73.6	73.7	3.1	83.7 (15.0 %)	11.7
physysical	61.5	344.0	62.2 (1.0 %)	62.2	62.2	3.6	68.1 (10.5 %)	8.6

Table 6.11: Results for small generated crane planning instances

We also tested the heuristics with a larger generated data set. This contains 60 transport jobs and the results are presented in Table 6.12. Here, the tendency of the small results approves. Even the quality of the worse Simulated Annealing run is better than the quality of the Tabu Search. Furthermore, the runtime difference is higher (21 s compared to nearly 500 s and 30 s compared to above 500 s)

	Simulated Annealing				Tabu Search	
	mean	min.	max.	t	objective	t
operational	1478.8	1472.3	1492.3	21.1	1524.1	498.3
physical	1327.6	1310.7	1355.6	30.1	1380.5	524.7

Table 6.12: Results for generated crane planning instances

6.6 Discussion of the results

In this section we give a short conclusion of the computational results of the crane planning.

First of all, one can see, that for our instances the proposed initial sortings do not strongly influence the final quality of our local search algorithms. With the tight runtime limits we considered, unfortunately Tabu Search was only for very few instances able to leave the local optima, which can also be reached by a best fit Hill Climber. So, Tabu Search

is only interesting, if much higher runtimes are allowed.
According to Simulated Annealing it turned out, that it is able to outperform the Hill Climber for the smaller data sets. But with increasing number of cranes and jobs, the runtime of Simulated Annealing increases and the solution quality is worse than that of the Hill Climber.
Furthermore, one can see that the complexity of the computation raises, if physical crane working areas are considered (compared to operational working areas). For the optimization in practice it might be interesting to use fixed job to crane assignments in combination with a neighborhood that changes crane assignments. This modeling has the advantage that a comparable solution quality can be achieved in less runtime. Here, the differences in terms of runtime for the different crane working areas are also not so large.
Finally, we have to say that if instances with more than 150 transport jobs have to be optimized and the runtime shall be very short, even the easiest local search procedures may not be fast enough. So, one may think about a decomposition approach, which first plans the cranes individually and afterwards integrates non-crossing constraints.

6.7 List of crane planning notation

$\mathcal{K} = \{1, \ldots, m\}$	set of RMGs (indices k, h)
Ω	set of transport jobs (indices i, j, l)
s_i, $\tilde{s_i}$	source x- and y-position of job i
g_i, $\tilde{g_i}$	sink x- and y-position of job i
φ_i	priority of transport job i
r_i	release date of transport job i
d_i	due date of transport job i
p_i	process time of transport job i
ω	uniform crane width
b	uniform transport time for spreader moves
d_{ij}	length of empty crane moves between transport jobs i and j
t_{ij}	time for empty move between transport jobs i and j
S_i	Start time of transport job i
0_k,$*_k$	dummy start and end job of crane k
e_i^{max},e_i^{min}	the minimum and the maximum x-positions of transport job i
Φ	set of job pairs that have to be considered to avoid crane crossings
t_{ij}^+,t_{ij}^-	time-lags between transport jobs i and j to avoid crossings
δ	minimum soft crane safety distance
Ψ	set of job pairs that have to be considered to avoid undercut of minimum soft crane safety distance
τ_{ij}^+,τ_{ij}^-	time-lags between transport jobs i and j to count soft crane distance violations
c_{ij}	spreader set-up cost, if job i is followed by j
σ_{ij}	number of crane turnarounds, if job i is followed j
$\mathcal{P}$	set of precedences relations

$\mathcal{L}$	set of transport jobs that are delivered or fetched by truck
a_k^{min},a_k^{max}	the minimum and maximum x-position that crane k can reach
$\theta(\varsigma)$	the travel time of a crane for distance ς
η_i	the direction of a transport job, indicated by the sign ($\eta_i = s_i - g_i$)
L	length of the terminal
w_τ	weighting factors for soft constraints

Table 6.13: List of crane planning notations

7 Conclusions

Various aspects of optimization in rail-road or continental intermodal transportation are of interest from a scientific point of view and getting more and more attraction from the scientific community. For the operational planning of rail-road intermodal transportation one of the most interesting aspects is the transfer of load units in rail-road terminals. In addition these planning problems are of high interest, since different companies start to build software solutions that support practitioners with optimized plans for the operational optimization in rail-road terminals.
Three subproblems of the operational planning in rail-road terminals are addressed in this thesis – storage planning, load planning (of trains) and crane planning. All subproblems are treated in a deterministic setting and the load planning is also done under different uncertainties. For all subproblems we were able to provide solution methods that obtain relatively good solutions in a short amount of runtime for the planning situation in a real-word setting. Even though we modeled all relevant constraints for typical German rail-road terminals in detail.

For the **storage planning** beside the optimization approaches for the practical planning problem we introduced a three-field notation to categorize storage problems in rail-road terminals. For several settings of the storage planning problem we provided complexity results. We can show that some practical problems can be solved in polynomial time, but most practical problems are NP-complete. For the practical situation in German rail-road terminals we developed different MIP-models and a fast heuristic, which organize the data representation of the storage area in trees.

For the **load planning** of trains we first introduced all the technical and physical details necessary to understand and model the practical railway-wagon restrictions. We presented different IP- and MIP-models. The best one, which models all practical restrictions, can be solved in a few seconds even with non-commercial MIP-solvers for realistic train sizes.

As the load planning in practice often has to be done before all load units are delivered to the terminal by trucks, we also conducted different robustness approaches for three types of practical uncertainties. For the **robust load planning** we provided MIP-models (where some can only be used as MIP-heuristic) for all types of uncertainties and achieved relatively large gains, within a runtime of some minutes.

Finally, we considered the **crane planning**, which actually plans the transfer of load units by the cranes in the terminal. Beside a detailed description and modeling of the non-crossing constraints in the situation of deviating source and sink x-positions of load units, we provided a MIP-formulation to exactly introduce the model. We developed again fast heuristics which provide good solutions in a runtime of less than one minute. For offline planning situations, where more time for the optimization is available, the heuristics can also be used in other settings (with greater runtimes) to even further improve the quality of solutions.

For further research we see different interesting directions. For the load planning this are on the one hand other robustness approaches (e.g. recoverable robustness). On the other

hand, with the increasing importance of planned or already built hub terminals, it would be interesting to analyze the multiple load planning. The multiple load planning considers the load planning at the terminals which load trains that run to hub terminals and the load planning in the hub terminal, where the load units of all trains are rearranged. So, the multiple load planning assigned each load unit to one slot on the way to the hub and potentially another slot on a train that leaves the hub. Here, an interesting question is, if an integrated optimization approach is needed or if with good strategies it is sufficient to do independent load planning at the involved terminals.
Especially for the storage and crane planning we think that an integrated planning has advantages. In an integrated planning of these two subproblems the total length of the crane moves (loaded and empty) could be optimized in a single method. This might lead to better solutions than the decomposition of the problems, where in the storage planning the total length of the loaded crane moves is minimized, ignoring the effects on the total length of the empty crane moves (which are optimized subsequently in the crane planning).
Another important research issue is the organization and planning of visits of trucks, which deliver or fetch load units. A major question is how much the transfer of load units can be optimized, if trucks do not "disturb" the terminal operations by randomly showing up. This could be organized by an appointment system, if this helps the terminal operator to organize the transfer more efficiently. Furthermore, a question is, if also the haulage companies can profit due to shorter turnaround times of the appointed trucks in the terminal.

Beside the open scientific questions it is fascinating how much optimization will be used in the terminal practice in the next years. We are sure that we could provide a good basis for fast optimization methods for several relevant operational optimization problems of rail-road terminals. Especially, for the new terminal MegaHub (in Hannover Lehrte) which will be run semi-automatically such optimization methods are needed.

References

[1] *AIMMS, the Modeling System.* Paragon Decision Technology, 1999.

[2] T. Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007.

[3] K. Alicke. Modeling and optimization of the intermodal terminal Mega Hub. *OR Spectrum*, 24(1):1–18, Februar 2002.

[4] K. Altmann, M. Lenk, and R. Hetzel. Das Betriebs- und Leitsystem für Umschlagbahnhöfe im Kombinierten Verkehr. *Eisenbahntechnische Rundschau*, 6:328–333, 2007.

[5] P. P. Alvarez and J. R. Vera. Application of robust optimization to the sawmill planning problem. *Annals of Operations Research*, pages 1–19, October 2011. doi:10.1007/s10479-011-1002-4.

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

[7] M. Avriel, M. Penn, and N. Shpirer. Container ship stowage problem: complexity and connection to the coloring of circle graphs. *Discrete Applied Mathematics*, 103(1):271–279, 2000.

[8] M-A. Aßbrock. *Integrierte Kran- und Lagerplanung in Straße-Schine-Containerterminals.* Master thesis, Universität Osnabrück, Fachbereich Mathematik/Informatik, 2014.

[9] A. Ballis and J. Golias. Towards the improvement of a combined transport chain performance. *European Journal of Operational Research*, 152(2):420–436, 2004.

[10] A. Ben-Tal, D. Bertsimas, and D. B. Brown. A soft robust model for optimization under ambiguity. *Operations Research*, 58:1220–1234, 2010.

[11] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization.* Princeton University Press, Princeton and Oxford, 2009.

[12] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming A*, 99:351–376, 2003.

[13] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[14] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming A*, 88:411–424, 2000.

[15] D. Bertsimas, V. Goyal, and X. A. Sun. A geometric characterization of the power of finite adaptability in multistage stochastic and adaptive optimization. *Mathematics of Operations Research*, 36(1):24–54, 2011.

[16] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[17] A. Bessas, S. Kontogiannis, and C. Zaroliagis. Robust line planning in case of multiple pools and disruptions. In A. Marchetti-Spaccamela and M. Segal, editors, *Theory and Practice of Algorithms in (Computer) Systems*, volume 6595 of *Lecture Notes in Computer Science*, pages 33–44. Springer, 2011.

[18] C. Bierwirth and F. Meisel. A fast heuristic for quay crane scheduling with interference constraints. *Journal of Scheduling*, 12(4):345–360, 2009.

[19] C. Bierwirth and F. Meisel. A survey of berth allocation and quay crane scheduling problems in container terminals. *European Journal of Operational Research*, 202:615–627, 2010.

[20] Y. M. Bontekoning, C. Macharis, and J. J. Trip. Is a new applied transportation research field emerging?–a review of intermodal rail-truck freight transport literature. *Transportation Research Part A: Policy and Practice*, 38(1):1–34, 2004.

[21] B. Borgmann, E. van Asperen, and R. Dekker. Online rules for container stacking. *OR Spectrum*, 32(3):687–716, 2010.

[22] N. Bostel and P. Dejax. Models and algorithms for the container allocation problem on trains in a rapid transshipment yard. *Transportation Science*, 32(4):370–379, 1998.

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

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

[25] P. Brucker and S. Knust. *Complex Scheduling*. Springer Berlin, 2006.

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

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

[28] F. Bruns, S. Knust, and N. V. Shakhlevich. Complexity results for storage loading problems with stacking constraints. *submitted*, 2014.

[29] Bundesamt für Güterverkehr. *Marktbeobachtung Güterverkehr, Bericht Herbst 2012*, 2013.

[30] Bundesministerium für Verkehr Bau- und Wohnungswesen. *Bericht des Bundesministeriums für Verkehr zum Kombinierter Verkehr*, 2001.

[31] Bundesministerium für Verkehr, Bau und Stadtentwicklung. *Prognose der deutschlandweiten Verkehrsverflechtungen 2025*, 2007.

[32] E. Carrizosa and S. Nickel. Robust facility location. *Mathematical Methods of Operations Research*, 58:331–349, 2003.

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

[34] P. Corry and E. Kozan. An assignment model for dynamic load planning of intermodal trains. *Computers and Operations Research*, 33:1–17, 2006.

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

[36] K. Corvers and R. Weißenburger. *Transport- und Lademittel im Kombinierten Verkehr*, 2007.

[37] R. Dekker, P. Voogd, and E. van Asperen. Advanced methods for container stacking. *OR Spectrum*, 28(4):563–586, 2006.

[38] A. Delgado, R. M. Jensen, K. Janstrup, T. Høyer Rose, and K. Høj Andersen. A constraint programming model for fast optimal stowage of container vessel bays. *European Journal of Operational Research*, 220(1):251–261, 2012.

[39] U. Dorndorf and F. Schneider. Scheduling automateed triple cross-over stacking cranes in a container yard. *OR Spectrum*, 32(3):617 – 632, 2010.

[40] A.L. Erera, J.C. Morales, and M. Savelsbergh. Robust optimization for empty repositioning problems. *Operations Research*, 57(2):468–483, 2009.

[41] S. Even and O. Kariv. An $O(n^{2.5})$ algorithm for maximum matching in general graphs. In *16th Annual Symposium on Foundations of Computer Science*, pages 100–112. IEEE, 1975.

[42] T. A. Feo and J. L. Gonzáles-Velarde. The intermodal trailer assignment problem. *Transportation Science*, 29(4):330–341, 1995.

[43] U. Fischer, R. Gomeringer, M. Heinzler, R. Kilgus, F. Näher, S. Oesterle, H. Paetzold, and A. Stephan. *Tabellenbuch Metall.* Europa Lehrmittel, 2011.

[44] M. Fischetti and M. Monaci. Light robustness. In R. K. Ahuja, R.H. Möhring, and C.D. Zaroliagis, editors, *Robust and online large-scale optimization*, volume 5868 of *Lecture Notes in Computer Science*, pages 61–84. Springer, 2009.

[45] H.N. Gabow. *Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs.* PhD Thesis, Department of Computer Science, Stanford University, Stanford, California, 1973.

[46] H.N. Gabow. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *Journal of the ACM*, 23(2):221–234, 1976.

[47] M.R. Garey and D.S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[48] F. Glover and M. Laguna. *Tabu search.* Kluwer Academic Publishers, 1997.

[49] M. Goerigk, M. Knoth, M. Müller-Hannemann, M. Schmidt, and A. Schöbel. The price of robustness in timetable information. In A. Caprara and S. Kontogiannis, editors, *Proceedings of ATMOS11*, volume 20 of *OpenAccess Series in Informatics (OASIcs)*, pages 76–87. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Germany, 2011.

[50] M. Goerigk and A. Schöbel. An empirical analysis of robustness concepts for timetabling. In T. Erlebach and M. Lübbecke, editors, *Proceedings of ATMOS10*, volume 14 of *OpenAccess Series in Informatics (OASIcs)*, pages 100–113. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Germany, 2010.

[51] M. Goerigk and A. Schöbel. A note on the relation between strict robustness and adjustable robustness. Technical report, Institute for Numerical and Applied Mathematics, University of Göttingen, 2012.

[52] J. A. González, E. Ponce, C. Mataix, and J. Carrasco. The automatic generation of transhipment plans for a train–train terminal: Application to the spanish–french border. *Transportation Planning and Technology*, 31(5):545–567, 2008.

[53] R. L. Graham, E. L. Lawler, J. K. Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics. v5*, pages 287–326, 1977.

[54] N. G. Hall, W. Kubiak, and S. P. Sethi. Earliness-tardiness scheduling problems, ii: Deviation of completion times about a restrictive common due date. *Operations Research*, 39(5):847 – 856, 1991.

[55] R. Hetzel. TaT - Neue Technologieansätze für automatisierbare Terminals im Kombinierten Verkehr. *Schienenverkehr – sicher, leise, effizient, InnoTrans, Berlin 2008*, 2008.

[56] Ilog SA. *ILOG CPLEX 11.0 User's Manual*, 2007.

[57] F. Jaehn. Positioning of load units in a transshipment yard storage area. *OR Spectrum*, 35(2):399–416, 2013.

[58] S. H. Jung and K. H. Kim. Load scheduling for multiple quay cranes in port container terminals. *Journal of Intelligent Manufacturing*, 17:479–492, 2006.

[59] J. Kang, K. R. Ryu, and K. H. Kim. Deriving stacking strategies for export containers with uncertain weight information. *Journal of Intelligent Manufacturing*, 17(4):399–410, 2006.

[60] K. H. Kim and Y. M. Park. A crane scheduling method for port container terminals. *European Journal of Operational Research*, 156:752–768, 2004.

[61] K. H. Kim, Y. M. Park, and K. R. Ryu. Deriving decision rules to locate export containers in container yards. *European Journal of Operational Research*, 124(1):89–101, 2000.

[62] T. Koch. *Rapid Mathematical Programming*. PhD thesis, Technische Universität Berlin, 2004. ZIB-Report 04-58.

[63] Kombiverkehr. *Frachtbehälter Einteilung nach Längenklassen UIC*, 2008.

[64] Y-C. Lai and C. Barkan. Options for improving the energy efficiency of intermodal freight trains. *Journal of the Transportation Research Board*, 1916:47 – 55, 2005.

[65] Y-C. Lai, C. Barkan, J. Drapa, N. Ahuja, J. Hart, P. Narayanan, C. Jawahar, A. Kumar, L. Milhon, and M. Stehly. Machine vision analysis of the energy efficiency of intermodal freight trains. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 221(3):353–364, 2007.

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

[67] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York, 1976.

[68] D-H. Lee, H. Q. Wang, and L. Miao. Quay crane scheduling with non-interference constraints in port container terminals. *Transportation Research Part E*, 44:124–135, 2008.

[69] J. Lehnfeld and S. Knust. Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research*, to appear.

[70] C. Liebchen, M. Lübbecke, R. H. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recoery, and railway applications. In R.K. Ahuja, R.H. Möhring, and C.D. Zaroliagis, editors, *Robust and online large-scale optimization*, volume 5868 of *Lecture Notes in Computer Science*. Springer, 2009.

[71] A. Lim, B. Rodrigues, and Z. Xu. A m-parallel crane scheduling problem with a non-crossing constraint. *Naval Research Logistics*, 54:115–127, 2007.

[72] R. Lougee-Heimer. The common optimization interface for operations research. *IBM Journal of Research and Development*, 47(1):57–66, 2003.

[73] F. Meisel. The quay crane scheduling problem with time windows. *Naval Research Logistics (NRL)*, 58(7):619–636, 2011.

[74] L. Moccia, J-F. Cordeau, and M. Gaudioso und G. Laporte. A branch-and-cut algorithm for the quay crane scheduling problem in a container terminal. *Naval Research Logistics*, 53:45–59, 2005.

[75] W.C. Ng. Crane scheduling in container yards with inter-crane interference. *European Journal of Operational Research*, 164(1):64–78, 2005.

[76] T. Park, R. Choe, S. M. Ok, and K. R. Ryu. Real-time scheduling for twin RMGs in an automated container yard. *OR Spectrum*, 32(3):593 – 615, 2010.

[77] W. B. Powell and T. A. Carvalho. Real-time optimization of containers and flatcars for intermodal operations. *Transportation Science*, 32:110–126, 1998.

[78] H. Rotter. New operating concepts for intermodal transport: The mega hub in Hanover/Lehrte in Germany. *Transportation Planning and Technology*, 27(5):347 – 365, 2004.

[79] SBB and HUPAC. *Neue Beladeschema für Wechselbehälter- und Containerwagen*, 2006.

[80] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden. *Variable Neighbourhood Descent for Planning Crane Operations in a Train Terminal*, volume 624 of *Lecture Notes in Economics and Mathematical Systems.* Springer Berlin Heidelberg, 2009.

[81] A.L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21:1154–1157, 1973.

[82] R. Stahlbock and S. Voß. Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52, 2008.

[83] S. Stiller. *Extending concepts of reliability. Network creation games, real-time scheduling, and robust optimization.* PhD thesis, TU Berlin, 2008.

[84] K. Subramani. Analyzing selected quantified integer programs. In David Basin and Michaël Rusinowitch, editors, *Automated Reasoning*, volume 3097 of *Lecture Notes in Computer Science*, pages 342–356. Springer, 2004.