MASTER PROJECT BUSINESS ANALYTICS

Complimentary Material to ORTEC Pallet and Load Building

Wouter Kool

1937189

VU Supervisors: René Bekker, Gerrit Timmer

ORTEC Supervisors: Joaquim Gromicho, Bobby Miller

VU University, Amsterdam

Faculty of Sciences

Master Business Analytics

De Boelelaan 1081a

1081 HV Amsterdam

ORTEC

Houtsingel 5

2719 EA Zoetermeer

July, 2014

Contents

1 Introduction **1**

2 Select stacks per container **2**

2.1 The data 2

2.2 The mathematical problem 3

2.3 Primary MIP formulation 8

2.4 Alternative formulation 13

3 Place stacks in container **23**

3.1 Objectives 23

3.2 Create and place blocks 26

3.3 Place stacks directly 30

A Select stacks model formulations **32**

A.1 Select stacks per container basic model 32

A.2 Elimination of stack symmetry 35

A.3 Select stacks per container desymmetrized model 36

A.4 Alternative formulation 38

A.5 Pricing problem 39

B Place stacks model formulations **41**

B.1 Place blocks in container model 41

B.2 Place stacks in container model 43

List of symbols **47**

Bibliography **50**

Chapter 1

Introduction

This document is complimentary to the thesis document ‘ORTEC Pallet and Load Building’, which we refer to as the main document. We assume that the reader has read the main document, in which we introduced the improved Pallet Grid Assignment (PGA) algorithm in Chapter 4. In this document, we work out in detail the two subproblems of the improved PGA algorithm. In Chapter 2 an MIP model is developed for the first subproblem of creating stacks for a single container. The formulation is rewritten to a column generation form and it is discussed how it can be solved. In Chapter 3 the methods for the second subproblem of placement of the stacks in the container are developed.

Chapter 2

Select stacks per container

This chapter describes the method that is used for creating and selecting the stacks for a single container. Section 2.1 contains a description of the data and introduces the notification we will use. In Section 2.2 a mathematical formulation for the objectives and the restrictions is defined. Section 2.3 introduces the primary MIP formulation for the problem and shows how it can be desymmetrized by considering unique pallets and defining an ordering for the stacks. Section 2.4 introduces an alternative MIP formulation that can be solved using column generation and partial application of branch and price, and works out this method.

Although an early variant of the primary MIP formulation has been implemented, experiments showed that the computation time was too large for any practically sized instance, even with the symmetry eliminated. This is why the alternative formulation using column generation is the one that was implemented as the final variant. This formulation is the one that is used in the main document to find the results.

2.1 The data

Let the pallets be given by $\mathcal{P} = \{1, ..., m\}$. Furthermore, the load consists of a set of orders $O = \{1, ..., s\}$. For each pallet $i \in \mathcal{P}$, the following properties are given:

l_i	the length
b_i	the width
h_i	the height of the pallet including the wood
$\hat{h}_i$	if wood may be removed, the height of the pallet with the wood removed, otherwise $\hat{h}_i = h_i$
w_i	the weight of the pallet including the wood
$\hat{w}_i$	if wood may be removed, the weight of the pallet with the wood removed, otherwise $\hat{w}_i = w_i$
p_i	the priority
$O_i \subset O$	the orders for which the pallet contains at least one product

Additionally, there is an $m \times m$ stackability matrix A where $a_{ii'} = 1$ indicates that pallet i can be stacked on pallet i'. For each order $j \in O$, let $\mathcal{P}_j \subseteq \mathcal{P}$ be the set of pallets that contain at least one product of order j, and let $O^m = \{j \in O : |\mathcal{P}_j| > 1\}$ be the set of orders that are divided over multiple pallets. Let the length, width and height of the container be given by L, B and H respectively. The maximum weight that is allowed in the container is W. We assume that the pallet dimensions are given according to the fixed orientation, so $b_i > l_i$ because the longest side is placed in the width of the container. We estimate the number of pallets that can be placed alongside the length of the container by dividing the length of the container L by the average length of a pallet $\frac{1}{m}\sum_{i\in\mathcal{P}} l_i$. Rounding down and assuming two pallets per row, we calculate the maximum number of floorspots K using

$$K = 2\left\lfloor \frac{L}{\frac{1}{m}\sum_{i\in\mathcal{P}} l_i} \right\rfloor. \tag{2.1}$$

2.2 The mathematical problem

Before we define the problem mathematically, let a *stack* of pallets be defined as a (sub)*set* $S \subseteq \mathcal{P}$. A stack is *feasible* if the following condition holds: there exists at least one ordering of the pallets in the stack such that it is feasible with respect to the stackability matrix and for which the stack fits in the height of the container if the wooden pallet is removed for all pallets where this is allowed except for the pallet on the bottom. If there are multiple orderings possible, we call these the *feasible arrangements* of the stack, but we consider the different arrangements as the same stack. This notion is important as otherwise we would require an ordered *tuple*, rather than a set, to represent the stack. For a single container, let a solution be given by a collection of stacks $S_1, ..., S_\theta$, such that the set of pallets loaded in that container is $\bigcup_{k=1,\dots,\theta} S_k$.

2.2.1 Objective

The objective is to select a number of feasible stacks such that it is likely that a feasible placement in the container exists and that the container is utilized 'as much as possible' while splitting as few orders as possible. The stacks should be selected from the must-go and may-go pallets as explained in Section 4.3 in the main document. Therefore, in the scope of the optimization, we consider $\mathcal{P}$ not to include the may-not-go pallets and define $\mathcal{P}^m \subseteq \mathcal{P}$ as the set of must-go pallets, such that the may-go pallets are in the set $\mathcal{P} \setminus \mathcal{P}^m$. In order to define a mathematical problem we will define a quantitative measure for the quality of a selection of stacks, which is a linear combination of quality measures for the VFR, the number of stacks, the number of orders that is split and the properties of the stacks. We explain the four measures after which the combined objective is defined.

Vehicle Fill Rate

As was described in the other document, we optimize a combination of weight and volume. To do so, we define the weight factor $0 < \chi < 1$ that represents the fraction of the VFR objective that should focus on weight. As we know that the must-go pallets must be loaded anyway, χ should only depend on the may-go pallets. Let $\text{BFD}(\mathcal{P})$ be the lower bound on the number of stacks found by the Best Fit Decreasing method, so the lower bound on the number of containers on volume for the may-go pallets is $\text{BFD}(\mathcal{P} \setminus \mathcal{P}^m)/K$ and the lower bound based on weight is $\sum_{i \in \mathcal{P} \setminus \mathcal{P}^m} w_i / W$, so we calculate the weight factor using Equation (2.2).

$$\chi = \frac{\frac{\sum\limits_{i \in \mathcal{P} \setminus \mathcal{P}^m} w_i}{W}}{\frac{\text{BFD}(\mathcal{P} \setminus \mathcal{P}^m)}{K} + \frac{\sum\limits_{i \in \mathcal{P} \setminus \mathcal{P}^m} w_i}{W}} \tag{2.2}$$

Let VFR^W and VFR^V be the VFR in terms of weight and volume, respectively. In order to justify the use of the weight factor, both measures should be on the same scale, so we normalize them (approximately) to the range $[0, 1]$. For the weight we can have almost 100% utility, so we let this value be proportional to the maximum weight in the container as in Equation (2.3).

$$\text{VFR}^W = \frac{\sum\limits_{i \in \bigcup_{k=1,\dots,\theta} S_k} w_i}{W} \tag{2.3}$$

Note that this value does not take into account that some pallets may be loaded without the wooden pallet such that the actual loaded weight is lower. However, as explained in Section 3.5 of the main document, this actual weight loaded is what should be considered, since shipping additional wood does not add any value. By taking the removal of the wood into account in the weight restriction but not in the objective, we may actually find a VFR little over 100% because the removal of wood allows to 'overload' the container. This way, automatically the removal of wood has a positive effect on the objective value.

In terms of volume, we typically cannot achieve 100% fill. Also, the length and width of a pallet are not really relevant, as the practical 'space' it consumes in a container is the space within the stack, which is only determined by its height. In that sense, the total height of pallets that can be loaded in a container is KH. However, the average height of minimal stacks for the may-go pallets is $\frac{\sum_{i\in\mathcal{P}\setminus\mathcal{P}^m} h_i}{\text{BFD}(\mathcal{P}\setminus\mathcal{P}^m)}$. As we are unlikely to do better than this value, in order to get a value for VFR^V which approximately scales between 0 and 1 we need to correct for this difference. Therefore we calculate the VFR for the volume using Equation (2.4).

$$\text{VFR}^V = \frac{\sum_{i\in\bigcup_{k=1,\ldots,\theta} S_k} h_i}{KH} \cdot \frac{\text{BFD}(\mathcal{P}\setminus\mathcal{P}^m)}{\sum_{i\in\mathcal{P}\setminus\mathcal{P}^m} h_i} = \sum_{i\in\bigcup_{k=1,\ldots,\theta} S_k} h_i \cdot \frac{\text{BFD}(\mathcal{P}\setminus\mathcal{P}^m)}{KH \sum_{i\in\mathcal{P}\setminus\mathcal{P}^m} h_i} \tag{2.4}$$

The value that we want to maximize is the weighted combination of the volume and weight according to the weight factor, as represented by Equation (2.5).

$$\text{VFR} = \chi\text{VFR}^W + (1-\chi)\text{VFR}^V \tag{2.5}$$

Number of stacks

Obviously, the number of stacks selected, which we will denote with θ, should not exceed the number of floorspots K. However, at any point we may have concluded during placement of the stacks that it is not actually possible to load K stacks, in which case the limit is decreased to a lower number which we denote with $\theta^{\max}$. If this is the first iteration for the container, we define $\theta^{\max} = K$. Next to that, we also define the *desired* number of stacks θ^*. If $\mathcal{P}^m \neq \mathcal{P}$, we have some may-go pallets and not all pallets can be loaded. Then obviously we want to load a full truck and for the best stability and weight distribution we use as much floorspots as possible, so $\theta^* = \theta^{\max}$. If $\mathcal{P}^m = \mathcal{P}$ we load all remaining pallets, but typically this is not a full truck. In that case we want to compress the load length by minimizing the number of stacks. However, it makes no sense to compress a heavy load disproportionately, so we define the desired number of stacks proportionally to the weight of the pallets as in (2.6). If the load is very light compared to its volume, this value is very low so optimizing towards the desired number of stacks equates to minimizing the number of stacks. If the value is used without rounding, the preference between $\lfloor\theta^*\rfloor$ and $\lceil\theta^*\rceil$ automatically follows from the objective function of minimizing the absolute difference $|\theta^* - \theta|$.

$$\theta^* = \sum_{i\in\mathcal{P}} w_i \frac{K}{W} \tag{2.6}$$

Order splitting

An order is split if at least one but not all of the pallets which have at least one product from an order are selected. We add a cost C^v for each order that is split, so if we denote with $\Delta = |\{j \in O^m : 1 \le |\mathcal{P}_j \cap (\cup_{k=1}^{\theta} S_k)| < |\mathcal{P}_j|\}|$ the number of orders that is split then we add cost $C^v\Delta$.

Properties of the stacks

With every stack we associate a cost $C(S)$:

$$\begin{aligned} C(S) \quad &= \quad C^l \left(\max_{i\in S} l_i - \min_{i\in S} l_i\right) + \\ & \qquad C^b \left(\max_{i\in S} b_i - \min_{i\in S} b_i\right) \end{aligned} \tag{2.7}$$

The two terms incur costs C^l and C^b for the *span* of the length and width, respectively. This is done to create homogeneous stacks with respect to these dimensions. The weights should be used to make distinctions in the importance of the homogeneity in the width and the length of the stack.

Combined objective

The VFR has to be maximized, while the deviation from the desired number of stacks and the ‘costs’ for the order splitting and the quality of the stacks have to be minimized. Applying weights C^{VFR}, C^θ and C^v for the first three components results in the objective (2.8). The costs of the stacks do not need an additional weight coefficient as this can be included in the coefficients C^l and C^b.

$$\max \quad C^{\mathrm{VFR}}\left(\chi\mathrm{VFR}^W + (1-\chi)\mathrm{VFR}^V\right) - C^\theta|\theta^* - \theta| - C^v\Delta - \sum_{k=1}^{\theta} C(S_k) \tag{2.8}$$

2.2.2 Restrictions

Multi-order restriction

As has been explained in Section 4.3.3 of the main document, the containers are loaded iteratively. We want to select in each container at least a percentage of pallets corresponding to multi-orders, to prevent greedily selecting all 'flexible' pallets first. Again, we have to deal with the weight and volume dimensions. Putting a restriction on a mix of both or on both dimensions individually is disadvantageous, as some orders may be heavy and some orders may be light, such that the restriction to select both a minimum weight forces orders to be split. Therefore, we identify the dimension in which the multi-orders take up most containers and put the restriction on this dimension. This may result in heavy multi-orders being selected for the first container, such that for the next container the volume is the limiting dimension in which case a minimum restriction on the volume results in light orders being selected.

Again, we only take the may-go pallets into consideration as the must-go pallets are selected anyway. Let $\{i \in \mathcal{P} \setminus \mathcal{P}^m : O_i \cap O^m \neq \emptyset\}$ be the set of may-go multi-order pallets, which is the set of pallets that contain at least one order that is divided over multiple pallets. Also, we define a function $N(\hat{\mathcal{P}})$ to calculate the lower bound on the number of containers for a set of pallets $\hat{\mathcal{P}}$ in (2.9).

$$N(\hat{\mathcal{P}}) = \max\left\{\frac{\mathrm{BFD}(\hat{\mathcal{P}})}{K}, \frac{\sum\limits_{i\in\hat{\mathcal{P}}} w_i}{W}\right\} \tag{2.9}$$

The average weight of the may-go multi-order pallets per container is $\frac{\sum\limits_{i\in\mathcal{P}\setminus\mathcal{P}^m : O_i\cap O^m\neq\emptyset} w_i}{N(\mathcal{P}\setminus\mathcal{P}^m)}$, so we should select at least this fraction of the weight in the next container. If we have some must-go pallets as well, we should correct for the space taken up by these, so the minimum weight of the may-go multi-order pallets to be selected can be calculated as in (2.10).

$$\omega^W = \begin{cases} (1 - N(\mathcal{P}^m))\frac{\sum\limits_{i\in\mathcal{P}\setminus\mathcal{P}^m : O_i\cap O^m\neq\emptyset} w_i}{N(\mathcal{P}\setminus\mathcal{P}^m)} & \text{if } N(\mathcal{P}\setminus\mathcal{P}^m) > 1, \frac{\mathrm{BFD}(\mathcal{P}\setminus\mathcal{P}^m)}{K} < \frac{\sum\limits_{i\in\mathcal{P}\setminus\mathcal{P}^m} w_i}{W} \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

Note that $N(\mathcal{P}^m) \leq 1$ by definition. If $N(\mathcal{P} \setminus \mathcal{P}^m) \leq 1$ all may-go pallets will fit in the next container, so we do not worry about selecting the multi-order pallets, so in that case we set $\omega^W = 0$, just as we do when weight is not the limiting factor.

For the volume dimension, we do the same, again based on the heights of the pallets. In contrast to the objective, we do not need to scale because we only put a restriction on a relative value. Therefore, we define ω^V as the minimum total height that should be selected from the may-go multi-order pallets according to (2.11).

$$\omega^V = \begin{cases} (1 - N(\mathcal{P}^m))\frac{\sum_{i \in \mathcal{P}\setminus\mathcal{P}^m : O_i \cap O^m \neq \emptyset} h_i}{N(\mathcal{P}\setminus\mathcal{P}^m)} & \text{if } N(\mathcal{P} \setminus \mathcal{P}^m) > 1, \frac{\text{BFD}(\mathcal{P}\setminus\mathcal{P}^m)}{K} \geq \frac{\sum_{i \in \mathcal{P}\setminus\mathcal{P}^m} w_i}{W} \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

Diversity in weight of stacks

In order to quantify the diversity of the weights of the stacks we would ideally use the variance. However, it is not possible to do this in a linear model. Therefore we sum the absolute deviations from a fixed value. This fixed value is the mean weight per stack if the container is loaded with the maximum number of stacks and the maximum weight: $\frac{W}{K}$. If we select fewer stacks, this also increases the flexibility for the weight distribution in the container, so we consider this as 'empty stacks' with weight 0 for an increase in the sum of the deviations.

$$\sum_{k=1}^{\theta} \left| \sum_{i \in S_k} w_i - \frac{W}{K} \right| + (\theta^{\max} - \theta)\frac{W}{K} \tag{2.12}$$

In the model, we require that the value in (2.12) is at least Π, with $\Pi = 0$ initially such that there is no restriction. If the selection of stacks is such that a solution with the COG in the feasible range (according to the lower bound for the rear axle weight) does not exist, then we increase the restriction by a percentage $d\Pi$. If we denote with Π^1 the value of (2.12) for the selected stacks, we select new stacks and require that (2.12) is at least $\Pi := (1 + d\Pi)\Pi^1$.

2.3 Primary MIP formulation

For readability, the formulation of the problem as an MIP and explanation of the individual constraints is not given in this chapter but in Appendix A.1. However, we give the decision variables as they define the basic structure of the formulation. Also, we explain the symmetry that results from this formulation and show how it can be removed by adding additional constraints and merging variables.

2.3.1 Decision variables

In order to introduce the decision variables, recall that we have the sets $\mathcal{P}$ for the must- and may-go pallets, of which $\mathcal{P}^m \subseteq \mathcal{P}$ are must-go. O^m is the set of orders that is divided over multiple pallets. Additionally, let $\mathcal{Q} = \{1, ..., \theta^{\max}\}$ be the set of (some potentially empty) stacks and $\mathcal{L} = \{1, ..., t\}$ the potential levels in a stack, so t is an upper bound on the number of pallets in a single stack, for instance $t = \lfloor \frac{\min_{i \in \mathcal{P}} \hat{h}_i}{H} \rfloor$ although a tighter bound may be used for a more compact formulation. Let the following decision variables be given:

y_i	$\begin{cases} 1 & \text{if pallet } i \text{ is in the container} \\ 0 & \text{otherwise} \end{cases}$	$i \in \mathcal{P}$
$x_{i\ell k}$	$\begin{cases} 1 & \text{if pallet } i \text{ is on level } \ell \text{ in stack } k \\ 0 & \text{otherwise} \end{cases}$	$i \in \mathcal{P},\ \ell \in \mathcal{L},\ k \in \mathcal{Q}$
u_j	$\begin{cases} 1 & \text{if order } j \text{ is fully in the container} \\ 0 & \text{otherwise} \end{cases}$	$j \in O^m$
v_j	$\begin{cases} 1 & \text{if order } j \text{ is partly in the container} \\ 0 & \text{otherwise} \end{cases}$	$j \in O^m$
z_k	$\begin{cases} 1 & \text{if stack } k \text{ is non-empty} \\ 0 & \text{otherwise} \end{cases}$	$k \in \mathcal{Q}$
π_k	$\begin{cases} 1 & \text{if the weight of stack } k \text{ exceeds } \frac{W}{K} \\ 0 & \text{otherwise} \end{cases}$	$k \in \mathcal{Q}$
ϖ_k^-	How much the weight of stack k is lower than $\frac{W}{K}$, or 0	$k \in \mathcal{Q}$
ϖ_k^+	How much the weight of stack k is higher than $\frac{W}{K}$, or 0	$k \in \mathcal{Q}$
τ_k^-, τ_k^+	The minimum and maximum length of pallets in stack k	$k \in \mathcal{Q}$
ν_k^-, ν_k^+	The minimum and maximum width of pallets in stack k	$k \in \mathcal{Q}$

The MIP formulation of this basic model is given in Appendix A.1. Although in terms of the objective the order of the pallets within the stacks is irrelevant, the formulation requires that the solution is found with an ordering of the pallets in each stack such that feasibility is guaranteed.

2.3.2 Symmetry

Although the formulation in Appendix A.1 is correct, solving the problem is very hard. That is because the formulation contains multiple types of *symmetry*:

- **Pallet symmetry**: suppose we have two pallets i and i' which have identical properties and correspond to the same order(s). Then variables $x_{i\ell k}$ and $x_{i'\ell k}$ are symmetric as they may be swapped in any solution to obtain an equivalent solution. In other words, in terms of the objective function it does not matter *which* of the symmetric pallets is in a stack.
- **Stack symmetry**: for two *non-empty* stacks k and k', the variables $x_{i\ell k}$ and $x_{i\ell k'}$ may be swapped (τ and ν should be swapped accordingly) to obtain an equivalent solution. In other words, in terms of the objective function it only matters which pallets are in the same stack, but it does not matter *which* stack that is.
- **Within-stack symmetry**: if there are multiple feasible orderings of the pallets within the stack, some pairs of variables $x_{i\ell k}$ and $x_{i\ell' k}$ may be swapped without changing the solution in terms of which pallets are in which stacks.

This symmetry causes problems when a Branch and Bound scheme is used, as branching on symmetric variables results in equivalent solutions being found with other variables. The number of equivalent solutions can be very large: with r stacks there are already $r!$ equivalent solutions only because of the stack symmetry. That is why in the next paragraphs we propose methods to eliminate most of the symmetry such that for each ‘real world solution’ the number of equivalent ‘model solutions’ is minimized.

Pallet symmetry

In order to eliminate pallet symmetry, let $\bar{\mathcal{P}} \subseteq \mathcal{P}$ be the set of *unique* pallets with respect to the relevant properties (define $\bar{\mathcal{P}}_j$ similarly), and let the number of pallets n_i be given for each $i \in \bar{\mathcal{P}}$, such that $\sum_{i\in\bar{\mathcal{P}}} n_i = |\mathcal{P}|$. From a mathematical perspective, we represent the pallets by the *multiset* $(\bar{\mathcal{P}}, n)$, where n_i is the *multiplicity* of $i \in \bar{\mathcal{P}}$.

Although we now have multiple of the same pallets, we do not need to redefine $x_{i\ell k}$ to work with the multiset of pallets because we can still only have one pallet on each level in each stack, so most of the constraints are not influenced. However we should let y_i denote the *number* of times that pallet $i \in \bar{\mathcal{P}}$ is in the container. Unfortunately, the constraints linking y_j with u_j and v_j have to be modified accordingly and as a result the integrality no longer comes for free. We could strengthen the formulation to make the integrality come for free by adding constraints $u_j + v_j \geq x_{i\ell k}$ for all $j \in O^m, i \in \bar{\mathcal{P}}_j, k \in Q$, but we will not do this as this is not transferable to the formulation we will develop in Section 2.4.

This reformulation eliminates the pallet symmetry as there are no longer multiple ways of representing the same solution using different indices for the pallets. Although this reformulation requires explicitly $u_j + v_j$ to be binary, this does not add to the complexity of the model, as the integrality will still come for free if all pallets in the order have $n_i = 1$. The non-integer values that require branching will only occur when $n_i > 1$, in which case the $x_{i\ell k}$ variables have been removed for at least one (original) i, resulting in a decrease of $t\theta^{\max}$ binary variables.

Stack symmetry

Stack symmetry arises because all stacks are equivalent, and therefore every permutation of the stacks in the solution gives an equivalent solution. This can be eliminated by defining an ordering on the stacks, and requiring that the solution that is found is ordered. In order to eliminate *all* symmetry, the ordering should be unique, that is, for every combination of two stacks that are different, the first stack *must* be either before or after the second, according to the ordering.

Looking at the model formulation, the most natural way to order the stacks is to sort them by the index of the bottom pallet, then the index of the second pallet, etc. However, this sorting depends on the arrangement of the pallets within the stack. Therefore we can still have an equivalent solution with the stacks in a different ordering if the pallets within the stacks are rearranged. We propose an ordering that does not depend on the arrangement of the pallets within the stack. Assuming that we eliminated the pallet symmetry, we order the stacks by the number of times they contain pallet 1, in descending order. Then, for all the stacks that have pallet 1 the same number of times, we order by the number of times of pallet 2, etc. Unless two stacks are equal, there is always a pallet by which we can order, so this ordering is unique.

Enforcing the model to represent stacks in this ordering means that for each pair of stacks k and $k+1$ the following should hold:

- $\sum\limits_{\ell \in \mathcal{L}} x_{1\ell k} \geq \sum\limits_{\ell \in \mathcal{L}} x_{1\ell(k+1)}$, ór
- $\sum\limits_{\ell \in \mathcal{L}} x_{1\ell k} = \sum\limits_{\ell \in \mathcal{L}} x_{1\ell(k+1)}$ and $\sum\limits_{\ell \in \mathcal{L}} x_{2\ell k} \geq \sum\limits_{\ell \in \mathcal{L}} x_{2\ell(k+1)}$, ór

- $\sum\limits_{\ell \in \mathcal{L}} x_{1\ell k} = \sum\limits_{\ell \in \mathcal{L}} x_{1\ell(k+1)}$, $\sum\limits_{\ell \in \mathcal{L}} x_{2\ell k} = \sum\limits_{\ell \in \mathcal{L}} x_{2\ell(k+1)}$ and $\sum\limits_{\ell \in \mathcal{L}} x_{3\ell k} \geq \sum\limits_{\ell \in \mathcal{L}} x_{3\ell(k+1)}$, ór

- $\vdots$

- $\sum\limits_{\ell \in \mathcal{L}} x_{i\ell k} = \sum\limits_{\ell \in \mathcal{L}} x_{i\ell(k+1)} \forall i \in \mathcal{P}$

Theoretically, this can be enforced by a single constraint by appropriate scaling of the terms. For this, define $\alpha_{|\bar{\mathcal{P}}|} = 1$ and $\alpha_i = (n_{i+1} + 1)\alpha_{i+1}$, so that $\alpha_i = \prod\limits_{i'=i+1}^{|\bar{\mathcal{P}}|} (n_{i'} + 1)$. Then the constraint is:

$$\sum_{i=1}^{|\bar{\mathcal{P}}|} \alpha_i \left(\sum_{\ell \in \mathcal{L}} x_{1\ell k} - \sum_{\ell \in \mathcal{L}} x_{1\ell(k+1)} \right) \geq 0, k \in \mathcal{Q} \setminus \{q\} \tag{2.13}$$

In Appendix A.2 it is explained how this constraint enforces the ordering. However, it is only theoretical, since for any practically sized instances α_1 will be so large that the system will suffer from numeric instability. Therefore we propose a different method using additional variables. Basically, we require $\sum\limits_{\ell \in \mathcal{L}} x_{i\ell k} \geq \sum\limits_{\ell \in \mathcal{L}} x_{i\ell(k+1)}$ for each pallet i, unless there is a pallet with a lower index for which the inequality is strict. In order to achieve that, we add a variable s_{ik} ($i \in \bar{\mathcal{P}}, k \in \mathcal{Q} \setminus \{q\}$) that *may* take up the slack in the constraint, and therefore is free to become at least equal to 1 if $\sum\limits_{\ell \in \mathcal{L}} x_{i\ell k} > \sum\limits_{\ell \in \mathcal{L}} x_{i\ell(k+1)}$. This slack may then be used (with a significantly large factor) by pallets with a higher index, so the constraint does not have to be satisfied if it is already satisfied by a pallet with a lower index. The constraint is then written as

$$\sum_{\ell \in \mathcal{L}} x_{i\ell k} + n_i \sum_{i'=1}^{i-1} s_{i'k} \geq \sum_{\ell \in \mathcal{L}} x_{i\ell(k+1)} + s_{ik}, i \in \bar{\mathcal{P}}, k \in \mathcal{Q} \setminus \{q\} \tag{2.14}$$

It is easy to see that if $\sum\limits_{\ell \in \mathcal{L}} x_{i\ell k}$ can only be smaller than $\sum\limits_{\ell \in \mathcal{L}} x_{i\ell(k+1)}$ if there is at least one $i' < i$ for which $s_{i'k}$ is non-zero. Furthermore, by induction it can be proved that if $\sum\limits_{\ell \in \mathcal{L}} x_{i'\ell k} = \sum\limits_{\ell \in \mathcal{L}} x_{i'\ell(k+1)}$ for all $i' < i$, then $s_{i'k} = 0$ for all $i' < i$, such that $\sum\limits_{\ell \in \mathcal{L}} x_{i\ell k}$ cannot be smaller than $\sum\limits_{\ell \in \mathcal{L}} x_{i\ell(k+1)}$. Therefore, these constraints also assure that the solution found is ordered. Although the most natural configuration that satisfies the constraints would be to have $s_{ik} = 1$ for $i = \min \left\{ i \in \bar{\mathcal{P}} : \sum\limits_{\ell \in \mathcal{L}} x_{i\ell k} > \sum\limits_{\ell \in \mathcal{L}} x_{i\ell(k+1)} \right\}$ and $s_{ik} = 0$ for all other i, this is not required, nor is the integrality of s_{ik}, only $s_{ik} \geq 0$. Although this is not required, we set the bound $s_{ik} \leq 1$ as this is sufficient.

Within-stack symmetry

In order to eliminate the symmetry within the stack that comes with the multiple possible arrangements of the pallets in the stack, we should eliminate the dependency of the model formulation on the ordering of the stack. However, then it is no longer possible to guarantee the feasibility of the stacks without enumerating all feasible (or infeasible) combinations of pallets in stacks. Although in this basic formulation the symmetry cannot be eliminated, in Section 2.4 we will develop a formulation that eliminates this type of symmetry naturally.

2.3.3 Desymmetrized model

The formulation of the problem if the techniques to eliminate the stack and pallet symmetry are applied can be found in Appendix A.3.

2.4 Alternative formulation

As an alternative to the standard MIP formulation, we can give an equivalent formulation that is a variant of the set partitioning formulation. This formulation is inspired by the Dantzig-Wolfe decomposition principle, that exploits the block-diagonal structure of the problem, which exists because the problem consists of smaller subproblems for each stack. The advantage of this formulation is that its relaxation is much tighter, which significantly contributes to an efficient branch and bound procedure.

2.4.1 Decision variables

Let $|\mathfrak{S}| = n$ and write the set of feasible stacks $\mathfrak{S}$ as $\mathfrak{S} = \{S_1, S_2, ..., S_n\}$ and define the corresponding set of indices as $\mathcal{N} = \{1, ..., n\}$. As each stack can be represented by a multiset, we write $S_k = (\bar{\mathcal{P}}, b_k), k \in \mathcal{N}$ such that the multiplicity vector b_k has elements $b_{ik} \in \{0, 1, ..., n_i\}$ that represent the number of times that pallet $i \in \bar{\mathcal{P}}$ is in stack S_k. Define the following decision variables:

d	The absolute difference between the actual and the desired number of stacks	
λ_k	$\begin{cases} 1 & \text{if stack } S_k \text{ is in the solution} \\ 0 & \text{otherwise} \end{cases}$	$k \in \mathcal{N}$
y_i	The number of times pallet i is in the container	$i \in \mathcal{P}$
u_j	$\begin{cases} 1 & \text{if order } j \text{ is fully in the container} \\ 0 & \text{otherwise} \end{cases}$	$j \in O^m$
v_j	$\begin{cases} 1 & \text{if order } j \text{ is partly in the container} \\ 0 & \text{otherwise} \end{cases}$	$j \in O^m$

λ_k and d replace all variables from the basic model that are related to the stacks. As the problem formulation without the pallet symmetry is preferred, this is the formulation that we give in Appendix A.4. In this formulation, $w(S_k) = \sum_{i \in S_k} w_i$ is the weight of stack k, comparable to the notation $C(S_k)$ for the cost of stack k. Note that the stack symmetry is eliminated naturally, as λ_k only determines whether or not a stack is selected, without assigning it a position or label. By letting b_{ik} be unique for each $k \in \mathcal{N}$ this formulation also eliminates the within-stack symmetry because the different arrangements of the pallets in the stack have the same representation and therefore are only one time in the formulation.

2.4.2 Column generation

Even with the different types of symmetry eliminated, the set $\mathcal{N}$ is extremely large. Therefore, it is impossible to consider all stacks explicitly. However, most of the stacks will not be part of an optimal solution and therefore are not required in the problem formulation. This enables us to use column generation: we begin with a small subset of columns (variables) and only add columns to the problem formulation if they contribute to finding the optimal solution. All possible columns are implicitly considered in the optimal solution if we can prove that there exists no column that would improve the solution.

In a minimization problem, a column would improve the solution if it has negative reduced costs. Therefore, the column generation subproblem, or *pricing problem*, is the problem of finding a column with negative reduced costs. This requires the use of dual variables, which unfortunately do not exist for the solution to an integer program. However, we can apply column generation to the *linear relaxation* of the problem, so we can develop the following heuristic column generation approach to find a solution for the integer problem:

1. Initialize the *restricted master problem* (RMP) with a subset of the columns. All variables d, y_i, u_j and v_j should be included but only a subset of the variables λ_k.

2. Solve the linear relaxation of the RMP by replacing constraint (15) in Appendix A.4, $\lambda_j \in \{0, 1, ...\}$, with $\lambda_j \geq 0$.

3. Obtain the dual variables from the optimal solution of the RMP and solve the pricing problem to find at least one column with negative reduced costs. If at least one column exists, add the column(s) to the RMP and go to step 2. Otherwise, go to step 4.

4. Solve the RMP with the integrality constraints $\lambda_j \in \{0, 1, ...\}$ and binary constraints $u_j \in \{0, 1\}$ and $v_j \in \{0, 1\}$. This is equivalent to solving the original problem with only the subset of columns that are in the RMP after all columns have been generated.

There are some additional issues that have to be taken care of when implementing this scheme. For instance, with the initial subset of columns the relaxation of the RMP may not be feasible, such that it cannot be solved and shadow prices cannot be obtained. Therefore, we relax constraints (7), (8) and (9) in Appendix A.4, by making them soft constraints with a high penalty, such that selecting no stacks at all is a feasible solution. The subset of columns that is in the initial RMP should be a subset of 'good' stacks that may be found using a heuristic.

Pricing problem

The pricing problem is the problem that has to be solved in order to find columns with negative reduced costs that are not already in the problem formulation. The only variable for which not all columns are in the formulation is λ, so the pricing problem is the problem of finding S_k such that the reduced costs for λ_k are negative. These reduced costs are the costs $C(S_k)$ for the stack itself minus the shadow prices for the constraints that affect λ_k, multiplied by the coefficient of λ_k in those constraints.

In the problem formulation in Appendix A.4, constraint (2), (3), (6) and (10) affect λ_k. Constraint (2) regards the number of times pallet i is in the stack and has, for each $i \in \bar{\mathcal{P}}$, as coefficient the number of times b_{ik} that the pallet is in the stack. Constraint (3) regards the number of stacks and therefore λ_k has a fixed coefficient of 1, both in the upper and lower bound constraint. Constraint (6) regards the weight of the stack and therefore has coefficient $w(S_k)$. Constraint (10) regards the absolute deviation of the weight from $\frac{W}{K}$ and therefore has coefficient $\left|w(S_k) - \frac{W}{K}\right|$.

Let ϕ_i, $i \in \bar{\mathcal{P}}$, be the dual values associated to constraints (2). The constraint (3) actually consists of three constraints, but since λ_k has the same coefficient, we may add the dual values and consider them as one, which we will denote with δ. In practice, one of the three dual values will be equal to δ and the other two will be 0, since only one constraint can be effective, or *basic*, at the same time. Then δ is the (potentially) negative shadow price of an additional stack. Finally, we let γ and φ be the dual values corresponding to constraints (6) and (10). Now ϕ_i is the shadow price for each pallet i in the stack, δ is the shadow price for the stack itself, γ is the shadow price for each unit of weight of the stack and φ is the shadow price for the deviation of the weight from $\frac{W}{K}$. Therefore the reduced costs for a stack $S_k = (\bar{\mathcal{P}}, b_k)$ are $C(S_k) - \delta - \sum_{i \in \bar{\mathcal{P}}} b_{ik}\phi_i - \gamma w(S_k) - \varphi \left|w(S_k) - \frac{W}{K}\right|$. The pricing problem (to find the column with the largest negative reduced costs) becomes:

$$rc^* = \min_{k \in \bar{N}} C(S_k) - \delta - \sum_{i \in \bar{\mathcal{P}}} b_{ik}\phi_i - \gamma w(S_k) - \varphi \left|w(S_k) - \frac{W}{K}\right| \tag{2.15}$$

If $rc^* < 0$, then S_{k^*} with $k^* = \arg\min_{k \in \bar{N}} C(S_k) - \delta - \sum_{i \in \bar{\mathcal{P}}} b_{ik}\phi_i - \gamma w(S_k) - \varphi \left|w(S_k) - \frac{W}{K}\right|$ is the column with most negative reduced costs.

The pricing problem may be regarded as a knapsack problem, because it basically concerns selecting a subset of pallets with maximum shadow prices that fit in the height of the container. However, the complexity of the additional constraints and the additional objectives depending on the combinations of pallets makes it impossible to use a standard algorithm, which is why we develop an MIP formulation.

MIP formulation The MIP formulation of the pricing problem is obtained by considering the standard MIP formulation of the problem for only one stack, with Equation (2.15) as objective function. For this, we remove the index k from all variables and we remove all constraints that are irrelevant for a single stack. Also we remove the 'global' constraints and objectives, as the contribution of the stack to these is represented through the shadow prices ϕ_i and γ in the objective. We may remove the variable y_i, so we remain with the following decision variables:

$x_{i\ell}$	$\begin{cases} 1 & \text{if pallet } i \text{ is on level } \ell \text{ in the stack} \\ 0 & \text{otherwise} \end{cases}$	$i \in \mathcal{P},\ \ell \in \mathcal{L}$
π	$\begin{cases} 1 & \text{if the weight of the stack exceeds } \frac{W}{K} \\ 0 & \text{otherwise} \end{cases}$	
ϖ^-	How much the weight of the stack is lower than $\frac{W}{K}$, or 0	
ϖ^+	How much the weight of the stack is higher than $\frac{W}{K}$, or 0	
τ^-, τ^+	The minimum and maximum length of pallets in the stack	
ν^-, ν^+	The minimum and maximum width of pallets in the stack	

The formulation of the problem is given in Appendix A.5.

Heuristic approach Solving of the pricing problem to optimality is only required to prove optimality of the solution by showing that no columns with negative reduced costs exist. However, it is computationally expensive to do this in every iteration if we may as well find a column with negative reduced costs by a heuristic. Although this may result in a higher number of iterations, the overall runtime is likely to be less because the heuristic column generation is (significantly) faster. If in any iteration the heuristic fails to find a column with negative reduced costs, we can solve the MIP formulation of the pricing problem to still find one or to prove that it does not exist.

The heuristic we propose is very simple and takes advantage of the business knowledge that in most of the stacks in the solution will consist of two pallets. We calculate the costs for all feasible combinations of two pallets one time *up front* and save these in an $|\bar{\mathcal{P}}| \times |\bar{\mathcal{P}}|$ matrix C, where $c_{ii'} = \infty$ if the stack is infeasible. Then during the column generation procedure we create a copy of the matrix (which we call $\hat{C}$) from which we subtract the shadow price δ from every entry and ϕ from the corresponding rows and columns. Also, we subtract γ multiplied with w_i for the rows corresponding to bottom pallets, and γ multiplied with $\hat{w}_{i'}$ from the top, since for that pallet the wood may be removed. Then for $\hat{C}$ it holds that $\hat{c}_{ii'} = c_{ii'} - \phi_i - \phi_l - \delta - \gamma(w_i + \hat{w}_{i'})$ and we let $\hat{c}^* = \min_{(i,i') \in \bar{\mathcal{P}}^2} c_{ii'}$ be the minimum of this *reduced cost matrix.* If $\hat{c}^* < 0$, we add all columns for which $\hat{c}_{ii'} = \hat{c}^*$. If not, we solve the pricing problem using the MIP formulation, in which we additionally require at least three pallets to be selected.

Depending on the problem instance, it may be that the number of potential stacks of three is limited, for instance if there are only a few pallets low enough to occur in a stack of three. In that case we may extend the heuristic for stacks of three, creating a ‘three-dimensional matrix’, or *tensor*, to store the pre-calculated costs. As the number of feasible entries compared to all possible combinations is low, this should be implemented in a sparse manner, for instance using a list that only stores the relevant entries. If all stacks of three are pre-calculated, we may require the MIP formulation to select at least four pallets.

It is very likely that the heuristic adds multiple columns in a single iteration. This increases the size of the RMP and may therefore increase the solution time of the relaxation, even though many columns are unused. For the sole purpose of finding the optimal value of the relaxation, this is disadvantageous. However, the addition of these columns may very well be beneficial when solving the final problem with integrality constraints, as (depending on the iteration in which they were found) they are *likely* to be good columns and therefore give more possibilities in finding a good integer solution. If only the single best column were added in each iteration, it may be that these columns together are not the best columns when searching for a good integer solution.

2.4.3 Branch and price

Solving the problem only using the columns that were generated while solving the LP relaxation does not guarantee that a good (or even feasible) solution is found. In order to find the optimal solution to the integer program, we should apply a branch and bound procedure to eliminate the fractional variables. However, after branching, it may be the case that as a result of the branching constraints new columns should be added as they have negative reduced costs. Therefore, we should apply column generation after branching, which results in a *branch and price* procedure. This is not straightforward as the branching constraints introduce additional dual variables that may not be compatible with the pricing problem.

This is illustrated by our model. If branching is done on λ_k, then we may have in one branch $\lambda_k \leq 0$, which prevents the stack at all from being in the solution. However, it is quite likely that the pricing problem would result in precisely stack S_k being found, as this stack would have been part of the optimal solution without the constraint. That means that we have to find the second optimal solution, and at depth k in the branching tree maybe even the k-th optimal solution, which is far from trivial. Moreover, the branching tree may grow very large because of the number of variables. Therefore, it is better to apply a branching scheme that is compatible with the pricing problem, such as the branching scheme that has been developed by Ryan and Foster (1981) for the Set Partitioning Problem. However, this is not easily implemented for our problem, since we have multi-sets rather than sets, and we do not require all elements to be in the partition.

Fortunately, experiments show that most of the values for λ_k are integral naturally. If not then imposing the integrality constraint does not severely influence the quality of the solution, since only a few variables need to be modified to resolve the fractional solution. Therefore, if the goal is to find a good solution rather than the optimal one, it may not be very beneficial to branch on λ_k. However, if the variables u_j or v_j concerning orders are fractional, imposing the integrality constraint may result in orders being included or excluded, heavily affecting the solution.

Branching

Branching on the variables v_j and u_j can be done quite efficiently. If significant costs are assigned to splitting an order, typically the number of order splits will be low, unless it is unavoidable to split multiple orders. Therefore the number of configurations for v and j corresponding to good solutions is low such that branches may be cut at a high level in the tree, because including, splitting or excluding additional orders results in a bad solution.

If for a certain j, $v_j = 0$ holds as a result from the first branching rule, then order j may not be split, and as a result from constraints (4) and (5) we have $y_i = n_i u_j$. In other words, from every product in order j we should select exactly a fraction u_j. If we have different quantities n_i of the different products, it is unlikely that this happens with a fractional value for u_j, so very often $u_j = 0$ or $u_j = 1$ results. In other words, order j is fully selected or not at all. In the other cases we may require some additional branching to resolve the fractional values for u_j. Since $u_j = 1$ may hold for multiple j, we branch for one variable at a time, preferring the one of which the fractional part of the value is closest to $\frac{1}{2}$.

If we have all v_j and u_j integral, there may still be some λ_k fractional, but as said before this can be resolved without to much compromising the solution. Therefore, we will consider this as a leaf node and solve the problem with the integrality constraints to find a new feasible solution. The advantage of using this branching scheme is that the branching on v_j and u_j can be enforced without incorporating the λ_k variables. Therefore, we do not have to take the branching constraints into account in the pricing problem: they are reflected in the shadow prices of the pallets, which will become zero if pallets cannot be selected because the corresponding orders

n_i / λ_k	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
1	1	1	0
1	0	1	1
1	1	0	1

Table 2.1: Example of a cycle in a fractional solution: the rows represent pallets and the columns represent stacks.

n_i / λ_k	1	$\frac{1}{2}$
1	1	0
2	1	2

Table 2.2: Example of a 'cycle' that occurs with pallets of which there are multiple.

are excluded.

2.4.4 Branch, price and cut

Due to the nature of the problem, it may occur at any point in the branch and price algorithm that a solution contains a structure as in Table 2.1. This structure is called a *cycle*, for the following reason: if the pallets would be nodes in a graph, and nodes corresponding to pallets in the same stack would be connected with an edge, then the stacks corresponding to non-zero variables would form a cycle. In practice, we only find cycles with an odd number of pallets[1]. The pallets are in stacks that are selected with fractional values. There may be many of these structures as typically most 'good' potential stacks consist of two pallets. The cycles do not only occur with pallets with multiplicity 1: the example in Table 2.2 is the cycle in Table 2.1 if pallets 2 and 3 are equal. In the branch and price algorithm, a solution with a cycle can be excluded by branching on values for λ_k, but we already discussed that it is not very efficient. We can eliminate the fractional solution without branching by the simple observation that for any three pallets, there cannot be more than two stacks that contain at least two of those three pallets. Otherwise formulated, we state that for any integer feasible solution λ, in the example in Table 2.1, it must hold that $\lambda_1 + \lambda_2 + \lambda_3 \leq 1$. This inequality is a *valid inequality* for the integer problem, but it is not part of the problem formulation. It is called a *cut* as it can be added to the problem formulation to 'cut off' the fractional solution. Adding this cut to the problem formulation and continuing the algorithm results in the solution being excluded without the need to create a branch. As the feasible region for the relaxation is smaller, the optimal value will be the same or higher (the problem is minimization) and is therefore a tighter lower bound on the

[1]Cycles with an even number of pallets are convex combinations of two integral solutions and therefore not found by a solver that only finds extreme solutions.

optimal solution of the integer problem.

The combination of the so called *branch and cut* algorithm and the branch and price algorithm is called *branch, price and cut.* This is more than trivially combining two well known techniques, as the addition of cuts to the problem formulation introduces new dual variables. In this example, after the cut is added the values of the variables λ_1, λ_2 and λ_3 may be limited by the cut, in which case the constraint corresponding to the cut is *basic* and has a negative shadow price (value of the dual variable). The negative shadow price may be interpreted as the decrease (improvement) in the objective value if the constraint is loosened. As a result of the shadow price, the column has a reduced costs ≥ 0, while it would have a negative reduced costs if this would not be taken into account. Therefore, without taking the shadow prices of the cut constraints into account, the pricing problem is very likely to find one of the columns in the cut, which are already in the formulation.

To maintain the validity of the branch and price algorithm, the dual variables should be added to the pricing problem. We will show how this can be done for the type of the cut which has been given for the example in Table 2.1, but first we introduce a generalization of this cut. Define $D \subset \bar{\mathcal{P}}$ as the set of unique pallets in the cut, such that $|\sum_{i \in D} n_i| = 2\alpha + 1$ for integer α. Then, using the same reasoning, we can have at most α stacks with 2 or more of the pallets in D such that $\sum_{k:\sum_{i \in D} b_{ik} \geq 2} \lambda_k \leq \alpha$. In this case, it means that also the shadow price for a cut constraint should be added to the objective value if the column should be part of the cut. In other words, letting π_j be the shadow price for cut j which consist of a subset of unique pallets $D \subset \mathcal{P}$, we have to add π_j to the objective if more than two pallets from D are in the stack. This may also mean that we have one of the pallets in D two times in the stack. We define the cut as only the set $D \subset \bar{\mathcal{P}}$ without multiplicity, because we always have to take into account all n_i pallets i because the individual ones cannot be distinguished in the cut.

Theoretically we can add all cuts to eliminate the fractional solutions that arise from cycles of pallets in stacks with two pallets. Moreover, we may think of ways to generalize the definition to also exclude fractional solutions with stacks of three. However, in general it is a tradeoff between the computational effort required to identify a cut and the added value. Experiments show that typically the most basic cuts with only three unique pallets show most added value in terms of a tighter bound. Larger cuts require more time to find and there are many more, such that excluding one fractional solution more often results in another fractional solution that is comparable. Also the returns are diminishing, because relatively each cut 'tightens' a constraint by $\frac{1}{2\alpha+1}$. Therefore, we only add cuts with a limited number of pallets and we limit the number of cuts added in each node in the branching tree.

Adapting the pricing problem

In order to include the shadow price for the cut, we introduce the additional binary variable μ_j that indicates if the stack contains at least two of the unique pallets in cut j, specified by $D_j \subset \bar{\mathcal{P}}$. We only add these variables for the cuts for which the shadow price π_j is non-zero, which we denote with $\mathcal{D}$. For each cut $j \in \mathcal{D}$, we add the term $-\mu_j \pi_j$ in the objective and we add the following constraint:

$$\sum_{i \in D_j} y_i \leq 1 + (t-1)\mu_j, \quad j \in \mathcal{D} \tag{2.16}$$

If $\mu_j = 1$ the constraint has no effect since at most t pallets can be in the stack. If $\mu_j = 0$ the stack should not be affected by the cut and therefore select at most one from the pallets in the cut. This constraint requires μ_j to be defined as binary variable, however we can enforce this binary status naturally by adding the following constraints:

$$x_{i\ell} + x_{i'\ell'} \leq 1 + \mu_j, \quad i, i' \in D_j, j \in \mathcal{D}, \ell, \ell' \in \mathcal{L}, \ell \neq \ell' \tag{2.17}$$

However, this will add $\sum_{j \in \mathcal{D}} \binom{t}{2} 2^{|D_j|}$ constraints, such that it may be more efficient to solve without these constraints and with the integrality constraint. In fact, these constraints are cuts and a good commercial solver may detect them and add them dynamically to the problem when required.

Chapter 3

Place stacks in container

This chapter describes the second step of the algorithm: the placement of the stacks within a single container. First we formalize the objectives in Section 3.1. After that, in Sections 3.2 and 3.3, we describe the two different methods for placing the stacks. Recall that the first method consists of creating blocks and placing them, while the second method places the stacks directly for additional flexibility, but this also leads to additional complexity. Therefore, we use this method only if the first method fails to result in an axle weight satisfying solution.

3.1 Objectives

In the first step, it is determined which products will be loaded in each truck, so for the second step the only objective is to load them 'as good as possible'. For this, we formulate two high-level requirements:

1. The *truck* should satisfy the axle weights

2. The load on the truck should be stable

3.1.1 Axle weights

At this point we know the weight that is loaded into the container, so we may translate the axle weight restrictions to a feasible interval for the COG as described in Appendix B.3 in the main document. When the axle positions can be adjusted, this COG interval depends on the actual axle positions but the interval is closest to the front of the container if both axle positions are closest to the front and vice versa. Therefore, the feasible interval for the COG is defined by the limits of the feasible COG regions for the extreme axle configurations in which both axles are either closest to the front or closest to the rear.

Let the feasible interval for the COG be defined by its lower bound COG^- and upper bound COG^+. Let the *desired* COG be the center of this interval, so $\text{COG}^* = \frac{\text{COG}^- + \text{COG}^+}{2}$. Then we define the following objective for the COG: it should be within the feasible interval and as close as possible to COG^*. Although the axle weight constraints are a hard constraint for the *truck*, we consider them as a soft constraint for the *container*, as we argued in Section 3.4.1 in the main document. An additional motivation for this is that, as we will see later in this chapter, we can only consider an *estimate* for the COG, which we denote with $\overline{\text{COG}}$.

Although the COG can only be estimated, the estimate still should be within the interval if possible. If not, it may be outside of the feasible COG range, but this should be avoided as much as possible and $\overline{\text{COG}}$ should be as close as possible to the interval. Overall we model the objective as a penalty C^ξ for the distance between $\overline{\text{COG}}$ and COG^* and a penalty $C^\Xi >> C^\xi$ for the distance from $\overline{\text{COG}}$ to the feasible interval, so this results in the objective in (3.1).

$$\min C^\xi |\overline{\text{COG}} - \text{COG}^*| + C^\Xi \max\{\overline{\text{COG}} - \text{COG}^+, \text{COG}^- - \overline{\text{COG}}, 0\} \tag{3.1}$$

If we only consider the axle weight objective, the second term is redundant since minimization towards COG^* also means the distance to the interval is minimized. However, by using the weights C^ξ and C^Ξ we can tradeoff the position of the COG within the interval against the stability objective, while we can highly penalize solutions outside the feasible range such that this takes precedence. With this objective, effectively the constraint on the COG interval is relaxed such that a feasible solution always exists.

3.1.2 Stability

The static stability of the load is mainly determined by the stackability of the individual stacks, so we do not consider it in this step. The actual dynamic stability is hard to model since this may depend on many unknown factors, such as the stiffness of individual pallets. As another example, a load may be more stable if each pallet is supported by a pallet that itself is sufficiently supported too. This can get very complex, so we will only consider the *direct* support of two adjacent stacks, either from the front, back or side. We formalize the objective as minimizing the *support deficiency*, which we will shortly define. The total support deficiency for all stacks should be minimized, where weights β^{f}, β^{r} and β^{s} are applied to distinguish the importance of support from the front, back and size, respectively. When a heavy loading pattern is used, we want to prevent long sequences of rows with only one pallet: we preferably alternate rows with two and one pallets. Therefore additionally we want to minimize the length of sequences with single centered pallets.

Support deficiency

We define the support deficiency in a direction as the difference between the height of the supporting pallet and the pallet that should be supported (according to the direction), if it is negative. If the supporting pallet is higher than the minimum support height, the support deficiency is 0, just as in the case the stack is supported by the container wall. If the stack is not at all supported, for instance at the rear of the truck, the support deficiency is the full height of the stack.

In order to define the support deficiency for the entire load, we sum for each row, or position, the *maximum weighted support deficiency*, which is the maximum of the front, rear or side support deficiency, weighted by coefficients β^{f}, β^{r} and β^{s}. Because bigger deficiencies are worse, we want to progressively penalize these. Ideally we would use a quadratic objective, but since this is not possible in a linear program, we add for each row an additional penalty if the maximum weighted support deficiency exceeds a threshold value τ.

Single centered pallet sequences

In order to minimize sequences of single centered pallets, we define a score for a heavy loading pattern. For each pallet that is placed as a single pallet in a row, we add to the score its position within the sequence, where the first pallet gets position 0. The sum of the values is quadratic in the length of the sequence, so if we minimize these values we prevent longer sequences as much as possible. In the rear of the container, a sequence should preferably be supported by a row of two pallets. Therefore, we penalize if we do not have such a row and end with a single pallet: we assign a value of 2 to the unsupported pallet in the rear and adjacent single pallets in the sequence get incrementing values. The quality measure is illustrated with three examples in Figure 3.1.

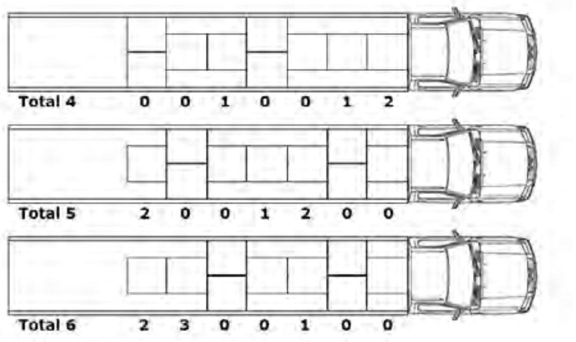

Figure 3.1: Examples of heavy loading patterns and their scorings: lower values are preferred.

3.2 Create and place blocks

This section describes the method to first create blocks and then place blocks in the container. The placing of blocks is only done if that is possible, that is, if the total length of the blocks that were created fits in the length of the container. If that is not the case, then, if the number of blocks is larger than $\frac{K}{2}$ as well, we know that it is not possible to place the stacks at all and we will go back to the step to create fewer stacks. If the total length exceeds the container length but the number of blocks is feasible, then we try the alternative method of directly placing stacks because this may result in different pairings of stacks such that the blocks fit in the length.

3.2.1 Creating blocks

For the creation of blocks of two stacks, we will use a (non-bipartite) matching algorithm. In order to do this, we need to define a cost for each pair of stacks. As each block will be placed as a single row in a truck, these pairs of stacks directly define the sideways support of the load. Indirectly, blocks of equal height stacks most likely result in better support in the front and rear direction too. For this reason, we take advantage of the fact that using the matching algorithm any cost can be defined for a pair to minimize the quadratic differences in heights of the stacks, such that we get equal height blocks as much as possible. Additionally, in order to make the blocks fit in the container, we want to minimize the differences between the lengths (sizes in lengthwise dimension of the container) of the stacks. We apply weights C^h and C^l to the objectives. Within the scope of this subproblem, let the stacks be represented by $\mathcal{S} = \{1, ..., m\}$ and for stacks $i, j \in \mathcal{S}$ let the heights be h_i and h_j and lengths be l_i and l_j respectively. Then we define a cost c_{ij} for pairing stack i with stack j as in (3.2).

$$c_{ij} = C^h \left(h_i - h_j\right)^2 + C^l \left(l_i - l_j\right)^2 \tag{3.2}$$

Not all pairs of stacks are feasible, as some combinations of stacks may not fit in the width of the container. Therefore, we consider the matching problem on a graph, where nodes represent stacks and edges represent combinations of stacks. We remove the edges for which the combination does not fit in the width, so the graph may not be complete. The first time the blocks are created the goal is to find a *minimum cost perfect matching* in the graph. A perfect matching matches all nodes, or all nodes except one if there is an odd number of nodes. That means that the minimum number of blocks is created. However, we may find that such a matching does not exist, in which case we want to minimize the number of stacks that is unmatched which is a *minimum cost maximum cardinality* matching problem. The unmatched nodes represent blocks that consist of a single stack. If the number of blocks that results this way is more than what fits in the container, we need to select less stacks in the first step.

We use an implementation[1] that finds a *maximum weighted matching*, so in order to use this to solve the minimum cost perfect matching problem we need to convert costs to rewards and add a large constant for each edge such that a higher cardinality matching always has a higher value than a lower cardinality matching. Recall that K is the number of floorspots in a container, so we can have at most $\frac{K}{2}$ blocks with a cost of at most $c^{\max} = \max_{i,j \in \mathcal{S}} c_{ij}$ so the multiplication of these values gives the upper bound on the costs for a matching that can be used as a large constants. This results in rewards r_{ij} for matching stacks i and j as in (3.3).

$$r_{ij} = \frac{K}{2} c^{\max} - c_{ij} \tag{3.3}$$

Although formally the set $\mathcal{S} = \{1, ..., m\}$ of stacks should be a multiset as we may have multiple identical stacks, we consider all pallets to be unique in the input for the matching algorithm. By doing so we introduce symmetry in the problem, but this is not a big concern because the problem can be solved in negligible time for practical instance sizes.

3.2.2 Placing blocks

After the blocks are created they have to be placed in the container. For this we assume that we have fixed positions in the container on which the blocks can be placed. Basically this is an

[1]The LEMON library: `http://lemon.cs.elte.hu/trac/lemon`

assignment problem, but we have the additional objectives for the COG, the support deficiency and the sequences of single centered pallets as described in Section 3.1. The placement of the blocks is found by solving an MIP model, of which the formulation is in Appendix B.1. The next paragraphs introduce the data in the scope of this problem, the decision variables and explain how the support is modeled and how the COG estimation in this formulation is derived.

The data

In the scope of the problem of placing the blocks in the container, let $\mathcal{B} = \{1, ..., m\}$ be the set of blocks. We have no multiplicity as we considered the stacks to be unique in the creation of the blocks and because it is unlikely that symmetry influences the algorithm significantly. Let $\mathcal{Q} = \{1, ..., m\}$ be the positions in the container, so that we have exactly one position per block. For each block $i \in \mathcal{B}$, the following data is derived from the result of the first step:

h_i^+	the height of the tallest stack in the block
h_i^-	the height of the lowest stack in the block
l_i	the length of the block, which is the length of the longest stack in the block
w_i	the weight of the block, which is the sum of the weights of the stacks in the block
n_i	the number of stacks in the block, 1 or 2

Decision variables

The decision variables are the following:

x_{ij}	$\begin{cases} 1 & \text{if block } i \text{ is placed on position } j \\ 0 & \text{otherwise} \end{cases}$	$i \in \mathcal{B}, j \in \mathcal{Q}$
s_j	The maximum weighted support deficiency of position j	$j \in \mathcal{Q}$
s_j^2	The excess of s_j above τ, or 0 if $s_j < \tau$	$j \in \mathcal{Q}$
δ_j	The contribution of block on position j to the scoring for sequences of centered stacks; 0 if position j contains a block with two stacks	$j \in \mathcal{Q}$
ξ	The distance between the estimated COG and the COG*	

Modelling support

In the model, in order to calculate the support deficiency we assume that in each block the tallest stack is placed at the left of the container, such that the tallest stacks in adjacent blocks always support each other. The lower stacks are placed in the right of the container. If there is only one stack in a block, then this stack is centered and therefore supports and is preferably supported by both the tallest and lowest stacks of the adjacent blocks, so we let $h_i^- = h_i^+$. This means that in terms of the data, it should always be that $h_i^+ \geq h_{i'}^+$ and $h_i^- \geq h_{i'}^-$ if blocks i and i' are placed adjacently, or the (weighted) deficiency should be represented by s_j. In order to model the penalty for large support deficiencies, we have the auxiliary variable $s_j^2 = \max\{s_j - \tau, 0\}$.

The modelling is done this way only to guarantee that a solution with the support deficiency exists, but the result may be unrealistic since all taller pallets in the left create a bad weight distribution in the width of the container. However, in practice we typically have some opportunities to swap blocks in which the lower stack could also support the adjacent taller stacks, for a more realistic result.

Modelling sequences of centered stacks

The variable δ represents the position of a centered stack or pallet in a sequence, optionally added with 2 if the sequence is unsupported in the rear of the container. This is basically implemented by the constraint $\delta_j \geq \delta_{j+1} + 1$ if a centered stack is on both position j and $j+1$, with some modifications to ensure that the value is 2 for the last position in the rear, if it contains a single centered stack.

Estimation of COG

Let $\bar{l} = \frac{1}{m}\sum_{i\in\mathcal{B}} l_i$ be the average length of a block and let $\bar{w} = \frac{1}{m}\sum_{i\in\mathcal{B}} w_i$ be the average weight. All blocks are placed adjacently, so the center of a block on position j will be approximately at a distance $(j - \frac{1}{2})\bar{l}$ from the front of the container. If we assume the COG of the block is its geometric center, then also the COG of the block is approximately at position $(j - \frac{1}{2})\bar{l}$. The COG of the load is the weighted average of the COG's of the blocks, where the 'weight' for each block i literally is its weight w_i as a fraction of the total weight $m\bar{w}$. Therefore, if block i is placed on position j in the container the contribution to the COG of the load is estimated by $\frac{w_i}{m\bar{w}}(j - \frac{1}{2})\bar{l}$.

It is possible to extend the model such that the position of block i in the container actually depends on the lengths of the blocks in front of i in the container. This may lead to a more accurate estimation if there is a large variance in the length of the blocks. However, the added complexity to the model is significant, while the estimate is still an estimate because of the reality gap. Therefore, we choose not to extend the model in this way.

3.3 Place stacks directly

If it is not possible to place the blocks in the container such that the estimated COG is within the feasible range, we use an alternate method to place the stacks directly in the container. We use an MIP model which is similar to the model used for placing blocks in the container, but we have separate variables to determine which stacks go on the left and which go on the right of the container. Again, we require that the taller stacks on each position are placed on the left of the container in order to model the support deficiency. For this alternate method, the COG interval constraint is implemented as a soft constraint, because we want to guarantee that a feasible solution exists. Again we have two paragraphs to introduce the data and the decision variables. The third paragraph discusses the differences between the model formulation for the placement of blocks and the model formulation to place stacks directly. The formulation of the MIP model to place the stacks directly is in Appendix B.2.

The data

In the scope of the problem of placing the stacks in the container we want to take into account the multiplicity, and therefore we let $\mathcal{S} = \{1, ..., m\}$ and $n_i, i \in \mathcal{S}$ define the multi-set $(\mathcal{S}, n)$ of stacks. This time, we let $\mathcal{Q} = \{1, ..., q\}$ be the set of positions *potentially* used in the container. On each position, there can be one or two pallets, and we have $q = \frac{K}{2}$ positions in the container. For each stack $i \in \mathcal{S}$, the following data is available:

l_i	the length of the stack
b_i	the width of the stack
h_i	the height of the stack
w_i	the weight of the stack

Decision variables

The decision variables are the following:

x^{l}_{ij}	$\begin{cases} 1 & \text{if stack } i \text{ is placed on position } j \text{ on the left side} \\ 0 & \text{otherwise} \end{cases}$	$i \in \mathcal{B}, j \in \mathcal{Q}$
x^{r}_{ij}	$\begin{cases} 1 & \text{if stack } i \text{ is placed on position } j \text{ on the right side} \\ 0 & \text{otherwise} \end{cases}$	$i \in \mathcal{B}, j \in \mathcal{Q}$
s_j	The maximum weighted support deficiency of position j	$j \in \mathcal{Q}$
s^2_j	The excess of s_j above τ, or 0 if $s_j < \tau$	$j \in \mathcal{Q}$
δ_j	The contribution of the stacks on position j to the scoring for sequences of centered stacks; 0 if position j contains zero or two stacks	$j \in \mathcal{Q}$
ω_j	The maximum length of the stacks on position j; 0 if there are no stacks on position j	$j \in \mathcal{Q}$
ξ	The distance between the estimated COG and the COG*	
Ξ	The distance between the estimated COG and the feasible COG interval	

Differences to placing blocks

In the placement of stacks directly in the container, we may either place one or two stacks in each position. Just as with the placements of blocks, as a way of modelling, we require that the taller stacks are on the left. Therefore, if we have only one stack, in the model it should be on the left, although in the solution it will be centered. Additionally, we do not require that all q positions are used, but for stability we require that the used positions are adjacent. The support constraints need to be reformulated: all stacks should always be supported by the left stack in adjacent positions, since this is the tallest stack on that position. Only if we have two adjacent positions with two stacks, the right stacks in the positions should provide mutual support, since the left stack cannot be placed in the middle for support. With the placement of blocks the feasibility is guaranteed because the blocks fit in the width and it is checked upfront whether the total length of the blocks fits in the container length. Because the stacks are placed directly, we have to check additionally that the pairs of stacks will fit within the width of the container and that the resulting blocks will fit in the length.

Appendix A

Select stacks model formulations

A.1 Select stacks per container basic model

Let $M^{\tau} = \max_{i \in \mathcal{P}} l_i$, $M^{\nu} = \max_{i \in \mathcal{P}} b_i$ and $M^{\varpi} = \max\{t \max_{i \in \mathcal{P}} w_i - \Pi, \Pi\}$. Then the MIP formulation of the basic model for selecting stacks per container is as follows:

$$
\begin{aligned}
\max \quad & C^{\text{VFR}} \left(\chi \frac{1}{W} \sum_{i \in \mathcal{P}} y_i w_i + (1-\chi) \frac{\text{BFD}(\mathcal{P} \setminus \mathcal{P}^m)}{KH \sum\limits_{i \in \mathcal{P} \setminus \mathcal{P}^m} h_i} \sum_{i \in \mathcal{P}} y_i h_i \right) \\
& - C^{\theta} \left(\sum_{k=1}^{\lfloor \theta^* \rfloor} (1 - z_k) + \sum_{k=\lceil \theta^* \rceil}^{\theta^{\max}} z_k \right) \\
& - C^{v} \sum_{j \in O^m} v_j \\
& - \sum_{k \in \mathcal{Q}} \left(C^{l} (\tau_k^+ - \tau_k^-) + C^{b} (\nu_k^+ - \nu_k^-) \right) && (1) \\
\text{s.t.} \quad & \sum_{k \in \mathcal{Q}} \sum_{\ell \in \mathcal{L}} x_{i\ell k} = y_i && i \in \mathcal{P} \quad (2) \\
& z_k \geq x_{i1k} && i \in \mathcal{P}, k \in \mathcal{Q} \quad (3) \\
& z_k \leq \sum_{i \in \mathcal{P}} x_{i1k} && k \in \mathcal{Q} \quad (4) \\
& z_{k+1} \leq z_k && k \in \mathcal{Q} \setminus \{\theta^{\max}\} \quad (5) \\
& \sum_{i \in \mathcal{P}} x_{i\ell k} \geq \sum_{i \in \mathcal{P}} x_{i(\ell+1)k} && k \in \mathcal{Q}, \ell \in \mathcal{L} \setminus \{t\} \quad (6)
\end{aligned}
$$

$$x_{i\ell k} + \sum_{i' \in \mathcal{P}} (1 - a_{i'i}) x_{i'(\ell+1)k} \leq 1 \qquad i \in \mathcal{P}, k \in \mathcal{Q}, \ell \in \mathcal{L} \setminus \{t\} \tag{7}$$

$$u_j + v_j \geq y_i \qquad j \in O^m, i \in \mathcal{P}_j \tag{8}$$

$$u_j \leq y_i \qquad j \in O^m, i \in \mathcal{P}_j \tag{9}$$

$$\sum_{i \in \mathcal{P}} \left(x_{i1k} h_i + \sum_{\ell \in \mathcal{L} \setminus \{1\}} x_{i\ell k} \hat{h}_i \right) \leq H \qquad k \in \mathcal{Q} \tag{10}$$

$$\sum_{k \in \mathcal{Q}} \left(\sum_{i \in \mathcal{P}} \left(x_{i1k} w_i + \sum_{\ell \in \mathcal{L} \setminus \{1\}} x_{i\ell k} \hat{w}_i \right) \right) \leq W \tag{11}$$

$$\tau_k^- - (1 - x_{i\ell k}) M^\tau \leq x_{i\ell k} l_i \leq \tau_k^+ \qquad i \in \mathcal{P}, \ell \in \mathcal{L}, k \in \mathcal{Q} \tag{12}$$

$$\nu_k^- - (1 - x_{i\ell k}) M^\nu \leq x_{i\ell k} b_i \leq \nu_k^+ \qquad i \in \mathcal{P}, \ell \in \mathcal{L}, k \in \mathcal{Q} \tag{13}$$

$$y_i = 1 \qquad i \in \mathcal{P}^m \tag{14}$$

$$\sum_{i \in \mathcal{P} \setminus \mathcal{P}^m : O_i \cap O^m \neq \emptyset} y_i w_i \geq \omega^W \tag{15}$$

$$\sum_{i \in \mathcal{P} \setminus \mathcal{P}^m : O_i \cap O^m \neq \emptyset} y_i h_i \geq \omega^V \tag{16}$$

$$\sum_{i \in \mathcal{P}} \left(x_{i1k} w_i + \sum_{\ell \in \mathcal{L} \setminus \{1\}} x_{i\ell k} \hat{w}_i \right) + \varpi_k^+ - \varpi_k^- = \frac{W}{K} \qquad k \in \mathcal{Q} \tag{17}$$

$$\sum_{k \in \mathcal{Q}} \left(\varpi_k^+ + \varpi_k^- \right) \geq \Pi \tag{18}$$

$$\varpi_k^+ \leq M^\varpi \pi_k \qquad k \in \mathcal{Q} \tag{19}$$

$$\varpi_k^- \leq M^\varpi (1 - \pi_k) \qquad k \in \mathcal{Q} \tag{20}$$

$$\tau_k^- \leq \tau_k^+ \qquad k \in \mathcal{Q} \tag{21}$$

$$\nu_k^- \leq \nu_k^+ \qquad k \in \mathcal{Q} \tag{22}$$

$$0 \leq u_j \leq 1 \qquad j \in O^m \tag{23}$$

$$0 \leq v_j \leq 1 \qquad j \in O^m \tag{24}$$

$$x_{i\ell k} \in \{0, 1\} \qquad i \in \mathcal{P}, \ell \in \mathcal{L}, k \in \mathcal{Q} \tag{25}$$

$$0 \leq y_i \leq 1 \qquad i \in \mathcal{P} \tag{26}$$

$$0 \leq z_k \leq 1 \qquad k \in \mathcal{Q} \tag{27}$$

$$\pi_k \in \{0, 1\} \qquad k \in \mathcal{Q} \tag{28}$$

$$0 \leq \varpi_k^-, \varpi_k^+ \leq M^\varpi \qquad k \in \mathcal{Q} \tag{29}$$

$$0 \leq \tau_k^-, \tau_k^+ \leq M^\tau \qquad k \in \mathcal{Q} \qquad (30)$$

$$0 \leq \nu_k^-, \nu_k^+ \leq M^\nu \qquad k \in \mathcal{Q} \qquad (31)$$

The objective (1) consists of four terms. The first term is the reward for the vehicle fill. The second term subtracts costs for the deviation from the desired number of stacks, which works because the non-empty stacks are adjacent. The third term subtracts the costs for orders that are split and the last term subtracts the costs for the spans for the length and width of each stack. Then we have constraints:

(2)	A pallet is in the container if it is in one stack at one level.
(3)	Stack k must be non-empty if pallet i is in stack k.
(4)	Stack k must be empty if there is no pallet in stack k.
(5)	If stack k is empty then stack $k+1$ must be empty.
(6)	If level ℓ is empty then level $\ell+1$ must be empty.
(7)	If pallet i is at level l prevent pallets i' that cannot stack on i to be at level $l+1$.
(8)	If pallet i is in the container, then corresponding orders must be in container fully or partly.
(9)	If order j is fully in the container, then corresponding pallets must be in the container.
(10)	The stacks must fit in the height of the container if wooden pallets are removed for levels 2 and higher.
(11)	The total weight of pallets in the container, if wooden pallets are removed for levels 2 and higher, should not exceed the maximum weight.
(12), (13)	Values of pallets in stacks must be between their minima and maxima.
(14)	Ensure that must-go pallets are in the container.
(15), (16)	Enforce that at least a total weight ω^W and a total volume, measured by the height, ω^V is selected from the may-go multi-order pallets. Either ω^W or ω^V should be zero so effectively there is one restriction.
(17)	$\varpi_k^+ - \varpi_k^-$ should be the difference between the weight of the stack and the average weight.
(18)	The sum of the deviations of the stacks should be at least the minimum.
(19), (20)	Force only one of ϖ_k^+ and ϖ_k^- to be nonzero according to π_k.

(21), (22)	Assure minimum smaller than maximum (required for empty stacks).
(23) - (31)	Define variables, x_{ilk} binary, binary status of u_j, v_j, y_i and z_k follows. π_k should be binary, while ϖ_k^-, ϖ_k^+, τ_k^-, τ_k^+, ν_k^- and ν_k^+ should be within their ranges.

The variables y_i in the formulation are only for ease of writing, as they may be substituted in each of the constraints according to their definition in (2).

A.2 Elimination of stack symmetry

The ordering of the stacks by the number of times that each pallet is in the stack can be enforced by including constraint (A.2.1). For readability of the proof, we write $w_{ik} = \sum_{\ell \in \mathcal{L}} x_{i\ell k}$. Recall that $\alpha_{|\bar{\mathcal{P}}|} = 1$ and $\alpha_i = (n_{i+1}+1)\alpha_{i+1}$, so that $\alpha_i = \prod_{l=i+1}^{|\bar{\mathcal{P}}|} (n_l + 1)$.

$$\sum_{i=1}^{|\bar{\mathcal{P}}|} \alpha_i (w_{ik} - w_{i(k+1)}) \geq 0, k \in \mathcal{Q} \setminus \{q\} \tag{A.2.1}$$

To show that the constraint works, we use the fact that $\alpha_i > \sum_{l=i+1}^{|\bar{\mathcal{P}}|} \alpha_l w_{lk}$. This follows by (backwards) induction on i with basis $\alpha_{|\bar{\mathcal{P}}|} = 1 > 0$:

$$\alpha_{i-1} = (n_i + 1)\alpha_i = n_i\alpha_i + \alpha_i > w_{ik}\alpha_i + \sum_{l=i+1}^{|\bar{\mathcal{P}}|} \alpha_l w_{lk} = \sum_{l=i}^{|\bar{\mathcal{P}}|} \alpha_l w_{lk}$$

Suppose $w_{1k} < w_{1(k+1)}$. This cannot happen because then $w_{1k} - w_{1(k+1)} \leq -1$ which implies

$$\sum_{i=1}^{|\bar{\mathcal{P}}|} \alpha_i (w_{ik} - w_{i(k+1)}) \leq -\alpha_1 + \sum_{i=2}^{|\bar{\mathcal{P}}|} \alpha_i (w_{ik} - w_{i(k+1)}) \leq -\alpha_1 + \sum_{i=2}^{|\bar{\mathcal{P}}|} \alpha_i w_{ik} < 0$$

However, if $w_{1k} > w_{1(k+1)}$, then $w_{1k} - w_{1(k+1)} \geq 1$ so

$$\sum_{i=1}^{|\bar{\mathcal{P}}|} \alpha_i (w_{ik} - w_{i(k+1)}) \geq \alpha_1 + \sum_{i=2}^{|\bar{\mathcal{P}}|} \alpha_i (w_{ik} - w_{i(k+1)}) \geq \alpha_1 - \sum_{i=2}^{|\bar{\mathcal{P}}|} \alpha_i w_{i(k+1)} > 0$$

In that case, the inequality will be satisfied independent of w_{ik} and $w_{i(k+1)}$ for $i > 1$. If $w_{ik} = w_{i(k+1)}$ then the first term equals zero so the inequality should be satisfied by the remaining terms, for which the same reasoning can be applied. Therefore, these constraints ensure that the solution found follows the ordering.

A.3 Select stacks per container desymmetrized model

The formulation of the problem with the pallet and stack symmetry eliminated is the following:

$$\begin{aligned}
\max \quad & C^{\mathrm{VFR}} \left(\chi \frac{1}{W} \sum_{i \in \bar{\mathcal{P}}} y_i w_i + (1-\chi) \frac{\mathrm{BFD}(\bar{\mathcal{P}} \setminus \bar{\mathcal{P}}^m)}{KH \sum\limits_{i \in \bar{\mathcal{P}} \setminus \bar{\mathcal{P}}^m} h_i} \sum_{i \in \bar{\mathcal{P}}} y_i h_i \right) \\
& - C^{\theta} \left(\sum_{k=1}^{\lfloor \theta^* \rfloor} (1 - z_k) + \sum_{k=\lceil \theta^* \rceil}^{\theta^{\max}} z_k \right) \\
& - C^{v} \sum_{j \in O^m} v_j \\
& - \sum_{k \in \mathcal{Q}} \left(C^l (\tau_k^+ - \tau_k^-) + C^b (\nu_k^+ - \nu_k^-) \right) && && (1) \\
\text{s.t.} \quad & \sum_{k \in \mathcal{Q}} \sum_{\ell \in \mathcal{L}} x_{i\ell k} = y_i && i \in \bar{\mathcal{P}} && (2) \\
& z_k \geq x_{i1k} && i \in \bar{\mathcal{P}}, k \in \mathcal{Q} && (3) \\
& z_k \leq \sum_{i \in \bar{\mathcal{P}}} x_{i1k} && k \in \mathcal{Q} && (4) \\
& z_{k+1} \leq z_k && k \in \mathcal{Q} \setminus \{\theta^{\max}\} && (5) \\
& \sum_{i \in \bar{\mathcal{P}}} x_{i\ell k} \geq \sum_{i \in \bar{\mathcal{P}}} x_{i(\ell+1)k} && k \in \mathcal{Q}, \ell \in \mathcal{L} \setminus \{t\} && (6) \\
& x_{i\ell k} + \sum_{i' \in \bar{\mathcal{P}}} (1 - a_{i'i}) x_{i'(\ell+1)k} \leq 1 && i \in \bar{\mathcal{P}}, k \in \mathcal{Q}, \ell \in \mathcal{L} \setminus \{t\} && (7) \\
& n_i (u_j + v_j) \geq y_i && j \in O^m, i \in \bar{\mathcal{P}}_j && (8) \\
& n_i u_j \leq y_i && j \in O^m, i \in \bar{\mathcal{P}}_j && (9) \\
& \sum_{i \in \bar{\mathcal{P}}} \left(x_{i1k} h_i + \sum_{\ell \in \mathcal{L} \setminus \{1\}} x_{i\ell k} \hat{h}_i \right) \leq H && k \in \mathcal{Q} && (10) \\
& \sum_{k \in \mathcal{Q}} \sum_{i \in \bar{\mathcal{P}}} \left(x_{i1k} w_i + \sum_{\ell \in \mathcal{L} \setminus \{1\}} x_{i\ell k} \hat{w}_i \right) \leq W && && (11) \\
& \tau_k^- - (1 - x_{i\ell k}) M^{\tau} \leq x_{i\ell k} l_i \leq \tau_k^+ && i \in \bar{\mathcal{P}}, \ell \in \mathcal{L}, k \in \mathcal{Q} && (12)
\end{aligned}$$

$$\nu_k^- - (1 - x_{i\ell k})M^\nu \leq x_{i\ell k} b_i \leq \nu_k^+ \qquad i \in \bar{\mathcal{P}}, \ell \in \mathcal{L}, k \in \mathcal{Q} \tag{13}$$

$$y_i = n_i \qquad i \in \bar{\mathcal{P}}^m \tag{14}$$

$$\sum_{i \in \bar{\mathcal{P}} \setminus \bar{\mathcal{P}}^m : O_i \cap O^m \neq \emptyset} y_i w_i \geq \omega^W \tag{15}$$

$$\sum_{i \in \bar{\mathcal{P}} \setminus \bar{\mathcal{P}}^m : O_i \cap O^m \neq \emptyset} y_i h_i \geq \omega^V \tag{16}$$

$$\sum_{i \in \mathcal{P}} \left(x_{i1k} w_i + \sum_{\ell \in \mathcal{L} \setminus \{1\}} x_{i\ell k} \hat{w}_i \right) + \varpi_k^+ - \varpi_k^- = \frac{W}{K} \qquad k \in \mathcal{Q} \tag{17}$$

$$\sum_{k \in \mathcal{Q}} \left(\varpi_k^+ + \varpi_k^- \right) \geq \Pi \tag{18}$$

$$\varpi_k^+ \leq M^\varpi \pi_k \qquad k \in \mathcal{Q} \tag{19}$$

$$\varpi_k^- \leq M^\varpi (1 - \pi_k) \qquad k \in \mathcal{Q} \tag{20}$$

$$\tau_k^- \leq \tau_k^+ \qquad k \in \mathcal{Q} \tag{21}$$

$$\nu_k^- \leq \nu_k^+ \qquad k \in \mathcal{Q} \tag{22}$$

$$u_j \in \{0, 1\} \qquad j \in O^m \tag{23}$$

$$v_j \in \{0, 1\} \qquad j \in O^m \tag{24}$$

$$x_{i\ell k} \in \{0, 1\} \qquad i \in \bar{\mathcal{P}}, \ell \in \mathcal{L}, k \in \mathcal{Q} \tag{25}$$

$$0 \leq y_i \leq n_i \qquad i \in \bar{\mathcal{P}} \tag{26}$$

$$0 \leq z_k \leq 1 \qquad k \in \mathcal{Q} \tag{27}$$

$$\pi_k \in \{0, 1\} \qquad k \in \mathcal{Q} \tag{28}$$

$$0 \leq \varpi_k^-, \varpi_k^+ \leq M^\varpi \qquad k \in \mathcal{Q} \tag{29}$$

$$0 \leq \tau_k^-, \tau_k^+ \leq M^\tau \qquad k \in \mathcal{Q} \tag{30}$$

$$0 \leq \nu_k^-, \nu_k^+ \leq M^\nu \qquad k \in \mathcal{Q} \tag{31}$$

$$\sum_{\ell \in \mathcal{L}} x_{i\ell k} + n_i \sum_{i'=1}^{i-1} s_{i'k} \geq \sum_{\ell \in \mathcal{L}} x_{i\ell(k+1)} + s_{ik} \qquad i \in \bar{\mathcal{P}}, k \in \mathcal{Q} \setminus \{q\} \tag{32}$$

$$0 \leq s_{ik} \leq 1 \qquad i \in \bar{\mathcal{P}}, k \in \mathcal{Q} \setminus \{q\} \tag{33}$$

Note that the subset with the must-go pallets is $(\bar{\mathcal{P}}^m, n)$, which means that the multiplicity n_i in the subset is equal to the multiplicity in the set $\bar{\mathcal{P}}$. This is because the elements represent equivalent pallets, so either all or none are must-go pallet (since this is based on the priority).

The constraints that are adapted (other than replacing $\mathcal{P}$ by $\bar{\mathcal{P}}$) or added are:

(8)	If pallet i is in the container at least 1 time, then corresponding orders must be in container fully or partly, which works because constraints (23) and (24) require u_j and v_j to be binary.
(9)	If order j is fully in the container, then all n_i corresponding pallets must be in the container.
(23), (24)	Require u_j and v_j to be binary.
(32)	Enforce the ordering of the stacks.
(33)	Define s_{ik} to be between 0 and 1 as integrality is not required.

A.4 Alternative formulation

$$\begin{aligned}
\max \quad & C^{\text{VFR}}\left(\chi\frac{1}{W}\sum_{i\in\bar{\mathcal{P}}} y_i w_i + (1-\chi)\frac{\text{BFD}(\bar{\mathcal{P}}\setminus\bar{\mathcal{P}}^m)}{KH\sum_{i\in\bar{\mathcal{P}}\setminus\bar{\mathcal{P}}^m} h_i}\sum_{i\in\bar{\mathcal{P}}} y_i h_i\right) \\
& - C^{\theta} d \\
& - C^{v}\sum_{j\in O^m} v_j \\
& - \sum_{k\in\mathcal{N}} C(S_k)\lambda_k && && (1) \\
\text{s.t.} \quad & \sum_{k\in\mathcal{N}} \lambda_k b_{ik} = y_i && i\in\bar{\mathcal{P}} && (2) \\
& \theta^* - d \le \sum_{k\in\mathcal{N}} \lambda_k \le \min(\theta^* + d, \theta^{\max}) && && (3) \\
& n_i(u_j + v_j) \ge y_i && j\in O^m, i\in\bar{\mathcal{P}}_j && (4) \\
& n_i u_j \le y_i && j\in O^m, i\in\bar{\mathcal{P}}_j && (5) \\
& \sum_{k\in\mathcal{N}} w(S_k)\lambda_k \le W && && (6) \\
& y_i = n_i && i\in\bar{\mathcal{P}}^m && (7) \\
& \sum_{i\in\bar{\mathcal{P}}\setminus\bar{\mathcal{P}}^m : O_i\cap O^m\neq\emptyset} y_i w_i \ge \omega^W && && (8) \\
& \sum_{i\in\bar{\mathcal{P}}\setminus\bar{\mathcal{P}}^m : O_i\cap O^m\neq\emptyset} y_i h_i \ge \omega^V && && (9) \\
& \sum_{k\in\mathcal{N}} \lambda_k \left| w(S_k) - \frac{W}{K} \right| \ge \Pi && && (10)
\end{aligned}$$

$$d \geq 0 \tag{11}$$

$$u_j \in \{0,1\} \qquad j \in O^m \tag{12}$$

$$v_j \in \{0,1\} \qquad j \in O^m \tag{13}$$

$$0 \leq y_i \leq n_i \qquad i \in \bar{\mathcal{P}} \tag{14}$$

$$\lambda_k \in \{0,1,...\} \qquad k \in \mathcal{N} \tag{15}$$

In this formulation, the constraints related to the stacks themselves have been removed from the formulation in Section A.3. The objective (1) has been adapted such that the costs for the deviation from the desired number of stacks are now calculated from the variable d and the costs of the stacks is the summation of the (fixed) costs $C(S_k)$ of the selected stacks. Also a number of constraints is adapted or added:

(2)	The number of times pallet i is in the container is the sum of the occurrences of pallet i in the selected stacks.
(3)	Enforce d to correspond to the deviation from the desired number of stacks.
(6)	The total weight of stacks in the container should not exceed the maximum weight.
(10)	The weight deviations are now fixed for the stacks and the sum of these deviations of the stacks selected should be at least the minimum.
(11), (15)	Define d to be at least 0 and λ_k to be integral. The integrality of d comes with the integrality of λ_k.

A.5 Pricing problem

The MIP formulation of the pricing problem is the following:

$$\begin{aligned} \min \quad & C^l(\tau^+ - \tau^-) + C^b(\nu^+ - \nu^-) \\ & - \delta - \sum_{i \in \bar{\mathcal{P}}} \sum_{\ell \in \mathcal{L}} x_{i\ell}\phi_i - \gamma \left(\sum_{i \in \bar{\mathcal{P}}} x_{i1} w_i + \sum_{\ell \in \mathcal{L} \setminus \{1\}} x_{i\ell} \hat{w}_i \right) - \varphi \left(\varpi^- + \varpi^+ \right) & (1) \\ \text{s.t.} \quad & \sum_{\ell \in \mathcal{L}} x_{i\ell} \leq n_i \qquad i \in \bar{\mathcal{P}} & (2) \end{aligned}$$

$$\sum_{i\in\bar{\mathcal{P}}}\sum_{\ell\in\mathcal{L}} x_{i\ell} \geq 1 \tag{3}$$

$$\sum_{i\in\bar{\mathcal{P}}} x_{i\ell} \geq \sum_{i\in\bar{\mathcal{P}}} x_{i(\ell+1)} \qquad \ell \in \mathcal{L}\setminus\{t\} \tag{4}$$

$$x_{i\ell} + \sum_{i'\in\bar{\mathcal{P}}} (1-a_{i'i})x_{i'(\ell+1)} \leq 1 \qquad i\in\bar{\mathcal{P}}, \ell\in\mathcal{L}\setminus\{t\} \tag{5}$$

$$\sum_{i\in\bar{\mathcal{P}}}\left(x_{i1}h_i + \sum_{\ell\in\mathcal{L}\setminus\{1\}} x_{i\ell}\hat{h}_i\right) \leq H \tag{6}$$

$$\tau^- - (1-x_{i\ell})M^\tau \leq x_{i\ell}l_i \leq \tau^+ \qquad i\in\bar{\mathcal{P}}, \ell\in\mathcal{L} \tag{7}$$

$$\nu^- - (1-x_{i\ell})M^\nu \leq x_{i\ell}b_i \leq \nu^+ \qquad i\in\bar{\mathcal{P}}, \ell\in\mathcal{L} \tag{8}$$

$$\sum_{i\in\mathcal{P}}\left(x_{i1}w_i + \sum_{\ell\in\mathcal{L}\setminus\{1\}} x_{i\ell}\hat{w}_i\right) + \varpi^+ - \varpi^- = \frac{W}{K} \tag{9}$$

$$\varpi^+ \leq M^\varpi \pi \tag{10}$$

$$\varpi^- \leq M^\varpi(1-\pi) \tag{11}$$

$$x_{i\ell}\in\{0,1\} \qquad i\in\bar{\mathcal{P}}, \ell\in\mathcal{L} \tag{12}$$

$$\pi\in\{0,1\} \tag{13}$$

$$0 \leq \varpi^-, \varpi^+ \leq M^\varpi \tag{14}$$

$$0 \leq \tau^-, \tau^+ \leq M^\tau \tag{15}$$

$$0 \leq \nu^-, \nu^+ \leq M^\nu \tag{16}$$

The pricing problem contains the part of the full model in Section A.3 that is relevant for a single stack. Only constraint (3) is added to ensure that at least one pallet is selected. The first part of the objective (1) is the fourth part of the objective of the full model for a single stack. Other parts of the objective for the full model are represented trough the shadow prices of the pallets. In the pricing problem, the second part of the objective contains the shadow prices: for the number of stacks, for the pallets selected, for the weight of the pallets (with optionally wood removed) and the for the absolute deviation in the weight from $\frac{W}{K}$. As the number of stacks is fixed and equal to 1, we may as well remove the constant $-\delta$ from the objective. However, this way the objective value equals the reduced costs of the column that is generated.

Appendix B

Place stacks model formulations

B.1 Place blocks in container model

$$\min \quad C^s \sum_{j\in\mathcal{Q}} s_j + C^{s^2} \sum_{j\in\mathcal{Q}} s_j^2 + C^\delta \sum_{j\in\mathcal{Q}} \delta_j + C^\xi \xi \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{Q}} x_{ij} = 1 \qquad i \in \mathcal{B} \tag{2}$$

$$\sum_{i\in\mathcal{B}} x_{ij} = 1 \qquad j \in \mathcal{Q} \tag{3}$$

$$\sum_{i\in\mathcal{B}} h_i^+ x_{ij} + \frac{s_j}{\beta^{\mathrm{f}}} \geq \sum_{i\in\mathcal{B}} h_i^+ x_{i(j+1)} \qquad j \in \mathcal{Q} \setminus \{m\} \tag{4}$$

$$\sum_{i\in\mathcal{B}} h_i^- x_{ij} + \frac{s_j}{\beta^{\mathrm{f}}} \geq \sum_{i\in\mathcal{B}} h_i^- x_{i(j+1)} \qquad j \in \mathcal{Q} \setminus \{m\} \tag{5}$$

$$\sum_{i\in\mathcal{B}} h_i^+ x_{i(j+1)} + \frac{s_j}{\beta^{\mathrm{r}}} \geq \sum_{i\in\mathcal{B}} h_i^+ x_{ij} \qquad j \in \mathcal{Q} \setminus \{m\} \tag{6}$$

$$\sum_{i\in\mathcal{B}} h_i^- x_{i(j+1)} + \frac{s_j}{\beta^{\mathrm{r}}} \geq \sum_{i\in\mathcal{B}} h_i^- x_{ij} \qquad j \in \mathcal{Q} \setminus \{m\} \tag{7}$$

$$\frac{s_m}{\beta^r} \geq \sum_{i\in\mathcal{B}} h_i^+ x_{im} \tag{8}$$

$$s_j^2 \geq s_j - \tau \qquad j \in \mathcal{Q} \tag{9}$$

$$\mathrm{COG}^* - \xi \leq \sum_{i\in\mathcal{B}} \sum_{j\in\mathcal{Q}} \frac{w_i}{m\bar{w}} \left(j - \frac{1}{2}\right) \bar{l} x_{ij} \leq \mathrm{COG}^* + \xi \tag{10}$$

$$\mathrm{COG}^- \leq \sum_{i\in\mathcal{B}} \sum_{j\in\mathcal{Q}} \frac{w_i}{m\bar{w}} \left(j - \frac{1}{2}\right) \bar{l} x_{ij} \leq \mathrm{COG}^+ \tag{11}$$

$$m \sum_{i \in \mathcal{B}: n_i > 1} \big(x_{ij} + x_{i(j+1)}\big) + \delta_j \geq \delta_{j+1} + 1 \qquad j \in \mathcal{Q} \setminus \{m\} \qquad (12)$$

$$2 \sum_{i \in \mathcal{B}: n_i > 1} x_{im} + \delta_m \geq 2 \qquad (13)$$

$$x_{ij} \in \{0, 1\} \qquad i \in \mathcal{B}, j \in \mathcal{Q} \qquad (14)$$

$$s_j \geq 0 \qquad j \in \mathcal{Q} \qquad (15)$$

$$s_j^2 \geq 0 \qquad j \in \mathcal{Q} \qquad (16)$$

$$\delta_j \geq 0 \qquad j \in \mathcal{Q} \qquad (17)$$

$$\xi \geq 0 \qquad (18)$$

The objective (1) contains four terms. The first term adds costs C^s for the maximum weighted support deficiency for each row, where the weights β^{f} and β^{r} for the front and rear support are enforced by scaling of s_j in the constraints. The second term adds additional costs C^{s^2} where the maximum weighted support deficiency exceeds τ. The third term adds costs C^δ for the sequences of blocks with only one stack or pallet. These costs are quadratic in the length of the sequence since for each additional 'single stack block' in a sequence the value of δ increases. The last term adds costs C^ξ for the distance of the estimated COG from the desired COG. The constraints enforce the following:

(2)	Each block should be on one position.
(3)	On each position there should be one block.
(4)	The highest stack in the block on position j should support the highest stack in the block on position $j + 1$, or s_j should be equal to the lack of support.
(5)	The lowest stack in the block on position j should support the lowest stack in the block on position $j + 1$, or s_j should be equal to the lack of support. This also works if one of both blocks has only one stack, since that stack is automatically also the lowest and that stack supports the stack in the other block since it is centered in the container.
(6)	Similar to constraint (4), but for the rear support.
(7)	Similar to constraint (5) but for the rear support.
(8)	The last block is never supported from the rear, so add lack of support for the tallest stack in the last block. This prevents that the last stack is too tall.
(9)	s_j^2 should represent how much s_j exceeds τ, or be zero otherwise.

(10)	Enforce that ξ represents the distance from the estimation of the COG to the desired COG.
(11)	Enforce that the estimation of the COG is within the interval.
(12)	δ_j should be $\delta_{j+1}+1$ unless the block on position j is a block with two stacks or a block with two stacks is behind it on position $j+1$.
(13)	On the last position there should be a block with two stacks or it should hold that $\delta_m = 2$.
(14) - (18)	Define x_{ij} to be binary and s_j, s_j^2, δ_j and ξ to be at least 0.

B.2 Place stacks in container model

$$\min \quad C^s \sum_{j\in\mathcal{Q}} s_j + C^{s^2} \sum_{j\in\mathcal{Q}} s_j^2 + C^\delta \sum_{j\in\mathcal{Q}} \delta_j + C^\xi \xi + C^\Xi \Xi \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{Q}} x^{\mathrm{l}}_{ij} + \sum_{j\in\mathcal{Q}} x^{\mathrm{r}}_{ij} = n_i \qquad i \in \mathcal{S} \tag{2}$$

$$\sum_{i\in\mathcal{S}} x^{\mathrm{l}}_{ij} \leq 1 \qquad j \in \mathcal{Q} \tag{3}$$

$$\sum_{i\in\mathcal{S}} x^{\mathrm{r}}_{ij} \leq 1 \qquad j \in \mathcal{Q} \tag{4}$$

$$\sum_{i\in\mathcal{S}} x^{\mathrm{l}}_{ij} \geq \sum_{i\in\mathcal{S}} x^{\mathrm{l}}_{i(j+1)} \qquad j \in \mathcal{Q}\setminus\{q\} \tag{5}$$

$$\sum_{i'\in\mathcal{S}:h_{i'}\geq h_i} x^{\mathrm{l}}_{i'j} \geq x^{\mathrm{r}}_{ij} \qquad i \in \mathcal{S}, j \in \mathcal{Q} \tag{6}$$

$$\sum_{i\in\mathcal{S}} x^{\mathrm{l}}_{ij} b_i + \sum_{i\in\mathcal{S}} x^{\mathrm{r}}_{ij} b_i \leq B \qquad j \in \mathcal{Q} \tag{7}$$

$$\omega_j \geq x^{\mathrm{l}}_{ij} l_i \qquad i \in \mathcal{S}, j \in \mathcal{Q} \tag{8}$$

$$\omega_j \geq x^{\mathrm{r}}_{ij} l_i \qquad i \in \mathcal{S}, j \in \mathcal{Q} \tag{9}$$

$$\sum_{j\in\mathcal{Q}} \omega_j \leq L \tag{10}$$

$$\sum_{i\in\mathcal{S}} h_i x^{\mathrm{l}}_{ij} + \frac{s_j}{\beta^{\mathrm{f}}} \geq \sum_{i\in\mathcal{S}} h_i x^{\mathrm{l}}_{i(j+1)} \qquad j \in \mathcal{Q}\setminus\{q\} \tag{11}$$

$$\sum_{i\in\mathcal{S}} h_i x^{\mathrm{l}}_{i(j+1)} + \frac{s_j}{\beta^{\mathrm{r}}} \geq \sum_{i\in\mathcal{S}} h_i x^{\mathrm{l}}_{ij} \qquad j \in \mathcal{Q}\setminus\{q\} \tag{12}$$

$$\frac{s_q}{\beta^r} \geq \sum_{i\in\mathcal{S}} h_i x^{\mathrm{l}}_{iq} \tag{13}$$

$$\sum_{i\in\mathcal{S}} h_i x^{\mathrm{l}}_{ij} + \frac{s_j}{\beta^{\mathrm{f}}} \geq \sum_{i\in\mathcal{S}} h_i x^{\mathrm{r}}_{i(j+1)} \qquad j \in \mathcal{Q}\setminus\{q\} \qquad (14)$$

$$\sum_{i\in\mathcal{S}} h_i x^{\mathrm{l}}_{i(j+1)} + \frac{s_j}{\beta^{\mathrm{r}}} \geq \sum_{i\in\mathcal{S}} h_i x^{\mathrm{r}}_{ij} \qquad j \in \mathcal{Q}\setminus\{q\} \qquad (15)$$

$$\sum_{i\in\mathcal{S}} h_i x^{\mathrm{r}}_{ij} + \frac{s_j}{\beta^{\mathrm{f}}} \geq \sum_{i\in\mathcal{S}} h_i x^{\mathrm{r}}_{i(j+1)} - H\left(1 - \sum_{i\in\mathcal{S}} x^{\mathrm{r}}_{ij}\right) \qquad j \in \mathcal{Q}\setminus\{q\} \qquad (16)$$

$$\sum_{i\in\mathcal{S}} h_i x^{\mathrm{r}}_{i(j+1)} + \frac{s_j}{\beta^{\mathrm{r}}} \geq \sum_{i\in\mathcal{S}} h_i x^{\mathrm{r}}_{ij} - H\left(1 - \sum_{i\in\mathcal{S}} x^{\mathrm{r}}_{i(j+1)}\right) \qquad j \in \mathcal{Q}\setminus\{q\} \qquad (17)$$

$$\sum_{i\in\mathcal{S}} h_i x^{\mathrm{r}}_{ij} + \frac{s_j}{\beta^{\mathrm{s}}} \geq \sum_{i\in\mathcal{S}} h_i x^{\mathrm{l}}_{ij} \qquad j \in \mathcal{Q}\setminus\{q\} \qquad (18)$$

$$s^2_j \geq s_j - \tau \qquad j \in \mathcal{Q} \qquad (19)$$

$$\mathrm{COG}^* - \xi \leq \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{Q}} \frac{w_i}{m\bar{w}} \left(j - \frac{1}{2}\right) \bar{l} \left(x^{\mathrm{l}}_{ij} + x^{\mathrm{r}}_{ij}\right) \leq \mathrm{COG}^* + \xi \qquad (20)$$

$$\mathrm{COG}^- - \Xi \leq \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{Q}} \frac{w_i}{m\bar{w}} \left(j - \frac{1}{2}\right) \bar{l} \left(x^{\mathrm{l}}_{ij} + x^{\mathrm{r}}_{ij}\right) \leq \mathrm{COG}^+ + \Xi \qquad (21)$$

$$m \sum_{i\in\mathcal{S}} \left(x^{\mathrm{r}}_{ij} - x^{\mathrm{l}}_{ij} + 1\right) + \delta_j \geq \delta_{j+1} + 2 - \sum_{i\in\mathcal{S}} \left(x^{\mathrm{r}}_{i(j+1)} + x^{\mathrm{l}}_{i(j+1)}\right) \qquad j \in \mathcal{Q}\setminus\{q\} \qquad (22)$$

$$2 \sum_{i\in\mathcal{S}} \left(x^{\mathrm{r}}_{ij} - x^{\mathrm{l}}_{ij} + 1\right) + \delta_m \geq 2 \qquad (23)$$

$$x^{\mathrm{l}}_{ij} \in \{0,1\} \qquad i \in \mathcal{S}, j \in \mathcal{Q} \qquad (24)$$

$$x^{\mathrm{r}}_{ij} \in \{0,1\} \qquad i \in \mathcal{S}, j \in \mathcal{Q} \qquad (25)$$

$$s_j \geq 0 \qquad j \in \mathcal{Q} \qquad (26)$$

$$s^2_j \geq 0 \qquad j \in \mathcal{Q} \qquad (27)$$

$$\delta_j \geq 0 \qquad j \in \mathcal{Q} \qquad (28)$$

$$\omega_j \geq 0 \qquad j \in \mathcal{Q} \qquad (29)$$

$$\xi \geq 0 \qquad (30)$$

$$\Xi \geq 0 \qquad (31)$$

The objective for this model is the same as for placing blocks in the container. However, in this model we want to guarantee feasibility, so the COG-constraint is implemented as a soft constraint. The violation Ξ is therefore penalized in the objective with the (high) costs C^{Ξ}.

(2)	Each stack should be on one position, either left or right.
(3), (4)	On each position, both left and right, there may be at most one stack.

(5)	If position j is empty, then position $j+1$ must be empty. Only for the left as the right must be empty if the left is empty by constraint (6).
(6)	The right position may only be used by a stack if a taller stack is on the left position.
(7)	The stacks placed next to each other must fit in the width of the container.
(8), (9)	The length of position j in the container must be greater than the length of the stacks on position j.
(10)	The total length of the positions in the container must fit in the length of the container.
(11)	The left stack on position j should front-support the left stack on position $j+1$.
(12)	The left stack on position $j+1$ should rear-support the left stack on position j.
(13)	The last stack is never supported from the rear so add lack of support for that.
(14)	The left stack on position j should front-support the right stack on position $j+1$. This is because there may not be a right stack on position j in which case the left stack is centered and provides the support. If there is a right stack, then the left stack is taller such that this constraint is automatically satisfied if constraint (16) is satisfied.
(15)	The left stack on position $j+1$ should rear-support the right stack on position j. The explanation is the same as with constraint (14).
(16)	The right stack on position j should front-support the right stack on position $j+1$ but only if there is a right stack on position j, otherwise the support is provided by the left stack.
(17)	The right stack on position $j+1$ should rear-support the right stack on position j but only if there is a right stack on position $j+1$, otherwise the support is by the left stack.
(18)	The right stack should support the left stack on the same position. In the other direction there is always sufficient support since the left stack is taller than the right one.
(19)	s_j^2 should represent how much s_j exceeds τ, or be zero otherwise.
(20)	Enforce that ξ represents the how far the estimation of the COG is from the desired COG.

(21)	Enforce that Ξ represents the how far the estimation of the COG is outside the interval.
(22)	δ_j should be $\delta_{j+1} + 1$ unless on position j there are two stacks or there are two stacks on position $j+1$. If there are no stacks on position $j+1$, then $\delta_j = 2$ if there is only one stack on position j.
(23)	On the last position it should hold that $\delta_m = 2$ if there is exactly one stack.
(24) - (31)	Define x^{l}_{ij} and x^{r}_{ij} to be binary and s_j, s^2_j, δ_j, ω_j, ξ and Ξ to be at least 0.

List of symbols

$a_{ii'}$	Entry of A: 1 if pallet i can be stacked on pallet i'
b_i	The width of pallet, a stack or a block i
b_k	The multiplicity vector for stack k
b_{ik}	The number of times unique pallet i is in stack S_k
$c_{ii'}$	Entry of C: the costs of a stack of pallet i and i'
c_{ij}	Cost of matching stack i and j as a block
$\hat{c}_{ii'}$	Entry of $\hat{C}$: the reduced costs of a stack of pallet i and i'
$\hat{c}^*$	Minimum of $\hat{C}$: the minimum reduced costs
h_i	The height pallet i including the wood, or the height of a stack or block i
$\hat{h}_i$	The height of pallet i with the wood removed, if wood may be removed, otherwise $\hat{h}_i = h_i$
h_i^+	The height of the tallest stack in a block
h_i^-	The height of the lowest stack in a block
l_i	The length of pallet, a pallet or a block i
m	The number of (unique) pallets, or the number of (unique) stacks or the number of blocks
n	The number of feasible stacks
n_i	The multiplicity of unique pallet i or unique stack i
p_i	The priority of pallet i
q	The number of positions in the length of the container
r_{ij}	The rewards for matching stacks i and j as a block
s	The number of orders
t	Maximum number of pallets in a stack

w_i	The weight of pallet i including the wood
$\hat{w}_i$	The weight of pallet i with the wood removed, if wood may be removed, otherwise $\hat{w}_i = w_i$
rc^*	The minimum reduced costs of a stack
A	Stackability matrix
B	Width of the container
C	Cost matrix for stacks of two pallets
C^b	Costs for each unit of span of the widths of pallets in a stack
C^l	Costs for each unit of span of the lengths of pallets in a stack, or stacks in a block
C^h	Costs for each unit of span of the heights of stacks in a block
C^s	Costs for support deficiency
C^{s^2}	Costs for support deficiency that exceeds the threshold τ
C^v	Costs for splitting an order
C^{VFR}	Reward for the combined VFR of the container
C^θ	Costs for deviation from the desired number of stacks
C^ξ	Costs for the distance from the COG to the desired COG
C^Ξ	Costs for the distance the COG is outside the feasible COG interval
$\hat{C}$	Reduced cost matrix for stacks of two pallets
D	Set of pallets in a cut
H	Height of the container
K	Number of floorspots of the container
L	Length of the container
O	The set of orders
$O_i \subset O$	the orders for which the pallet contains at least one product
S, S_k	Subset of pallets representing a stack
S_{k^*}	Stack with minimum reduced costs
W	Maximum weight allowed in the container
COG^*	The desired COG
COG^-	Lower bound for the COG interval
COG^+	Upper bound for the COG interval
$\mathcal{B}$	The set of blocks
$\mathcal{L}$	The potential levels in a stack

$\mathcal{N}$	Set of indices of $\mathfrak{S}$
$\mathcal{P}, \bar{\mathcal{P}}$	The set of (unique) pallets
$\mathcal{P}_j, \bar{\mathcal{P}}_j$	The set of (unique) pallets that contain a product of order j
$\mathcal{P}^m, \bar{\mathcal{P}}^m$	The set of (unique) must-go pallets
$\mathcal{Q}$	Set of (potential) stacks or set positions in the container
$\mathcal{S}$	Set of stacks that has to be loaded in the container
$\mathfrak{S}$	Set of all feasible stacks
β^{f}	Factor for importance of front support
β^{r}	Factor for importance of rear support
β^{s}	Factor for importance of side support
γ	Shadow price for the weight of a stack
δ	Shadow price for a stack
φ	Shadow price for the deviation of the weight of a stack from $\frac{W}{K}$
χ	The weight factor for the weight/volume VFR
τ	Threshold for additional penalty if the maximum weighted support deficiency exceeds value
ϕ_i	Shadow price for pallet i in a stack
π_j	Shadow price for cut j
θ	Number of stacks in the container
$\theta^{\max}$	Maximum number of stacks to load in the container
θ^{*}	The desired number of stacks to load in the container
ω^{W}	Minimum total weight for the may-go multi-order pallets selected in the container
ω^{V}	Minimum total height for the may-go multi-order pallets selected in the container
Δ	The number of orders that is split
Π	The minimum diversity in weight of stacks selected in the container
$\mathrm{BFD}(\hat{\mathcal{P}})$	Number of stacks according to the BFD method for a set of pallets $\hat{\mathcal{P}}$
$C(S_k)$	The costs of stack S_k
$N(\hat{\mathcal{P}})$	Lower bound on number of containers for a set of pallets $\hat{\mathcal{P}}$
$w(S_k)$	The weight of stack S_k

Bibliography

D. M. Ryan and B. A. Foster. An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling*, pages 269–280, 1981.