# ORTEC Pallet and Load Building

Wouter Kool

1937189

(3D model of a container)

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

# Preface

This thesis has been written as the final part of the Master Program Business Analytics at the VU University in Amsterdam. It describes the results of the research that has been done at the Pallet and Load Building department of ORTEC. The research focuses on an important part of the business problems of ORTEC's customers: the problem of finding the optimal assignment of pallets to containers and placement of these pallets within these containers.

As part of this research I have been at ORTEC's business unit in Atlanta, Georgia, United States. What I have learned during my three month visit is described as part of this thesis. The other part concerns the development of a new algorithm for load building. An outline of this algorithm is given in this document, but all technical details are described in a complimentary document.

I would like to thank everyone who has in some way contributed to this work. First of all, I want to thank ORTEC as a company for giving me the opportunity to go to the US and helping me to organize everything that was needed for my visit. The people at the Atlanta business unit have been very welcoming and I am grateful for the time they invested in sharing with me their knowledge. I would like to thank my ORTEC supervisors, Joaquim Gromicho in the Netherlands and Bobby Miller in the US, for their help. Finally, I would like to thank my VU supervisor René Bekker for his useful feedback during the writing of this thesis, and Gerrit Timmer for being the second reader.

# Summary

ORTEC Pallet and Load Building (OPLB) is the successor of ORTEC LoadDesigner: a software product that implements container loading algorithms to allow customers to optimize the *vehicle fill rate* (VFR) of truck loads. Traditionally, loading algorithms are *construction heuristics* that repeatedly 'build up' a solution item by item, which makes it hard to incorporate global constraints. ORTEC started the development of the *Pallet Grid Assignment* (PGA) algorithm to better deal with such constraints. This alternative uses the assumption that pallets are loaded in a $2 \times 2 \times n$-grid to simplify the problem, such that a 'global' approach can be used. The goal of this research is to find out if and how the PGA algorithm can be used for customers in the US. A field study is done in the US and based on that the algorithm is improved.

In the study on the US situation we give important insights regarding different topics relevant for load building, such as weight distribution, loading patterns, stackability, stability and priorities of products. Most important for the PGA algorithm are the considerations when simultaneously dealing with axle weight restrictions and stability. Additionally, we identify business opportunity if the requirements for priorities of products are reformulated such that different priorities can be mixed.

In this thesis, the PGA algorithm is improved by relaxing the grid assumption in the third dimension, which means that the algorithm can create stacks with more than two pallets. In order
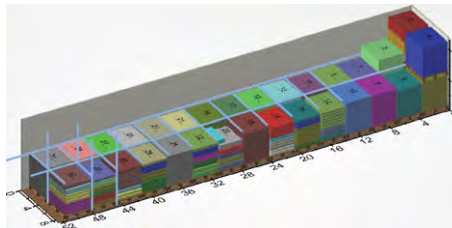
Figure 1: Example of pallets loaded in an imaginary 'grid'.

to be able to create stacks of more than two pallets, the 'improved PGA algorithm' integrates the problems of creating stacks and assigning pallets, or stacks, to containers. Therefore the first of the two subproblems is to create stacks for each container. This problem is solved using a model that creates stacks for each container, one at a time. The model is solved using column generation with partial application of branch and price. The second subproblem is the placement of the stacks within the container, which is solved using Mixed Integer Program (MIP) models. Based on the study in the US, the model takes into account axle weight restrictions and stability of the load, also with dynamic loading patterns. Optionally the stacks are first matched in pairs of two, in order to create blocks for an easier placement problem.

We compare the results of the improved PGA algorithm and the construction heuristic in LoadDesigner in terms of the *Vehicle Fill Rate* (VFR) of the last container. A lower rate is preferred since then more products are in the full containers. For most instances in the data set, the results in terms of VFR are similar. This is satisfying because the implementation in LoadDesigner is the result of many years of experience in load building. For some instances the PGA algorithm performs better than the construction heuristic. There is one scenario in which the VFR in the last container is lower using the construction heuristic, but in that scenario the axle weight restrictions are violated, while the PGA algorithm satisfies them.

Not all requirements, such as complex grouping constraints, can be efficiently implemented with the PGA algorithm, which means that in practice it should be used in parallel with the construction heuristic. Nevertheless we think that the PGA algorithm has high potential. For appropriate scenarios, it achieves results similar to the construction heuristic, while the construction heuristic uses multiple iterations. The PGA algorithm uses a single iteration and solves each subproblem only once, unless the second subproblem is infeasible. The method for the selection of stacks is very efficient, but the method for placing the stacks can be improved. This is one of the directions for future research.

# Contents

# Chapter 1

# Introduction

ORTEC is a Dutch-based software and consulting company that is one of the largest global suppliers of advanced planning solutions. One of these solutions is *ORTEC Pallet and Load Building* (OPLB), which is the topic of this research. OPLB is a software product that implements container loading algorithms to allow customers to optimize the *vehicle fill rate* (VFR) of truck loads and therefore minimize the number of trucks.

In order for ORTEC to remain an important player on the market, ORTEC continuously works on improving its products. In this regard, OPLB has been developed as the successor of LoadDesigner, a product that has evolved from many years of experience in load building. Compared to LoadDesigner, the biggest improvement of OPLB is that it allows for complete configuration of the algorithm without changing the software itself. This is achieved by generalizing a wide scope of generally accepted loading methodologies, which are called *construction heuristics*. As LoadDesigner also implements construction heuristics, OPLB generalizes the methods in LoadDesigner. The main benefit of OPLB is in terms of maintainability for ORTEC, as OPLB no longer requires to have many different versions to meet specific requirements of different customers. Moreover, thanks to the abstraction it is possible to use parameter tuning technologies to systematically develop optimal algorithms specific to each customer's situation (van Dijk, 2014).

## 1.1   Background

At this moment, there are still many customers that use LoadDesigner, as from a customer's perspective there is little benefit in migrating to OPLB compared to the efforts to do so. This is

especially the case for large customers, as migrations are typically preceded by a long trajectory of implementing and testing. If OPLB would, at least in one respect, be able to do significantly better than LoadDesigner, this would increase the added value of implementing OPLB for the customer. Moreover, new customers may be attracted by the increased functionality.

A major opportunity lies in some of the customer requirements. The construction heuristics implemented in LoadDesigner and the current version of OPLB 'build up' a solution item by item and repeat this process with some randomization, to end up with the best solution that was found. These type of 'build up' approaches allow to take into account many item-level requirements, such as items or orders that should (or should not) be shipped together (in close proximity) in a container, or be stacked. However, it is hard to incorporate global requirements, such as weight distribution and stability, as the influence of a single item placement is unclear until the container is fully loaded.

In the US, loaded trucks must comply with axle weight regulations because they may be denied access to certain states or get significant fines otherwise. Especially for large customers this is problematic as they ship mainly full (heavy) truckloads and stability requirements are usually enforced by loading taller or stacked pallets in the front of the container, causing an axle weight violation. Some customers have expressed their need for a better solution with regard to axle weight restrictions. Although a construction heuristic can be configured to accept only solutions that satisfy axle weight restrictions, this may result in poor quality solutions in terms of the *Key Performance Indicators* (KPI's) and there is no guarantee that a solution is found at all.

## 1.2   A new approach

In order to be able to optimize truck loads while actually taking into account the global requirements, ORTEC has developed an idea for an alternative approach that takes advantage of additional assumptions based on the situation in the US. Using these assumptions, the problem is simplified such that it may be solved using a 'global' approach rather than building up a solution. Basically the idea is that items are always on pallets and, under the assumption of standard pallet sizes, these pallets can be placed in an imaginary $2 \times 2 \times n$-grid (see Figure 1.1), such that the loading problem becomes an assignment problem of pallets to *cells* in a container. This is why we refer to this new approach as the *Pallet Grid Assignment* (PGA) algorithm. In the algorithm, the size of the assignment problem is reduced by one-to-one matching pallets
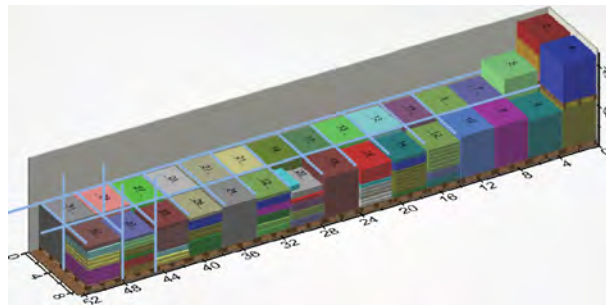
Figure 1.1: Example of pallets loaded in an imaginary 'grid'.

to create *stacks* of two pallets. Then matching is used to create pairs of stacks, called *blocks*, which is done either before or after the pallets are assigned to containers. As a *proof of concept*, ORTEC has created a very basic implementation of the PGA algorithm, which is described in Appendix A.

Some of the assumptions that have been used by the proof of concept implementation of the PGA algorithm do not hold in practice. For instance, in some cases a stack of three pallets is required in order to find a good solution. ORTEC requires to know whether there are other assumptions that are invalid or whether there are additional assumptions that can be used. In other words, ORTEC needs to know how load building is used in practice. This gives insight in how the PGA algorithm should be adapted in order to make it usable in practice.

## 1.3   Research goal

The goal of this research is to find out if and how the PGA algorithm can be used beneficially for customers in the US. Therefore, the validity of the assumptions has to be investigated and the assumptions that do not hold should be reformulated. Based on this the proof of concept should be further worked out as a first step towards the development of an algorithm that can be used to solve some of the loading problems for customers in the US. Finally, the quality of the solutions provided by the resulting algorithm should be compared to the traditional construction heuristic in order to get insights in the potential benefits for customers.

## 1.4   Research questions

In order to meet the research goal and to define the scope of this research, we formulate a number of research questions:

1. What relevant previous research has been done?

2. What are the different concepts that are relevant for load building in the US?

3. How can the PGA algorithm be improved?

4. How well does the improved PGA algorithm perform compared to the construction heuristic?

Chapter 2 answers the first question in the form of a literature review. This literature review discusses related work on the Container Loading Problem, but also some more general models that are related to the methods used as part of the improvements that are made to the PGA algorithm. The second question will be answered by a number of sections in Chapter 3, each discussing a requirement or concept that appears with load building in the US. Chapter 4 answers the third question. It first discusses the differences between the PGA algorithm and the construction heuristic. Then for each requirement or concept in Chapter 3 the potential contribution to the complexity of the problem for the PGA algorithm is balanced against the practical relevance. Simultaneously, the chapter discusses which requirements are considered in this thesis and based on this the improved PGA algorithm is developed. The algorithm consists of two steps, where the first step iteratively creates stacks for a single container and the second step places the stacks in the container. For readability the technical explanation is in a separate but complimentary document (Kool, 2014). As an answer to the final question, Chapter 5 compares the quality of solutions created by the improved PGA algorithm with the results from the construction heuristic in LoadDesigner. Finally, Chapter 6 concludes the research with summarizing what has been learned from the study in the US, stating the potential of the improved PGA algorithm and pointing directions for future research.

# Chapter 2

# Literature review

This chapter discusses the literature that is relevant for our research. Section 2.1 discusses the container loading problem in general and describes attempts that have been made to include additional constraints. Section 2.2 describes the set partitioning problem and column generation as a solution method. This is relevant as we model the problem of creating stacks for a single container as a variant of the set partitioning problem with some additional constraints, and we use column generation to solve the model. Finally, Section 2.3 discusses the generalized assignment problem. This problem has a number of variants, of which some are very similar to the problem of placement of stacks in the container.

## 2.1 The Container Loading Problem

This section discusses the Container Loading Problem (CLP). First the typology of the wider family of Cutting and Packing problems is discussed. Then, different solutions to the CLP are discussed, as well as attempts to include weight distribution considerations and other additional requirements and an exact method for the CLP.

### 2.1.1 Typology of Cutting and Packing problems

The CLP is part of the wider family of *Cutting and Packing* (C&P) problems, which are problems concerned with packing small items in or cutting them from large objects. Because problems in this family have appeared under various names in the literature, Dyckhoff (1990) introduced a typology that can be used for a systematic classification based on the basic logical structure of

the problems. Dyckhof defines four characteristics of a C&P problem:

- **Dimensionality** $(1/2/3/N)$: the physical dimensionality of the problem

- **Kind of assignment** (B/V): whether to assign a selection of small items to all large objects (B) or all small items to a selection of large objects (V)

- **Assortment of large objects** (O/I/D): whether there is one object (O), multiple objects with identical figures (I), or multiple objects with different figures (D)

- **Assortment of small items** (F/M/R/C): whether there are a few small items of different figures (F), many items of many different figures (M), many items of relatively few different figures (R), or items of congruent figures (C)

Taking a value for each characteristic a problem is denoted by a fourth-tuple $\alpha/\beta/\gamma/\delta$. Using this typology, the container loading problem is referred to as either $3/V/I/$ or $3/B/O/$. The former variant concerns packing a fixed set of items in as few containers as possible, and the latter packing as many items as possible in a single container.

Wäscher et al. (2007) build on this typology to define an improved typology, by adding a fifth criterium for the shape of the small items. They suggest a name for six basic problem types defined by combinations of the properties dimensionality, kind of assignment and assortment of small items, as is displayed in Figure 2.1. Each basic problem type is worked out in intermediate problem types, depending on the other properties, while problems with additional constraints are considered as variants. In this framework, the container loading problem for a fixed set of items falls within the category of (three-dimensional) bin packing problems.

## 2.1.2   Solutions to the Container Loading Problem

The CLP has been extensively studied in the literature, but mainly in the context of practical applications. Bischoff and Ratcliff (1995) argue that the approaches developed are only applicable to a fraction of the wide scope of situations that occur in practice. They give a number of practical requirements that play an important role in the development of these methods, and discuss the implications of these requirements for the theoretical work of developing solution methods to the problem. As a result, they express the need for a way to compare different approaches and describe a reproducible mechanism to generate different types of data sets that can be used to evaluate different loading methodologies. In their paper, they also describe two loading methodologies, each with a different focus. The first one takes into account the stability
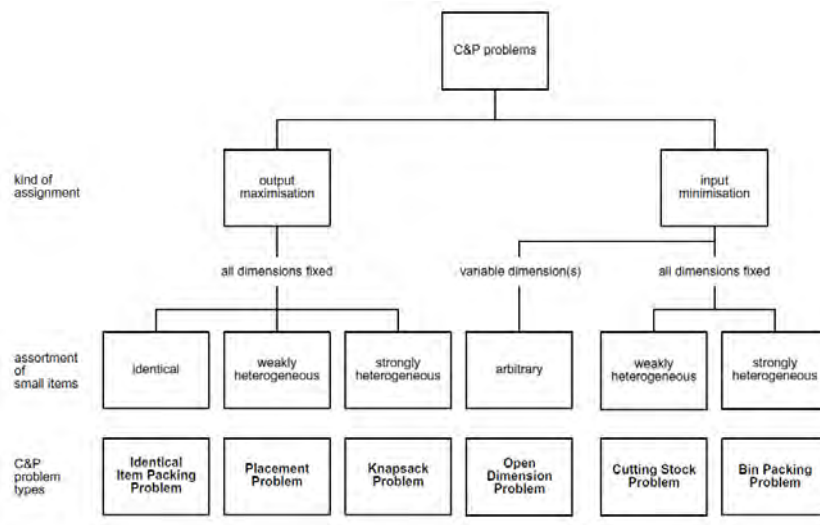
Figure 2.1: Basic problem types of Cutting and Packing problems according to Wäscher et al. (2007).

of the load, for which they define a measure. The other allows for multiple vehicle stops by packing different sections for different destinations. Both methods are compared in terms of volume utility using their own benchmark set. Despite the additional practical requirements, the results in terms of space utilization are comparable to other methods that do not consider these requirements.

Pisinger (2002) discusses the *Knapsack Container Loading Problem* (KCLP), that concerns packing the highest volume of items in a single container with fixed dimensions. As the problem is NP-hard, optimal solutions can only be obtained for small sized instances. Therefore, research has focused on the development of heuristic approaches that can be used in practice. Most of these heuristics somehow build up a solution by iteratively placing an item in a certain arrangement. Pisinger classifies them as wall building algorithms, stack building algorithms, guillotine cutting algorithms and cuboid arrangement algorithms. This classification is based on the shape of the arrangements in which the items are placed.

### 2.1.3 Weight distribution in container loading

One important practical constraint that is only occasionally referred to in the literature, is the weight distribution. Most studies have been done in the context of aircraft loading, in which case

the weight distribution is considered in one dimension. This means that the lengthwise center of gravity has a desired target. As an example, Brosh (1981) considers the (fractional) cargo distribution over multiple bays within a plane, using a non-linear model which is solved by solving a sequence of linearized models that converge to the optimum. Amiouny et al. (1992) develop two more general heuristics that are able to pack blocks into a bin so that their center of gravity is as close as possible to a target. Davies and Bischoff (1999) study the weight distribution in the context of container loading and develop an approach based on the general container loading algorithm proposed by Gehring et al. (1990). Davies and Bischoff adapt this algorithm, which consist of building layers, such that blocks of $K$ layers can be created resulting in a more stable load. These blocks can then be reordered to achieve an optimal weight distribution. For this, they suggest complete enumeration, or a randomized approach if the number of options is too large. The results are compared to the heuristics of Amiouny et al.. They conclude that their approach is a good way to achieve a satisfying weight distribution, while the results are comparable with other methods in terms of space utilization and stability.

### 2.1.4   Multiple objectives

Other attempts have been made to to consider multiple objectives in cutting and packing problems. Wäscher (1990) develops a Linear Program (LP) based approach to cutting stock problems with multiple objectives. Imai et al. (2006) discuss the loading of a container ship based on two criteria: ship stability and the number of container rehandles required. The stability is evaluated in terms of the center of gravity and the second criterium may be generalized as the executability of the loading plan. They develop a Mixed Integer Program (MIP) with multiple objectives and show, using numerical experiments, that the solutions by the formulation are useful and applicable in practice.

### 2.1.5   Exact optimizations

Most research in the field of container loading has focused on the development of heuristics. This is because, due to the complexity of the problem, the applicability of exact methods is limited to small instances only. However, exact methods are interesting as they could be used for solving smaller subproblems in a heuristic approach, and for testing the quality of heuristic methods on smaller instances. Hifi et al. (2010) give an exact MIP formulation for the three-dimensional bin packing problem, and introduce a set of valid inequalities to strengthen the LP-relaxation of the

problem. These inequalities are based on the lower bound for the parallel-machine scheduling problem defined by Eastman et al. (1964). As a result, they are able to find tighter lower bounds for the problem, and they show how this significantly improves the performance of the *Branch and Bound* procedure they use to solve the MIP. The inequalities may be added to a wide range of similar problems to strengthen the formulations.

## 2.2 Set Partitioning Problem

The *Set Partitioning Problem* (SPP), also known as the equality-constrained Set Covering Problem (SCP), is a problem that has been widely studied in the literature, thanks to its many applications, for example in vehicle routing and flight crew scheduling. The problem is as follows: given subsets $S_j \subset S$ with associated costs $c_j$, $j \in J$, find a collection of these subsets such that it is a partition of $S$ at minimal cost. The SPP has the following Integer Linear Program (ILP) formulation:

$$
\begin{aligned}
(\text{SPP}) \quad \min \quad & \sum_{j \in J} c_j x_j \\
\text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j = 1 \qquad i \in S \\
& x_j \in \{0, 1\} \qquad j \in J
\end{aligned}
$$

where $a_{ij} = 1$ if element $i$ is in $S_j$ and 0 if $i \notin S_j$. $x_j$ is the decision variable that indicates whether subset $S_j, j \in J$, is in the collection of subsets that forms the partition of $S$. In the typical example of flight crew scheduling, the set $S$ represents the set of all flight legs that have to be operated during some fixed planning period. Each $S_j, j \in J$, represents a possible tour for a crew. As the number of possible tours is usually very large, only a small subset of the tours is generated, for which it is likely that they are a part of the optimal solution.

Balas and Padberg (1976) wrote a survey on the SPP, in which they list a large number of applications and summarize the most important algorithms that have been developed for the problem. They give a set of rules that can be used to reduce the size of the problem and describe six different approaches to solve the problem:

1. Implicit enumeration

2. Simplex-based cutting plane methods

3. Column generating algorithms

4. Hybrid primal cutting planes/implicit enumeration algorithms

5. Symmetric subgradient cutting planes

6. Set partitioning via node covering

We discuss the column generating algorithm, as this is the method that we use to solve the variant of the SPP in this thesis.

## 2.2.1   Column generation

In the ILP formulation, each set corresponds to a variable with a column in the constraint matrix. As opposed to solving the set partitioning with fixed columns, using a column generation approach allows for the dynamic addition of columns to the problem, based on a subproblem which is called the *pricing problem*. The column generation approach is discussed in general by Barnhart et al. (1998). They suggest that the method can be used to solve huge integer programs, which may arise when the Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960) is applied to the linear formulation of the problem. This may be done for different reasons: the reformulation of the MIP may have a tighter LP relaxation, it may allow for elimination of symmetry and it may provide a decomposition of the problem into a master and subproblems. These subproblems may have natural interpretations for which it becomes possible to include important additional constraints.

   The idea of column generation is that, in some LP problems, many columns can be left out of the (master) problem formulation as the associated variable will be 0 in the optimal solution. In terms of Linear Programming, the variable is *non-basic*. Therefore, the problem is solved with only a subset of the columns and the optimality of the solution is checked by solving a subproblem that 'generates' new columns for which the corresponding variable would be non-zero, or *basic*. If these columns are found, they are added to the problem and the problem is reoptimized, while if no such columns exist the solution is optimal. Generation of the column is done by solving the pricing problem, which has, in many applications, a natural interpretation. For example, in the *Cutting Stock Problem*, the pricing problem is to find a new cutting pattern which has negative reduced costs. This problem is actually a Knapsack problem (Vance et al., 1994).

**Branch and price**

Clearly, column generation is not only applicable to the SPP. It can also be used to solve a general Linear Program, or even an MIP. However, if used for solving an MIP, the column generation procedure interferes with the default branch and bound scheme (Lawler and Wood,

1966). Using Branch and Bound, fractional solutions are eliminated by adding constraints if a branch is created. Because of the additional constraints it may be the case that, after branching, a column prices out favorably and should be basic, while it is not present in the master problem. Therefore Barnhart et al. suggest a *branch and price* method in which columns are also generated at different branches in the search tree. This way it is possible to solve MIP problems to proved optimality.

Barnhart et al. discuss the SPP because it is the typical application of a column generation procedure. They make the important remark that if any 'subcolumn' of a column defines a feasible column at a lower cost, the SPP is equivalent to the SCP. In that case, the SCP formulation of the problem is preferred, because its LP relaxation is more stable and easier to solve and the step from a solution for the relaxation to a feasible solution for the MIP is trivial. As an example, they describe the *Vehicle Routing Problem*: the SPP formulation of the problem would require each customer to be visited *exactly* once, while the SCP formulation requires each customer is visited *at least* once. However, an optimal solution to SCP is also optimal to the SPP as a customer that is visited more than once can be removed from a route for a decrease in cost.

Column generation has been successfully applied as a solution method to a wide variety of other problems. Lübbecke and Desrosiers (2005) reviewed the contributions of different authors to the use of column generation for different applications. A relevant example is Vanderbeck (1999): he discussed the development of a branch and price method for various cutting stock and bin packing problems. He describes the column generation problem and the branching scheme and demonstrates the efficiency of the resulting algorithm.

## 2.3 Assignment problems

Kuhn (1955) wrote a paper that describes the Hungarian method for solving the *classical assignment problem* of persons to jobs. After that, a wide variety of related problems have been studied. Cattrysse and Van Wassenhove (1992) wrote a survey of algorithms for the *Generalized Assignment Problem* (GAP), which concerns an optimal assignment of jobs to agents, where agents can handle multiple jobs as long as they are not overloaded. The formulation of the

problem is:

$$(GAP) \quad min \quad \sum_i \sum_j c_{ij} x_{ij}$$

$$s.t. \quad \sum_j a_{ij} x_{ij} \leq b_i \qquad i \in I$$

$$\sum_i x_{ij} = 1 \qquad j \in J$$

$$x_{ij} \in \{0,1\} \qquad i \in I, j \in J$$

In this formulation, $c_{ij}$ is the cost of assigning job $j$ to agent $i$, $a_{ij}$ is the capacity requirement of job $j$ when assigned to agent $i$ and $b_i$ is the capacity of agent $i$. The decision variable $x_{ij}$ is 1 if job $j$ is assigned to agent $i$ and 0 otherwise.

Most algorithms for solving the GAP are based on Branch and Bound techniques with different methods for obtaining the bounds. Cattrysse and Van Wassenhove present a review of these methods and describe some variants of the problem, of which the most relevant for our research is the Multi Constraint Generalized Assignment Problem (MCGAP). This problem is formed by replacing the capacity constraints $\sum_j a_{ij} x_{ij} \leq b_i, i \in I$, by the constraints $\sum_{j \in J} a_{ijk} x_{ij} \leq b_{ik}, i \in I, \ k \in K$, where $K$ is the set of resource types and $a_{ijk}$ and $b_{ik}$ represent the required and available capacities for resource type $k$. Gavish and Pirkul (1991) develop different heuristic solution procedures and an efficient Branch and Bound procedure with reasonable computation times.

For the GAP, many more contributions to the literature have been made, each with their own focus. For example, Hallefjord et al. (1993) discuss the solution of large scale GAPs by aggregating the problem into a smaller problem, for which a solution is found by an approximation technique. A disaggregation approach is used to generate a feasible solution for the full problem. Savelsbergh (1997) develops a branch-and-price algorithm for the problem based on its formulation as an SPP (see Section 2.2.1). A more general contribution that aids the development of algorithms for the GAP is made by Gottlieb and Rao (1990) as they derive three classes of valid inequalities for the GAP. These inequalities can be used to strengthen the LP relaxation of the problem. In addition, they prove that a basic fractional solution to this relaxation can be eliminated by a facet defining inequality.

More than 50 years after the initial publication of Kuhn, Pentico (2007) wrote a survey listing the most useful formulations of the numerous variations to the (generalized) assignment problem. The intention was to make it easier to find relevant literature during the development of new variations. Among these variations are the multiple resource GAP, which is actually the MCGAP, and the classical assignment problem with side constraints, which is discussed in

Section 2.3.1.

## 2.3.1 Classical assignment problem with side constraints

The classical assignment problem with side constraints arises when adding the constraints (2.1) to the formulation.

$$\sum_{i \in I} \sum_{j \in J} a_{ijk} x_{ij} \leq b_k, \quad k \in K \tag{2.1}$$

$K$ is again the resource set, $a_{ijk}$ is the required capacity of resource type $k$ for job $j$ by agent $i$ and $b_k$ is the available capacity of resource type $k$. These constraints differ from the constraints in the MCGAP because the resource available $b_k$ is not per agent[1]. Mazzola and Neebe (1986) develop a Branch and Bound scheme for the problem, as well as an effective heuristic that finds a solution on average 0.8% from the optimum. Foulds and Wilson (1999) use a Branch and Bound algorithm for a special problem that concerns the assignment of tools to slots in a tool carousel in order to maximize the valuations of tools that are placed adjacently, with a special set of side constraints. Each tool can be placed in one slot in the carousel, which is comparable to the problem of assigning pallets to slots in a container; the adjacency valuations can be compared to valuations for products that should be close together in the container. Caron et al. (1999) discuss the assignment problem with the special class of seniority and priority constraints, which imply that agents/jobs cannot be unassigned if any agent/job of a lower seniority/priority is assigned. They show that using a greedy heuristic, as is often done in practice, the number of assignments may reduce more than 50% and that an optimal assignment can be obtained by solving the classical assignment problem with a proper scaling of the coefficients. Volgenant (2004) continues on this work and shows that the problem can be solved by successive re-optimization of a linear assignment problem of increasing size, which is of lower complexity than the approach with scaling of the coefficients.

---

[1] In the classical (one-to-one) assignment problem, a limited resource capacity per agent would simply restrict certain assignments. This may be modeled as a sufficiently large cost, such that the problem can be regarded as the unrestricted classical assignment problem

# Chapter 3

# US situation

This chapter contains a summary of the most important topics or concepts that are relevant for load building, especially in the US. It is a consolidation of knowledge gained from different documents, internal meetings and conversations with ORTEC's business consultants in the US. The chapter has been divided into different sections, where each section deals with a single topic or concept. As an introduction, we discuss how ORTEC's load building solution is used in the business process in Section 3.1. Then, we discuss the optimization of VFR for different types of shipments in Section 3.2 and in Sections 3.3 and 3.4 we discuss different types of *loading units* and the weight restrictions that come as a result of regulations for trucks on the road. In Sections 3.5 to 3.8, we discuss topics regarding the way a load is built: methods of loading, loading patterns, stackability and stability. Then, in Sections 3.9 to 3.11, we discuss some additional restrictions such as compatibility of products, priorities of different products and grouping and sequencing constraints. Finally, in Section 3.12 we will discuss three additional concepts that, if successfully implemented, on a higher level can help to reduce overall shipment costs: vendor managed inventory, transportation forecasting and transportation cooperation.

## 3.1 Load building in the business process

At most customers, ORTEC's load building solution is used at a high level to determine the number of trucks required for a load, because of the seamless integration with their *Enterprise Resource Planning* (ERP) system. Upon order intake, the software may be used for *sizing* of orders, which is estimating the number of trucks. The estimate may directly be used for *up-*

*selling*, which is explained in Section 3.2.3. Even if orders are sized as 'full' truckloads, in the planning they are mixed with other orders as this still may increase the VFR. However, mixing orders should be done with care: a customer does not expect two half-full trucks to be delivered if he orders one full truckload.

For the actual planning, the solution can be used in different manners. For internal shipments or for some customers, optimization of long-term *demand* can be done rather than optimization of fixed orders. Based on the results, orders are created that correspond to full truckloads. For fixed orders, a possible planning can be made for a longer time window to get an impression of the expected workload, although this may be subject to change if new orders arrive. For a shorter time window, the planner may decide to *release* a number of trucks, either from a separate optimization or from the long-term plan, which means they are sent to the warehouse for execution.

At the warehouse level the *Warehouse Management System* (WMS), or another downstream system, determines and executes the actual plan that is used to load the individual trucks. This may be taken over from ORTEC's solution, but in some cases the system has its own implementation, for instance to realize efficient warehouse execution. In any case the software systems should be aligned as much as possible, since in case a feasible solution by ORTEC's solution cannot be realized at the warehouse level, a pallet needs to be *cut* from the truck. Cutting of pallets is very costly as the products need to be delivered later, and the VFR of the truck decreases.

Even if the actual loading plan from ORTEC's solution is not directly used in the warehouse, it is important that it is feasible because otherwise the warehouse will not be able to load the products. However, constraints to ensure feasibility should not be overrestricting compared to the real situation, since that limits the possibility for optimization.

## 3.2   Types of shipments

Generally, we can distinguish three types of shipments: *inbound shipments*, *interplant shipments* and *customer shipments*. Inbound shipments are shipments from vendor to plant or vendor to distribution center (DC). Interplant shipments are from plant to DC or from DC to DC. Customer shipments are from DC to customer. Figure 3.1 graphically represents the different types of shipments. This section discusses the optimization of VFR for the three types of shipments. Although some customers also use ORTEC's software to optimize VFR for inbound or customer
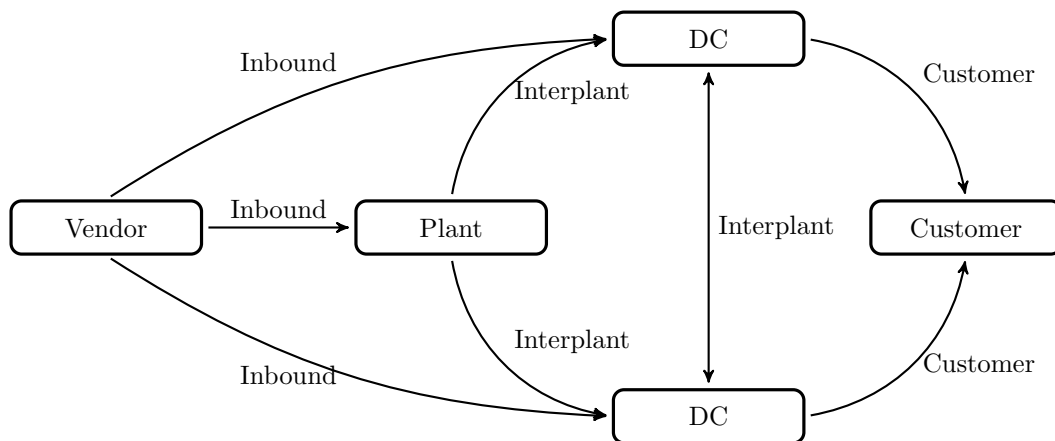
Figure 3.1: Illustration of different types of shipments.

shipments, the major use is for interplant shipments.

### 3.2.1   Inbound shipments

Inbound shipments consist of products or materials that are used for production at the plant or packing at the DC's. The inbound shipments arriving at a company are initiated by the purchase orders (PO's) the company sends to its vendors. These PO's can be optimized so that they fit nicely in a certain number of trucks. This way of using load building is indirect and rather advanced, as the shipment is done by the vendor and not by the company itself. However, it gives the company strategic advantages in negotiating shipping costs with the vendor if it can show that the PO's fit within a certain number of trucks.

### 3.2.2   Interplant shipments

Interplant shipments are usually large and there is some degree of flexibility which allows for the possibility to build loads that fully utilize trucks. Full truckloads are shipped from one DC (or plant) to another, so there are no stops in between. Each combination of an origin and a destination is called a *lane* and for a large vendor it is not unusual to have multiple dozens of trucks on a single lane each day. Depending on the lane, a single truck can cost thousands of dollars, so the benefits from a little increase in VFR are significant.

### 3.2.3 Customer shipments

Customer shipments are the most complex type of shipments as they may contain a large number of products in small quantities. Moreover, these shipments may have sequencing constraints (see Section 3.11) if orders are to be delivered at different locations. A customer shipment does not necessarily fill up a truck, or in the case of multiple trucks, fill up the last truck. In that case *up-selling* techniques, such as offering a discount, may be used to increase the size of the order for a better utilization, or even *down-selling* may be applied if that results in the need for a truck less. On the long term, the customer would require the same products, so this does not result in an increase or decrease in sales. However, the total number of trucks reduces as every order is shipped with a higher VFR. This concept is sometimes referred to as *order building* and is limited by practical considerations. Most important is that the customer must be able to temporarily accept higher (or lower) levels of inventory for certain products. In some industries, this may be very costly, for instance when products have to be held refrigerated. Furthermore, these types of products (foods) are usually perishable and may reach their expiry date even before they are sold because of the increased level of inventory.

In some cases, small sized orders may be consolidated in a shipment with other orders at a later time. Then load building can be used on the individual order, but the goal is not to minimize the number of trucks, but to pack the load such that the number of *floorspots* (number of pallets on the ground) is minimized.

## 3.3 Loading units

The terms *vehicles*, *trucks* and *containers* are sometimes used interchangeably. In most contexts this is valid, for example when talking about minimizing the number of vehicles, trucks or containers. However, it is good to make the distinction clear: a truck is a special type of vehicle onto which a container, the actual loading unit, is attached or may be loaded. The container may be (on) a *trailer*, which is an unmotorized vehicle that can be pulled by a truck. A *semi-trailer* is a trailer with only one (tandem) axle and, therefore, is not only pulled by the truck but also rests on the truck. Although most loading units are containers that are shipped over the road, there are also *intermodal* containers that are (partly) shipped by rail or ship. In the context of load building, the term vehicle is only used in the term *vehicle pool*, which is the set of loading units that is available for optimization.

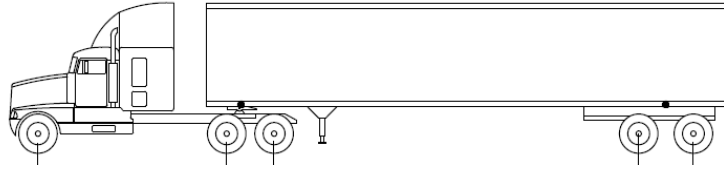The vehicle pool can consist of different loading units. Although most containers have stan-

Figure 3.2: A truck and semi-trailer combination with five axles. The black dots represent the imaginary axles considered.

dardized dimensions, the length of containers on the road may vary within the limitations of legislation, see Appendix B.1. For intermodal containers, there are different sizes which have different costs of shipping. Containers can also differ in types of freight. In the foods industry, for example, there are containers for dry food and containers that can be refrigerated or even frozen to ship perishable goods. In other industries there are different types. Special types of containers are typically more expensive but obviously required for some types of products. Intermodal shipments are generally cheaper but take longer, so they can be used depending on the time window in which products have to be delivered.

A loading unit or container is characterized by its loading space, which is a box with fixed length, width and height. For most containers, there is also a maximum weight that may be loaded. If the container is on a truck, also the positions of two axles are given and the maximum weight allowed on each of those axles is known. In practice, there are usually more than two axles, but it suffices to consider two (imaginary) axles in terms of requirements considering the weight distribution (see Section 3.4).

For a typical truck and semi-trailer combination, the imaginary axles are at the *kingpin* connection between the semi-trailer and the truck and at the center of the rear tandem axle, as shown in Figure 3.2. In most cases, both the position of the kingpin connection and the position of the rear tandem axle can be adjusted within a small range. The only scenario where it is not sufficient to consider two axles is with *double semi-trailers*, when a single truck has two trailers with each only one (tandem) axle. In that case, the trailers have a leverage effect on each others axle weights, but this configuration is rarely seen in the US.

In most scenario's, load building is done for a single container type, with as goal minimizing the number of containers, which is equivalent to maximizing the VFR of the 'full' containers, and minimizing the utilization of the last one. If the vehicle pool contains different types of trucks with different numbers available, we have two options. The vehicles may be ordered, so that we

first have to use all trucks of one type, then the second, etc. Then the goal still is to minimize the number of trucks. However, if we are able to select the vehicles that we want to use, then part of the optimization is to select the best mix of trucks to ship the load. Each truck should be associated with a certain cost, which is why this is referred to as *cost optimization*.

## 3.4 Maximum weight and weight distribution

For containers that are shipped on the road, there are a lot of restrictions that have to do with the weight and the weight distribution. Each truck is allowed to have a *gross weight*, the weight of the truck and the load combined, of at most 80,000 pounds. Depending on the length of the truck, this maximum gross weight can actually be lower as a result of *axle weight restrictions*. These restrictions are defined by legislation and imply how the weight must be distributed along the horizontal lengthwise dimension. Also in the other dimensions the weight distribution has to be considered. In the widthwise horizontal dimension it is important that the center of gravity (COG) of the load is as close as possible to the geometric center, in order to prevent rollovers. In the vertical dimension it is important that the COG is as low as possible, both for individual stacks to minimize the possibility that they collapse (see Section 3.8) and for the entire load in order to prevent truck rollovers.

### 3.4.1 Axle weight restrictions

The maximum weight and the weight distribution requirements in terms of axle weight restrictions are a result of the *Federal Bridge Law* (FBL), which is described in Appendix B.2. The FBL actually limits the weight on each group of axles of the vehicle, where more axles or a larger distance between the axles may result in a higher weight limit. Satisfying the FBL is, theoretically, equivalent to satisfying the maximum gross weight and the maximum weights on two (imaginary) axles, which is illustrated by the calculations in Appendix B.4.

As mentioned previously, some adjustments can be made in the positions of the kingpin connection and the rear tandem axle. However, this does not mean that both imaginary axles on the container can be moved: the kingpin has a fixed position on the *container* whereas the connection on the *truck* may be moved. The rear tandem axle position can actually be moved on the *container*. Moving of the rear tandem changes the weight distribution *of the load* over the rear tandem (of the container) and the kingpin, while moving the kingpin changes how the weight *on the kingpin* is distributed over the axles of the truck. This means that, depending on

the kingpin position, the allowed maximum weight on the kingpin according to the FBL may vary as well. There is one position for the kingpin for which the allowed weight is maximized, so usually the position is regarded fixed at this position with the corresponding maximum weight. In some cases it may be required to move the kingpin connection further to the back to make space for equipment attached to the front of the container, such as a refrigeration unit.

When considering a single container, satisfying the axle weight limits is equivalent to having the COG in a lengthwise interval, but this interval depends on the weight that is loaded in the container: a heavier load results in a smaller interval. The interval may be somewhat enlarged because of the flexibility in the rear tandem axle position, although in certain states the distance to the kingpin is limited by legislation (see Appendix B.1). In general, the constraint should be formulated as a maximum gross weight and two axle weight restrictions, rather than as a fixed COG interval based on the maximum gross weight. This is because a COG interval may be overrestricting in case not the maximum weight is loaded.

Axle weight restrictions are important, as failure to comply with the FBL may result in fines of several hundreds of dollars per truck. Even worse is when an overloaded truck is denied access to a certain route or state and therefore would be required to return to the warehouse to unload part of its load. In Section 3.8 we describe how axle weight restrictions are most problematic in combination with stability requirements when the load is moderately heavy, such that there are some stacks of pallets and some single pallets.

**Axle weight restrictions: the reality gap**

Although axle weight restrictions should be hard constraints for the *trucks* that go on the road, it is difficult to translate them into hard constraints for the 'axle weights' for the *containers*. That is because the weight of the actual truck that hauls the container may vary, not only because of physical properties but also due to some external factors. Even more important is that the actual way of loading heavily influences the axle weights, even if the arrangement is as planned. The COG's of individual pick pallets vary by the way they are built and the rotation in which they are placed in the container. As another example, if only a single pallet has (unplanned) overhang (see Section 3.6.1) or is simply placed with just a little slack, the entire load after that pallet shifts which heavily influences the axle weights, even if the shift is only a couple of inches. Even more influencing the axle weight may be the scenario where it is required to swap pallets if they are not available at the time of loading because of warehouse operations. All these factors together make that there is a difference between the theoretical axle weights and the real axle

weights if the truck is loaded. This difference we refer to as the 'reality gap'.

Because of the reality gap, the maximum weight and the axle weights that are used for planning are typically not the theoretical values (see Appendix B), but values based on warehouse experience on what weight in practice can be legally loaded on a truck. In other words, there is a 'safety margin' in the values. Additionally, as a rule of thumb there should be a slack of at least a few hundred pounds in the sum of the axle weight limits compared to the gross maximum weight, since otherwise the feasible COG range gets very small.

Obviously, even within the axle weight restrictions, the load should be planned such that the weight distribution over the axles is 'as good as possible', as this makes it more likely that the loaded truck will satisfy the axle weights in practice. However, especially if the safety margin in the maximum weight is small, there is no way that a loading plan that satisfies the axle weight restrictions guarantees that the loaded truck will satisfy the legal axle weight restrictions as well. On the other hand, a (small) violation of an axle weight limit in the plan does not automatically imply that the truck is overloaded on its axles too. Maybe the size of the reality gap could be estimated, dependent on the type of the load, but it is very hard to overcome the reality gap in general and this makes it unrealistic to force the axle weights as a hard constraint to the single pound.

Although we argue that because of the reality gap it is not realistic to enforce the axle weight constraints in a hard manner, there are some limits. In a rare but conceivable case where we have no or limited stacking possibilities and there is little diversity in pallet weights, even the extreme solution of loading all heavy pallets to the front may not be close to satisfying the weight restriction for the rear axle. As a result of the limited diversity, the COG is close to the center, while legislation forces the rear axle to be close to the kingpin. As the kingpin position on the container is fixed, this means that the rear axle must be moved forward and therefore be closer to the center, such that it is overloaded. This is illustrated in Figure 3.3. In this scenario, the only solution is to load a greater diversity in weights of products in the container, or, when this is not possible, not to load the maximum gross weight at all.

## 3.5 Methods of loading

Generally, two methods of loading can be used:

- *Bulk loading*: products are placed directly in a container without pallets. Items can be stacked one by one.
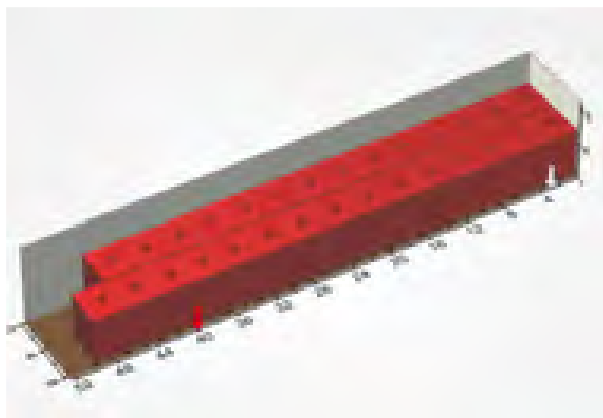
Figure 3.3: Example of an overloaded rear axle if there is no diversity in the pallet weights.

- *Palletized loading*: products are placed on wooden pallets and these pallets are loaded in
  the container. Pallets may be stock pallets with a fixed number of identical products or
  pick pallets with different products.

## 3.5.1   Bulk loading

Figure 3.4 gives an example of bulk loading products in a container. Bulk loading results in
better volume utilization but is more labour intensive and therefore happens in countries where
labour costs are low and shipping costs are high. Also, in such countries the availability of
forklifts at customers is unusual, such that bulk loading is the only option. In the US, bulk
loading may happen when the load is very diverse and palletizing is not beneficial, but this is
not what ORTEC's software is typically used for.

## 3.5.2   Palletized loading

Figure 3.5 gives an example of palletized loading. With palletized loading, all products are
placed on a pallet before they are loaded in a container. *Stock pallets* are pallets with only one
product (SKU) in a fixed quantity and have fixed dimensions based on the quantity. This is
how the products come from the plant. *Pick pallets* are pallets with combinations of multiple
products. They are usually built up of full layers of products with optionally a number of cases
on top. Alternatives to the so called *layered pallets* are *column pallets*, on which each product
is accessible without unstacking, and *sandwich pallets*, in which a wooden or cardboard pallet is
inserted between different layers, so that layers of at least four products can be stacked, rather
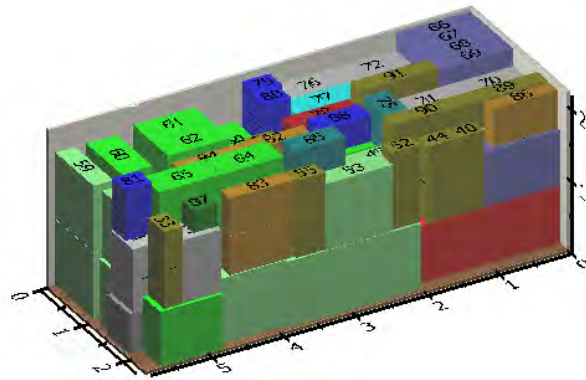
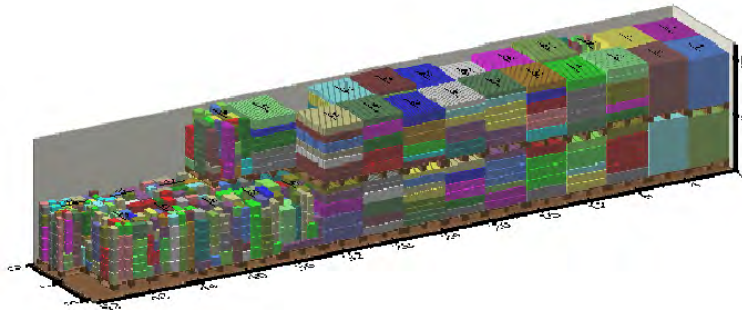Figure 3.4: Bulk loading of products in a container.



Figure 3.5: Palletized loading of pallets in a container.

than only full layers. The layered pallets are the most commonly used. In some businesses, typical orders consist of less-then-a-layer quantities per product, such that the layers consist of mixtures of products. These products are not necessarily equal in height, which may result in unstable stacking. In order to make a stable unit, pallets are usually *shrink-wrapped* as in Figure 3.6.

Palletized loading occurs with all types of shipments. In the case of customer shipments, there may be more pick pallets and therefore smaller pallets if they cannot be stacked too high. For interplant and inbound shipments, there will be more stock pallets. Although these stock pallets have the same length and width, the height may differ with different products. Frequently, pallets are stacked up to a height of two, but in case of very tall stock pallets this may not be possible. In order to achieve a good VFR it may be beneficial to split pallets (stock or pick pallets), to load layers of products on top of full stock pallets. However, this complicates the

Figure 3.6: Shrink-wrapping of a pallet.

executability in the warehouse and a customer may require that each part of a split pallet has its own wooden pallet, such that actually more weight needs to be shipped.

In some cases, pallets may be stacked *woodless*, which means that the wooden pallet itself is removed if a pallet is stacked on top of another one. This saves weight and allows for more space for on top loading. However, the wood of the pallet on the floor should not be removed. Removal of wood mainly happens in Europe and some other countries. Also in the US some customers do this.

If the number of wooden pallets varies, either because of pallet splitting or woodless stacking, the VFR KPI's should be defined with care. This is because a higher weight does not necessarily mean a better solution if it only consists of additional wood. In the end, the goal is to ship as much products as possible so the weight of the actual products (with or without wood) should be considered, regardless of whether the wood is removed or not.

Traditionally, palletizing is done as a single step prior to loading. This two step approach has a major disadvantage: in the palletizing step groups of products (see Section 3.11) may end up on the same pallet, which forces them to be on the same truck (if the groups cannot be split). This may be suboptimal in terms of VFR, so it would be better if the algorithm would be able to try different 'palletizations' for the best loading result.

## 3.6   Loading patterns

The loading pattern defines the two-dimensional arrangement of the floorspots on the floor of the truck. On each floorspot a pallet or stack of pallets can be placed. The loading patterns differ for different types of containers and regions. In the US, typically all pallets are placed in
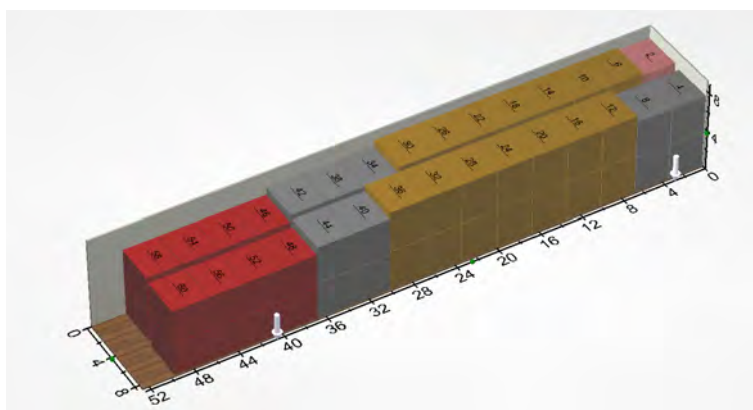
Figure 3.7: Example of the default loading pattern.

the same orientation. A default US stock pallet has the size of 40 by 48 inches, where a truck typically has a width of 8.5 feet (102 inches), such that two pallets fit lengthwise in the width of the truck. Depending on the length of the truck, there may be some space left at the end, but this cannot be better utilized by a different loading pattern, since rotation of two pallets in a row only results in $2 \times (48 - 40) = 16$ inches slack to the side.[1] The loading pattern in which pallets are placed in a container lengthwise in rows of 2 is illustrated in Figure 3.7.

There are cases in which the width of the container is less than 8 feet, for instance with refrigerated containers with a thick isolated wall (see Section 3.3) or intermodal containers (see Section 3.3). In these containers, pallets should be placed widthwise at one or even both sides of the container. If only one of the pallets needs to be rotated, this is done using a *pinwheeling* pattern. Depending on the customer's loading capabilities and stability requirements, this may either be done using, what we will call, *full* or *partial* pinwheeling, as illustrated in Figures 3.8 and 3.9. Full pinwheeling results in a higher stability (see Section 3.8) while using partial pinwheeling more pallets can be loaded.

Although the majority of shipments in the US happens with the default US stock pallets, there are some special types of shipments for which smaller pallets are used. Very often this combines with other special constraints.

---

[1]This situation is different from Europe, where the pallet size of 32 by 48 inches allows for a row of three rotated pallets.
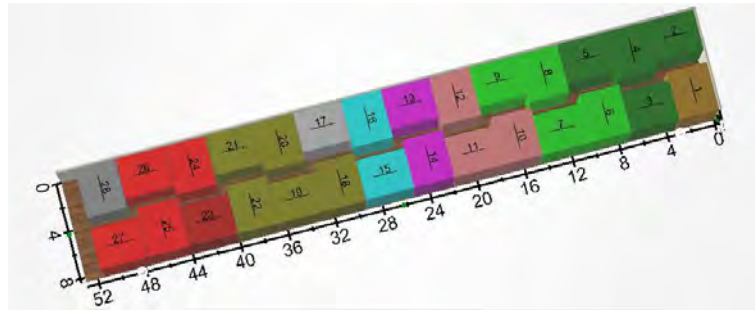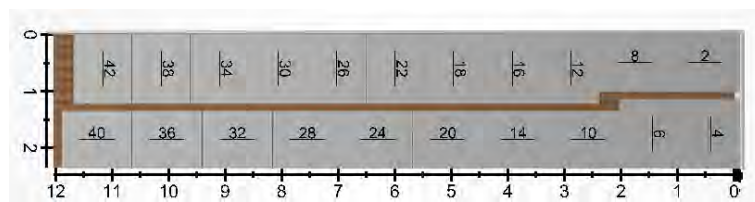
Figure 3.8: Example of full pinwheeling.



Figure 3.9: Example of partial pinwheeling.

### 3.6.1   Overhang

Overhang occurs when the length or width of a layer of products on a pallet exceeds the length or width of the pallet. Overhang may cause problems when used with the loading pattern of two pallets next to each other. With a 102 inch truck, there is a slack of 6 inches, so there is not much of a problem if pairs of pallets can be made such that their combined overhang does not exceed 6 inches, although such overhang causes limitations in the physical arrangement of the pallets. If more than half of the pallets have more than 3 inches overhang, or when there are pallets that individually have overhang more than 6 inches, enforcing the fixed orientations would result in only one pallet being placed in a single row. If the load of the truck is limited by the weight rather than the volume or space on the floor (the number of floorspots), this is not a problem as all pallets can still be placed without all floorspots to be occupied. However, it is very likely that this is not the case, because pallets with overhang are created to maximize the volume utility of individual layers. In that case, using pinwheeling is the only option to place the pallets. As an additional complexity, the overhang cannot always be calculated upfront, as it depends on the way the warehouse employee actually builds the pallet in the end.
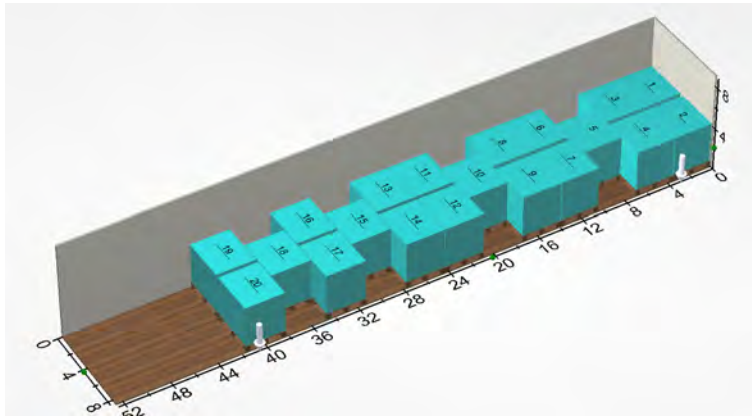
Figure 3.10: Example of a heavy loading pattern for 20 pallets.

### 3.6.2 Heavy load

As mentioned before, in the case that the load is heavy, the weight restricts the number of pallets that are placed in a truck, rather than the volume or the number of floorspots. Then it may not be necessary to occupy all floorspots to place the pallets. Moreover, if the load is extremely heavy it may not even be *possible* to occupy all floorspots, since the number of pallets in the container may very well be lower than the number of floorspots. If the pallets are placed two by two according to the loading pattern, this results in the load not reaching to the back end of the truck. In that case, most of the (heavy) load rests on the front axle, which likely results in an axle weight violation (see Section 3.4.1). It is not possible to simply place a number of pallets in the back of the truck, as these may shift while moving. Therefore, a special loading pattern is required in which in some rows pallets are deliberately placed individually. An example for 20 pallets is given in Figure 3.10.

### 3.6.3 Warehouse executability

In order for execution of the loading plan to be efficient in the warehouse, it should be sufficiently simple. In the default loading pattern, pallets are always placed in the same grid, and correct execution of the plan is equivalent to placing the right pallets (or stacks) on the right (fixed) floorspot. In the warehouse this is typically achieved by sequencing the pallets and placing them one by one in the truck. In the case of a heavy load, one has to use the special loading pattern in order to place the pallets correctly, even if the pallets are sequenced. In order to achieve this at the warehouse level, there are two options:

- Always use the same loading patterns for fixed numbers of pallets

- Communicate the specific pattern for each load to the warehouse

Which method works best may vary for different warehouses. In some warehouses, employees may be trained or refer to fixed charts to load the pallets according to the fixed loading pattern, depending on the number of pallets. In other warehouses, they may use information in the form of codes[2] that tell them where to place each pallet, in which case there is no reason to stick to a fixed pattern. In terms of stability (see Section 3.8), however, some additional thought has to be put in dynamically creating a loading pattern.

In some cases with heavy loads, employees may not even use the exact loading plan, but simply load the truck according to their own experience and the fixed loading pattern. This is possible because with a heavy load with few pallets, there are no instable stacks of pallets. Each pallet is approximately the same height, or at least sufficiently stiff to be supported by a smaller pallet. Therefore, actually *any* configuration of the pallets within the truck is feasible, as long as the pallets are placed adjacently. In order to satisfy other constraints, such as axle weights, when slack in the axle weights is sufficient, employees only need to make sure not to place all heavy pallets together at one side of the container. The insight that feasible loading in this scenario may be trivial is also beneficial when designing an algorithm to do so.

There are many more scenario's that prevent the possibility to exactly execute the loading plan. For instance, problems with picking may result in pallets arriving at the dock too late (in the wrong sequence) such that another pallet is loaded first as the loading process cannot wait. Also, as a result of unexpected overhang, it may not be possible to place a pallet at its desired location. Typically warehouse employees have some flexibility to actually load the pallets and are trained to do so while taking into account considerations such as axle weights.

## 3.7   Stackability

Stackability mostly has to do with *static stability*. This is the basic form of stability and means building a load that it is stable and does not collapse when the container is (un)loaded. Stackability data is available in the form of a *stackability matrix*, which tells what combinations of products, or pallets, can be stacked. Typically, this stackability data is not maintained on the product level, but for *stackability groups*, such as 'light', 'heavy' or 'dangerous'. Therefore, it

---

[2]Examples of codes are L01, R02 and M03 for left, right and middle position in rows 1, 2 and 3.

mainly induces 'rules of thumb' such as that heavy products should not be stacked on light products and that dangerous products should not be stacked at all. That is one of the reasons why the stackability matrix only represents the stackability *qualitatively*: it tells which groups of products can be stacked, but it does not tell how many times. The only exception is on the diagonal of the matrix, where the number represents how often the products in the group can be stacked on top of itself, although for most cases this is unlimited. Depending on the product groups, there may be different reasons why stacking would not be allowed:

- Physical considerations, for instance products may be fragile or odd shaped such that they cannot be stacked.

- Weight considerations, such that heavy products are not stacked on light products.

- Qualitative considerations, for instance liquids may leak (when they fall from the stack) and damage (valuable) goods below it. A customer may decide not to stack these products because of the risk.

If the stackability matrix follows from a combination of these reasons, it is unclear whether transitivity holds: if A may be stacked on B and B may be stacked on C, then may A stack on C? The answer to the question comes from the stackability matrix itself, and will be yes in most cases, but whether or not A may stack (directly?) on C, one could question whether a stack of A on top of B on top of C is feasible? Currently, the algorithm will allow the stack of A-B-C to be created, which means that only *direct stackability* is considered.

Within the pallets, the stackability has to be considered in a way similar to bulk loading. As a result of the stackability matrix, a typical pallet will be built up with heavy to light products from bottom to top. Next to the stackability matrix, also a *support coefficient* is taken into account which requires that each product is in contact with the surface of the products below it (which may differ in height a few millimeters) for a minimum percentage.

With palletized loading, after the pallets are created, the stackability has to be translated in terms of pallets which means that multiple stackability groups on the same pallet should be consolidated. There may be a specific stack group for mixed pallets, or the stackability of two pallets is checked by every combination of stackability groups in the two pallets. In terms of pallets, considering direct stackability is sufficient for most customers since most stacks consist of two pallets. If a third pallet is stacked, it is usually very small and light and, although the bottom pallet supports two pallets, their combined weight is not significantly higher than the

weight of a single taller pallet. When stacking pallets, also the support coefficients should be considered.

For some customers there is a custom implementation that also considers maximum weight. This means that, next to the stackability matrix, a maximum weight is calculated that each pallet can bear. In the case of stock pallets, a lower bound for this value is determined by the number of times it can stack on itself. For pick pallets, the value is determined by the weakest layer, which is the layer that can bear the least weight when subtracting the weight of the layers on top of it from its *maximum weight per layer*. This maximum weight per layer is estimated by calculating the weight the bottom layer in the bottom stock pallet bears if the pallet is stacked the maximum number of times.

## 3.8   Dynamic stability

*Dynamic stability* means that all pallets should be placed in the container such that they cannot move when the truck moves, due to safety reasons and to prevent damage to the products. Pallets may move as a result of forces that are exerted on them when the truck accelerates, brakes or turns left or right. For this reason, all pallets should be supported from all sides. Arguably the largest force is the force that results from a sudden brake of the truck, implying that it is more important for pallets to be (fully) supported from the front than from the back. Typically it is attempted to create a load with a *decreasing load height*. Depending on the speed and sharpness of a turn, also sideways forces may be significant so it is important that pallets or stacks of pallets of the same height are placed next to each other.

In the default US loading pattern (see Section 3.6), a large part of the support is achieved by placing the pallets adjacently from front to back. As the load rarely exactly reaches the back door, it should always be secured at the back, typically using a leash. In a heavy loading pattern there are some rows of two pallets that are only supported by a single pallet from the front. Each of the two pallets has either only the left or the right half supported (see Figure 3.10). Depending on where the center of gravity of the pallet is, it may rotate such that the unsupported half moves forward if the truck brakes. This may be prevented by fixing the two pallets along the width of the truck with some material, or separating the rows with a board or leash. Additionally, a pallet that is centered may shift left or right as it has no side support. This risk increases if there is a longer sequence of rows with a single centered pallet. Ideally each centered pallet is 'squeezed' between two rows of two pallets. If this is not possible, then the

length of each sequence should be limited as much as possible. The last row preferably consists of two pallets.

It is even more important that pallets stacked on other pallets are sufficiently supported. Stacks of pallets are usually not shrink-wrapped together, so a pallet may fall off the pallet below if it shifts while the pallet below does not. With a heavy load, this is irrelevant as we have no stacks. With a light load, the volume is maximized so we have all stacks as close to the maximum height as possible, so all stacks can support each other. Most problematic is the scenario in between: the load is heavy, but not extremely heavy, so we have some stacks and some unstacked pallets. According to the decreasing load height and the desired side support, we preferably load all stacks first and then the single pallets, but this is likely to result in an axle weight violation. As an acceptable alternative, we may place the tallest stacks in the middle if we can sufficiently support them from the front and back. However, if there is little difference in the pallet heights, this may only be feasible if it is possible to additionally secure the stacks in the middle. For instance, a supporting board may be placed between two rows, but preferably this is avoided as it results in additional handling costs. Depending on the products, some customers may prefer to take the risk of transportation damage rather than doing the costly additional securing. Figure 3.11 illustrates the different possibilities, where the first arrangement is most preferred. The last arrangement is the most unstable one and should be avoided as much as possible.

In an attempt to create a 'smooth' load with only small height differences, sometimes additional pallets are placed under a stack so it becomes a little taller, but this is difficult with weight limitations. Also, allowing stock pallets to split increases the possibility to create stacks of different heights which enhances the flexibility in creating a smooth load. However, this requires additional handling as the stock pallets should be unwrapped and the split pallets wrapped again, since they should be stiff to provide support. In any case, the final stability of the load is hard to predict and depends on many (unknown) factors, such as the stiffness of the individual pick pallets.

To minimize the probability that a stack collapses, the center of gravity should be as low as possible. This way, the heavier forces are exerted lower, where the support is better as there are more pallets on that level. For more on weight distribution, we refer to Section 3.4.
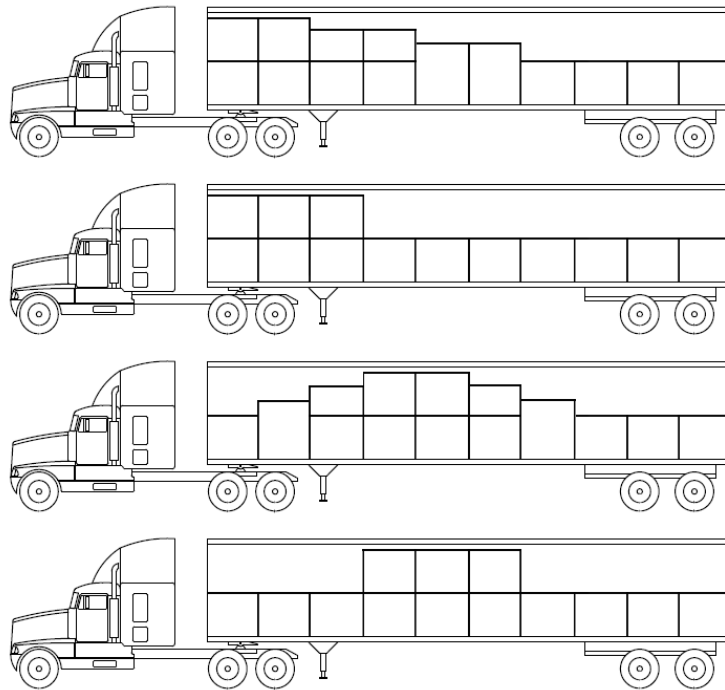
Figure 3.11: Possible arrangements in which stacks can be loaded in a truck, ordered by their preference in terms of stability.

## 3.9 Compatibility

In some cases products may be incompatible, which means that they cannot go on the same pallet or truck, depending on the degree of incompatibility. Examples are products that contaminate other products or dangerous goods. In some cases products may go on the same truck, but must have at least a pallet spot distance in between. The compatibility may depend on the length of the travel, for instance if products are only allowed to be packed together for a certain amount of time. Since the length of travel is fixed for the load it may be regarded constant.

Another type of compatibility is the compatibility between products and vehicle types, typically for perishable foods that must go on refrigerated or frozen containers. In some cases, so called *dry food* may also go on refrigerated or even frozen containers if the cooling does not destroy the product.

## 3.10 Priorities

Products may have priorities. A product of a higher priority should be delivered earlier than a product with lower priority. Usually a priority corresponds to a latest shipping date. This means that the priority of a truck is determined by the highest priority of the products in it. It is desired to have fewer trucks with higher priority, for different reasons:

- Higher priority shipping may be more expensive as it has to be done sooner.

- Higher diversity in priorities results in greater flexibility to divide the workload over time.

- If a truck has a lower priority, it can be dispatched later and orders arriving late may still be added to the truck for a better VFR.

The requirement to deal with priorities is usually translated in the following constraint, which has two equivalent formulations:

1. A product may not be on a truck if there is a higher priority product on a lower priority truck.

2. For each priority, there may be only one truck which has products of both that priority and lower priorities.

This results in a load where the first trucks only contain priority 1 (the highest priority) products, then there is a truck with priority 1 filled up with priority 2, then a number of priority 2 trucks,

etc. This is very intuitive, as it prevents lower priority products being preferred over higher priority products, but it may be suboptimal, which is why in some cases *backfilling* is done.

### 3.10.1   Backfilling

Backfilling is the placement of lower priority products on a higher priority trucks, if it is not possible to place higher priority products on the same truck. The most likely scenario is *on top loading* of small pallets or layers of products where no products of higher priority would fit. Backfilling increases the VFR of the trucks while it does not deteriorate the solution with respect to priorities.

### 3.10.2   Mixing of priorities

The fact that backfilling can be used to improve the solution after it is 'optimized' actually indicates that it is not the real problem that is optimized. After all, after the backfilling is done, we may no longer satisfy the 'requirement' that a lower priority product cannot be loaded on a higher priority truck, so it is not a real requirement. As an alternative, we may formulate the requirement as to minimize the number of trucks, and especially to minimize the number of priority 1 trucks, then the number of priority 2 trucks etc. But does it actually matter whether we have one truck with 100% priority 1 products and one with 50/50% priority 1 and 2 or two trucks with a distribution 75/25%? If we are allowed to mix we may increase the VFR, especially if the types of the loads for priority 1 and 2 differ in whether they mostly consume the weight or the volume of a truck.

Figure 3.12 gives a small example of how mixing of priorities can decrease the number of trucks, without changing the priorities of the trucks, even when backfilling is not possible. For a more detailed explanation we refer to Appendix C. Also, the appendix gives an example that goes a step further and illustrates how it is possible to save a truck in total if we accept to increase the priority of another truck, that is, we have the option to 'trade' two priority 2 trucks for one priority 1 truck. In a less ideal scenario we may not save a truck, but reduce the utilization of the last truck.

Mixing of priorities may be subject to additional constraints. For example, the warehouse should be able to handle mixed priorities. If products come from a plant, lower priority products may not yet have been produced and therefore may not be available. In the current scenario, this is not a problem since usually only a number of the highest priority trucks get *released*, which
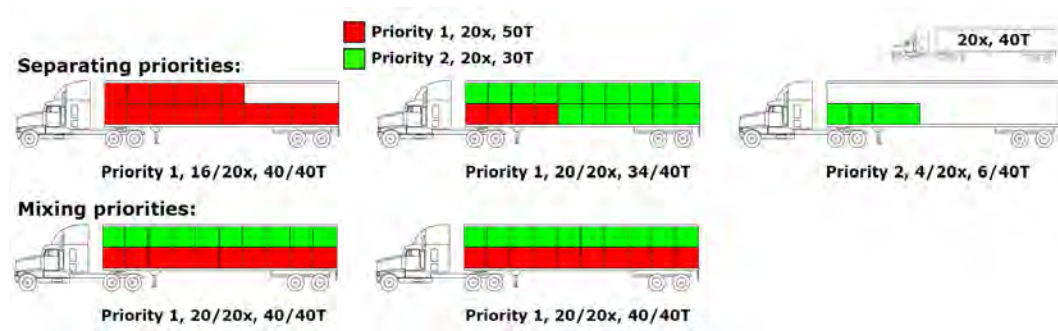
Figure 3.12: Example of how allowing priorities to mix can reduce the number of trucks without increasing the number of trucks per priority.

correspond to a limited execution time window (see Section 3.1). The products on the rest of the trucks will be consolidated with the new orders for the next run. However, knowing that this is how the plan is actually used allows us to think even one step further while taking, for example, the availability constraint into account. Traditionally, the requirement is to minimize the number of trucks per priority, but an alternative could be to minimize the total number of trucks while delivering the products in time. In this case, there is a limited capacity or there are costs for sending out a number of trucks each day.

Let each product have an earliest availability date (or be readily available) and a latest shipping date, and let each day have a maximum number of trucks that can be sent, or a cost function depending on the number of trucks for that day. Then the problem is to find a mix of trucks over the dates that minimizes the total costs or number of trucks while each product is placed on a truck such that it is shipped after its availability date and before its latest shipping date. If we expect orders to be added for the next day, this should be taken into account in the number of trucks available for that day.

Being able to mix priorities heavily complicates the problem from an algorithmic point of view, since containers can no longer be loaded one priority at the time. However, the business value may be significant. It allows customers more flexibility in mixing heavy and light products, leading to a decrease in the number of trucks since each truck is no longer full on either weight or volume, but as full as possible in both dimensions.

## 3.11    Grouping and sequencing

Depending on the type of shipment, a load may be comprised of *relations*, *parties* and *records*, which are groups of products. Relations and parties can represent anything, but in the typical example a relation corresponds to a physical stop and a party to an individual order. Records are always individual item lines. The data is on the item level but represents a tree structure, meaning that each record corresponds to a single party, and each party corresponds to a single relation such that records corresponding to the same party must have the same relation.

If we refer to relations, parties and records as *levels* of grouping, groups from each level may be required not to be split into different trucks. Such a requirement may be hard, in which case the group is never split, or soft, in which case some groups may be split but are preferably kept together. Furthermore the groups may be required to be consolidated within the truck, optionally with some distance tolerance.

Parties and relations may be subject to *sequencing* constraints, requiring them to be in a fixed order across and within the trucks. Within each truck the order of the parties may be required to be reversed, if the sequence is the preferred *unloading* sequence rather than the *loading* sequence.

In the current implementation, sequencing constraints on relations and parties are incompatible with priorities since the priorities are on the item level and are implemented as a sequencing constraint too, even though priorities may be mixed within a single truck. Another shortcoming of the current implementation is that there are only two levels of grouping that have freedom in what they represent (relations and parties) and these are often insufficient to model the business scenario. For instance, when we do not only have stops and orders, but each stop consists of multiple *doors*, a third level is required. Also, the requirement to ship certain orders together could be implemented as an additional level 'order groups' not to be split, just as the requirement to sort by the location on the warehouse could be an additional level.

For the greatest flexibility to model any business scenario, ideally the grouping levels would be dynamically configurable. In that case each level could have different options, such as:

- Do not split across different loading units, optionally as soft constraint

- Keep together within loading unit, optionally with distance tolerance

- Sequence: globally, within loading unit or within parent group, optionally reversed

This way of configuring grouping levels with sequencing also generalizes the concept of sequenced priorities.

## 3.12   Additional concepts

This section discusses three additional concepts that are not directly related to loading individual trucks, but are related to load building and may help to decrease the overall transportation costs for a company if they are successfully implemented.

### 3.12.1   Vendor managed inventory

In some cases and for some products, the vendor manages inventory for its customers. This way, the vendor is responsible for the stock level at its customer, and initiates replenishment based on its own forecasts. This *vendor managed inventory* (VMI) allows for better utilization of trucks as it allows to fill up trucks with (non-VMI) products. However, this VMI has a number of disadvantages:

- The vendor incurs costs for the resources needed to manage the inventory at the customer.

- The customer may not end up with the stock he thinks he needs based on his own experience and marketing strategy.

- To keep inventory costs low, customers may want low quantities delivered more often, while the vendor preferably supplies larger quantities less often.

The alternative in which the customer manages his own inventory is suboptimal in terms of vehicle utilization. Therefore, there is the tendency for a shift towards a solution in which both parties cooperate. That way, the customer gets more freedom to manage his own inventory, but agrees to take into account what is important to the vendor.

### 3.12.2   Transportation forecasting

Load building is sometimes used in the context of transportation forecasting. Transportation is expensive and usually it is cheaper to lock in capacity earlier. Therefore, it is important to be able to create a good forecast for the load that has to be shipped. In order to get a good estimation of the number of trucks that is required, the load building algorithm should be robust such that the dependence on the actual composition of the (forecasted) load is not too strong. Locking in too much capacity is very costly as expensive trucks would remain unused. On the other side, when insufficient capacity is locked in additional capacity has to be allocated at higher cost.

### 3.12.3   Transportation cooperation

A concept that is new in the field of load building is *transportation cooperation.* Transportation cooperation is best illustrated by an example. Suppose that two vendors are located near each other, or are even in the same warehouse. This is not uncommon when warehouse activities are outsourced to a *third party logistics provider* (3PL). If both vendors need to ship half a truck, they could use a shipping provider that allows to ship *Less than Truck Load* (LTL) and be charged by weight, but this is very expensive and for half a truck the vendor would normally still be better off paying for *Full Truck Load* (FTL) shipping. However, if both vendors need to ship to the same customer, or at least to customers nearby, they could combine their loads and together share one FTL shipment. Intuitively, it would be fair to share the costs of the truck equally, as they each use half of the truck. But how should half a truck be interpreted: with respect to volume or weight? And what if the orders are not fixed, but each vendor makes a certain profit for each pallet on the truck? From a theoretical perspective, the problem of determining each vendor's fair share in the profits or costs is a problem of game theory. Although the scenario sounds somewhat idealistic, it is actually very realistic, for instance when two food providers share a warehouse and serve the same supermarket.

# Chapter 4

# The PGA algorithm

This chapter describes how the PGA algorithm could be improved such that it is more applicable in practice and gives an outline of the improved PGA algorithm that is created as part of this thesis. A complimentary document contains a detailed technical explanation of the algorithm (Kool, 2014).

The chapter is structured as follows. First, Section 4.1 discusses the differences between the PGA algorithm and the construction heuristic and explains why it is not desirable to consider all requirements for the PGA algorithm if it is used in parallel with the construction heuristic. Section 4.2 describes for the requirements following from Chapter 3 what is expected to be the relevance for the PGA algorithm. This section also defines which of the relevant requirements we take into consideration with the improved PGA algorithm, since not all requirements can be taken into account because of lack of data within the research environment and the limited time window. Finally, Section 4.3 contains the outline of the improved PGA algorithm we develop.

From a practical standpoint the improved PGA algorithm developed in this thesis is not directly ready to be used in practice. Some requirements need additional development, but the improvements that are already made to the algorithm are significant. Furthermore, we point the direction in which further research should focus. However, even with additional developments, the PGA algorithm will not be applicable in all scenarios, which means that in practice it should always be used in parallel with the construction heuristic.

## 4.1   PGA algorithm versus construction heuristic

The benefits of the PGA algorithm mainly come from the fact that the loading problem is simplified by a number of assumptions. For the current implementation the assumptions are the following:

- Pallets are created in an initial phase, prior to loading.

- The orientation in which pallets are placed in the container is predefined.

- All pallets have the same dimensions, except for small differences in overhang.

- Pallets can be stacked, with a maximum of two.

- Stacks can be placed in two columns in the containers.

- The number of rows of pallets can be estimated from the pallet and container dimensions.

- Weight limitation is on two axles, which is equivalent to a total weight restriction and a lengthwise interval for the center of gravity (COG), depending on the weight loaded.

For a description of the current implementation, we refer to Appendix A. From the assumptions follows the use of the grid pattern, such that the loading problem is discretized and can be considered as an assignment problem of pallets to cells. Depending on whether the pallets are considered individually or first combined to stacks or blocks, the assignment problem is in respectively three, two or one dimension(s). In any of the cases, the discretization enables the use of special models, such as Mixed Integer Programs (MIP's) to solve different parts of the problem. The benefit of this should be either in terms of computation time or a better quality solution, compared to the construction heuristic.

In the construction heuristic, many specific constraints can be implemented relatively easily by limiting the options for placements during the construction of the solution. However, when using the PGA algorithm, it may be far from trivial or not even possible to implement some constraints without destroying the special properties required for some of the models. In most cases, each constraint increases the complexity for the models used in the PGA algorithm while the construction heuristic may actually benefit from the limitations in number of options. As a result, the potential of the PGA algorithm over the construction heuristic decreases when the number of constraints increases. If a set of constraint results in most of the potential of the PGA algorithm being lost, there is little reason to use the PGA algorithm for such cases. It may be

decided not to use the PGA algorithm for those constraints at all, and therefore, they do not need to be implemented.

The challenge is to decide which constraints and requirements to implement and which ones to leave out. Each requirement that is left out induces the assumption that it is 'irrelevant' to the cases that can be solved using the PGA algorithm. The relevance of the requirements is determined by the frequency that they occur, and the extend to which cases they occur with benefit from using the PGA algorithm.

It is known that the construction heuristic performs well in terms of VFR. Therefore, the potential of PGA in terms of a higher VFR is small, although small improvements can be interesting if achieved on a larger scale. More likely, there is benefit in combining a high VFR with a better satisfaction of global requirements. In this sense, satisfying the axle weights restrictions in combination with stability requirements is the most important element. This also means that the relevance of the PGA algorithm is low for requirements that typically occur with cases where it is unlikely that an axle weight violation occurs or that the PGA algorithm is able to find a better solution with respect to stability.

In terms of computation time it is hard to make an estimate about which cases would benefit from using the PGA algorithm. The construction heuristic is very fast, but the quality of the solution depends on the number of iterations and, therefore, on the computation time. When using the PGA algorithm, there are no iterations, but the computation time depends on the size of the instance. This may be limited by not solving the subproblems to optimality.

## 4.2 Requirements for the PGA algorithm

This section discusses the tradeoff between the complexity and benefits of different requirements when considered in the context of the PGA algorithm. Based on this, we describe which requirements are potentially more relevant to the PGA algorithm. Additionally, we define the requirements that we take into account in the development of the improved PGA algorithm. The structure that is used in this section corresponds to the structure in Chapter 3. The requirements are based on the situation in the US so they do not generalize to other regions. Again, the improved PGA algorithm does not consider all relevant requirements because of lack of data and time limitations.

**Load building in the business process**   We only consider the optimization of orders. The optimization of demand can be regarded as the optimization of one large order.

**Types of shipment**   There is no requirement on the type of shipment, although this is induced by the other requirements. The PGA algorithm will be most applicable to interplant shipments, as these shipments usually consist of full and heavy truckloads and are therefore concerned with axle weight restrictions. Customer shipments are more likely to have custom requirements that are not compatible with the PGA algorithm. The solution may be used to optimize inbound shipments, but in this case the customer is usually less concerned with axle weight restrictions and stability requirements, as the customer is not the one to ship the load.

**Loading units**   The algorithm should be able to deal with different types of loading units, although this may be less relevant as typically we would only have a single type for interplant shipments. Therefore, in the algorithm that we develop we assume that there is only one type. This could be quite easily adapted to different types of loading units if they are sequenced. In the sense of cost optimization, further research has to be done on how to dynamically select the different types of loading units, although the algorithm we will develop may be used as a subprocedure in an approach that searches through the limited space of possible combinations of loading units.

**Maximum weight and weight distribution**   The PGA algorithm should take the maximum weight into consideration as a hard constraint. However, to a certain limit, we will consider the axle weights as a soft constraint. The weight should be distributed as good as possible over the axles according to the axle weights, but they may be violated if not otherwise possible. The fact that the axle weight limits are an estimation including a safety margin may be used to justify that the axle weight of the load itself may also be approximated, rather than calculated exactly. As a secondary requirement, the center of gravity should be as close as possible to the center in the width and as low as possible in the height.

**Methods of loading**   Naturally, the method of loading that is used with the PGA algorithm is palletized loading. That means that it is assumed that all products are on pallets, which are either stock pallets or pick pallets created in a step prior to the loading in the containers. The PGA algorithm only considers the loading phase and there is no interaction between the two phases. Therefore, pallets are given as input and they cannot be split.

**Loading patterns** The PGA algorithm should be able to deal with both the default loading pattern and the pinwheeling pattern. Although most shipments are with the default loading pattern, the number of refrigerated and intermodal shipments that require pinwheeling is significant. It should be possible to implement this pattern with the PGA algorithm as the pinwheeling patterns can be considered as grids too, only with half of the pallets rotated based on the position. However, some issues have to be resolved, such as that the number of stacks depends on the variant of the pinwheeling pattern used. This will be out of the scope of this thesis. Therefore, the algorithm that we develop assumes that all pallets are loaded according to the default loading pattern: lengthwise in the width of the container. On each row there should be one or two pallets or stacks, such that the algorithm dynamically creates a heavy loading pattern if required. Using the default loading pattern implies the assumption that all pallets are, except for overhang, the same size in the length and widthwise dimension.

**Stackability** We concern stackability only for pallets and in terms of direct stackability between two pallets. Maximum weight on top is most likely a requirement that is relatively easy to implement in the PGA algorithm, but we will not consider this as the required data is not available. The pallets may be stacked multiple times within the height of the container, so the algorithm should be able to generate a solution with stacks with any number of pallets. It is not required to assume that all pallets are approximately the same height. However, the algorithm may take advantage of the assumption that in practice most stacks will consist of one or two pallets, and some stacks of three pallets, while stacks with four or more pallets are very rare. The algorithm should stack pallets woodless if this is allowed, which may or may not be the case for different pallets.

**Dynamic stability** The PGA algorithm should incorporate the stability of the solution, by maximizing the support of all stacks in all directions. In order to do this, a measure should be defined for the quality of the solution in terms of support. Additionally, in a heavy loading pattern, the length of sequences of centered pallets should be minimized. We will incorporate both of these requirements in the development of the improved PGA algorithm.

**Compatibility** The PGA algorithm should take into account compatibility between different products and between products and vehicles, but we will not consider these requirements because the data is not available. This means that we assume that all products are mutually compatible and compatible with the single vehicle type.

**Priorities**   In terms of priorities, we take the restrictions into account as sequencing constraints between containers. Within a single container, the priorities may be mixed. Although there may be business opportunities in mixing the priorities in different containers, it should first be investigated whether this is really feasible from an operational point of view. Also, to do mixing of priorities additional data is required. For the algorithm we develop we do not consider the option of mixing priorities.

**Grouping and sequencing**   Preferably the PGA algorithm would be able to deal with multiple levels of grouping. By default, these groups are preferred to be shipped in the same container, but this is not a hard requirement, although it can be set as such. In some cases the groups are also required to be consolidated within the container. This is a typical example of a constraint that is hard to implement using PGA: almost the only way to implement that groups are together within the container is by 'placing' them one by one, which is effectively what the construction heuristic does. Therefore, we assume that, with this requirement, the best way to find the solution is using the construction heuristic. We do not consider grouping constraints within the container and as a logical result we do not consider sequencing constraints. In the algorithm we develop, we consider grouping, but only on one level, as this is the data that is available. Although this level can represent any type of group, we assume it concerns orders for the remainder of this thesis.

**Additional concepts**   The PGA algorithm does not need to consider VMI, as this is typically only backfilled after the load has been created. Transportation forecasting and transportation cooperation are not relevant to the load building algorithm itself.

## 4.3   The improved PGA algorithm

This section gives the outline of the improved PGA algorithm we develop. The technical details and formulations of the models are worked out in the complimentary document. First, we give a small introduction of the algorithm.

As followed from the previous sections, the main difference between the algorithm that is outlined in this section and the algorithm described in Appendix A is the stacking of more than two pallets. Although this may seem like a small change, the impact from an algorithmic point of view is significant. Stacking more than two pallets does no longer allow to use a one-to-one

matching algorithm for creating stacks. Moreover, if volume is the limiting factor, it is no longer possible to estimate upfront the number of pallets that will fit in a container. This is the NP-hard 1-dimensional bin packing problem, even if stackability constraints are not considered. This makes the strategy of first selecting pallets per container difficult.

An alternative to the method of first selecting pallets per container, is the method of first creating stacks or even blocks of pallets (see Appendix A). This method is more naturally compatible with the stacking of more than two pallets. However, experiments with this method have shown that it is very hard to foresee how the stacks should be created in order to be able to create trucks with a high VFR using the stacks. As a small example, consider the case where we have less than a truck of priority 1 pallets which, for some reason, cannot be stacked on top of anything. Then pallets of other priorities should be stacked on top, but the total weight or volume should be such that the stacks or blocks still fit within a single truck. As a result of the stacks, there may be need for an additional truck, or the truck may not be fully utilized. Although one could think of ways to estimate this upfront, there is also additional complexity from other constraints, such as orders that should not be split. Almost the only way to incorporate all constraints is to actually know which stacks go into which container.

The best way to overcome the problems inherent with both strategies is to combine them in an algorithm that simultaneously creates stacks and assigns them to containers, which is the method we propose. It is likely very complex to do this for all containers together, so we will load the containers one by one. For each container, we create a number of stacks from the pallets not yet loaded such that they will, in most of the cases, fit in the container. This is done using an MIP model that we solve using column generation and partial application of branch and price. As the loading is done one by one, implementing the priority constraints as a sequence is trivial and orders can be kept together as good as possible. Furthermore, we will introduce additional constraints to avert the negative affects of the greediness of the solution and formulate the objective to optimize a mix of the weight and volume utility that most likely results in an overall optimal result.

After the stacks for a single container are created, they need to be placed on floorspots within the container to maximize the stability while taking into account the axle weight constraints. In order to do so we define a measure for the stability and create a model to optimize this. We propose two different methods: the first methods uses one-on-one matching to create blocks of stacks which then, using an MIP model, can efficiently be sorted in the container. The other method places the stacks directly in the container, again using an MIP model. The last method

is computationally more expensive than the first one and may be used if it is impossible to find an axle weight satisfying solution using the blocks, which is likely if the slack in the axle weights over the maximum gross weight is very small.

Algorithm 1 contains the basic outline of the algorithm: it shows how the two different subproblems are solved iteratively to load containers one by one. The remainder of this section contains a description of the different components and is structured as follows. Section 4.3.1 formulates the main objective of maximizing VFR for the first subproblem that creates stacks for a single container. Section 4.3.2 describes how the constraints for this subproblem should be defined in order to satisfy the priority requirements. Section 4.3.3 explains how the requirement to preferably not split orders can be incorporated when containers are loaded one by one. Section 4.3.4 gives a brief overview of the two options for the second subproblem on how to place the stacks within the container and discusses when they should be used. Finally, Section 4.3.5 brings everything together in a more detailed global description of the algorithm.

---

**Algorithm 1** The improved Pallet Grid Assignment algorithm.

---
1: **procedure** IMPROVEDPGA(palletsToLoad)
2:     **while** palletsToLoad $\neq \emptyset$ **do**
3:                                                                ▷ Subproblem 1: selecting stacks for a container
4:         stacks ← SelectStacksForSingleContainer(palletsToLoad);
5:                                                                ▷ Subproblem 2: placing the stacks in the container
6:         blocks ← CreateBlocks(stacks);
7:         blocks ← PlaceBlocksInContainer();
8:         **if** COGIntervalIsNotSatisfied() **then**
9:             blocks ← placeStacksInContainer();                          ▷ Create blocks by placing stacks
10:         **end if**
11:                                                       ▷ Subtract pallets in blocks placed in container from pallets to load
12:         palletsToLoad ← palletsToLoad \ PalletsInBlocks(blocks);
13:     **end while**
14: **end procedure**

---

## 4.3.1   Objective

The main objective of load building is clear: to minimize the number of containers or, equivalently, to maximize the VFR. The latter objective can perfectly be used in an approach that loads containers one by one. The only question is how to define VFR: with respect to volume or weight? This depends on the problem. Obviously, if a load is very heavy the loading is limited by the weight and it makes no sense to optimize the volume utilization, just as a very light load

should not be optimized on weight utilization. However, situations in between are less obvious. If a load is limited by weight on average but has a relatively high volume too, optimizing only by weight for the first container may deteriorate result for the remaining containers if too little volume is selected such that the remaining load is limited by volume. This leads to an overall suboptimal weight and volume utilization. Therefore, we propose to optimize a weighted combination of weight and volume, where the coefficient for each dimension is proportional to the lower bound on the number of containers. This follows as it is more important to optimize the most limiting factor, which is the one that requires the largest number of containers. It is important to note that the ratio can differ per priority and therefore coefficients should be adjusted according to the priorities that are being loaded, see Section 4.3.2.

**Calculation of lower bound for number of containers**

In terms of weight, the lower bound for the number of containers is easily calculated by dividing the total weight of the pallets by the maximum allowed weight per container. This is quite accurate because in practice the weight utilization can get very close to $100\%$[1]. In terms of volume the utilization is significantly lower due to the indivisibility of pallets in three dimensions. This differs from instance to instance based on actual pallet dimensions and stackability restrictions, so we cannot use a 'typical' volume utilization factor to determine an approximate lower bound for the number of containers. However, using the default loading pattern we are quite confident on the number of stacks that can be loaded in a container, so if we know the number of stacks we can calculate a lower bound for the number of containers quite easily. As mentioned before, finding the minimum number of stacks is NP-hard, but for our purpose of constructing a lower bound using an easy heuristic is sufficient. In particular, we use the *Best Fit Decreasing* (BFD) method (Johnson et al., 1974). The result found using the BFD method is within a factor $\frac{9}{11}$ (+4) from the optimum, so when constructing a lower bound, formally we should correct for this. However, this bound is theoretical and typically the result is much closer to the optimum. Also, in any practical scenario it is quite unlikely that the final number of stacks that is created is lower than the number of stacks found by the BFD method, especially if additional stackability constraints are added. Therefore, we use the number of stacks resulting from the BFD method as a lower bound, from which the lower bound (on volume) for the number of containers can be calculated. For a more accurate result, we do not round the number of containers when

---

[1] Only if there is very little diversity in the weights of pallets, there may be too little flexibility to select the maximum weight, or axle weight restrictions may require to load less weight.

calculating the ratio for the volume and weight coefficients.

## 4.3.2   Priorities

In order to satisfy the priority requirements, the priorities should be loaded one by one. That means we may only load a pallet with a lower priority if all pallets of a higher priority are loaded. This can be incorporated in the model with additional constraints. However, since the number of priorities is limited we may also efficiently try the different options for the priorities that are loaded in the container and enforce, by constraints, that pallets are loaded accordingly. The options are uniquely determined by setting the lowest priority on the truck, say $\hat{p}$. It is clear that all pallets with a higher priority, so for which $p_i < \hat{p}$, *must* be loaded since otherwise pallets with priority $\hat{p}$ may not be on the truck. All pallets for which $p_i = \hat{p}$ *may* be loaded and all pallets for which $p_i > \hat{p}$ *may not* be loaded since $\hat{p}$ is the lowest priority in the truck.

We always want to load as much on a truck as possible, while we do not allow priorities to be mixed. Therefore, we try to maximize the value of $\hat{p}$, such that all pallets with higher priority are loaded. In order to do so, we construct an upper bound on the value of $\hat{p}$. Then we try to create a truck with stacks that contain all higher priority pallets, and if we fail to do so we decrease $\hat{p}$ by one. This we continue until we are able to load all must-go pallets, which works because if $\hat{p}$ is equal to the highest priority, there are no 'must-go' pallets so a feasible solution always exists.

### Upper bound on lowest priority in container

The upper bound on the lowest priority on the container is constructed using lower bounds for the 'cumulative number of trucks' per priority. For each priority, the cumulative number of containers is the number of containers that is required to load all pallets which have a priority equal to or higher than that priority. This number of containers is the minimum of the lower bounds for the number of containers with respect to the weight and volume, which can be found using the method as described in Section 4.3.1. The first priority that 'cumulatively' requires more than one container to load can be used as an upper bound $p_{UB}$ for $\hat{p}$. This is because for any value $\hat{p} > p_{UB}$ all pallets up to priority $p_{UB}$ are must-go, while they do not fit in the container as the lower bound for the number of containers is more than one. If there is no priority that cumulatively requires more than one container, we set $\hat{p}$ to the lowest priority $+1$, such that all pallets are must-go.

### 4.3.3 Orders

Orders should be kept together as much as possible. This can be enforced as a hard constraint by selecting orders rather than individual pallets for each container. This is equivalent to enforcing that either all or no pallets from an order are selected in each container. However, this can lead to poor utilization or even infeasibility if there are many large orders. Also orders may be forced being in the same container because there are pallets that contain multiple orders. Therefore, we implement the requirement as a soft constraint or objective to minimize the number of orders that is split. Even with the constraint being soft, we should put some thought in the way we implement this. When loading the first containers, it is likely that the pallets that are selected are 'independent' pallets (pallets that do not contain parts of orders), or pallets from small orders, because these give most flexibility in maximizing the VFR without splitting orders. As a result it may happen that for the last containers only large orders remain, such that splitting is unavoidable. At the same time, there may be alternate solutions that achieve the same VFR with splitting fewer orders.

The problem is a result of the greedy characteristics of the approach, taking only locally optimal decisions when pallets are selected. In order to prevent this, we restrict a solution in which not enough *multi-order pallets*, which are pallets with products from *multi-pallet orders*, are selected. This is done using the following observation. Suppose the percentage of multi-order pallets is $x$. Assuming that we are able to fill containers 100%, the average fill of containers by multi-order pallets is also $x$, so at least one of the containers should have a percentage of at least $x$ with multi-order pallets. We can decide to select this container first by restricting that the percentage of fill by multi-order pallets is at least $x$. This increases the likelihood of larger orders being selected. If strictly more than $x$ percent of multi-order pallets is selected, the percentage of multi-order pallets and thus the restriction decreases for the remaining containers.

We should not regard the percentage in terms of numbers of pallets but with respect to the weight or volume, so we should again apply a method to incorporate both. Also the restrictions should be compatible with the must-go and may-go status of the pallets. The method is further worked out in Section 2.2 of the complimentary document.

### 4.3.4 Placement within container

After the stacks are created for a single container, they should be placed within the container in order to ensure that they fit and to find the best solution with respect to axle weights and

stability. As the weight of the load is fixed after the stacks have been selected, the axle weights constraints can be translated to a feasible range for the Center Of Gravity (COG). In order to find a stable solution with a COG in the feasible range we have two options:

- First blocks of two stacks are created, which are then sorted within the container.

- Stacks are placed directly in the container.

As mentioned before, the first method creates blocks using a matching algorithm, which can be done in negligible time. The placement of the blocks is a one-dimensional matching problem of the blocks to slots in the container, with some side constraints. As the number of blocks does typically not exceed fifteen, this can be efficiently solved using the MIP formulation. However, the creation of blocks limits the flexibility in the placement of the stacks and as a result it may not be possible to construct a solution that has a COG in the feasible range. In that case the second formulation can be used. This problem formulation is typically too complicated to solve to optimality, but the best quality feasible solution within a time limit may be used. This formulation will balance the maximization of the stability of the load with optimizing the weight distribution within the feasible COG range. For the stability, we define a measure for the support of the stacks and a measure to score the quality of the dynamic heavy loading pattern. If no solution with the COG in the feasible range exists, the model will give priority to minimizing the distance from the COG of the load to the feasible range.

If, because of overhang, it is not possible to load all stacks on a container, the maximum number of stacks that should be selected is decreased by the number of stacks that could not be placed and new stacks are created. Also, as mentioned in Section 3.4.1, it may happen that it is not possible at all to find a solution where the COG is in the feasible range, as the rear axle will always be overloaded a result of a lack in diversity in the weights of the stacks. To estimate upfront whether that will be the case, we construct a lower bound for the weight on the rear axle by sorting the stacks by weight and calculating the rear axle weight if the stacks are loaded, heaviest first, in rows of two from the front. If this lower bound exceeds the maximum, we create new stacks, but add the restriction that the diversity in weights of the stacks is a percentage greater than the diversity of the current stacks.

## 4.3.5   Algorithm description

The algorithm will fill the containers one by one, taking into account the objectives and restrictions as described in the previous subsections. We start with all pallets that have to be loaded,

and determine for each priority the lower bound for the cumulative number of containers, from which we find an upper bound on the lowest priority that is loaded in the container. Then iteratively we try all priorities from the upper bound to 1 to use as lowest priority while loading the container, although typically the first iteration is sufficient. In each iteration, the statuses of the pallets to load are defined according to the lowest priority, and the maximum number of stacks is initialized according to the number of floorspots in a container. The steps described below are repeated until an arrangement of the stacks in the container is found, or the restrictions on the maximum number of stacks or the weight diversity have become such that the problem of selecting stacks for the container is infeasible. In that case, we continue with a higher priority as lowest priority, updating the statuses of the pallets that should be loaded.

If a feasible set of stacks is created according to the maximum number of stacks, we first check if the COG can be in the feasible range by calculating the lower bound for the rear axle weight. If not, we increase the restriction on the diversity of the weights of the stacks, but otherwise we try to find a feasible arrangement by first creating the blocks. If not all stacks can be matched in blocks, we decrease the maximum number of stacks accordingly and restart with the selection of stacks. Also if the total length of the blocks does not fit in the length of the container, we decrease the maximum number of stacks and retry, but this is very rare as there is usually enough slack at the rear of the container. If the blocks are feasible, we try to sort them in the container. If this does not give a feasible solution with respect to the COG range, we invoke the method to place the stacks directly.

Algorithm 2 contains the top-level pseudocode of the improved PGA algorithm that we have described.

---

**Algorithm 2** The improved Pallet Grid Assignment algorithm.

---

1: **procedure** IMPROVEDPGA(palletsToLoad)
2:     **while** palletsToLoad $\neq \emptyset$ **do**
3:         $p_{UB} \leftarrow$ GetPriorityUB();
4:         **for** $\hat{p} \leftarrow p_{UB}, p_{UB} - 1, ..., 1$ **do**
5:             maxNoStacks $\leftarrow 2K$;
6:             **while** stacks $= \emptyset$ **do**
7:                 $\Pi \leftarrow 0$;
8:                 stacks $\leftarrow$ SelectStacksForSingleContainer(maxNoStacks, $\Pi$, $\hat{p}$);
9:                 **if** stacks $= \emptyset$ **then**
10:                     break;                                                   ▷ Not possible, go to next priority
11:                 **end if**
12:                 **while** GetRearAxleWeightLB(stacks) > maxRearAxleWeight **do**
13:                     $\Pi \leftarrow$ GetWeightDiversityOfStacks(stacks)·$(1 + d\Pi)$;
14:                     stacks $\leftarrow$ SelectStacksForSingleContainer(maxNoStacks, $\Pi$, $\hat{p}$);
15:                     **if** stacks $= \emptyset$ **then**
16:                         break 2;                           ▷ Not possible, break two levels to go to next priority
17:                     **end if**
18:                 **end while**
19:                 blocks $\leftarrow$ CreateBlocks(stacks);
20:                 **if** BlocksFitInLength(blocks) **then**
21:                     blocks $\leftarrow$ PlaceBlocksInContainer();
22:                     **if** COGIntervalIsNotSatisfied() **then**
23:                         blocks $\leftarrow$ placeStacksInContainer();                           ▷ Create blocks by placing stacks
24:                     **end if**
25:                 **else**                                         ▷ Decrease number of stacks and try again
26:                     maxNoStacks $\leftarrow$ min(maxNoStacks - 1, maxNoStacks - (|blocks| - $K$));
27:                     stacks $\leftarrow \emptyset$;
28:                     blocks $\leftarrow \emptyset$;
29:                 **end if**
30:             **end while**
31:             **if** stacks $\neq \emptyset$ **then**
32:                 break;                                                        ▷ Solution found, break priority loop
33:             **end if**
34:         **end for**
35:                                          ▷ Subtract pallets in blocks placed in container from pallets to load
36:         palletsToLoad $\leftarrow$ palletsToLoad \ PalletsInBlocks(blocks);
37:     **end while**
38: **end procedure**

---

# Chapter 5

# Results

This chapter contains the results that follow from comparing the construction heuristic with the improved PGA algorithm. Section 5.1 describes the experimental set-up and the parameter settings. Section 5.2 graphically represents the overall results. Since it is hard to take all aspects into account in the overall results, Section 5.3 discusses five scenarios in more detail. For each scenario, the loading results from both methods are visualized and discussed. Finally, Section 5.4 discusses the computation times of both algorithms.

## 5.1 Experimental set-up

In order to compare the quality of the construction heuristic and the improved PGA algorithm, we tested both methods on a set of instances, which is described later in this section. For the construction heuristic the implementation in LoadDesigner is used with default settings. The PGA algorithm was implemented in MATLAB and uses the Gurobi solver to solve the MIP's. The parameter settings used for the PGA algorithm are listed at the end of this Section.

Since the PGA algorithm only considers the step after the pallets have been created, ideally the comparison would be done on the same pallets. However, LoadDesigner and OPLB (which runs the PGA algorithm) have their own implementation. In many cases the pallets are very similar, especially because most input data already contains full stock pallets, but there may be minor differences in the results as a consequence of this step. The results of the order splitting constraints however strongly depend on the way orders are consolidated on the pallets. Therefore, we will not compare the quality of the solutions in this sense.

For both methods, the LoadDesigner software is used to visualize the results and calculate the KPI's that are compared: the weight VFR, the volume VFR, and the front and rear axle weights as a percentage of their maxima. The axle weights depend on the axle positions. The optimal positions for the planned load are determined by LoadDesigner automatically, regardless of the method that was used. In the visualizations, the pallets are colored according to their priority.

We use a data set of 22 real instances that are collected from two different customers in the US. The data set is a mix of instances with load sizes varying from two to nine containers. Most of the instances contain moderately heavy loads, but there are some light loads as well. The majority of the instances consists of multiple priorities. Some instances have been slightly adapted to make them compatible with the PGA algorithm.

Table 5.1 contains the parameter settings for the PGA algorithm that were used in the experimental set-up. The parameter settings are found by experiments during the development of the algorithm. Most of the settings represent weights for parts of the objectives in the subproblems, so the values represent the relative importance of the different parts of the objectives. The parameters are scaled according to the unit of the value they correspond to. The costs $C^v$ for order splitting are set to 0 because grouping is not considered. For the construction heuristic, the default settings from the LoadDesigner implementation are used with 750 iterations.

## 5.2  Overall results

For the data set we used, the number of containers was equal for both methods for all instances. Therefore, we can use the VFR of the last container as a KPI. This is useful as it gives only one value per instance, while it contains sufficient information: a lower VFR in the last container implies a higher VFR in the other (full) containers. Therefore, in the scope of the last container, a *lower* VFR is preferred. In the algorithm the weighted combination of the volume and weight VFR is used as an optimization criterium, but in this section we report the maximum of both. This is because the coefficient for the weighted combination depends on the algorithm state and the value has no clear intuitive meaning, while using the maximum means that the dimension that takes up most 'space' in the container defines how full it is.

Figure 5.1 is a scatter plot of the VFR of the last container using the construction heuristic against the result using the PGA algorithm. The horizontal axis represents the construction heuristic and the vertical axis the PGA algorithm, so a point below the diagonal corresponds to

| Parameter | Value | Explanation |
|---|---|---|
| $C^{\mathrm{VFR}}$ | $10^5$ | The VFR is in the range $(0,1)$, so 1% VFR gives 1000 reward. |
| $C^\theta$ | 100 | Costs of 100 for each deviation from the desired number of stacks. Mostly relevant for the last container. |
| $C^v$ | 0 | Grouping is not considered. |
| $C^l$ | 2 | The length of the pallets is in $mm$. |
| $C^b$ | 1 | The width of the pallets is in $mm$. Less important because typically there is slack in the width of the container. |
| $C^h$ | 4 | The heights of the stacks is in $mm$. More important to have equal height of stacks than equal length. |
| $C^\xi$ | 20 | The COG is in mm. |
| $C^\Xi$ | 1000 | There should be a high penalty if the COG is outside of the range. |
| $C^s$ | 1 | The support is in mm. |
| $C^{s^2}$ | 4 | The support is in mm, heavy penalty for support deficiency higher than threshold. |
| $\beta^{\mathrm{f}}$ | 1.5 | Front support, measured in mm for each stack. |
| $\beta^{\mathrm{r}}$ | 0.5 | Rear support is less important than front and side support. |
| $\beta^{\mathrm{s}}$ | 1 | Side support in terms of importance in between of front and rear support. |
| $\tau$ | $0.2H$ | Set the support deficiency threshold to 0.2 times the container height (less than a half pallet in a stack of two). |
| $d\Pi$ | 0.03 | Increase diversity in weights of stacks by 3% if stacks provide no feasible axle weight solution. |

Table 5.1: Parameter settings used in the experiments.
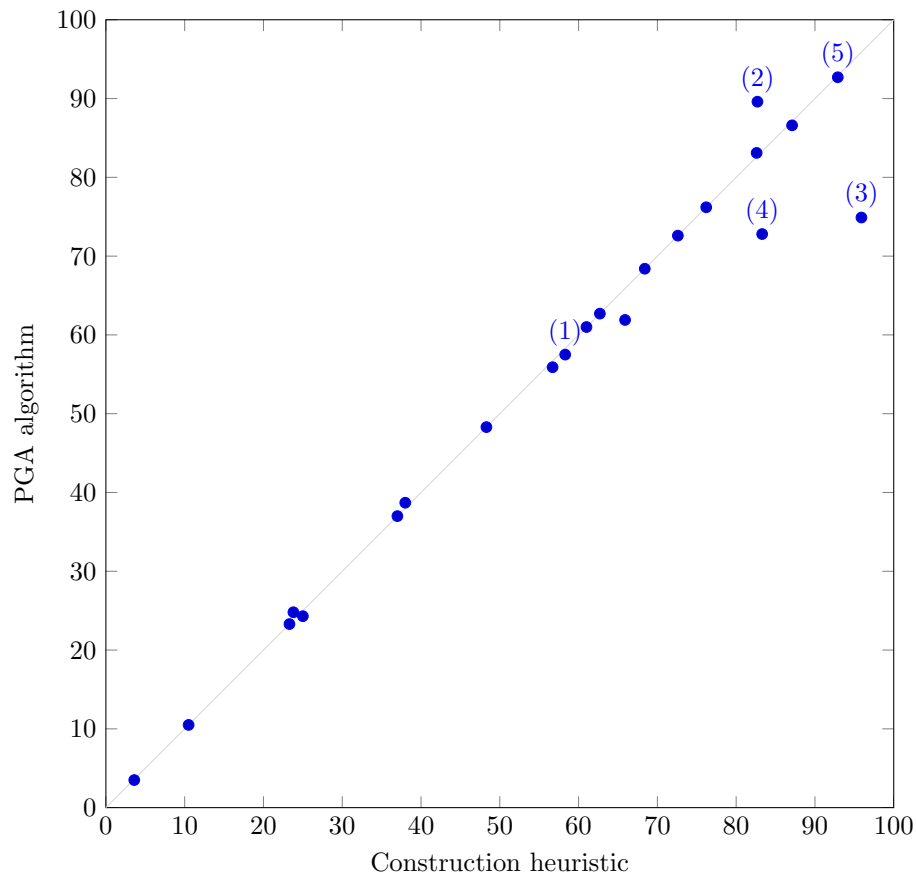
Figure 5.1: Scatter plot of the maximum of the weight and volume fill rate of the last container for each instance: the horizontal and vertical axes represent the results of the construction heuristic and the PGA algorithm respectively.

an instance for which the PGA algorithm performs better than the construction heuristic.

In the scatter plot it can be observed that most entries are on the diagonal, which means that in terms of the VFR the construction heuristic and the PGA algorithm perform similar. This is a good result since the construction heuristic in LoadDesigner, as a result of many years of experience, is known to be very efficient in terms of VFR. There is one instance, labeled with (2), for which the construction heuristic results in a lower VFR of the last container and thus seems to perform better. However, this is at the cost of an axle weight violation in one of the full containers; see scenario 2 in Section 5.3. For two other instances, a better result in terms of VFR is found by the PGA algorithm. These instances, labeled with (3) and (4) also correspond to the scenarios in Section 5.3. Instances (1) and (5) have similar results in terms of VFR, but

are discussed because they illustrate other differences between the construction heuristic and the PGA algorithm.

For the instances in the data set that are not discussed in the scenarios, the results using the construction heuristic and the PGA algorithm are similar.

## 5.3 Five scenarios

This section contains five scenarios that illustrate a number of differences between the construction heuristic and the PGA algorithm. Scenarios 1 and 2 show how two different axle weight violations are resolved by using the PGA algorithm and illustrate how the limited number of stacks for the last container affects the solution. Scenarios 3 and 4 show how the PGA algorithm results in less weight being loaded in the last container. Scenario 5 illustrates how the dynamic heavy loading pattern found by the PGA algorithm is different from the fixed patterns used in the construction heuristic for an improved stability.

### 5.3.1    Scenario 1

Table 5.2 illustrates scenario 1. From this scenario it can be seen that using the PGA algorithm, approximately 1% can be saved from the last container, in terms of both weight and volume. This is because more weight is loaded in the first container. At the same time, the axle weight violation in the second container is resolved as a result of the dynamic loading pattern that the PGA algorithm creates. A last difference is the way the load is placed in the last container, but this is a result of a choice in implementation. Some versions of LoadDesigner have an option to do 'length compression' which will give a result similar to the PGA algorithm. From a practical perspective, the arrangement of the pallets in the last container is of little relevance, because typically the last truck will not be released. Visually, the result from the PGA algorithm indicates more clearly that the last container is not full.
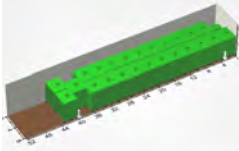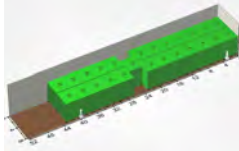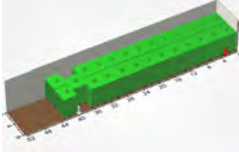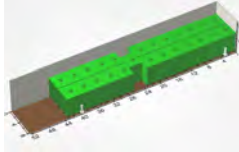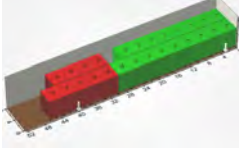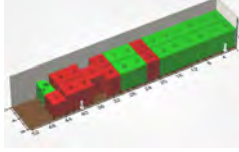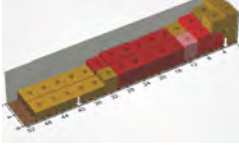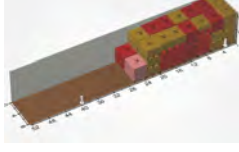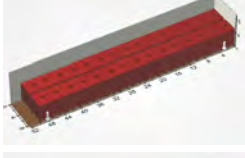
| Container | Construction heuristic | | PGA algorithm | |
|---|---|---|---|---|
| 1 | | | | |
| Weight | 98.2 % | | | 99.5 % |
| Volume | 47.6 % | | | 48.0 % |
| Axles (f/r) | 99.8/96.6 % |  |  | 99.6/99.3 % |
| 2 | | | | |
| Weight | 99.5 % | | | 99.5 % |
| Volume | 48.0 % | | | 48.0 % |
| Axles (f/r) | **102.5**/96.5 % |  |  | 99.6/99.3 % |
| 3 | | | | |
| Weight | 99.2 % | | | 98.7 % |
| Volume | 50.3 % | | | 50.9 % |
| Axles (f/r) | 99.1/99.4 % |  |  | 98.8/98.7 % |
| 4 | | | | |
| Weight | 58.3 % | | | 57.5 % |
| Volume | 44.0 % | | | 43.1 % |
| Axles (f/r) | 53.4/63.2 % |  |  | 82.8/32.4 % |

Table 5.2: The construction heuristic versus the improved PGA algorithm for scenario 1.

### 5.3.2 Scenario 2

Table 5.3 illustrates Scenario 2. In this scenario it can be seen that using the construction heuristic the rear axle is overloaded for the first container, because there is no diversity in the weights of the pallets. Using the PGA algorithm, two pallets form this container are moved to the second container.
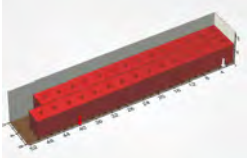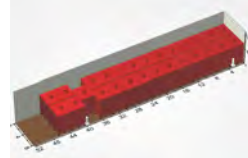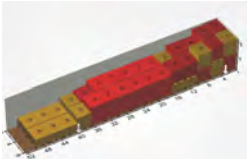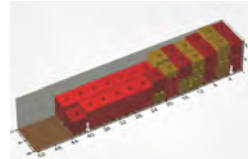
| Container | Construction heuristic | | PGA algorithm | |
|---|---|---|---|---|
| 1 | | | | |
| Weight | 99.2 % | | | 92.3 % |
| Volume | 55.2 % | | | 51.4 % |
| Axles (f/r) | 85.0/**113.3** % | | | 86.2/98.4 % |
| 2 | | | | |
| Weight | 82.7 % | | | 89.6 % |
| Volume | 56.3 % | | | 60.1 % |
| Axles (f/r) | 76.4/89.1 % | | | 89.5/89.7 % |

Table 5.3: The construction heuristic versus the improved PGA algorithm for scenario 2.

### 5.3.3   Scenario 3

Scenario 3 in Table 5.4 consists of a large load of 9 containers with little diversity in the sizes of the pallets, but some diversity in the weights. Using the PGA algorithm, significantly more weight is loaded in the second and third container, which results in a decrease in weight loaded in the last container of over 20%!
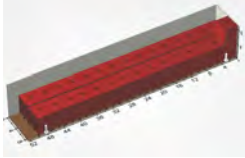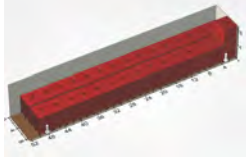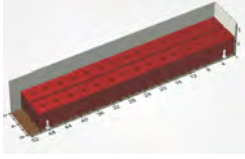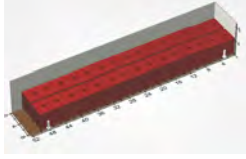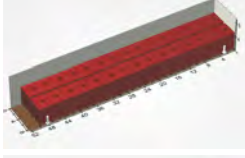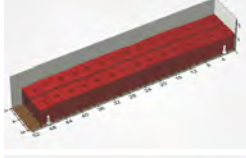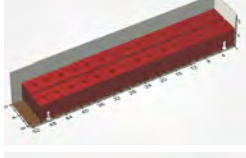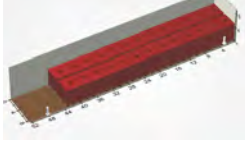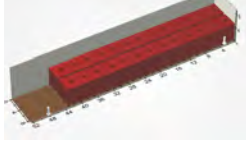
| Container | Construction heuristic | | PGA algorithm | |
|---|---|---|---|---|
| 1 | | | | |
| Weight | 98.4 % | | | 99.4 % |
| Volume | 50.5 % | | | 50.5 % |
| Axles (f/r) | 83.8/80.2 % | | | 91.3/74.4 % |
| 2 | | | | |
| Weight | 91.8 % | | | 99.8 % |
| Volume | 49.0 % | | | 48.6 % |
| Axles (f/r) | 74.7/78.3 % | | | 91.7/74.6 % |
| 3 | | | | |
| Weight | 87.5 % | | | 99.4 % |
| Volume | 48.9 % | | | 48.0 % |
| Axles (f/r) | 74.2/71.7 % | | | 91.3/74.4 % |
| 4 - 8 | | | | |
| Weight | 86.4 % | | | 86.4 % |
| Volume | 48.9 % | | | 48.9 % |
| Axles (f/r) | 74.2/69.9 % | | | 74.2/69.9 % |
| 9 | | | | |
| Weight | 95.9 % | | | 74.9 % |
| Volume | 41.0 % | | | 42.4 % |
| Axles (f/r) | 84.4/75.4 % | | | 73.4/51.4 % |

Table 5.4: The construction heuristic versus the improved PGA algorithm for scenario 3.

### 5.3.4 Scenario 4

Scenario 4 is another scenario that illustrates how the VFR in the last container decreases using the PGA algorithm, because more weight is loaded into the full containers, especially in container 3. The results are comparable in terms of weight distribution and stability.
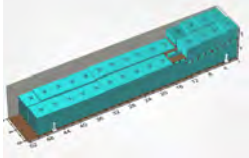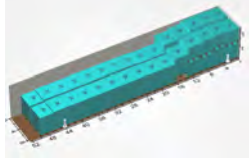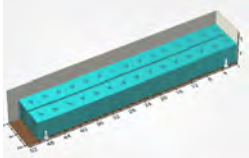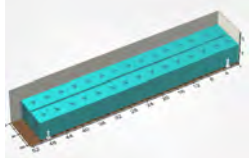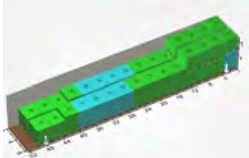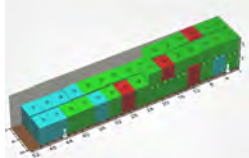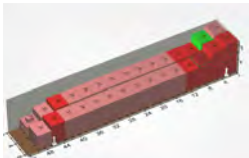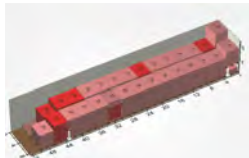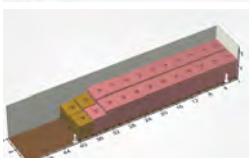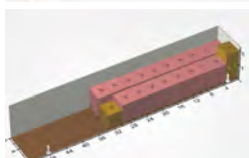
| Container | Construction heuristic | | PGA algorithm | |
|---|---|---|---|---|
| 1 | | | | |
| Weight | 98.2 % | | | 99.9 % |
| Volume | 62.1 % | | | 64.0 % |
| Axles (f/r) | 93.3/98.6 % | | | 99.0/96.2 % |
| 2 | | | | |
| Weight | 85.4 % | | | 85.7 % |
| Volume | 52.9 % | | | 52.5 % |
| Axles (f/r) | 83.2/83.5 % | | | 91.9/75.6 % |
| 3 | | | | |
| Weight | 92.0 % | | | 99.9 % |
| Volume | 60.5 % | | | 67.8 % |
| Axles (f/r) | 94.0/85.7 % | | | 99.6/95.7 % |
| 4 | | | | |
| Weight | 99.1 % | | | 99.8 % |
| Volume | 57.3 % | | | 53.5 % |
| Axles (f/r) | 97.3/96.3 % | | | 99.2/95.9 % |
| 5 | | | | |
| Weight | 83.3 % | | | 72.8 % |
| Volume | 41.2 % | | | 36.1 % |
| Axles (f/r) | 99.2/63.7 % | | | 99.7/42.5 % |

Table 5.5: The construction heuristic versus the improved PGA algorithm for scenario 4.

### 5.3.5   Scenario 5

Scenario 5 in Table 5.6 shows how the dynamic loading pattern implemented in the PGA algorithm results in a more compact pattern. This pattern maximizes the side support, because there are fewer pallets that are individually in a row. The solution remains feasible with respect to the axle weights, because the rear axle position has moved forward. If this is not desired, the warehouse may decide to load the pallets differently, but the feasibility is guaranteed and the PGA algorithm suggests the most stable load plan. For this scenario, the results of the PGA algorithm and the construction heuristic are similar in terms of VFR.
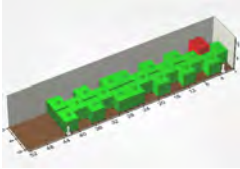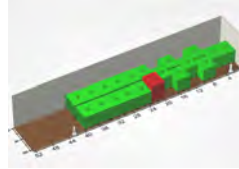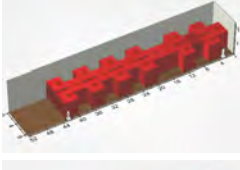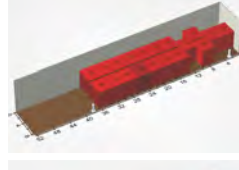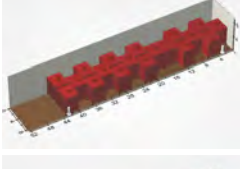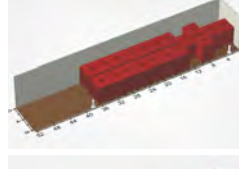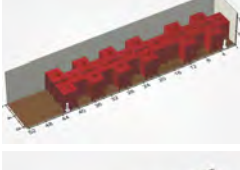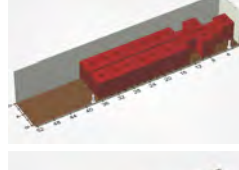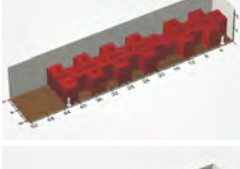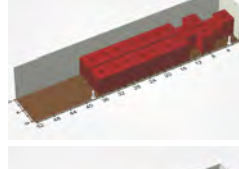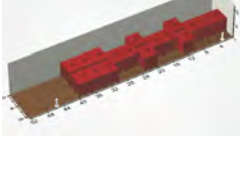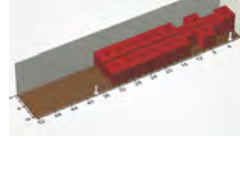
| Container | Construction heuristic | | PGA algorithm | |
|---|---|---|---|---|
| 1 | | | | |
| Weight | 99.4 % | | | 99.4 % |
| Volume | 24.8 % | | | 24.8 % |
| Axles (f/r) | 99.4/94.1 % | | | 100.0/93.5 % |
| 2 | | | | |
| Weight | 99.7 % | | | 99.7 % |
| Volume | 30.4 % | | | 30.4 % |
| Axles (f/r) | 99.4/94.9 % | | | 98.9/95.3 % |
| 3 - 5 | | | | |
| Weight | 99.7 % | | | 99.7 % |
| Volume | 30.4 % | | | 30.4 % |
| Axles (f/r) | 99.4/94.9 % | | | 98.9/95.3 % |
| 6 | | | | |
| Weight | 99.7 % | | | 100.0 % |
| Volume | 30.4 % | | | 30.1 % |
| Axles (f/r) | 99.4/94.9 % | | | 99.3/95.3 % |
| 7 | | | | |
| Weight | 100.0 % | | | 100.0 % |
| Volume | 30.1 % | | | 30.1 % |
| Axles (f/r) | 99.6/95.0 % | | | 99.3/95.3 % |
| 8 | | | | |
| Weight | 92.9 % | | | 92.7 % |
| Volume | 21.6 % | | | 21.9 % |
| Axles (f/r) | 99.2/81.7 % | | | 99.4/81.0 % |

Table 5.6: The construction heuristic versus the improved PGA algorithm for scenario 5.

## 5.4   Computation times

In terms of computation time, the different methods are hard to compare. This is because they heavily depend on the settings used. For the construction heuristic, actually any computation time can be used depending on the number of iterations that is set as input. For the instances in our data set, with 750 iterations, the calculation time was in the order of 10 to 60 seconds, depending on the size of the instance. For the PGA algorithm, the computation time is similar, although most of the time is consumed by solving the MIP model for the second subproblem of placing the stacks within the container. This time may be limited by not solving the model to optimality, such that the arrangement is suboptimal, but feasible. The first subproblem is for most cases solved within a second or a number of seconds, so the selection of the stacks for a container is very fast.

# Chapter 6

# Conclusions & future research

This chapter contains the conclusions of this research and lists a number of directions for future research. The two most important contributions are an overview of the situation in the US and the improvement of the PGA algorithm.

**US situation**  Summarizing what we have learned from the three month visit, we provided an overview that gives important insights in the different topics that are relevant for load building in the US. For ORTEC, these insights help to improve the business understanding of the OPLB development team. Additionally, we identified a number of things that could be improved, not only for the PGA algorithm but for the load building software in general:

- The formulation of the axle weight constraints should take into account that the algorithm can only calculate an estimation of the axle weights due to the 'reality gap'.

- The two step approach with palletized loading may cause that pallets are created that result in a suboptimal solution during the loading step. Ideally there would be some interaction between the different steps.

- The VFR KPI should only concern actual product weight and volume if the number of pallets is variable because of woodless stacking or pallet splitting.

- The VFR can be improved if the priority requirements are reformulated such that the priorities are allowed to be mixed.

- Parties and relations as dynamic grouping levels are insufficient to model every business scenario. Ideally it would be possible to configure three or more dynamic grouping levels.

**PGA algorithm**   We improved the PGA algorithm by adding the possibility to create stacks of more than two pallets. In order to do this, we integrated the steps of assigning pallets to containers and creating stacks, by creating stacks per container. We developed a method to place the stacks in the container while taking into account the stability and axle weight constraints. As a result, the improved PGA algorithm can be used for a wider scope of scenarios, which brings it a considerable step closer to practical implementation.

From the results it follows that the improved PGA algorithm has high potential. In many cases, the algorithm is able to find results that are comparable to the results from the construction heuristic in LoadDesigner, while in some cases the results are actually improved. This is satisfying, since the implementation in LoadDesigner has evolved from many years of experience in the development of optimal load building algorithms. The PGA algorithm achieved the same performance but it may be further improved, for example using parameter tuning methodologies. Currently, in a practical scenario, the PGA algorithm should be used in parallel with the construction heuristic. This is because it cannot deal with all requirements, such as complex grouping constraints. However, if implemented in a production environment, the algorithm could already replace the construction heuristic for a number of scenarios.

The PGA algorithm was originally designed for a better satisfaction of the global constraints, because of the global assignment. The results indeed improved, especially with respect to the axle weight restrictions, but the improvements were only partially the result of the global assignment. However, the benefit resulting from the global selection of pallets for each container turned out to be significant: for some scenarios there was a major decrease in utilization of the last container, such that more products are loaded in the other containers.

Even when the PGA algorithm does not strictly improve the results using the construction heuristic, it is interesting that similar results are achieved *in a single iteration.* In contrast with the construction heuristic, the computation time is not caused by multiple iterations, but by different components of the algorithm. It has been observed that in this sense the method used for selection of stacks for a single container is very efficient. The method requires relatively little computation time, and enables the use of column generation with branch and price to solve a powerful MIP model, to make an optimal selection of pallets. The placement of the pallets is feasible with high probability, because the stackability of the pallets is guaranteed.

In the algorithm that was developed, the placement of the stacks takes up most of the time. The reason is that, as a result of the first step, the stacks are fixed and therefore provide less flexibility in finding a good solution. This means that the method used for the placement could

still be improved, for instance by somehow allowing the stacks to be modified. Alternatively, the MIP model could be replaced by a construction heuristic to be especially developed for placing pallets in a single container. The quantifications of the KPI's that were introduced for the stability may be used in the development of such a construction heuristic. We discuss this and other topics for future research in the next paragraph.

**Future research**   This paragraph contains a number of topics that could be directions for future research at the Pallet and Load Building department of ORTEC. The first four are technical and concern the PGA algorithm and, in some cases, also the construction heuristic directly. Improvements for these topics may result in an algorithm that provides better quality solutions or supports more requirements. The last two topics are more business related: in those cases it has to be investigated if there really is added value for the business. For each topic, we give a short description of the difficulties and the added value.

- **Pinwheeling patterns**: in the US, pinwheeling patterns are relevant for refrigerated and intermodal containers. In order to use full and different partial pinwheeling patterns with the PGA algorithm, an additional decision is introduced. This is the decision which pinwheeling pattern should be used for each container. It should be investigated how this decision can be included in the algorithm.

- **Loading containers simultaneously**: instead of loading the containers one by one, the model could be extended to load all containers simultaneously. Especially with grouping constraints it could be beneficial to make a global assignment rather than a local assignment for each container. Using column generation makes the model larger, so it has to be investigated wether this is possible and what the computational performance would be.

- **Placement of pallets in container**: a different method for placement of pallets within the containers could improve the stability of the load. For the best result, the method should be able to modify the stacks. Possible methods could be an advanced column generation method to generate stacks for each position in the container, or a variant of the construction heuristic. Using the latter makes it possible to additionally consider grouping within the container, which, together with the model to assign stacks to the container, makes it possible to consider different types of grouping and sequencing constraints.

- **Integrating palletizing/loading step**: the fixed configuration of the pallets may be suboptimal during loading, especially if grouping constraints are considered or there are

stock pallets with odd heights. An option is to iteratively try different pallet configurations for the best loading result, or to create the pallets dynamically during loading. For instance, in the improved PGA algorithm, the stacks may be created directly from unpalletized products, using an adaption of the palletizing algorithm.

- **Mixing of priorities**: mixing of priorities may be business potential for improving the VFR, but it has to be studied whether this really is the case and what the operational constraints are. From an algorithm standpoint, the complexity of the problem increases because of the additional decision how many trucks to use for each priority, but implementation may be feasible if containers are loaded simultaneously (see further research on 'loading containers simultaneously').

- **Axle weights estimation**: the accuracy of the calculated axle weight may be estimated depending on the load type. Based on the accuracy, it might be possible to set the limit for the axle weights and total weight dynamically. This way, the VFR can be increased if there is little risk in violating the axle weights. Alternatively, fixed limits may be used in combination with an estimated distribution or confidence interval for the axle weight and the requirement that the axle weight is within the limits with a high probability, for instance 95% or 99%.

# Appendix A

# Current PGA algorithm

This appendix describes the implementation of the PGA algorithm that is currently used as proof of concept. Currently ORTEC is working on an algorithm that exploits the assumptions in section A.1 to efficiently place pallets in an imaginary $2 \times 2 \times n$-grid such that the solution is better with respect to global constraints that are hard to satisfy using the construction heuristic. The algorithm mainly consists of the following two steps:

- **Placement**: creating containers with a number of $2 \times 2$ blocks of pallets per container. For this, two different strategies can be used:

  - **First select per container**: first the pallets are divided over the containers using an MIP model after which the pallets are placed inside the container. For each container, the pallets are combined into blocks of $2 \times 2$, which are then placed in the container.

  - **First create blocks**: first pallets are combined into blocks of $2 \times 2$ which are then divided over the containers using an LP.

- **Post steps**: sorting the blocks for each container, to satisfy constraints such as axle weight distribution and stability (a decreasing load height).

## A.1  Assumptions

The assumptions that are made by the current implementation of the PGA algorithm are the following:

- Pallets are created in an initial phase, prior to loading.

- The orientation in which pallets are placed in the container is predefined.

- All pallets have the same dimensions, except for small differences in overhang.

- Pallets can be stacked, with a maximum of two.

- Stacks can be placed in two columns in the containers.

- The number of rows of pallets can be estimated from the pallet and container dimensions.

- Weight limitation is on two axles, which is equivalent to a total weight restriction and a lengthwise interval for the center of gravity (COG), dependent on the weight loaded.

## A.2    Placement

### A.2.1    First select per container

Using the **First select per container** strategy pallets are divided over the different containers. This is done in an iterative process in which in every iteration pallets are selected to fill a single container using a LP. This is done per priority: first containers are filled one by one with orders of highest priority. After that, the remaining space in the last container is filled with orders of the second priority, etc. Orders (referred to as groups) are always kept together, but if an order consists of too many pallets to fit in a single container, it is split. After that, it can be enforced (by setting a very high objective value to the pallets in the order) that the rest of the order is loaded in the next container, but the default behavior is that the rest is considered as a smaller order and thus may or may not be selected for the next container, dependent on the LP criteria.

After the pallets to load in the next container are selected, the container is loaded (see the paragraph on loading a single container). If not all pallets fit, at least one pallet is removed according to a strategy (see the paragraph on removing pallets) and the container is loaded again, until all pallets fit.

**Load single container**

For loading of a single container there are three different strategies, but the most important consist of first stacking and then pairing stacks to blocks, after which the blocks are placed in the container. Both steps are done by application of an algorithm for maximum weighted matching (non-bipartite), thus from the entire population pairs are selected. Some matchings

are infeasible, for instance because of stackability constraints or the container height/width. The feasible matches have costs which depend on the difference in dimensions, priority or orders on the pallets. The matching results in stacks and the stacks are again matched, but with different costs, to form blocks. The blocks are placed in the container in their natural order, as they will be sorted during the post steps (section A.3).

**Removing pallets**

If the container is loaded, the total length of the pallets left and right is calculated. If pallets do not fit, at least one pallet is removed according to a strategy. Multiple strategies (for instance removing groups or by priority) can be configured that are executed sequentially. After a strategy is executed that removes at least one pallet, the container is loaded again.

### A.2.2 First create blocks

Using the **First create blocks** strategy pallets are first created in blocks using the same methodology as described in the paragraph on loading a single container. After that blocks are divided over the containers, using an LP that is similar to the LP used for selecting the pallets per container.

## A.3 Post Steps

After the pallets are placed in the container, a number of post steps is applied. The target of these steps is to make the load well spread over the container surface and to make the axle weight distribution as good as possible. As the load should be stable, both statically and dynamically, the higher stacks are preferably loaded in front of the container. Different options are implemented for the post steps. One example is a heuristic sorting of the blocks according to their height (by creating buckets of blocks with similar height) while considering the axle weight restriction. The other example is an LP that minimizes the distances of the real center of gravity (COG) to the preferred COG while maintaining a decreasing load height.

# Appendix B

# US legislation

## B.1 Types of trucks

Generally two types of trucks are distinguished: *California Legal Trucks* and *Interstate "STAA" trucks*, of which the latter are larger and heavier. In California, the different types of trucks are allowed to different routes according to the Caltrans Truck Route Map [1]. Because of the different cabin sizes there are different rules regarding the lengths of these trucks with a semitrailer[2]:

- California Legal Truck: the length of the semitrailer is unlimited, but

  - the KPRA (king-pin-to-rear-axle) distance should not be over 38 feet for 1 rear axle or 40 feet for 2 rear axles and the overall length should not be over 65 feet.

- Interstate "STAA" truck: the overall length is unlimited, but

  - the length of the semitrailer should not be over 48 feet, ór

  - the length of the semitrailer should not be over 53 feet and the KPRA distance should not be over 38 feet for 1 rear axle or 40 feet for 2 rear axles.
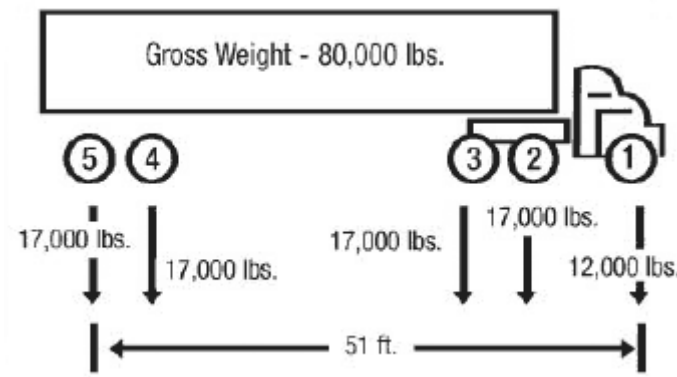
---

[1]`http://www.dot.ca.gov/hq/traffops/engineering/trucks/truck-length-routes.htm`
[2]`http://www.dot.ca.gov/hq/traffops/engineering/trucks/truckmap/truck-legend.pdf`

Figure B.1: Truck with axle weights.

## B.2  Federal Bridge Law

Equation (B.1) describes the Federal Bridge Law (FBL) that restricts the weight any set of axles on a motor vehicle may carry on the Interstate highway system[3].

$$W = 500 \left( \frac{LN}{N-1} + 12N + 36 \right) \tag{B.1}$$

In this formula, the following variables are used:

- $W$ = the overall gross weight on any group of two or more consecutive axles to the nearest 500 pounds.

- $L$ = the distance in feet between the outer axles of any group of two or more consecutive axles.

- $N$ = the number of axles in the group under consideration.

In addition to this formula, a single axle is restricted to a weight of 20,000 pounds and tandem axles less than 96 inch apart may bear 34,000 pounds together. The gross vehicle weight is limited to 80,000 pounds. Figure B.1 gives an example of a truck and semi-trailer combination with a distribution of the weight over the axles that satisfies the FBL.

---

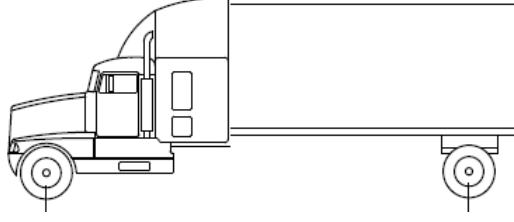[3]http://ops.fhwa.dot.gov/freight/publications/brdg_frm_wghts/index.htm

Figure B.2: Example of a truck with two axles.

## B.3    Calculations - Two axles

In case of a single fixed truck with two axles (Figure B.2) we can translate the restrictions on the axle weights to an allowed range in which the center of gravity should be. For this, let $x_1$ and $x_2$ be the positions of the front and rear axle and $x_{\text{COG}}$ the position of the center of gravity. Let $w_{\text{tot}}$ be the total weight and $w_1$ and $w_2$ the weights on the front and rear axle. Then, we might consider the truck as a lever with the front axle as fulcrum. The total weight $w_{\text{tot}}$ exerts a force downwards at position $x_{\text{COG}} - x_1$, while the axle exerts a force corresponding to a weight $w_2$ upwards at position $x_2 - x_1$, so using the concept of equal moments we have

$$(x_{\text{COG}} - x_1)w_{\text{tot}} = (x_2 - x_1)w_2. \tag{B.2}$$

If the rear axle is allowed to bear $w_2^{\max}$ at most, then $w_2 \leq w_2^{\max}$ results in

$$x_{\text{COG}} \leq (x_2 - x_1)\frac{w_2^{\max}}{w_{\text{tot}}} + x_1. \tag{B.3}$$

Similarly we find, with a maximum front axle load of $w_1^{\max}$, that

$$x_{\text{COG}} \geq -(x_2 - x_1)\frac{w_1^{\max}}{w_{\text{tot}}} + x_2. \tag{B.4}$$

Also, we may have minimum required loads $w_1^{\min}$ and $w_2^{\min}$, which result in Equations (B.3) and (B.4), so that we find

$$\max \left\{ \begin{array}{l} -(x_2 - x_1)\frac{w_1^{\max}}{w_{\text{tot}}} + x_2, \\ (x_2 - x_1)\frac{w_2^{\min}}{w_{\text{tot}}} + x_1 \end{array} \right\} \leq x_{\text{COG}} \leq \min \left\{ \begin{array}{l} (x_2 - x_1)\frac{w_2^{\max}}{w_{\text{tot}}} + x_1, \\ -(x_2 - x_1)\frac{w_1^{\min}}{w_{\text{tot}}} + x_2 \end{array} \right\}. \tag{B.5}$$

## B.4    Calculations - Three or more axles

In case we have three or more axles, we can no longer calculate the axle weights for each of the axles using equations for equal moments, because the resulting system of equations does not have
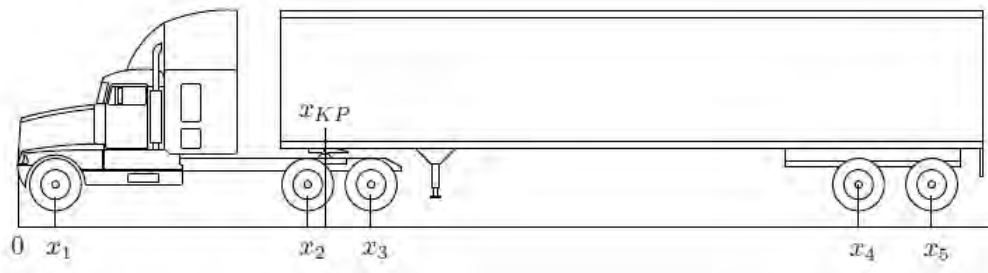
Figure B.3: Example of a truck with five axles.

a unique solution. In terms of physics, the system is *statically indeterminate* (Carpinteri, 2002) and properties of the material should be taken into account to solve the theoretical problem of calculating the axle weights. The exact solution may be found using structural analysis methods, but we will not elaborate on those.

In practice this result means that the weight that actually works on each axle depends on the stiffness/rigidity of the container. However, the relevance of the individual axle weights is only in the context of the FBL. Therefore we do not require an exact solution if we can show that, with certain assumptions, satisfaction of the FBL is equivalent to the container (the trailer) satisfying weight constraints on two (imaginary) points. In that case, the restriction can be formulated as an interval for the COG as in Section B.3.

We have a closer look at a typical truck and semi-trailer combination with 5 axles, which can be seen in Figure B.3. The truck has a steering axle and a tandem axle while the trailer only has a tandem axle. The trailer is connected to the truck at one point, which is the *kingpin*.

First we look at the trailer. If, for the moment, we assume that the tandem axle is a *bogie*[4] we may consider it as a single axle. Naturally, this assumption is invalid so we may not assume an equal weight distribution over the two axles, but for the total weight restriction over the tandem it still holds that we can regard the tandem as a single axle. According to the FBL this total weight is restricted to 34,000 lbs while each single axle may bear 20,000 lbs. That means that in practice the true weight on each axle is allowed to vary between 14,000 and 20,000 lbs. Under the assumption that the trailer is horizontal and rigid it is very unlikely that this restriction is not satisfied if the total weight on the tandem does not exceed 34,000 lbs. Therefore in practice the only restriction is the total weight on the tandem, which may then be considered as a single

---

[4]A bogie is carriage that connects a pair of axles to the vehicle (trailer) at a single point. As this point is in the middle between the axles, the weight is evenly distributed (assuming the point is a hinge) over the axles.

axle.

On the front side of the trailer we have a kingpin-connection with the truck. As this is the only place at which the trailer exerts its weight to the truck, we may again regard this as a single axle. The maximum weight that may rest on the kingpin depends on the maximum weight the truck can bear at that point while satisfying the FBL for all of its wheel groups. Assuming that we can find this maximum weight, the problem of loading the container such that the overall combination satisfies the FBL is equivalent to loading the container such that the maximum weights on the kingpin and the tandem axle are satisfied. In that case we can find a range for the COG as in section B.3.

### B.4.1 Maximum weight on the kingpin

For finding the maximum weight on the kingpin, we again consider the tandem axle of the truck as a single axle. That is, we regard axles 2 and 3 as a single axle located at $\frac{x_2+x_3}{2}$. Let $w_{\text{KP}}$ be the weight on the kingpin. We will use a calculation similar to the one in section B.3 to derive how this weight is distributed over the axles. As the truck cabin itself also has a weight we define separately $w_{\text{KP},i}, i = 1, 2, 3$ as the weight of the kingpin that is distributed to axle $i$. Let $w_{\text{C},i}, i = 1, 2, 3$ be the weight distribution of the weight $w_{\text{C}}$ of the truck (cabin) without a trailer over its axles so that the total weight for each axle is $w_i = w_{\text{C},i} + w_{\text{KP},i}, i = 1, 2, 3$. For ease of notation we write $w_{2+3} = w_2 + w_3$ for the weight on the tandem axle, with similar notations for the components C and KP. Considering axle 1 as a lever, we find

$$(x_{\text{KP}} - x_1)w_{\text{KP}} = \left( \frac{x_2 + x_3}{2} - x_1 \right) w_{\text{KP},2+3} \tag{B.6}$$

and similarly

$$\left( \frac{x_2 + x_3}{2} - x_{\text{KP}} \right) w_{\text{KP}} = \left( \frac{x_2 + x_3}{2} - x_1 \right) w_{\text{KP},1} \tag{B.7}$$

from which we can derive

$$w_1 = w_{\text{C},1} + w_{\text{KP},1} = w_{\text{C},1} + \frac{\frac{x_2+x_3}{2} - x_{\text{KP}}}{\frac{x_2+x_3}{2} - x_1} w_{\text{KP}} \tag{B.8}$$

and

$$w_{2+3} = w_{\text{C},2+3} + w_{\text{KP},2+3} = w_{\text{C},2+3} + \frac{x_{\text{KP}} - x_1}{\frac{x_2+x_3}{2} - x_1} w_{\text{KP}}. \tag{B.9}$$

If the trailer (with load) has as a total weight $w_{\text{TRL}}$, then the weight on the tandem axle 4 and 5 is

$$w_{4+5} = w_{\text{TRL}} - w_{\text{KP}}. \tag{B.10}$$

Next we apply the FBL to derive restrictions on $w_{\mathrm{KP}}$. Remember that for a single axle the limit is 20,000 lbs, for a tandem 34,000 and for three or more axles Equation (B.1) holds, while the gross weight of the combination ($w_{\mathrm{C}} + w_{\mathrm{TRL}}$) is restricted to 80,000 lbs. The derivation and the resulting restrictions from applying the FBL to the most important axle groups are in Table B.1. The first column gives the group of axles, the second gives the maximum weight allowed for that group according to the FBL, the third gives the mathematical restriction on the axle weights that follows from this and the last column gives the restriction for the weight on the kingpin. Note that not all axle combinations are worked out, as satisfaction of the combinations worked out will mean in practice that the restrictions for the other groups are satisfied as well[5].

We can combine the restrictions on the kingpin to find an interval in which $w_{\mathrm{KP}}$ must be in order to satisfy the FBL. This interval is given in Equation (B.11).

$$
w_{\mathrm{TRL}} - 34,000 \leq W_{\mathrm{KP}} \leq \min \Big\{ \quad
\begin{aligned}
& (20,000 - w_{\mathrm{C},1}) \frac{\frac{x_2+x_3}{2} - x_1}{\frac{x_2+x_3}{2} - x_{\mathrm{KP}}}; \\
& (34,000 - w_{\mathrm{C},2+3}) \frac{\frac{x_2+x_3}{2} - x_1}{x_{\mathrm{KP}} - x_1}; \\
& 500 \left( \tfrac{3}{2}(x_3 - x_1) + 72 \right) - w_{\mathrm{C}}; \\
& \frac{\frac{x_2+x_3}{2} - x_1}{x_{\mathrm{KP}} - \frac{x_2+x_3}{2}} \left( 500 \left( \tfrac{4}{3}(x_5 - x_2) + 84 \right) - w_{\mathrm{C},2+3} - w_{\mathrm{TRL}} \right) \quad \Big\}
\end{aligned}
$$

$$(\text{B.11})$$

---

[5]http://ops.fhwa.dot.gov/freight/publications/brdg_frm_wghts/index.htm

| Axles | Max lbs | Restriction on axles | Restriction on kingpin |
|-------|---------|----------------------|------------------------|
| 1 | 20,000 | $\begin{aligned} w_1 &= w_{C,1} + \frac{\frac{x_2+x_3}{2}-x_{KP}}{\frac{x_2+x_3}{2}-x_1} w_{KP} \\ &\leq 20,000 \end{aligned}$ | $w_{KP} \leq (20,000 - w_{C,1}) \frac{\frac{x_2+x_3}{2}-x_1}{\frac{x_2+x_3}{2}-x_{KP}}$ |
| 2 - 3 | 34,000 | $\begin{aligned} w_{2+3} &= w_{C,2+3} + \frac{x_{KP}-x_1}{\frac{x_2+x_3}{2}-x_1} w_{KP} \\ &\leq 34,000 \end{aligned}$ | $w_{KP} \leq (34,000 - w_{C,2+3}) \frac{\frac{x_2+x_3}{2}-x_1}{x_{KP}-x_1}$ |
| 4 - 5 | 34,000 | $\begin{aligned} w_{4+5} &= w_{TRL} - w_{KP} \\ &\leq 34,000 \end{aligned}$ | $w_{KP} \geq w_{TRL} - 34,000$ |
| 1 - 3 | $500\left(\frac{3}{2}(x_3 - x_1) + 72\right)$ | $\begin{aligned} w_{1+2+3} &= w_C + w_{KP} \\ &\leq 500\left(\frac{3}{2}(x_3 - x_1) + 72\right) \end{aligned}$ | $w_{KP} \leq 500\left(\frac{3}{2}(x_3 - x_1) + 72\right) - W_C$ |
| 2 - 5 | $500\left(\frac{4}{3}(x_5 - x_2) + 84\right)$ | $\begin{aligned} w_{2+3+4+5} \\ = w_{C,2+3} + w_{KP,2+3} + w_{TRL} - w_{KP} \\ = w_{C,2+3} + \frac{x_{KP}-x_1}{\frac{x_2+x_3}{2}-x_1} w_{KP} + w_{TRL} - w_{KP} \\ = w_{C,2+3} + \frac{x_{KP}-\frac{x_2+x_3}{2}}{\frac{x_2+x_3}{2}-x_1} w_{KP} + w_{TRL} \\ \leq 500\left(\frac{4}{3}(x_5 - x_2) + 84\right) \end{aligned}$ | $\begin{aligned} w_{KP} &\leq \frac{\frac{x_2+x_3}{2}-x_1}{x_{KP}-\frac{x_2+x_3}{2}} \cdot \\ &\left(500\left(\frac{4}{3}(x_5 - x_2) + 84\right) - w_{C,2+3} - w_{TRL}\right) \end{aligned}$ |
| 1 - 5 | $\min\left\{500\left(\frac{5}{4}(x_5 - x_1) + 96\right); 80,000\right\}$ | $\begin{aligned} w_{1+2+3+4+5} \\ = w_C + w_{TRL} \\ \leq \min\left\{500\left(\frac{5}{4}(x_5 - x_1) + 96\right); 80,000\right\} \end{aligned}$ | - |

Table B.1: Table with restrictions on the weight on the kingpin resulting from the Federal Bridge Law.

# Appendix C

# Priorities examples

This appendix contains two examples that illustrate how mixing of priorities in trucks can be beneficial. The first example illustrates an increase in VFR and a decrease in the number of trucks, without increasing the number of trucks for each priority. The second example illustrates how an increase in VFR and decrease in the total number of trucks can be achieved as a tradeoff against an increase in the priority of (at least) one truck.

## C.1 Example 1

Table C.1 contains example 1 in which we have two priorities. For each priority, we have exactly one truckload in terms of volume, while the first priority has a weight of 125 % of a full truckload and the second priority has a weight of 75 % of a full truckload. The table shows the number of trucks per priority, with and without priority mixing. The example is illustrated by Figure C.1, where for illustration purposes we assume that a truck can hold 20 units of load and 80 tons. From the example it is clear that we achieve a better VFR because we are able to better mix weight and volume by mixing the priorities. However, this can be achieved without increasing the priority for any truck, solely by better utilising (in terms of weight) the 'leftover priority 1 space' in the second truck, which has to be priority 1 anyway. Note that the result without mixing priorities cannot be improved by backfilling!

|                                  | Priority 1 | Priority 2 | Total |
|----------------------------------|:----------:|:----------:|:-----:|
| Volume                           | 100%       | 100%       | 200%  |
| Weight                           | 125%       | 75%        | 200%  |
| # trucks without priority mixing | 2          | 1          | 3     |
| # trucks with priority mixing    | 2          | 0          | 2     |

Table C.1: Example 1, volume and weight as percentages of full trucks per priority and number of trucks with and without priority mixing.



Figure C.1: Example of how allowing priorities to mix can reduce the number of trucks without increasing the number of trucks per priority.

|  | Priority 1 | Priority 2 | Total |
|---|---|---|---|
| Volume | 100% | 100% | 200% |
| Weight | 75% | 125% | 200% |
| # trucks without priority mixing | 1 | 2 | 3 |
| # trucks with priority mixing | 2 | 0 | 2 |

Table C.2: Example 2, volume and weight as percentages of full trucks per priority and number of trucks with and without priority mixing.
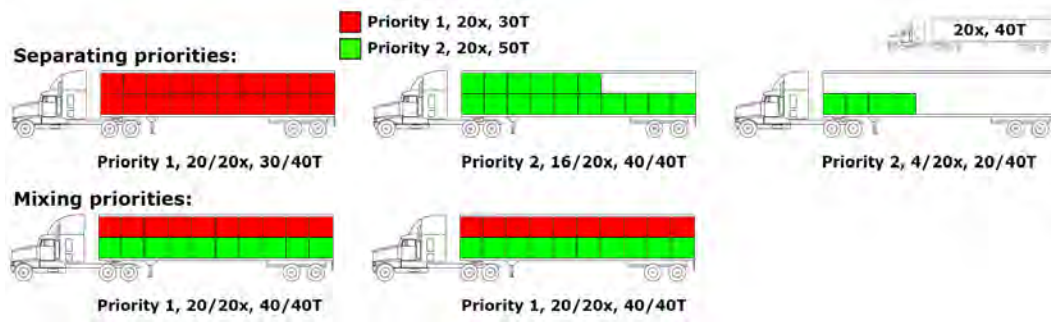


Figure C.2: Example of how allowing priorities to mix can reduce the number of trucks by increasing the priority of a truck.

## C.2    Example 2

Table C.1 shows example 2 which is basically constructed by switching the priorities, or the weights of the loads for the priorities, in example 1. Figure C.2 illustrates how the number of trucks again can be reduced, however this time at the cost of increasing the priority of the second truck. Note that the result with mixing is basically the same as in example 1, only priority 1 is illustrated on top because this is the lighter load.

# Glossary

| | |
|---|---|
| **C & P** | Cutting and Packing |
| **CLP** | Container Loading Problem |
| **COG** | Center of gravity |
| **DC** | Distribution center |
| **ERP** | Enterprise Resource Planning |
| **FBL** | Federal Bridge Law |
| **GAP** | Generalized Assignment Problem |
| **ILP** | Integer Linear Program |
| **KCLP** | Knapsack Container Loading Problem |
| **KPI** | Key Performance Indicator |
| **LP** | Linear Program |
| **MCGAP** | Multi Constraint Generalized Assignment Problem |
| **MIP** | Mixed Integer Program |
| **OPLB** | ORTEC Pallet and Load Building |
| **PGA** | Pallet Grid Assignment |
| **PO** | Purchase order |
| **SCP** | Set Covering Problem |
| **SPP** | Set Partitioning Problem |
| **VFR** | Vehicle fill rate |
| **WMS** | Warehouse Management System |

# Bibliography

S. V. Amiouny, J. J. Bartholdi III, J. H. Vande Vate, and J. Zhang. Balanced loading. *Operations Research*, 40(2):238–246, 1992.

E. Balas and M. W. Padberg. Set partitioning: A survey. *SIAM review*, 18(4):710–760, 1976.

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

E. E. Bischoff and M. Ratcliff. Issues in the development of approaches to container loading. *Omega*, 23(4):377–390, 1995.

I. Brosh. Optimal cargo allocation on board a plane: a sequential linear programming approach. *European Journal of Operational Research*, 8(1):40–46, 1981.

G. Caron, P. Hansen, and B. Jaumard. The assignment problem with seniority and job priority constraints. *Operations Research*, 47(3):449–453, 1999.

A. Carpinteri. *Structural mechanics: a unified approach*. CRC Press, 2002.

D. G. Cattrysse and L. N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260–272, 1992.

G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

A. P. Davies and E. E. Bischoff. Weight distribution considerations in container loading. *European Journal of Operational Research*, 114(3):509–527, 1999.

H. Dyckhoff. A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159, 1990.

W. L. Eastman, S. Even, and I. Isaacs. Bounds for the optimal scheduling of n jobs on m processors. *Management science*, 11(2):268–279, 1964.

L. Foulds and J. M. Wilson. On an assignment problem with side constraints. *Computers & industrial engineering*, 37(4):847–858, 1999.

B. Gavish and H. Pirkul. Algorithms for the multi-resource generalized assignment problem. *Management Science*, 37(6):695–713, 1991.

H. Gehring, K. Menschner, and M. Meyer. A computer-based heuristic for packing pooled shipment containers. *European Journal of Operational Research*, 44(2):277–288, 1990.

E. S. Gottlieb and M. Rao. The generalized assignment problem: Valid inequalities and facets. *Mathematical Programming*, 46(1-3):31–52, 1990.

Å. Hallefjord, K. O. Jörnsten, and P. Värbrand. Solving large scale generalized assignment problemsan aggregation/disaggregation approach. *European Journal of Operational Research*, 64(1):103–114, 1993.

M. Hifi, I. Kacem, S. Nègre, and L. Wu. A linear programming approach for the three-dimensional bin-packing problem. *Electronic Notes in Discrete Mathematics*, 36:993–1000, 2010.

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

D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3(4):299–325, 1974.

W. Kool. Compimentary material to ortec pallet and load building. Master's thesis, VU University, Amsterdam, 2014.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations research*, 14 (4):699–719, 1966.

M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

J. B. Mazzola and A. W. Neebe. Resource-constrained assignment scheduling. *Operations Research*, 34(4):560–572, 1986.

D. W. Pentico. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793, 2007.

D. Pisinger. Heuristics for the container loading problem. *European journal of operational research*, 141(2):382–392, 2002.

M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45(6):831–841, 1997.

T. van Dijk. Tuning the parameters of a loading algorithm. Master's thesis, University of Twente, 2014.

P. H. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser. Solving binary cutting stock problems by column generation and branch-and-bound. *Computational optimization and applications*, 3(2):111–130, 1994.

F. Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Mathematical Programming*, 86(3):565–594, 1999.

A. Volgenant. A note on the assignment problem with seniority and job priority constraints. *European journal of operational research*, 154(1):330–335, 2004.

G. Wäscher. An lp-based approach to cutting stock problems with multiple objectives. *European Journal of Operational Research*, 44(2):175–184, 1990.

G. Wäscher, H. Haußner, and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130, 2007.