Solving a highly constrained multi-level container loading problem from practice

Division of Optimization, Department of Mathematics, Linköping University

Jonas Olsson

Bachelor Thesis: 16 hp
Level: G2
LiTH-MAT-EX--2017/01--SE

Supervisor: Nils-Hassan Quttineh
Examiner: Torbjörn Larsson
Linköping, February 2017

Abstract

The container loading problem considered in this thesis is to determine placements of a set of packages within one or multiple shipping containers. Smaller packages are consolidated on pallets prior to being loaded in the shipping containers together with larger packages. There are multiple objectives which may be summarized as fitting all the packages while achieving good stability of the cargo as well as the shipping containers themselves.

According to recent literature reviews, previous research in the field have to large extent been neglecting issues relevant in practice. Our real-world application was developed for the industrial company Atlas Copco to be used for sea container shipments at their Distribution Center (DC) in Texas, USA. Hence all applicable practical constraints faced by the DC operators had to be treated properly. A high variety in sizes, weights and other attributes such as stackability among packages added complexity to an already challenging combinatorial problem.

Inspired by how the DC operators plan and perform loading manually, the *batch concept* was developed, which refers to grouping of boxes based on their characteristics and solving subproblems in terms of partial load plans. In each batch, an extensive placement heuristic and a load plan evaluation run iteratively, guided by a Genetic Algorithm (GA). In the placement heuristic, potential placements are evaluated using a scoring function considering aspects of the current situation, such as space utilization, horizontal support and heavier boxes closer to the floor. The scoring function is weighted by coefficients corresponding to the chromosomes of an individual in the GA population. Consequently, the fitness value of an individual in the GA population is the rating of a load plan.

The loading optimization software has been tested and successfully implemented at the DC in Texas. The software has been proven capable of generating satisfactory load plans within acceptable computation times, which has resulted in reduced uncertainty and labor usage in the loading process. Analysis using real sea container shipments shows that the GA is able to tune the scoring coefficients to suit the particular problem instance being solved.

Keywords: container loading; 3D packing; real-world application; heuristic; genetic algorithm

Acknowledgements

Several people have made valuable contributions to this project and some of them are acknowledged below. My sincere gratitude is also forwarded to everyone else at Atlas Copco that have provided support in any way. The project has been truly exciting and an incredible learning experience for me.

First of all I would like to thank Thomas Dahlgren for connecting me with the DC. This project would not have happened without your favorable introduction of me to management at the DC. Second, I would like to thank Ian Hale for giving me the opportunity to join the business analysis team at the DC for an internship the summer of 2015, and for initiating this project. Your articulation of the business need was the seed which led to the development of the software.

Further, I would like to devote enormous gratitude to Steven Vogel and Andrew Kirchner, who commonly are referred to as the *DC operators* throughout the thesis. Your support, feedback and help in testing the software in practice has without doubt been crucial for this project. Your tireless attitude and elaborate responses to my questions and ideas have been vital for the development of the algorithms in the software. I would particularly like to highlight Steven's feedback in terms of specific comments on flawed or undesirable suggestions of placements in load plans, which has driven significant improvements in the algorithms.

Moreover, I would like to thank Tinto Skaria and Matthew Scott for continuous help related to the enterprise software and Microsoft Access. Your assistance has been crucial for making the software operational within a very limited time frame. I would also like to thank Steve Bialas for very important help in the deployment phase, including end-user testing. Further, I would like to express sincere gratitude to Juan Cordova and Michael Moriarty, who commonly are referred to as the *customer service representatives* throughout the thesis. Your feedback on the end-user experience and positive attitude to try and start using the software in practice is much appreciated.

Second to last, I would like to devote immense gratitude to William Switzer for incredible mentorship, sponsorship and hospitality. Your mentorship in terms of advice and encouragement have not only helped me in this project, but also developed me a lot on both a professional and a personal level. I am so thankful for your sponsorship in terms of promoting the project and advocating for my return the summer of 2016. The project involved much ambiguity and many stakeholders to persuade, and your guidance combined with trust in me has been key to what we have accomplished.

Finally, I would like to thank Nils-Hassan Quttineh and Torbjörn Larsson at Linköping University for great mentorship in the work with the thesis. Your guidance have been essential to present the project in a comprehensible manner and it has enhanced my understanding of the concepts used, as well as developed awareness of potential improvements in future work. Your engagement in the project, as well as tutoring interesting courses in operations research at Linköping University, have truly been a great source of inspiration.

To all mentioned above, and everyone else that have supported the project in any way: thank you!

Jonas Olsson

Contents

1 Introduction

Atlas Copco and the Mining and Rock Excavations Service division are introduced in Section 1.1. The Distribution Center in Texas, USA, is introduced in Section 1.2. The former process for loading of sea containers is described in Section 1.3, which leads to the development of the optimization software announced in Section 1.4. The following chapters of the thesis are outlined in Section 1.5.

1.1 Atlas Copco and MRS division

Atlas Copco is a Swedish industrial company providing sustainable productivity solutions to businesses across in a wide range of industries. The company was founded in 1873 and has today over SEK 100 billion in revenue and more than 43 000 employees globally. To reflect the variety of industries, the company is divided in five business areas. Mining and Rock Excavation Technique (MR) is the second largest business area and provides equipment for drilling and rock excavation, including rock drills as well as related consumables and services.

Figure 1. A surface drilling rig - one of numerous products offered by Atlas Copco Mining and Rock Excavation Technique. Source: Mining & Construction (2013)

Mining and Rock Excavations Service (MRS) is one of six divisions within the MR business area. In order to supply spare parts to mining and construction companies across the globe, MRS has three Distribution Centers (DCs) located in Örebro (Sweden), Allen (Texas, USA) and Nanjing (China).

1.2 Distribution Center in Allen, Texas, USA

The DC in Allen, a northern suburb of Dallas, holds inventory and ships spare parts to a large number of customers using several modes of transport. The DC has about 15 employees in the office and about 70 employees in the warehouse. Their customers include end-customers as well as customer centers (CCs) which are Atlas Copco sales offices located closer to end-customers. Air-freight is their most common mode of transport for customers located overseas, followed by sea container shipping which is slower but less expensive for larger volume shipments. The DC currently ships up to 20 trucks per day (many of which go to an airport) and up to 5 sea containers per week. The value of all packages in one sea container may vary from below USD 50,000 to above USD 250,000.

1.3 Loading of sea containers

For some overseas customers packages for multiple orders are accumulated over a period of time and shipped all at once in one or two 20 ft. or 40 ft. intermodal containers. The containers are ordered as soon as the predicted volume utilization is high enough, or if the customer requests their orders to be shipped immediately.

Prior to the implementation of the solution presented in this thesis, the task of predicting volume utilization and planning how to load the containers was entirely manual. Due to significant variety in sizes, weights and other attributes such as stackability among packages, these activities were often too complicated to do without actually manually attempting to perform the loading. Hence, the loading process for sea containers included an activity called "pre-staging", which meant performing a test load on the warehouse floor. This was necessary to predict if the container would be full enough, and also to ensure that all packages would fit in the container.

Figure 2. Completed pre-staging of a 40 ft. sea container

Pre-staging exercises were performed repeatedly by the DC operators before giving the proper customer service representative a "go ahead" to order the containers. The containers would then show up at the DC, and the DC operators would usually have only one hour available to load the container before departure.

1.4 Loading optimization software

With the ambition to reduce uncertainty as well as labor usage in the shipping process, DC management desired a solution that would support more efficient activities. More specifically, management wanted to eliminate delays in ordering containers while also being confident about fitting all packages in the number of containers ordered, with as little excess capacity as possible. Due to the complexity and diversity between shipments, the possibility to simply use historical metrics of volume utilization was quickly ruled out. It was concluded that the loading would have to be predetermined rather than predicted, and these discussions led to the development of container loading optimization software. It was determined that an "off-the-shelf" solution would likely require

too much tailoring to fit the unique business needs, so it was decided that an in-house loading optimization software should be developed.

The software ultimately became a decision support tool in the loading process that fully eliminated the labor associated with pre-staging. The customer service representative was able to run the software repeatedly as new orders were received, to know quickly when containers should be ordered. By removing their dependence on "go ahead" from DC operators, potential delays were indeed reduced. The ambition was to be able to load containers right away based upon load plans generated by the software with only minor adjustments by the DC operators upon loading. Acceptable adjustments includes minor changes of placements as well as the use of bands, straps and airbags to secure the cargo.

Throughout the development and deployment of the software, many different benefits were identified by various stakeholders. To improve volume utilization of the containers was not the main reason for the endeavor. The expectation with the software was only to achieve utilization as good as without the software. The five benefits listed below to improve processes and reduce costs were recognized by management at the DC:

1. Reduce labor usage by eliminating pre-staging (test loading) activities
2. Reduce lead time by eliminating delays in ordering containers
3. Reduce damage during transport by having DC operators focus more on securing cargo rather than making all cargo fit in containers
4. Improve customer service by providing visual load manifests to identify exactly where an item is located within a large, densely packed, container
5. Improve safety by ensuring stability and capacity of containers

This thesis presents the design of the software with an emphasis on the mathematics and algorithm design, but a brief chapter is also dedicated to the implementation of the software in practice.

1.5 Outline of following chapters

In Chapter 2 the container loading problem considered in this thesis is defined. The input to the problem and the output from the problem are elaborated upon. In Chapter 3 the literature on container loading is studied, preceded by a brief review of commercial optimization software. Assessments of our problem types and constraints applicable to our problem facilitate search for promising concepts in past publications. In Chapter 4 the proposed method for solving our container loading problem is thoroughly explained. First, the total approach is outlined. Second, the major parts of the overall algorithm are treated in further detail. In Chapter 5 the implementation of the software in practice is treated, focused on the user interfaces. In Chapter 6 results and analysis are presented. Real shipments are used to analyze the performance of the proposed method, which include relating computational experiments to actual loadings. In Chapter 7 conclusions are presented, confirming both that the approach has been proven successful and that the purpose of the software was fulfilled.

2 Problem definition

The container loading problem considered in this thesis is defined in Section 2.1. The input to the problem in terms of packages to be loaded, available shipping container types and available consolidation pallet types are described in more detail in Section 2.2. Goals for loading are outlined, followed by an initial itemization of objectives, in Section 2.3. The purpose of the optimization software linked to the output from the problem, i.e. a solution, is elaborated upon in Section 2.4.

2.1 Overview of problem

The container loading problem considered in this thesis can be described as determining *placements* of a set of *packages* within one or multiple *shipping containers*. This can also be phrased as generating a *load plan*. It is a multi-level loading problem since smaller packages are consolidated on pallets (commonly called *consolidation pallets*) prior to being loaded in the shipping containers, together with larger packages. A package may contain one or multiple items (spare parts) depending on what the customer has ordered, when the items are packed, as well as the characteristics of the items. The packing and loading hierarchy is illustrated in Figure 3, and is explained further below.

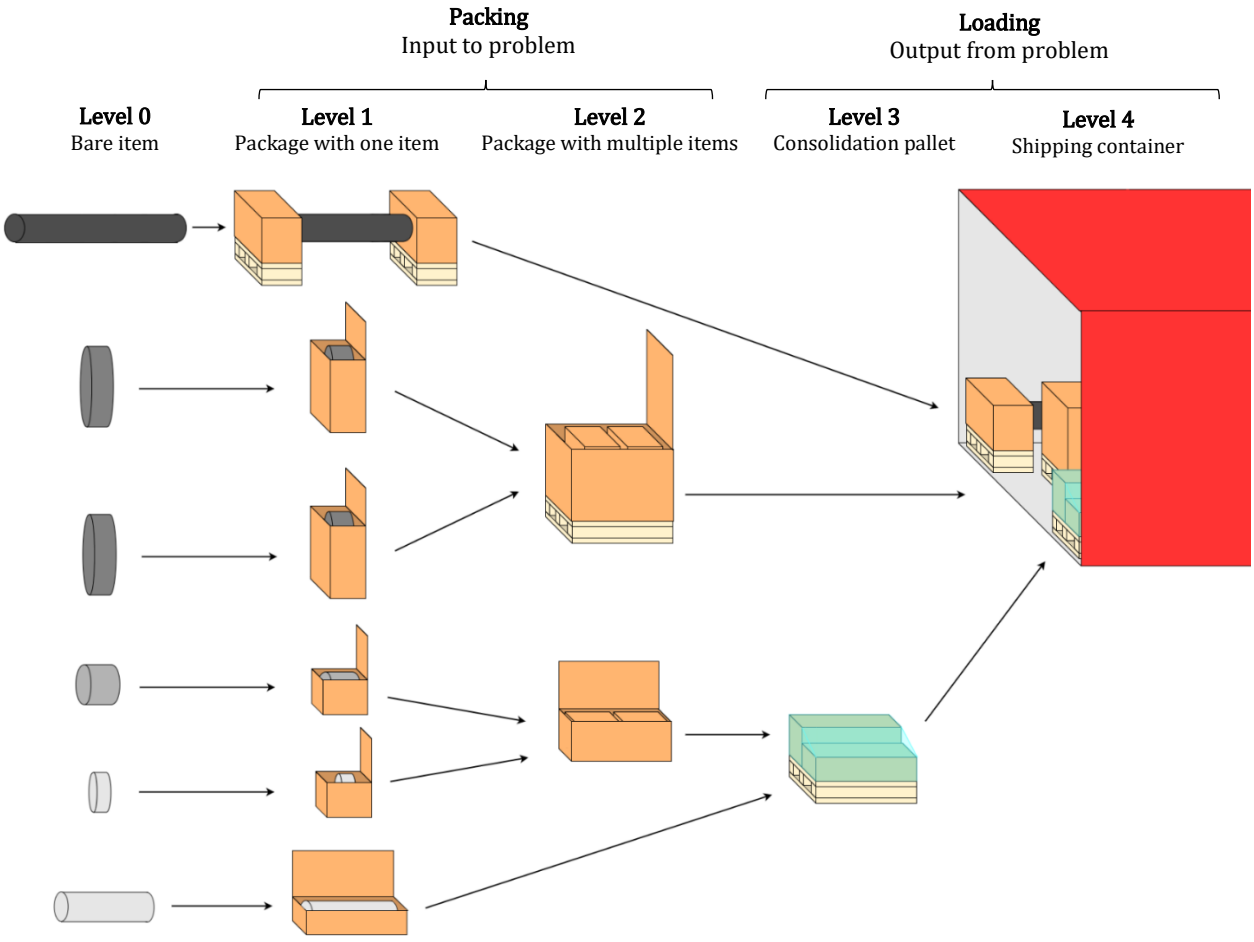

Figure 3. Entire packing and loading hierarchy

Items are received by the DC from the supplier at either at level 0 or level 1 as illustrated above in Figure 3. Items received at level 0 are packed individually upon receiving, to ensure that items are at level 1 when put into stock. There are however some exceptions for special items that are still at level 0 when put into stock, for example very small items (e.g. nuts and bolts) which are not ordered in standard quantities by customers.

Smaller level 1 packages are packed together with other (similar or different) items for the same customer, resulting in level 2 packages. After this packing, level 2 packages are moved to shipping locations in the warehouse. Any level 1 packages too large or too heavy to be considered for level 2 are moved directly to these shipping locations. The level 1 and level 2 packages available at the shipping locations serve as input to the multi-level loading problem. At level 3, smaller packages are consolidated on consolidation pallets prior to being loaded into a shipping container.

The reason why both level 2 and level 3 exist is that packages for a given customer sometimes are accumulated over a period of time before being loaded in a container. This means that packages that end up in the same container might be packed several different days, sometimes even during different weeks. Therefore packages that could have been consolidated at level 2 during packing are instead consolidated at level 3 during loading. At level 4, packages from level 1 and level 2 as well as consolidation pallets are loaded into the shipping container. Both level 3 and level 4 is considered to be loading and hence considered as the output from the multi-level loading problem.

In total, the input consists of three components: packages to be loaded, available shipping container types and available consolidation pallet types. Since it is only provided which types are available, the number of containers as well as the number of pallets to use must also be selected for each type. Therefore the total problem can be expressed as an integrated container selection and loading problem.

When selecting containers and pallets, the objective is to minimize volume capacity or cost. When loading, there are multiple objectives, which can be summarized as: fitting all packages while achieving good stability of the cargo as well as the shipping containers themselves. These loading objectives are constrained by many practical considerations. A solution is called a *load plan*, and consists: which shipping containers and consolidation pallets to use, and placements (position and orientation) of the packages within these.

Prior to the development and deployment of the software presented in this thesis, load plans were generated manually through test loading exercises called pre-staging as described in Section 1.3. The intent with the loading optimization software was to replace the manual work associated with pre-staging. The following sections of this chapter describe in more detail the conditions and goals for loading, as well as what the load plans generated by the software consist of and how they are used by the DC operators.

2.2 Conditions for loading

In this section the three components of input to the problem are described in more detail. These are packages to be loaded, available shipping container types and available consolidation pallet types.

2.2.1 Packages

The first component of the input is the cargo to be loaded. This typically entails packages for multiple orders from the same customers, to be shipped to a single destination overseas. A package may contain one or multiple items. The attributes of packages are therefore partly inherited from the

content, and partly based on the packaging material used to accommodate the content. For multiple reasons, the number of packages in a container may vary significantly, from less than 10 to over 100. Most shipments however consists of 20-60 packages.

One reason for the differences in number of packages is that volume utilization of containers may be sacrificed if the customer desires shipment prior to a full container accumulating. Another reason is due to a large variety in sizes, weights and other attributes such as stackability among packages. The variety among packages is partly a result of vast differences among items, since the DC distributes all kinds of spare parts for a drilling rig. Figure 4 exemplifies how characteristics of packages may vary.

Figure 4. Packages with a large variety of sizes, weights and other attributes

Packaging options

When level 1 packages are packed together with other (similar or different) parts resulting in level 2 packages, there are 25 packaging options available for the DC operators, as listed in Table 15 in Section 9.1. The high number of options enables packing items of all different shapes, sizes and weights without too much excess space being generated as a result. In the current packing process, the DC operator makes the decision of which packing options to use for the items that have been assigned to the operator. Typically, the weight and the size of the items limits the number of rational options significantly, but still leaves several reasonable options to choose from based upon individual judgment and preference. One important parameter in making the choice is whether the packaging has a pallet attached, which is required for packages over 80 lbs. (36 kg). 10 of all 25 packaging options have a pallet attached.

The 25 packaging options do not however limit the packages to 25 standard sizes, since 5 of the 25 options are "non-standard" options. These 5 non-standard options allow for packaging to be customized to better accommodate one or multiple parts. The customization may be cutting down the height of a standard size box, or increasing the height of a "telescopic" box. 5 of the standard options have telescopic possibility. For example, if the height of a standard "GREEN SHORT-45x30x21-Telescope" would be increased, the packaging would be changed to a non-standard "Pallet". Further regarding customization of packages, the non-standard option "CRATE" is for bulky parts and means to build customized crates of wood from scratch, which is the case in Figures 7-10. These non-standard packaging options create an infinite amount of possibilities in size and weight, and hence is another reason leading to the variety among packages.

Sample of packages

To demonstrate the variety among packages, a set of 661 packages from 20 complete sea container shipments to four different customers has been compiled. These 20 shipments are presented in more detail in Section 6.1. The average number of packages per shipment was 33, with 11 packages at lowest and 64 packages at highest. The average number of packages per size, i.e. unique combination of length, width and height, per shipment was 1.9, with 1.2 at lowest and 2.5 at highest. In total there were 178 sizes of packages, hence an average of 3.7 packages per size over all shipments, with 1 at lowest and 53 at highest. Further, 308 of the 661 packages (47%) had non-standard packaging.

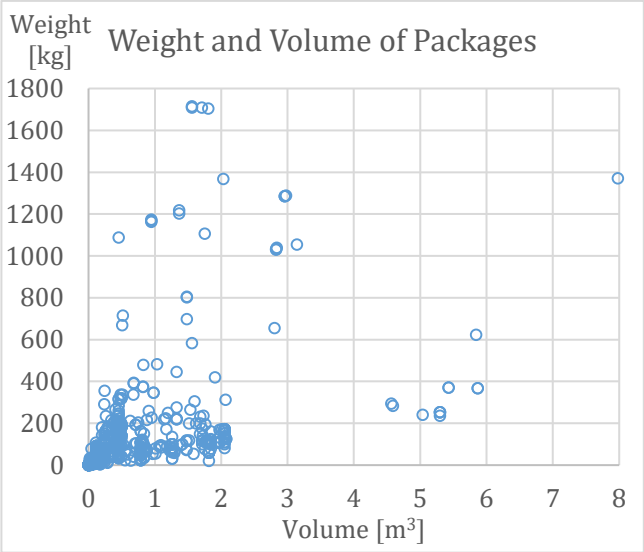

Figure 5. Weight and volume for sample of 661 packages

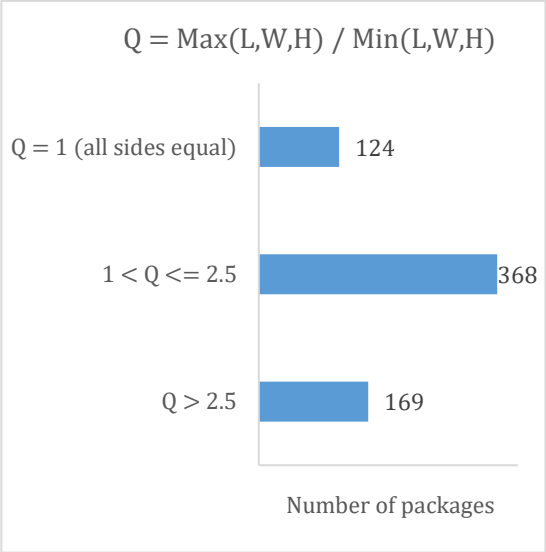

Figure 6. Ratio between largest and smallest dimension (Length, Width and Height) for sample of 661 packages

Considering weight and volume as in Figure 5, 462 of the packages (70%) are less than 200 kg and 1 m^3. If doubling these upper bounds to 400 kg and 2 m^3, then 595 of the packages (90%) are within the limits. As can be seen in Figure 5, many of the outliers in the sample are far outside even the latter limits, with the highest weight at 1714 kg and the highest volume at 7.980 m^3.

Figure 6 is an attempt to illustrate the differences in shapes among packages. Q is the ratio between the largest and smallest dimension of a package's outer length, width and height. Given the definition, the smallest possible value of Q is 1, and occurs if all sides are equal. This was the case for 124 of the packages (19%). Packages with Q greater than some threshold could be considered as long or slim, dependent upon the third dimension (the one that is neither maximum nor minimum). 169 of the packages (26%) had a Q value greater than 2.5, and the three most significant outliers had Q values of 25.4, 19.3 and 19.2, respectively. The use of Q to describe the shape of a package is just to illustrate a concept, and not necessarily terminology recognized by the employees at the DC.

Other attributes for packages

In addition to the dimensions and weights there are other attributes for packages that have to be considered for. Edge Crush Test (ECT) value and thickness of a package's walls enable stacking strength calculations using the McKee Formula (see Example 3 in Section 4.4.2). If a package has hazardous content, it must be easy accessible, which may be interpreted as no other packages placed above or in front. If a pallet is attached to a package, the orientation of the pallet runners are relevant

for two reasons: First because a forklift is used when loading, and second because packages without a pallet should preferably not be placed on the container floor due to the risk of moisture. This is due to the fact that the containers travel by sea.

All attributes mentioned thus far are inherited from either the packaging or the items within the package. The DC operators have the possibility to override some of these inherited attributes, as well as a few other assumptions by submitting *exceptions* for packages. Exceptions can be submitted to change the inherited pallet orientation, as well as reduce or increase the inherited stacking strength for packages which are fragile or extra strong. Exceptions can also be submitted to reject the assumption that a package can be stacked on other packages (instead of being placed on the floor) as well as the opposite, namely that other packages can be stacked on the package. Figures 7-10 are examples of such exceptions. The number of packages for which exceptions are applicable also vary greatly from shipment to shipment, from zero packages with exceptions to exceptions for almost all packages. Exceptions for the 20 shipments treated above are presented in Table 13 in Section 6.1.

Figure 7. Ok to stack similar, but not different, package above

Figure 8. Not ok to stack any package above, item inside is glass

Figure 9. Only to be placed on floor due to weight

Figure 10. Only to be placed on floor due to fluid, which is inherently prone to leaking

2.2.2 Shipping containers

The second component of the input is the container types which can be used to ship the packages as sea freight. When selecting which specific shipping containers to use, the objective is to minimize volume capacity or cost. Typically, one or two containers are enough to accommodate the packages to be shipped. Most commonly, only one container at a time is considered.

Which container types to choose from depends on infrastructure restrictions on the route from the DC to the customer's site as well as the availability of the container itself from the container supplier. For example, there are customers for which 20 ft. containers are the only option, while for other customers it is possible to choose between using 20 ft. or 40 ft. containers.

Sometimes very long packages also limit the choice of container. For example, a package which exceeds the length of a 20 ft. container may lead to the need for a 40 ft. container, even though this results in a lower volume utilization. If, however, a 20 ft. container is the only option due to infrastructure restrictions, the long part could be shipped using a different mode of transport, for example air freight and then by truck from the airport. Whenever containers as sea freight are used, they will also be complemented by truck transportation to and from the harbor.

In total, there are four intermodal container sizes, and in addition to their own internal dimensions, the dimensions of the door as well as the weight capacity are relevant. In practice it is also necessary to account for the fact that the container ceiling or floor may be dented, thereby reducing the amount of available space slightly.

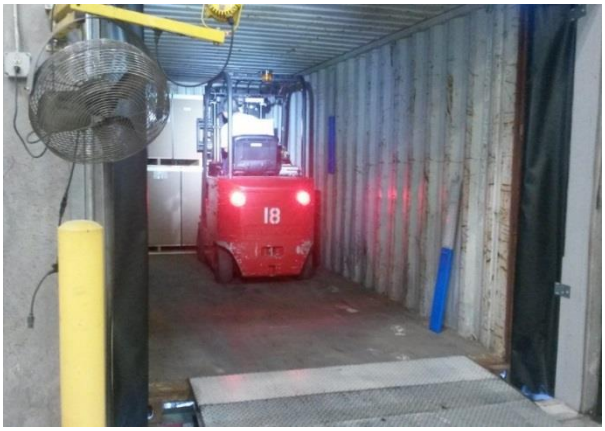

Figure 11. A 20 ft. container being loaded from the dock at the DC

Figure 12. 40 ft. containers.
Source: Wikimedia Commons (2007)

Figure 13. A container ship - this one is operated by Maersk Line
Source: Wikimedia Commons: Buonasera (2014)

Table 1. Shipping container data including dimensions and weight capacities

Container	Weight Capacity [kg]	Inner Length [m]	Inner Width [m]	Inner Height [m]	Inner Volume [m^3]	Door Width [m]	Door Height [m]
20' container	28200	5.867	2.352	2.385	32.911	2.343	2.280
40' container	26600	12.032	2.352	2.385	67.494	2.343	2.280
40' high-cube container	26580	12.000	2.311	2.650	73.490	2.280	2.560
45' high-cube container	25600	13.556	2.352	2.698	86.022	2.343	2.585

2.2.3 Consolidation pallets

The third component of the input is the types of consolidation pallet that can be used to consolidate multiple smaller packages, prior to being loaded into the container together with larger packages. A consolidation pallet can be either an open box on a pallet or an empty pallet wrapped in plastic to prevent the packages form sliding. When selecting which consolidation pallets to use, the goal is to minimize volume capacity. Typically one to three consolidation pallets are enough to accommodate the smaller packages for one container. Most commonly, one consolidation pallet is sufficient. When the number of small packages is equal or close to zero no consolidation pallet is needed. Which types to choose from depend upon what packaging material is available in the warehouse, as well as preference of open box versus plastic wrapping.

In total, there are four consolidation pallet types, two are open boxes on pallets and two are empty pallets. In the case of multiple consolidation pallets, it is typical that all are of the same type for convenience, however this is not absolutely necessary. In Table 2, the open boxes are the ones ending with “bottom” and the empty pallets the ones ending with “SW”. Both the open boxes and the empty pallets have an outer and inner height specified, however in practice the heights are not necessarily fixed. In the open box case, the walls may be cut down if needed to fit heightwise. In the empty pallet case, there are no walls and the height provided in Table 2 is based upon a “rule of thumb” at the DC in which stacks on pallets should not exceed five feet in height.

Figure 14. An open box on a pallet used as a consolidation pallet

Table 2. Consolidation pallet data including dimensions, weight capacities and tare weights

Consolidation Pallet	Weight Capacity [kg]	Tare Weight [kg]	Outer Length [m]	Outer Width [m]	Outer Height [m]	Inner Length [m]	Inner Width [m]	Inner Height [m]	Inner Volume [m^3]
GREEN-45x30x38-bottom	1329	19	1.143	0.762	0.965	1.113	0.732	0.935	0.762
YELLOW-45X45X38-bottom	1176	25	1.143	1.143	0.965	1.113	1.113	0.935	1.158
EUR-pallet-47X32X6-SW	1000	25	1.200	0.800	1.524	1.200	0.800	1.524	1.463
US-pallet-48X40X6-SW	1000	25	1.219	1.016	1.524	1.219	1.016	1.524	1.887

2.3 Goals for loading

For a given selection of shipping containers and consolidation pallets, the goals for loading can be categorized as follows:

Goal 1) Fit all packages into the shipping containers
Goal 2) Achieve stability of the cargo, i.e. prevent sliding and falling
Goal 3) Achieve stability of the containers, i.e. center of gravity close to middle and bottom
Goal 4) Avoid awkward placements, e.g. orientation of package opposite to orientation of pallet runners, package with pallet not reachable by forklift, and package without pallet on floor

These goals are elaborated upon in Section 2.3.1 and an initial itemization of objectives is developed in Section 2.3.2.

2.3.1 Outline of goals

Goal 1 is simply to fit all packages in the selected shipping containers. If only one or a few packages cannot fit with a given selection of shipping containers, there are possible alternatives presented. One alternative is to increase the container capacity by selecting a different combination of containers. Another alternative is to ship the packages that did not fit by using another mode of transport such as air freight combined with trucks. A third option is to leave the leftover packages behind, and wait for more packages to accumulate for the next shipment at a later point in time.

Goal 2 is to reduce the risk of damage during transport caused by packages sliding and falling. Adequate stability of cargo can be achieved by using consolidation pallets for smaller packages, as well as having packages and consolidation pallets supported on the sides from either the wall of the container, or other cargo. This reduces the need for securing packages using bands, straps and airbags, which saves time as well as cost of materials spent on loading.

Figure 15. Securing packages from sliding by using bands

Goal 3 is for transportation safety purposes, reducing the risk of accidents while the container is on the route from the DC to the customer's site. For example, a heavy load that is distributed unevenly could impact the driving of the truck transporting the container to the harbor. A potential worst case scenario in this regard is a container with almost all weight on one side widthwise causes a large truck to flip over when driving through a sharp curve.

Goal 4 is similar to Goal 2, since the impact of not achieving this is more time and material being spent on loading. If the orientation of a package is opposite the orientation of the pallet runners, the DC operator would potentially have to get off the forklift and rotate the package by hand. If a package with a pallet cannot be reached by the forks of the forklift, the DC operator would potentially have to get off the forklift and place the package by hand. Another potential option for these two scenarios, if the package is to be placed on top of a previous package, is to place the latter package on top of the former package outside the shipping container with more leeway, and then lift both packages into the shipping container together. Further, if a package without a pallet has to be placed on the floor, and the risk for moisture is considered to be significant, the DC operator would potentially have to find some material to place below for protection.

Goals 1 through 3 are considered to be more important than Goal 4. In practice, Goal 1 will be the main concern, whereas Goal 2 and 3 are of secondary concern. It is up to the DC operator to judge whether sufficient stability of the cargo and container is achieved, which typically translates into no known risk for damage of the cargo or accident during transport. Goal 4 would typically be sacrificed to achieve all other goals. The goals are constrained by many practical considerations, many of which are related to the attributes of packages described in Section 2.2.1. There are also other considerations in addition to these, for example sufficient support below a package. Constraints are further treated in Section 3.4.

2.3.2 An initial itemization of objectives

The following itemization of objectives can be made based on the goals in Section 2.3.1:

Goal 1) Fit all packages into the shipping containers

- 1.1 Number of leftover boxes lower
- 1.2 Volume of leftover boxes lower
- 1.3 Weight of leftover boxes lower

Goal 2) Achieve stability of the cargo, i.e. prevent sliding and falling

- 2.1 Use consolidation pallets for smaller packages
- 2.2 All sides of packages and consolidation pallets supported

Goal 3) Achieve stability of the containers, i.e. center of gravity close to middle and bottom

- 3.1 Center of gravity closer to center lengthwise
- 3.2 Center of gravity closer to center widthwise
- 3.3 Center of gravity closer to floor

Goal 4) Avoid awkward placements, e.g. orientation of package opposite to orientation of pallet runners, package with pallet not reachable by forklift, and package without pallet on floor

- 4.1 Packages with pallet not rotated
- 4.2 Packages with pallet reachable by forklift
- 4.3 Packages without pallet not on floor

These objectives are further treated in Chapter 4.

2.4 Purpose of software

A solution to the multi-level loading problem is called a load plan. A load plan consists of which shipping containers and consolidation pallets to use, and placements (position and orientation) of the packages within these. Prior to the development and deployment of the software presented in this thesis, load plans were generated (but not necessarily documented in detail) manually through test loading exercises called pre-staging as described in Section 1.3. The ambition to reduce uncertainty as well as labor usage in the shipping process led to the idea of a software that would eliminate the need for pre-staging. If the software could generate satisfactory load plans and be used by the DC operators in practice it would completely replace the pre-staging exercise. Hence the purpose of the software came to be replacing pre-staging.

The forklifts used by the DC operators are equipped with small laptops as seen in Figure 16. These laptops were identified as a key element to enable the DC operators to use load plans generated by the software. The laptops would be able to display visual step-by-step instructions, suggesting the placement of each package. By having these visual instructions in the forklift, the DC operators would be able to load containers immediately without pre-staging. Also, the customer service representative would be able to run the software repeatedly as new orders were received, to know as soon as possible when containers would be full and thus should be ordered. By removing their dependence on "go ahead" from DC operators, potential delays in ordering containers would be reduced.

Figure 16. Laptop in the forklift can be used to visualize step-by-step loading instructions

3 Previous work

Preceded by a brief review of commercial optimization software in Section 3.1, the literature study consists of four main sections. First, a brief overview of the literature in Section 3.2. Second, an assessment of our problem types based on five classification criteria in Section 3.3. Third, an assessment of the constraints applicable to our problem based on a framework with 10 categories of constraints in Section 3.4. Fourth, both general and specific concepts we found promising for our approach in Section 3.5. The assessments of problem types and constraints help to narrow down the number of past publications to examine in the search of promising concepts.

3.1 A note on commercial optimization software

To develop an in-house software rather than purchasing an off-the-shelf solution was a directive from management, as explained in Section 1.4. The rationale was that a commercial software was anticipated to require too much tailoring to fit their needs. Despite this directive we want to gain some awareness about commercial offerings. This is partly just out of curiosity, and partly to be able to answer inevitable questions about our competition in terms of software already available on the market.

Hence we conduct a search for commercial offerings, without the intention to be comprehensive. We find an article in the magazine Inbound Logistics by Douglas (2008) helpful as a starting point. Douglas (2008) describe a loading optimization software from Ortec that was implemented at SCA Tissue North America, a part of Svenska Cellulosa Aktiebolaget. According to Ortec (2016) their software solutions are offered as stand-alone, custom-made or embedded in the enterprise software SAP. ORTEC Load Building consists of three modules for multi-level optimization. These are carton optimization (order picking into cartons), pallet optimization (order picking onto pallets), and load optimization (loading into trucks, sea containers, train vehicles and air cargo containers). Ortec (2016) state that their software can handle both rectangular and cylindrical items and constraints including “physical constraints such as orientation, load weight, allowed overhang”, “different packing strategies”, “stacking rules”, “compatibility rules”, “grouping and sequencing rules” and “center of gravity / axle weight calculation”. Further, Ortec (2016) emphasize that ORTEC Load Building can be combined with ORTEC Route Scheduling to also solve the vehicle routing problem.

Douglas (2008) also list other optimization software including MaxLoad Pro by TOPS, Cube-IQ by MagicLogic and CargoWiz by Softtruck. By continuing to search using keywords identified on the suppliers’ websites we find many other offerings, including AutoO_2 by Transportation | Warehouse Optimization, Cargo Optimizer by Dreamsofts Optimization and CubeMaster by Logen Solutions. Similar to Ortec these other suppliers seem to offer a mix of configurations, involving both stand-alone applications, customization as well as various level of integration with enterprise software like SAP. MagicLogic (2016) state that “over 50% of our clients are using versions of Cube-IQ that were specifically modified for them” which links back to what DC management anticipated regarding tailoring. These other suppliers also offer multi-level optimization such as packaging (including design of new packaging), palletization and loading. All of the reviewed suppliers list many types of constraints that their software are capable of handling. There are also several other handy features promoted, for example drag-and-drop to edit a generated load plan, to just name one.

In general, we find plenty of information regarding features and benefits on the supplies’ websites, but less information on the inner workings of their algorithms. Some of the suppliers’ websites refer to academic research in a broader sense, but references to specific solution approaches or particular

publications are scarce. We also have difficulty finding academic research that refers to commercial software.

Based on this brief review of commercial optimization software for container loading we note:

- There seem to be plenty of attractive commercial optimization software available
- Customization of software seems to be common, as anticipated by DC management
- Commercial software seem to be related to academic research to some degree, but in order to assess the extent of the interchange, primary research on this subject itself would be necessary

3.2 The literature on container loading

This section offers a brief overview of the literature on container loading optimization. The overview is based on two recent literature review publications. The first review by Bortfeldt and Wäscher (2013) is focused on constraints in container loading. The second review by Zhao, Bennell, Bektaş and Dowsland (2016) is instead concentrated on algorithms for solving container loading problems. Zhao et al. (2016) make it clear that their work is intended to be complementary to that of Bortfeldt and Wäscher (2013).

3.2.1 Identifying the field of research

Bortfeldt and Wäscher (2013) define container loading problems as three-dimensional *small items* (called *cargo*) that have to be assigned (packed into) three-dimensional rectangular *large objects* (called *containers*) such that some objective function is optimized with respect to applicable constraints. Certain geometric constraints such as the cargo being entirely inside the containers as well as the cargo not intersecting will always exist, while the presence of other more practical constraints depends on the application.

Bortfeldt and Wäscher (2013) note that most publications on container loading only treat rectangular small items, and therefore introduce *boxes* as a better describing naming for cargo. Zhao et al. (2016) also use the term boxes. We therefore adopt *boxes* and use it instead of *packages* throughout this chapter as well as in Chapter 4 on our approach, to facilitate discussions about promising concepts found in the literature. We make exceptions from this and still refer to *packages* when the context is directly related to the DC.

By skimming the references provided in Bortfeldt and Wäscher (2013) as well as Zhao et al. (2016) we notice that there are many different naming used for problems in this area of research. For example the terms three-dimensional, 3D, packing, bin packing, container packing, pallet loading are commonly used in titles of publications in addition to container loading. We mainly stick to using *container loading* for convenience.

Bortfeldt and Wäscher (2013) use the term loading pattern to name a solution to a container loading problem. Zhao et al. (2016) use the term packing pattern instead, or simply pattern. We acknowledge that pattern seems to be a fairly established term for naming a solution, but we mainly use *load plan* in the succeeding chapters since this term is more familiar at the DC.

Bortfeldt and Wäscher (2013) recognize that their definition of a container loading problem also holds for trucks as well as pallets that are limited to be loaded up to a certain height. In line with this comparison Zhao et al. (2016) argue that arranging boxes into a container, truck or pallet is one of the more complex packing problems due to the presence of real-world constraints.

Regarding the practical relevance of the field Zhao et al. (2016) argue that container loading is an important factor in transportation of goods. Bortfeldt and Wäscher (2013) consider container loading to be a pivotal function for operating supply chains efficiently as it impacts both costs and service. Considering the practical importance of this research, we are not surprised by the acknowledgement by Bortfeldt and Wäscher (2013) that there are plenty of research on container loading problems. Bortfeldt and Wäscher (2013) as well as Zhao et al. (2016) also recognize a growth in the number of publications within recent years. Zhao et al. (2016) note that recent research also examine integrating packing with routing and scheduling problems. Based on our problem definition routing and scheduling is not relevant for us now, but it might become interesting for future extensions of our problem.

3.2.2 Solving problems in practice

When it comes to solving container loading problems Zhao et al. (2016) conclude that heuristic algorithms are more common as approach than exact algorithms (i.e. based on mathematical formulations) and suggest two reasons for this. First, a difficulty in representing possible loading patterns and practical constraints in models. Second, if a representation is possible, the formulation typically scales badly with increasing number of boxes and containers. Despite these challenges, Zhao et al. (2016) recognize that encouraging progress has been made in this area of the research. Bortfeldt and Wäscher (2013) also acknowledge that progress has been made in the development of exact and approximation algorithms, but at the same time they argue that heuristic algorithms, and particularly meta-heuristics, will remain the most important class of algorithms for solving container loading problems in practice in the foreseeable future. Bortfeldt and Wäscher (2013) state that in the foreseeable future, only heuristic algorithms will be able to provide solutions of sufficient quality, within acceptable computation times, for problem instances of relevant sizes, if constraints are present. Therefore we make the decision to only focus on heuristic algorithms for our approach moving forward.

Bortfeldt and Wäscher (2013) emphasize that loading patterns typically have to satisfy several constraints at the same time to be feasible in practice. Despite the high number of publications on container loading problems both Bortfeldt and Wäscher (2013) and Zhao et al. (2016) conclude that research on problems dealing with several practical constraints simultaneously is deficient. Zhao et al. (2016) argue that only a small portion of papers consider real-world constraints properly and that most papers instead focus on only a few specific constraints. Similarly Bortfeldt and Wäscher (2013) argue that research have been neglecting issues relevant to container loading in practice and instead has been dealing with standard problems. Since our software have to deal with many practical constraints we have a great opportunity to make a contribution to the field.

Both Bortfeldt and Wäscher (2013) and Zhao et al. (2016) identify one potential cause for these shortcomings in the research to be that the same data sets, which lack practical elements, frequently have been used to develop and test new approaches. Realistic and challenging problem instances would help moving the research forward according to both Bortfeldt and Wäscher (2013) and Zhao et al. (2016). We realize that the availability of real problem instances from the DC capturing relevant constraints is fundamental for our development efforts.

3.3 Assessment of our problem types

Both Bortfeldt and Wäscher (2013) and Zhao et al. (2016) rely on a typology for classifying container loading problem types developed by Wäscher, Haußner and Schumann (2007). The work by Wäscher et al. (2007) consist of categorization criteria as well as names for the problem categories.

We examine their typology in order to classify and name our problem. This exercise facilitates our further study of the literature. In particular it is helpful in Section 3.4 when reviewing the evaluation by Bortfeldt and Wäscher (2013) on how constraints are dealt with in past publications, as these are distinguished based on problem types. Additionally, it facilitates potential referring to our work.

The work by Wäscher et al. (2007) is not limited to container loading problems but rather all types of cutting and packing problems. Wäscher et al. (2007) use five criteria to categorize all cutting and packing problem types. We use their five criteria, but present them in a slightly different order compared to their paper. We note that the terms small items and large objects are used by Wäscher et al. (2007) similarly to how they are used in the definition of container loading problems by Bortfeldt and Wäscher (2013).

3.3.1 Five criteria

The first criteria we address is *dimensionality* and we conclude that our problem is three-dimensional.

The second criteria we address is *shape of small items*. The three alternatives provided by Wäscher et al. (2007) are rectangular, circular and irregular items. We decide to treat our problem as only rectangular items. Given Figures 7-10 in Section 2.2.1 this might not be an obvious simplification, but it is both desirable and necessary. It is desirable because it enables a more straightforward approach and also because there are more publications to find inspiration from. Recall the note by Bortfeldt and Wäscher (2013) that most publication only treat rectangular items. It is necessary because the available dimension data for packages at the DC only consist of length, width and height regardless if the package is fully rectangular or not. Further, the simplification is supported by the DC operators who estimate it to cause negligible reduction in utilization of space. It is motivated by that the majority of packages are either fully or almost fully rectangular, while stacking on the remaining non-rectangular packages often is limited anyway. Dealing with only rectangular items enables us to alternate the use of *small items* and *large objects* with *boxes* and *containers* respectively in this section as well.

Wäscher et al. (2007) note that in the literature rectangular items are usually assumed to be laid out orthogonally, meaning that the boxes are always placed parallel to the inner walls of the container. This simplification is also supported by the DC operators, as they very rarely gain anything from rotating a package other than 90 degrees, since most packages are more or less rectangular.

The third criteria we address is *assortment of small items* and the three possible cases declared by Wäscher et al. (2007) are identical, weakly heterogeneous or strongly heterogeneous. No strict threshold for weakly versus strongly heterogeneous is provided. Weakly heterogeneous is described as “small items can be grouped into relatively few classes (in relation to the total number of items), for which the items are identical with respect to shape and size”. We refer back to Section 2.2.1 and argue that the description for weakly heterogeneous typically does not match our instances, and hence we deal with a strongly heterogeneous assortment of boxes.

The fourth criteria we address is *kind of assignment*. For this criteria the two options are output maximization or input minimization. For output maximization problems the selection of large objects is fixed and not sufficient to accommodate all small items according to the definition by Wäscher et al. (2007). For input minimization problems on the contrary, the selection of large objects is variable and all small items have to be accommodated. This links back to Goal 1 in Section 2.3.1. Simply put, both cases are possible in practice at the DC and this may change from time to time. To exemplify

this, suppose a realistic scenario where a set of packages is considered to be loaded in one 20 ft. container and all but a few packages are able to fit. In this scenario it is typically the customer service representative that have to decide whether two 20 ft. containers (or one 40 ft. container) should be used, or if some packages should be left behind and/or shipped using another mode of transport. Similarly, for consolidation pallets, suppose another realistic scenario where all but a few small packages are able to fit in one consolidation pallet. In this scenario it is typically the DC operator that have to decide whether two consolidation pallets (or one larger consolidation pallet if available) should be used, or if some small packages should be placed directly in the container. Therefore we conclude that both output maximization and input minimization are applicable to our problem.

Finally, the fifth criteria we address is *assortment of large objects*. This criteria is divided in two sub criteria. The first sub criteria is if the number of large objects is one or several. As described in Section 2.2.2 and 2.2.3 most commonly only one container and one consolidation pallet is enough. However, this is not always the case, both multiple containers and multiple consolidation pallets do occur. Hence, we cannot make a strict distinction here. The second sub criteria is for the one large object case, whether dimensions are fixed or variable, and for the several large object case, whether objects are identical, weakly heterogeneous or strongly heterogeneous. In the one large object case we first consider containers and conclude that dimensions are fixed since an intermodal container has no variable dimensions. We then consider consolidation pallets and recall that height is variable as described in Section 2.2.3. With the desire to develop a more general approach we decide to still treat both levels of our multi-level problem as fixed dimensions. We consider the maximum heights of each type of consolidation pallet, but in our approach we remain aware of the possibility to have lower heights. Regarding the second sub criteria for the several large object case, we most commonly deal with identical containers as well as identical consolidation pallets, as described in Section 2.2.2 and 2.2.3. However, this is also not always the case, for example scenarios with two different types of containers or two different types of consolidation pallets are possible. Hence we deal with either an identical or a weakly heterogeneous assortment of large objects in the several large objects case

3.3.2 Conclusion on problem types

Wäscher et al. (2007) also provide names for the problem types based on their system for classification. Given our assessment of the five criteria there are five possible names of our three dimensional rectangular items problem. These are Single Knapsack Problem (SKP), Multiple Identical Knapsack Problem (MIKP), Multiple Heterogeneous Knapsack Problem (MHKP), Single Bin Size Bin Packing Problem (SBSBPP) and Multiple Bin Size Bin Packing Problem (MBSBPP). As we most commonly deal with either one container or multiple identical containers, the three most suitable classifications for our three dimensional rectangular items problem are Single Knapsack Problem (SKP), Multiple Identical Knapsack Problem (MIKP) and Single Bin Size Bin Packing Problem (SBSBPP). Our reasoning is illustrated in Figure 17, with a representation slightly different from that Wäscher et al. (2007), but still the very same criteria.

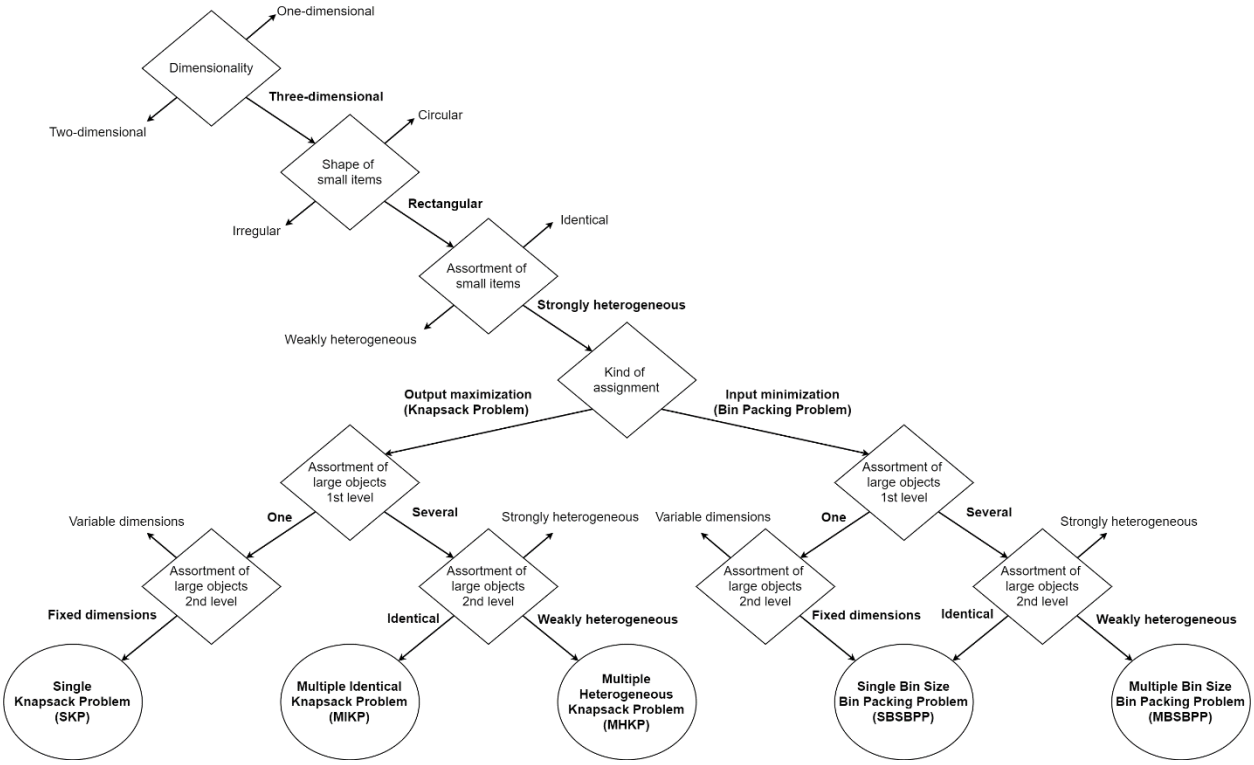

Figure 17. Categorization of our problem using the criteria provided by Wäscher et al. (2007)

3.4 Assessment of our constraints

The work by Bortfeldt and Wäscher (2013) includes a framework for classifying container loading constraints and a comprehensive evaluation of 163 papers published in mainly English from 1980 to 2011. The framework consists of 10 categories of constraints distributed over five category groups, as outlined in Table 3. The evaluation includes assessment of problem types using names proposed by Wäscher et al. (2007), and which of the 10 categories of constraints that are applicable. The classification of our constraints not only make it easier to search for inspiration for our approach, but it also help us to faster identify all relevant considerations that we have to deal with. Additionally, it facilitates potential referring to our work.

Table 3. Categories for container loading constraints provided by Bortfeldt and Wäscher (2013)

Category group number	Category group	Category number	Category
1	Container-related constraints	1	Weight limits
		2	Weight distribution constraints
2	Item-related constraints	3	Loading priorities
		4	Orientation constraints
		5	Stacking constraints
3	Cargo-related constraints	6	Complete-shipment constraints
		7	Allocation constraints
4	Positioning constraints	8	Positioning constraints
5	Load-related constraints	9	Stability constraints
		10	Complexity constraints

Bortfeldt and Wäscher (2013) clarify that item-related for category group 2 refers to one individual item while cargo-related for category group 3 refers to a set of items. Further, positioning for category group 4 concerns positioning of items within containers and load-related for category group 5 concerns the final arrangement of the items.

Bortfeldt and Wäscher (2013) also note that constraints may occur as either hard or as soft constraints. Any hard constraints must be feasible for a loading pattern to be feasible, while violations of soft constraints are tolerated, at least within certain limits. Based on discussions with the DC operators we concur with this note, as the operators describe certain scenarios as unacceptable, while other scenarios as acceptable but not preferred.

We address the 10 categories with the intention to better understand what they mean and to assess if they are applicable to our problem.

3.4.1 Container-related constraints

Category 1 *weight limits* refers to that the total weight of boxes loaded in a container should not exceed a certain weight limit of the container. This is applicable to our problem since weight capacities are available for both shipping containers and consolidation pallets.

Category 2 *weight distribution constraints* considers how the weight of the cargo is distributed across the container. Uneven weight distribution may cause damages and accidents during handling and transportation. The weight distribution is recognized by the DC operators as a concern and hence applicable to our problem.

3.4.2 Item-related constraints

Category 3 *loading priorities* is only relevant to output maximization problems as it refers to how boxes should be prioritized if the containers are not sufficient to accommodate all boxes. This is to some extent relevant for us and we refer back to our reasoning about the kind of assignment criteria in Section 3.3.1. Based on discussion with customer service representatives and management at the DC we conclude that whenever needed prioritization should be done manually rather than automatically by our software. For example, the customer service representative should rather be removing packages from the input than entering prioritization rules for the packages. Therefore we conclude that loading priorities is not applicable to our problem.

Category 4 *orientation constraints* refers to how orientation of boxes may be restricted both in vertical and horizontal direction. Bortfeldt and Wäscher (2013) exemplify that violation in vertical direction may cause damage of the content as well as the packaging. Horizontal direction may matter for boxes with pallets attached when loading using a forklift, if the pallet runners are only going in one direction (two-way entry rather than four-way entry). We note that if a pallet is attached then only one vertical direction makes sense. Given only orthogonal placements as stated in Section 3.3.1 there are six potential orientations for boxes. Bortfeldt and Wäscher (2013) define five different cases to describe to what extent these six orientations of boxes are restricted. We prefer to instead define the six orientations numbered 1-6 using *base* to describe vertical orientation and *rotation* to describe horizontal orientation. To facilitate further discussion we illustrate this in Figure 18 using a fixed three dimensional axis and fixed notation of the length, width and height of the box.

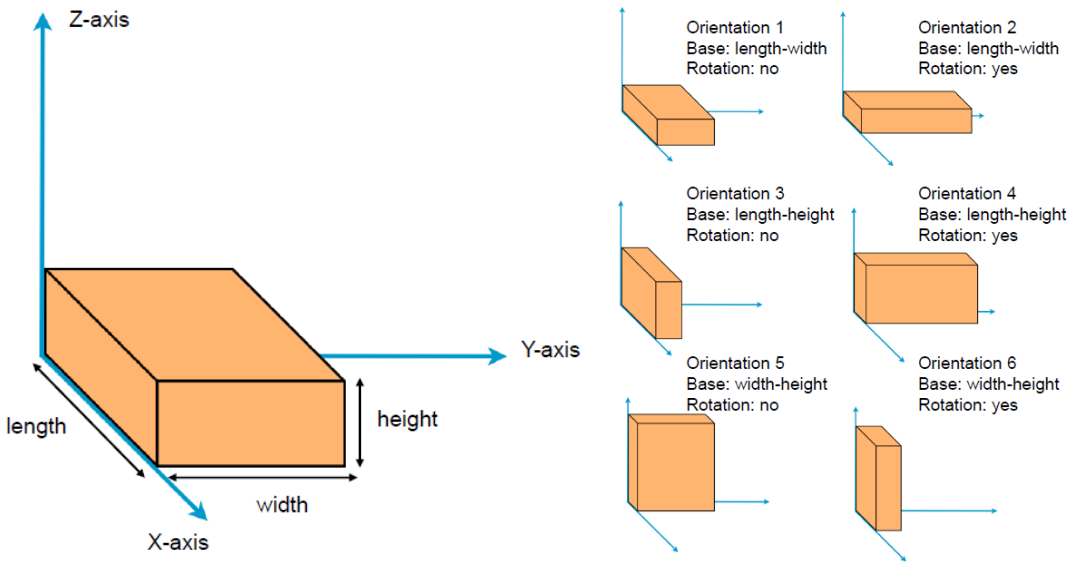

Figure 18. Possible orientations of three dimensional rectangular items

We typically have both boxes with pallet attached and boxes without pallet attached, as indicated in Section 2.2.1.

For boxes with pallet attached we define that the pallet is always adjacent to the length-width base. Forklifts are used at the DC when loading and according to the DC operators boxes with pallet attached are always placed so that the pallet is facing downwards (in contact with the container floor or packages below). Hence we conclude that only orientation 1 and 2 are relevant here.

For pallet runners in both directions (four-way entry) we note that both orientation 1 and 2 are acceptable. For pallet runners in one direction (two-way entry) it seems at a first glance reasonable to only allow one orientation. However, based on discussions with the DC operators we identify two exceptions when packages with pallets attached may be placed in the container oriented opposite to the direction of the pallet runners. The first exception is for long packages such as the cylinder in Figure 7 which are partly pushed into the container using the forks on the forklift. For very long packages this might be necessary due to the package only fitting in the container lengthwise, while in other scenarios it is not required but beneficial to better utilize the container volume. The second exception is for any package light enough to be rotated either by hand or using the forklift. This links back to objective 4.1 in Section 2.3.2 as it is not desirable from a loading efficiency standpoint with respect to time. Therefore it is only used when expected to be necessary from a volume utilization standpoint, e.g. to fit all packages. A rule of thumb among the DC operators is that packages up to 200 lbs. (91 kg) may be rotated by hand. Altogether we realize that the direction of the pallet runners relative the base dimensions (length and width) of a box is essential to handle orientation constraints. To remark this insight we define three possible pallet orientations as illustrated in Figure 19.

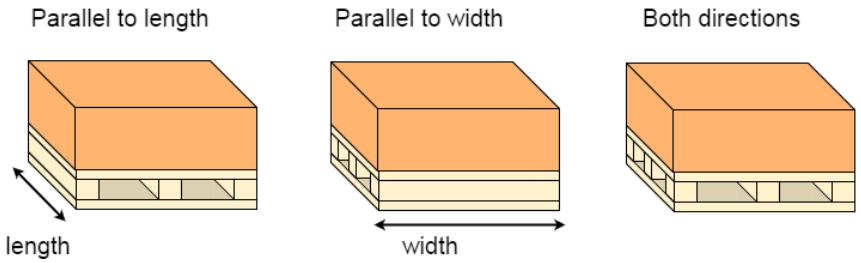

Figure 19. Possible pallet orientations for boxes with pallet attached

For boxes without pallet attached it is necessary to determine whether only orientation 1 and 2 should be acceptable, or if all six orientations should be acceptable. Based on discussion with the DC operators we realize that during packing on level 1 and level 2 as illustrated in Figure 3 the operators may make assumptions about the vertical direction upon loading. For example, an operator may assume that the package will be loaded according to any marking on the packaging saying "this way up", "this side up" or a symbol with arrows pointing upwards. From the DC operators we also learn that deviating from the vertical direction without knowledge about the content and the packaging could occasionally cause damage of parts during transport. However, the operators acknowledge that on rare occasions boxes are turned over to better fit or to achieve better stability of cargo. In these occasions it is desirable to inspect the content as well as the packaging first, but whether it is done properly is very dependent on the operators performing the loading. To avoid potential risks for damage of parts we decide, in agreement with the DC operators as well as management, to only accept orientation 1 and 2 for boxes without pallet attached.

Category 5 *stacking constraints* considers how boxes can be placed (stacked) on top of each other. Bortfeldt and Wäscher (2013) comment that *load bearing constraints* is also used to name this category. Four factors are mentioned by Bortfeldt and Wäscher (2013) to have a significant impact on how much weight or pressure a box may endure before bursting. Bortfeldt and Wäscher (2013) note that the purpose of stacking constraints is to avoid damaging boxes and based on discussions with the DC operators we conclude that this is one of the aspects considered to be most important for our problem.

The first factor is the strength of the packaging, which is dependent on the construction as well as the material of the box itself. The second factor is the content of the box, for example whether it is filled with solid content that can help withstand the load above. The third factor is related to the way of stacking, for example how many edges (or corners) that is used by a lower box to support a box above. Bortfeldt and Wäscher (2013) note that much of the strength of a box comes from the side walls and refer to Edge Crush Test (ECT) as a measure related to this. The fourth factor is related to the transportation and includes aspects such as the duration of the transport, the humidity throughout the route, etc.

The DC operators recognize all four factors to be relevant. However, the operators refer to common sense when it comes to evaluating feasible placements, but acknowledge that they likely are using the four factors either consciously or subconsciously. Bortfeldt and Wäscher (2013) mention several ways in which stacking constraints are dealt with in the literature. Approaches include rules for stacking based on classifications of boxes such as fragility, number of boxes, weight of boxes, density of boxes and pressure as in weight units per area unit.

From initial experiments which entailed pre-staging (test loading) together with DC operators we learned that the third factor, namely the way of stacking, have a significant impact on feasible

placements. For example, we learned that in many situations support from two corners is almost as good as support from four corners, while support from only one corner dramatically reduce how much weight that can be loaded above a given package. We also learned that for cases with support from less than four corners, plenty of the strength from the walls of the box below is gained if the box above almost reach out to the other edges. Hence it matters how close the box above is to reach out to the other edges of the box below in these cases. In Figure 20 below we illustrate a few examples of various level of support with respect to corners and edges.

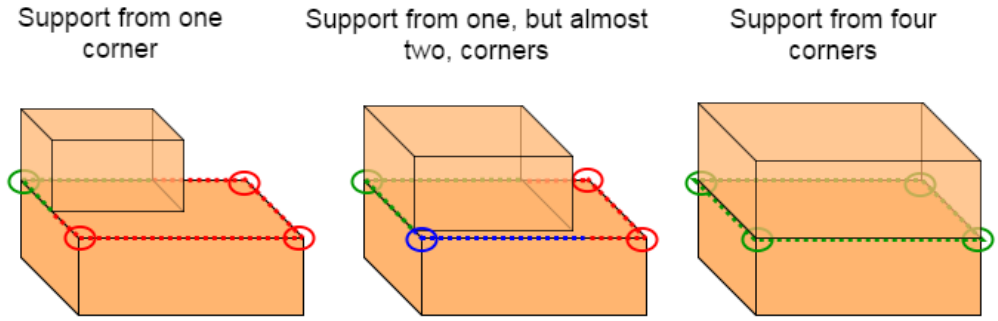

Figure 20. Examples of various level of support from box below

Since Bortfeldt and Wäscher (2013) only mention ECT in passing, we review a guide on ECT provided by Fibre Box Association (2008), a non-profit trade association representing corrugated packaging manufacturers in North America. ECT is described as a measure of a physical property of a corrugated board and is typically expressed in units of force per unit of length (i.e. 1 ECT = 1 lbs. per inch). The ECT value alone does not answer how much weight or pressure a box may endure before bursting, but it can be used combined with other parameters to estimate stacking strength. Fibre Box Association (2008) state that the McKee Formula developed by McKee, Gander and Wachuta (1963) is despite its age commonly used by the industry, including in commercial optimization software, to relate ECT values to stacking strength.

A key parameter in the McKee formula is the environmental factor. This factor is necessary to capture aspects such as the way of stacking, the duration of the transport, the humidity throughout the route, etc. Related to the way of stacking, Fibre Box Association (2008) note that the strength of a box is in particular focused in the corners. We note that ECT is a measure for corrugated board, while we also have crates of wood, as mentioned in Section 2.2.1. However, the crates of wood are typically stronger than packaging of corrugated board according to the DC operators.

Based on the above we conclude that stacking constraints is applicable to our problem, including consideration to the way of stacking. The McKee formula seems promising in order to relate ECT values to stacking strength, and therefore we treat the McKee Formula further in Example 3 in Section 4.4.2.

Finally, exceptions to reject the assumption that other packages can be stacked on a package as mentioned in Section 2.2.1 may either be considered as this category or category 8 (see Section 3.4.4).

3.4.3 Cargo-related constraints

Category 6 *complete-shipment constraints* is similar to loading priorities in the sense that it is only relevant for output maximization problems in which by definition all cargo cannot fit. Complete-shipment constraints refers to a practical consideration that a subset of boxes may represent a functional or administrative entity. A functional entity could for example be that pieces in multiple boxes are supposed to be assembled at the customer site. An administrative entity could for example be an order. We note that a functional entity may, but does not have to, correspond to an administrative entity, and vice versa. If complete-shipment constraints is applicable then all or none of the boxes in a subset must be loaded. In other words, if one box in a subset is loaded, then all other boxes must be loaded, and if one box cannot be loaded, no box should be loaded. Bortfeldt and Wäscher (2013) note that if several containers are loaded there are two cases, namely if boxes have to be loaded in the same container or just loaded in any of the containers. Both functional and administrative entities are relevant considerations for us according to customer service representatives at the DC. However, based on discussions with both customer service representatives and management we conclude that this should be handled manually rather than automatically by our software. For example the customer service representative should be adding packages to, and removing packages from, the input in sets made up of whole orders. Hence we conclude that complete-shipment constraints is not applicable to our problem.

Category 7 *allocation constraints* is relevant for problems with multiple containers and is quite similar to complete-shipment constraints as it also deals with subsets of boxes. Boxes may have to be loaded in the same container due to requirements from the customer, or may not be allowed to be loaded together, for example if the content of the boxes is not suitable to be transported together. Bortfeldt and Wäscher (2013) mention food with perfume as an example of the latter. Similar to complete-shipment constraints we conclude that this should be handled manually rather than automatically by our software. Hence we conclude that allocation constraints is not applicable to our problem.

3.4.4 Positioning constraints

Category 8 *positioning constraints* restricts the location of boxes within a container. Bortfeldt and Wäscher (2013) distinguish between absolute positioning constraints and relative positioning constraints. *Absolute positioning constraints* refers to how individual boxes should or should not be located relative the container, e.g. a specific position or space. Such restrictions may be caused by the size, the weight or the content of the box. *Relative positioning constraints* refers to how boxes should or should not be located relative each other. Such restrictions may be related to loading and unloading considerations and for example applicable when loading problems are integrated with vehicle routing problems. Bortfeldt and Wäscher (2013) note that positioning constraints often are combinations of the absolute and the relative sub categories.

We identify multiple positioning constraints from Section 2.2.1. First, we have one relative positioning constraint in that packages with hazardous content have to be easy accessible, which may be interpreted as no other packages placed above or in front of it. Second, the exceptions “not ok to stack a different package above” and “not ok to stack a similar package above” may also be considered as relative positioning constraints. Third, we have two absolute positioning constraints. The first absolute positioning constraint is that packages without pallet attached should preferably not be placed on the container floor due to the risk of moisture. The second absolute positioning constraint is packages with exception submitted as “must be placed on the floor”, for example due to liquid or weight. Therefore we conclude that positioning constraints is applicable to our problem.

3.4.5 Load-related constraints

Category 9 *stability constraints* considers the risk that boxes do not remain at the initial position in the container e.g. due to falling or shifting. Bortfeldt and Wäscher (2013) note that unstable loads may cause damage of cargo as well as injuries of people during loading, transportation and unloading. Based on discussions with DC operators as well as customer service representatives we conclude that this is considered to be one of the most important aspects of our problem. It links back to Goal 2 in Section 2.3. Bortfeldt and Wäscher (2013) also acknowledge this to be one of the important issues in container loading, but also recognize that despite this it is not commonly considered explicitly in publications. Bortfeldt and Wäscher (2013) comment that many authors instead simply argue that good stability is a consequence of high space utilization.

Bortfeldt and Wäscher (2013) distinguish between vertical stability and horizontal stability. *Vertical stability* is described as preventing boxes from falling down while the container is not moving (i.e. withstand gravity). Bortfeldt and Wäscher (2013) note that one way this is dealt with in the literature is to require either total or partial support of the base area of a box. If a box is placed on the container floor the support will always be 100% of the base area, while for a box placed on top of one or several other boxes the support may be below 100% which then implies overhang. *Horizontal stability* is described as preventing boxes from shifting while the container is moving (i.e. withstand inertia). In practice this is achieved by a combination of boxes receiving support from adjacent boxes and the container walls as well as additional securing. As mentioned in Section 2.3.1 material used at the DC to secure packages consist of bands, straps and airbags. Bortfeldt and Wäscher (2013) note that one way horizontal stability is dealt with in the literature is to measure contact with the four side surfaces of each box. In Figure 21 below we exemplify how one may conceive vertical as well as horizontal stability.

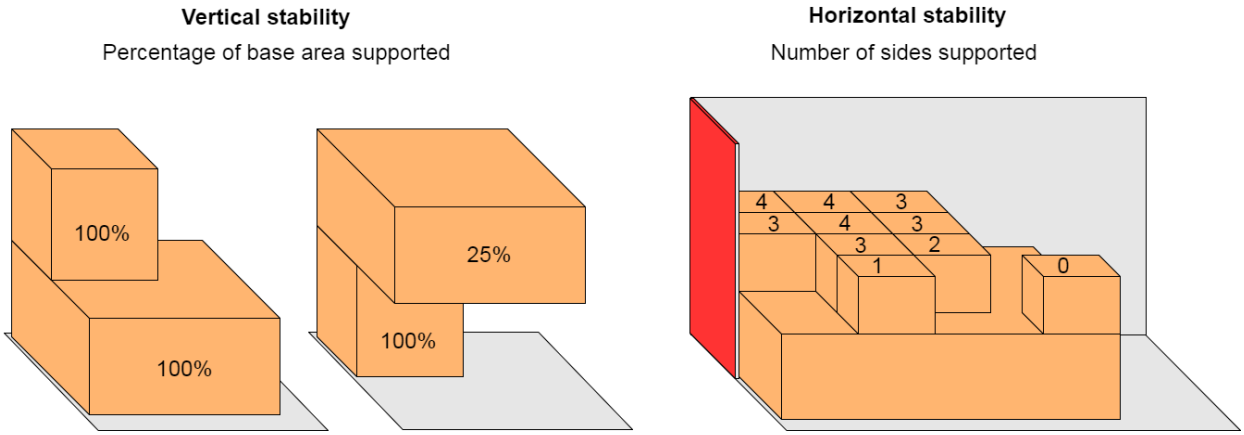

Figure 21. Examples of various level of vertical and horizontal stability

Based on discussion with DC operators we conclude that both vertical stability and horizontal stability are very important for us. In particular horizontal stability is something that the operators are very conscious about. Placements resulting in interlocking of packages is praised as a key aspect when loading. Towers of packages is recognized as something that if possible should be avoided. The DC operators stress the importance of common sense when it comes to evaluating feasible placements. We conclude that stability constraints is applicable to our problem.

Category 10 *complexity constraints* refers to any limitations of technological and human resources. Bortfeldt and Wäscher (2013) stress that solutions must be visualized and easy to understand for

the operators doing the loading so that the loading procedure does not become too time consuming. This links back to Section 2.4 about the small laptops in the forklift that should be used to display visual step-by-step instructions. We realize that this aspect is critical for us, but we question whether this should be considered as a constraint or instead a more general concern related to the loading process. Either way, based on our initial experiments which entailed pre-staging (test loading), we identify another consideration related to forklifts that clearly is a complexity constraint. More specifically, for a package to be placed above one or several other packages using a forklift, the forks should preferably be able to reach the intended position. Figure 22 below illustrates a scenario in which the intended position cannot be reached.

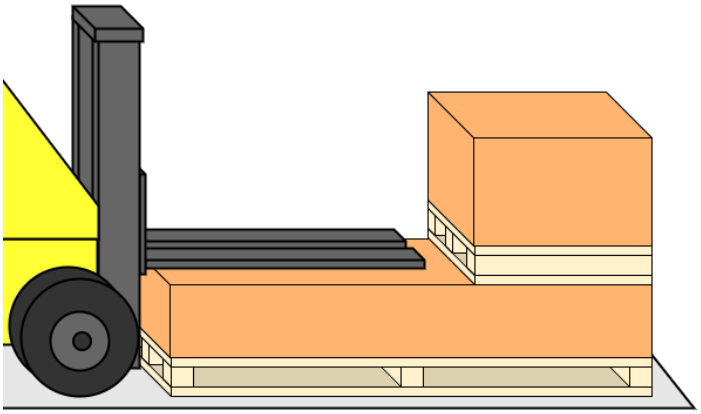

Figure 22. Example of intended position not reachable by forklift

Potential options if a position cannot be reached are mentioned under Goal 4 in Section 2.3.1. We conclude that complexity constraints is applicable to our problem.

3.4.6 Summary of applicable constraints

The classification of our constraints using the categories provided by Bortfeldt and Wäscher (2013) is summarized in Table 4. In total 7 of the 10 categories of constraints are applicable to our problem. We note that only 2 of the 163 publications (1.2%) evaluated by Bortfeldt and Wäscher (2013) consider 7 constraints, and no publication consider more than 7 constraints. This confirms our anticipation mentioned in Section 3.2.2 that we are facing many practical constraints relative publicly available container loading research.

Table 4. Categories for container loading constraints applicable to our problem

Category number	Category	Applicable to our problem
1	Weight limits	Yes
2	Weight distribution constraints	Yes
3	Loading priorities	No
4	Orientation constraints	Yes
5	Stacking constraints	Yes
6	Complete-shipment constraints	No
7	Allocation constraints	No
8	Positioning constraints	Yes
9	Stability constraints	Yes
10	Complexity constraints	Yes

3.5 Promising concepts for our approach

We define criteria to filter among the 163 publications evaluated by Bortfeldt and Wäscher (2013) based on the assessment of our problem types in Section 3.3 and the assessment of our constraints in Section 3.4. The following are our filtering criteria, preceded by motivations:

Avoid problems that are not relevant for us given the assessment of our problem types:

- Consider only three dimensional rectangular items SKP, MIKP, MHKP, SBSBPP, MBSBPP

More likely to be a problem from practice like ours:

- Include at least four constraint categories

Most important constraints for our problem:

- Include both stacking and stability constraints

By applying the criteria the 163 publications are narrowed down to 15. We find these 15 publications interesting to further examine in order to identify promising concepts for our approach. However, first we assemble some general ideas about algorithms for solving container loading problems from the work by Zhao et al. (2016).

3.5.1 General concepts

Zhao et al. (2016) claim that in many publications problems with several containers are tackled by simply extending models for one container. We have decided to solely focus on heuristic algorithms, as stated in Section 3.2.2. Zhao et al. (2016) note that all heuristic approaches include a mechanism to place boxes in the containers and call this *placement heuristic*. Zhao et al. (2016) further note that many heuristic approaches also include some kind of *improvement heuristic* and mention Genetic Algorithm (GA), Tabu Search (TS) and Simulated Annealing (SA) as commonly used meta-heuristics. For more information on meta-heuristics we refer to the extensive work by Talbi (2009) titled Metaheuristics: From Design to Implementation.

Regarding placement heuristics Zhao et al. (2016) distinguish between wall building, layer building and other approaches. Other approaches include block building as well as boxes placed one at a time. In wall building walls of boxes are created lengthwise in the container and in layer building layers of boxes are created heightwise in the container. In block building multiple boxes are arranged to form a block that may be treated as one larger box. Zhao et al. (2016) claim that wall building and layer building are most common when boxes are weakly heterogeneous, while other approaches are more common for strongly heterogeneous boxes. In Section 3.3.1 we concluded that our boxes are strongly heterogeneous, and therefore we decide to focus on other approaches than wall building and layer building.

Zhao et al. (2016) distinguish that placement heuristic adopt either static sequencing or dynamic sequencing. This refers to the order in which boxes are placed in the container, i.e. the apparent question after each placement, namely which box to place next. In static sequencing the ordering of boxes may be determined before the first placement as it is not impacted by the following placements. On the contrary, in dynamic sequencing the ordering of boxes is updated after each placement, meaning that the choice which box to place next may be dependent on previous placements. For both static and dynamic sequencing some sort of scoring is typically used to generate the ordering, the

differences lie in which criteria that may be used for the scoring. For example, static criteria is typically focused on characteristics of all boxes while dynamic criteria also may include aspects such as available spaces (unutilized volume) as well as characteristics of remaining boxes after each placement. Zhao et al. (2016) argue that the sequencing can be critical for the performance of a given approach. We intend to pay careful attention to this aspect, and intuitively we find dynamic sequencing more appealing.

Regarding where to place a box Zhao et al. (2016) mention that evaluating combinations of boxes, orientations and available spaces (unutilized volume) or boxes, orientations and placement points are two commons methods. We find these methods promising.

Considering improvement heuristics Zhao et al. (2016) explain the benefit as broadening the search space and make neighborhood moves in order to find better solutions. This is commonly accomplished using meta-heuristics such as GA, TS and SA that guide the placement heuristics. Based on examples provided by Zhao et al. (2016) we realize that the meta-heuristics may be used in many different ways. We therefore limit our study of meta- heuristics to the 15 publications we have selected from the evaluation by Bortfeldt and Wäscher (2013). From these publications we also identify more specifics related to placement heuristics.

3.5.2 Specific concepts

Based on an initial assessment of the relevance for our work, we decide to not further examine 11 of the 15 publications. We briefly elaborate on the reasons for excluding these 11 publications in this segment. Gendreau, Iori, Laporte and Martello (2006), Fuellerer, Doerner, Hartl and Iori (2010), Iori and Martello (2010) as well as Bortfeldt (2012) all treat integrated vehicle routing and container loading problems. For understandable reasons the loading approaches in these publications are not presented with as much detail compared to other publications treating pure loading problems. Bortfeldt and Gehring (2001), Gehring and Bortfeldt (2002) as well as Ceschia and Schaerf (2013) propose placement heuristics based on wall building or layer building, which we decided earlier in this section to not adopt. The work of Balakirsky, Proctor, Kramer, Kolhe and Christensen (2010) concerns palletizing rather than container loading and hence more relevant for consolidation pallets. We find loading of shipping containers much more important in our problem and we prefer to begin with an approach mainly intended for this. Gehring and Bortfeldt (1997) present an approach based on building and placing box towers. In Section 3.4 we noted that towers of packages is recognized by the DC operators as something undesirable. Makarem and Haraty (2010) suggest an approach with four agents, one for each of the activities packing, tracking, loading and unloading. We find this quite interesting, but are discouraged as neither Zhao et al. (2016) nor Bortfeldt and Wäscher (2013) mention agent based approaches in their reviews. Lastly, the work by Bortfeldt and Gehring (1999) is written in German and excluded simply based on our convenience as non-German speakers.

Out of the four remaining publications we favor the work by Techanitisawad and Tangwiwatwong (2004) in which the loading approach is thoroughly explained. We find several aspects of their approach promising, while we also appreciate that it seems to offer good flexibility for modifications. Techanitisawad and Tangwiwatwong (2004) tackle an integrated container selection and container loading problem. The problem is defined as to determine a set of containers that minimize cost and a loading pattern for each container that maximize volume utilization subject to practical loading constraints.

Techanitisawad and Tangwiwatwong (2004) propose an integrated procedure based on GAs in which two main heuristics called container selection and container loading are solved iteratively

until all boxes are loaded. Both the container selection and the container loading are driven by GAs, but with different design of chromosomes, fitness functions, etc. The container selection is designed to minimize cost constrained by volume and weight of boxes. We note that cost easily may be replaced with volume capacity. We also note that a GA for our container selection problem is redundant as we typically deal with only one or a couple of shipping containers and consolidation pallets with only one or a couple of different types of each to choose from. Further, Techanitisawad and Tangwiwatwong (2004) design the container loading heuristic using two evaluation functions with weighted coefficients to first select position and second select box for next placement. The chromosomes of each individual in the container loading GA represents the actual values of these in total 18 weighted coefficients.

Step 1: Select a loading position k with coordinates (x_k, y_k, z_k) among available points such that $\min_k P_k = p_1 \times x_k^2 + p_2 \times y_k^2 + p_3 \times z_k^2 + p_4 \times x_k \times y_k + p_5 \times x_k \times z_k + p_6 \times y_k \times z_k + p_7 \times x_k + p_8 \times y_k + p_9 \times z_k$ where $p_1, \dots, p_9$ are weighted coefficients and also chromosome values.

Step 2: Select a box i among remaining boxes to be loaded at the selected position k such that $\max_i B_i = \sum_{j=1}^{9} b_j \times factor_{i,j}$ where $b_1, \dots, b_9$ are weighted coefficients and also chromosome values. Each $factor_{i,1}, \dots, factor_{i,9}$ reflects characteristics of box i related to either size, weight, feasible orientations, stackability or how many similar boxes there are.

We recognize that this is static sequencing according to the definitions by Zhao et al. (2016) as none of the factors are dependent on previous placement. In other words, all box evaluation factors may be calculated before the first placement and remain fixed throughout the whole loading procedure, which clearly also is the case in the algorithm outlined by Techanitisawad and Tangwiwatwong (2004). As stated earlier in this section, we find dynamic sequencing more appealing. However, we assume that the concept of GA with weighted coefficient as chromosomes is appropriate not only for static sequencing but also for dynamic sequencing.

In the container loading heuristic proposed by Techanitisawad and Tangwiwatwong (2004) the boxes are placed one at a time, into one container at a time, without any type of wall building or layer building mechanism. Upon evaluation of a box, with a given orientation at a certain point, the constraints evaluated are weight capacity of container, stacking (based on density of boxes) as well as vertical stability, in addition to basic constraints such as inside container and not intersecting other boxes. Techanitisawad and Tangwiwatwong (2004) acknowledge as a concluding remark that the lack of consideration to horizontal stability may cause damage of boxes when containers are being moved and suggest further work to address this issue. As noted in Section 3.4.5, category 9, this is one of our primary concerns.

When there are no more feasible placements remaining the container stability in terms of center of gravity is evaluated. The fitness of a GA individual corresponding to one loaded container is calculated as $container\ volume\ utilization \times penalty\ factor$ where the $penalty\ factor$ is based on the container stability. Based on Section 2.3.2 we realize that a fitness function for our problem have to incorporate more aspects than container volume utilization and container stability. More specifically, the fitness function proposed by Techanitisawad and Tangwiwatwong (2004) do not include stability of cargo and awkward placement in the final load plan.

In the three other publications remaining after the initial scanning we also identify some other promising ideas.

Lin, Chang and Yang (2006) also use a GA and propose an overall procedure with clustering of boxes based on capacity of containers, dimensions of boxes and unloading sequence of boxes. The overall algorithm consists of two stages, namely a grouping procedure and a packing procedure, which are executed iteratively. The grouping procedure consist of determining a number of groups, assigning boxes to these groups and selecting potential containers for each group. In the packing procedure the containers are packed one at a time using only one container per group of boxes. If none of the potential containers selected for a given group of boxes were sufficient to accommodate all boxes in the group then the entire packing procedure is terminated and the grouping procedure is called for a new attempt to group the boxes. Lin et al. (2006) state that the number of containers typically range from 4 to 12 for their problem instances, which is very different from our problem with typically one or two shipping containers and consolidation pallets respectively. However, we do like the idea of grouping boxes when considering the large variety of packages at the DC as described in Section 2.2.1. Grouping is in fact already practiced with the use of consolidation pallets, as described in Sections 2.1 and 2.2.3. Based on discussions with the DC operators we also note that to think about groups of packages beyond consolidation pallets actually reflects the thought process when loading in practice. For example, large and heavy packages with better strength and stackability are typically placed first.

To continue on the topic of imitating the thought process in practice we find the two heuristics proposed by Wang, Guo, Chen, Zhu and Lim (2010) very interesting. Wang et al. (2010) state that two heuristics called Deepest-Bottom-Left-Fill and Maximum Touching Area mimic how loading is done in practice. In the Deepest-Bottom-Left-Fill heuristic boxes are loaded one at a time by placing the current box as close to the inner-bottom-left corner of the container as possible. In the Maximum Touching Area heuristic placements are selected based on maximizing the total contact area of the evaluated box which is defined as the sum of areas for all six sides that are adjacent to either other boxes or the inner walls of the container. Based on discussions with the DC operators we note that these two strategies are similar to how they are thinking, either consciously or subconsciously, when loading a container. We find it intuitive that rewarding contact with all sides of boxes contribute to achieve horizontal stability.

Tarantilis, Zachariadis and Kiranoudis (2009) propose six variants of packing heuristics called BackLeftLow, LeftBackLow, MaxTouchingAreaW, MaxTouchingAreaNoWallsW, MaxTouchingAreaL and MaxTouchingAreaNoWallsL. As the names convey these heuristics are similar to the two suggested by Wang et al. (2010). Tarantilis et al. (2009) argue that by favoring a packing structure it is more likely to obtain feasible load plans. We conclude that it seems appropriate to consider how boxes are placed relative the inner-bottom-left corner of the container as well as to what extent boxes are touching other boxes or the inner walls of the container. Finally, we recognize that these represents examples of dynamic sequencing.

3.5.3 Summary of promising concepts

Promising concepts for our approach obtained from the literature are summarized below. It is worth emphasizing that with Zhao et al. (2016) as source of inspiration we refer to the accumulated knowledge in their literature review. Recognition should also be forwarded to the authors behind the work that Zhao et al. (2016) have studied.

Inspired by Zhao et al. (2016)

- Place boxes one at a time rather than using a wall building or layer building mechanism
- Consider potential placements as combinations of either Box-Orientation-Point or Box-Orientation-Space

Inspired by Zhao et al. (2016), Wang et al. (2010), Tarantilis et al. (2009)

- Dynamic sequencing of boxes rather than static sequencing, including consideration of available spaces (unutilized volume), characteristics of remaining boxes, positioning relative inner-bottom-left corner of the container, as well as to what extent the sides of boxes are touching other boxes or the container walls

Inspired by Techanitisawad and Tangwiwatwong (2004)

- Genetic Algorithm as improvement heuristic with chromosomes representing values of weighted coefficients in evaluation function for placement heuristic

Inspired by Lin et al. (2006)

- Grouping of boxes

4 Approach

The proposed method for solving our container loading problem is thoroughly explained. At an overarching level, the explanation is organized in three blocks. First, the entire approach is outlined in Section 4.1, including the batch concept and the overall algorithm which iteratively solves four subproblems (batches). Second, the major parts of the overall algorithm are treated in further detail in Sections 4.2-4.5. These major parts are container selection, load plan evaluation, placement heuristic and Genetic Algorithm. Third, it is summarized how objectives and constraints are handled within the load plan evaluation and the placement heuristic in Section 4.6.

4.1 Outline of approach

In Section 4.1.1 the batch concept is explained, which is a fundamental design element of our total approach. In short, the batch concept refers to grouping of boxes based on their characteristics and solving subproblems in terms of partial load plans.

In Section 4.1.2 our overall algorithm is outlined, which iteratively solves four subproblems (batches). In total there are two variants of container selection procedures and four variants of container loading procedures. Both *consolidation pallets* and *shipping containers* are referred to as simply *containers* when appropriate, as mentioned in the chapter preface. The overall algorithm is presented in a schematic which also shows how the succeeding Sections 4.2-4.5 are related to each other. Section 4.2 treats the container selection while Section 4.3-4.5 are related to container loading. In brief, complete enumeration is used for the container selection and an extensive placement heuristic guided by a Genetic Algorithm (GA) is used for the container loading. The placement heuristic is deterministic, while the GA is stochastic. The container loading procedure is designed with two different types of objective functions: one to describe how smart a placement is expected to be, and the other to describe how good a complete load plan is.

4.1.1 Batch concept

The *batch concept* alludes to the meaning of the word batch as arrange things in groups (to be handled together), as well as the term batch production in manufacturing. By the batch concept we refer to grouping of boxes based on their characteristics and solving subproblems in terms of partial load plans.

The rationale was discussed in Section 3.5.2, inspired by the grouping approach proposed by Lin et al. (2006). In brief, grouping of boxes is already practiced at the DC with the use of consolidation pallets, and it also reflects the thought process of DC operators when loading in general. For example, large and heavy packages with better strength and stackability are typically placed first.

The total problem outlined in Section 2.1 is broken down into four subproblems and four corresponding batches are defined. These are named *Batch 1*, *Pre-check*, *Batch 2* and *Batch 3*. The differences in naming is intended to reflect that actual placements of boxes only occur in the first, third and fourth subproblem, while the second subproblem is a test that generates input parameters for the third and the fourth subproblem.

At the start of Batch 1 neither shipping containers nor consolidation pallets have been selected, and hence no packages have been loaded. At the end of Batch 3 both shipping containers and consolidation pallets have been selected, consolidation pallets have been arranged, and everything have been loaded into the shipping containers. The procedure is illustrated in Figure 23.

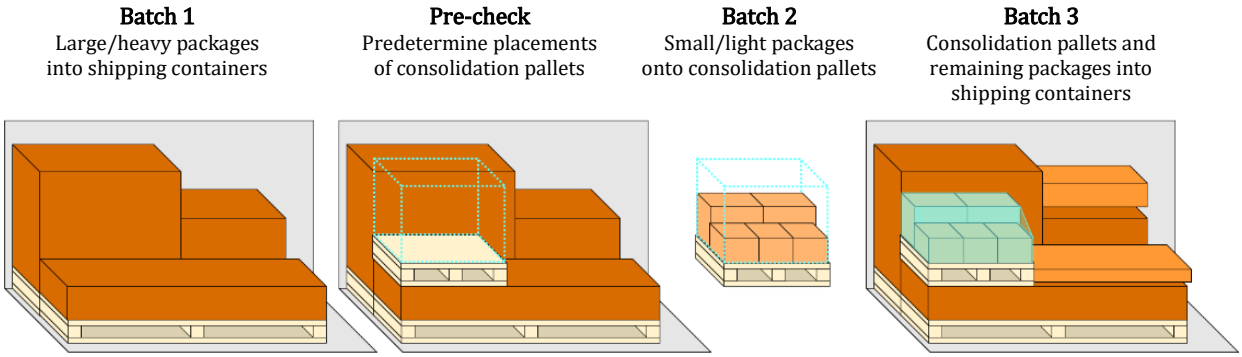

Figure 23. Batch concept - partial load plans generated by solving four subproblems (batches)

Each batch concerns a partial load plan and each batch is dependent on the previous batches, with exception for Batch 1 which is started with a blank load plan. Only one partial load plan is transferred from one batch to the next. At the end of Batch 3 the load plan is finally complete.

Grouping of boxes

The grouping of boxes is done before Batch 1 begins. All boxes are assigned to either Batch 1, Batch 2 or Batch 3. There are two types of batch reassignments that may occur, namely that boxes initially assigned to Batch 1 or Batch 2 are reassigned to Batch 3. This occurs if the particular batch is terminated by the user prior to all boxes being loaded, as opposed to allowing an increase of capacity by changing/adding shipping container or consolidation pallet (depending on which batch).

The criteria for the initial grouping involve weight, dimensions and three of the additional attributes that were presented in Section 2.2.1. These three attributes are if the box has a pallet attached, if the box is required to be placed on the container floor, and if the box is hazardous. More specifically, each box is assigned to one batch according to the criteria and sequence below that have been developed together with the DC operators. Some numerical values are from Sections 2.2.1 and 2.2.3.

Assign box to Batch 1 if <u>at least one</u> of the following conditions are true

- Weight of the box is greater than 36 kg (80 lbs.)
- Maximum dimension of the box is greater than 1.5 m
- The box has a pallet attached
- The box is required to be placed on the container floor

Assign box to Batch 2 if <u>all</u> of the following conditions are true

- Weight of the box is less than 36 kg (80 lbs.)
- Maximum dimension of the box is less than 1.113 m
- Maximum base dimension (length or width) of the box is less than 1.113 m
- Minimum base dimension (length or width) of the box is less than 0.732 m
- Volume of the box is less than 0.3 m^3
- The box has not a pallet attached
- The box is not required to be placed on the container floor
- The box is not hazardous

Assign box to Batch 3 if neither criteria for Batch 1 nor Batch 2 are satisfied.

What happens in each batch

Figure 24 is a complementary illustration to Figure 23 to support the explanation of what happens in each batch which follows below.

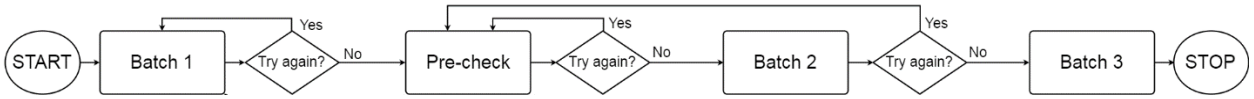

Figure 24. Schematic of batch concept - four subproblems (batches) solved iteratively

Batch 1 begins with selection of shipping containers among available types. Packages assigned to Batch 1 are then loaded into the shipping containers. If the limit for number of attempts is reached without successfully fitting all packages in the selected shipping containers, the capacity is increased by reselecting a combination of shipping containers with more capacity. If instead the batch is terminated by the user prior to capacity being added, the remaining packages not yet loaded are reassigned to Batch 3.

Pre-check begins with selection of consolidation pallets among available types. The selected consolidation pallets (still empty) are then test loaded into the shipping containers. For the test loading the tare weight of the consolidation pallets are considered together with a minimum height only sufficient to accommodate the Batch 2 boxes with lowest height. During the test loading the maximum possible weight (due to stacking strength constraint) as well as maximum possible height (due to container ceiling or other package above limiting) is registered given the position in the shipping container, and in relation to packages loaded in Batch 1. If there are empty consolidation pallets that do not fit inside the shipping container a new combination is selected and test loaded. If all consolidation pallets fit, or if there is no relevant combination left to try, all successfully test loaded consolidation pallets are transferred to Batch 2. In this transfer, the weight capacity as well as inner height are updated for consolidation pallets according to limitations inherited from the predetermined placement. Additionally, the consolidation pallets are given the attributes that no other packages can be stacked on top of them, neither similar nor different ones. This is because the consolidation pallets have no solid lid protecting the packages, as indicated in Section 2.2.3. The predetermined placements will be executed in Batch 3 when the consolidation pallets are no longer empty.

In Batch 2 the packages assigned to Batch 2 are loaded onto the consolidation pallets received from the preceding Pre-check. If the limit for number of attempts is reached without successfully fitting all packages on the consolidation pallets, the procedure restarts at Pre-check with a new selection of consolidation pallets followed by test loadings. If instead Batch 2 is terminated by the user prior to initiating another Pre-check, or if there is no relevant combination of consolidation pallets left to try, the remaining packages not yet loaded are reassigned to Batch 3.

In Batch 3 the consolidation pallets that are filled with packages from Batch 2 are loaded into the shipping containers at the positions predetermined in Pre-check. Since all constraints have been checked during Pre-check, all consolidation are guaranteed to be loaded. Any remaining packages that either were assigned to Batch 3 from the start, or that were reassigned to Batch 3 from Batch 1 or Batch 2, are also loaded into the shipping containers. If the limit for number of attempts is reached without successfully fitting all the remaining packages, the execution is terminated anyway and the user is provided with the best load plan available.

Based on experiments with real problem instances we know that this is typically not an issue if the remaining packages in Batch 3 only consist of packages that were initially assigned to Batch 3, given the criteria for grouping. Typically it is also not an issue to fit some packages that were reassigned from Batch 2 to Batch 3. However, packages that were reassigned to Batch 3 from Batch 1 after not being able to fit during Batch 1 despite many attempts are (as expected) very unlikely to fit during Batch 3. Such packages will most likely remain leftover, i.e. not included in the complete load plan.

Further information on experiments and results are provided in Chapter 6, but we would like to remark already now that Batch 1 typically is by far the most challenging batch for real problem instances. To clarify, the challenge entirely lies in the loading. The selection is very straight forward as the number of shipping containers and consolidation pallets are typically very low, as described in Section 2.2.2 and 2.2.3 respectively.

We would like to make some final comments on the benefits with generating partial load plans rather than loading all packages at once, i.e. to only have one batch. First, it reflects how the DC operators plan and perform loading manually, which is helpful when discussing the software with DC operators as well as other personnel at the DC. Second, it facilitates design with various objectives and constraints for different stages of the loading. We have found this to be very valuable in order to generate solutions that are feasible in practice. This aspect becomes apparent in Tables 5, 6, 7 and 8 in Sections 4.3.2, 4.3.3, 4.4.2 and 4.4.3 respectively. Third, by breaking up the total problem in subproblems the time complexity is reduced. During each attempt to generate a load plan fewer potential placements need to be evaluated. This aspect becomes more apparent when discussing our placement heuristic algorithm in more detail in Section 4.4.1.

4.1.2 Overall algorithm

The overall algorithm is presented in three steps. The first two steps result in conceptual modules that are integrated in the third step.

In the first step the batch concept is generalized to a design in which two variants of container selection procedures and four variants of container loading procedures run iteratively. Figure 26 is the conceptual module from this step.

A fairly simple complete enumeration algorithm is used for the container selection. We directly refer to Section 4.2 which treats the selection algorithm sufficiently. Instead, the second step is devoted to a high level explanation of our container loading approach. Zhao et al. (2016) present heuristic approaches as combinations of placements heuristics and improvement heuristics, as discussed in Section 3.5. Our extensive placement heuristic guided by a Genetic Algorithm is presented according to this. Figure 27 is the conceptual module from this step.

Finally, in the third step the modules are integrated according to Figure 25 in order to construct the schematic of the overall algorithm in Figure 29. In Figure 29 it is also outlined how Sections 4.2-4.5 are related to each other.

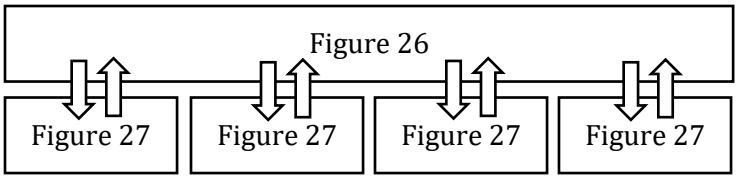

Figure 25. Integration of conceptual modules to construct schematic of overall algorithm

Generalization of batch concept

Recall that *consolidation pallets* and *shipping containers* may be referred to as simply *containers*. Also recall that *boxes* may be used instead of *packages* as suggested by the literature.

The generalization of container selection is fairly straight forward as both consolidation pallets and shipping containers are selected to minimize capacity or cost, constrained by volume and weight of boxes, as indicated in Sections 2.2.2 and 2.2.3 respectively. The generalization of container loading is less intuitive given that the height of consolidation pallets may be altered between Pre-check and Batch 2, as opposed to shipping containers that have fixed dimensions, as described in Section 4.1.1. This aspect was addressed already in Section 3.3.1 on the topic of assortment of large objects, where it was stated that we desired to develop a general approach. During Pre-check consolidation pallets are test loaded with the height set to the minimum height that would be needed to at least accommodate the smallest boxes in Batch 2, as explained in Section 4.1.1. In particular it was mentioned that the maximum possible height of the consolidation pallet given the position in the shipping container is "registered", and that the inner height of the consolidation pallets is "updated" based on this.

To articulate the test loading in a more general way, consider Pre-check to be loading of boxes (empty consolidation pallets with minimum relevant height) into containers (shipping containers). Similarly, consider Batch 2 to be loading of boxes (packages) into containers (empty consolidation pallets with altered inner height). The remaining articulations of Batch 1 and Batch 3 are more intuitive. Consider Batch 1 to be loading of boxes (packages) into containers (shipping containers). Finally, consider Batch 3 to be loading of boxes (filled consolidation pallets and packages) into containers (shipping containers).

Given the generalization to container selection and container loading for each batch, Figure 24 from Section 4.1.1 can be extended to Figure 26 below, which is the first conceptual module needed to construct the schematic of the overall algorithm:

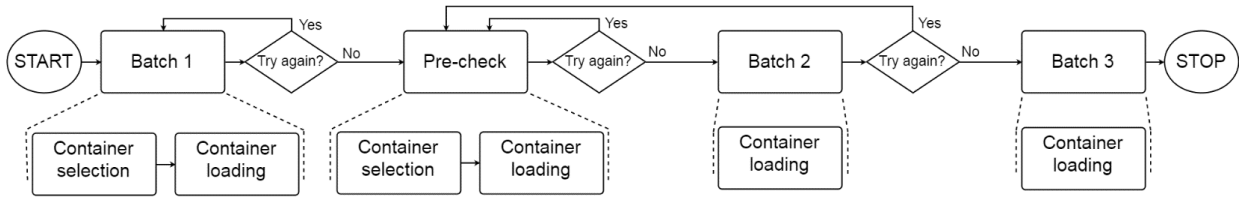

Figure 26. Batch concept generalized - the first conceptual module

Given the generalization of container loading to be loading of boxes into containers it is now possible to discuss container loading in a broader sense.

Containers are loaded to maximize an objective function designed to describe to what extent the objectives for loading listed in Section 2.3.2 are achieved for a complete load plan. The constraints for loading assessed in Section 3.4 are designed as either hard or soft constraints, or merely incorporated in an objective function. Further, the constraints apply for either a complete load plan or immediately upon a placement. For example, weight distribution (center of gravity) is not set until the last box is placed, while stacking feasibility is sufficient to check upon the placement of each box. Most of our constraints can be checked directly upon placement.

A few different types of mechanisms are needed based on the above. First, a mechanism to place boxes. Second, a mechanism to evaluate how good a load plan is when finished. Third, a mechanism that guides the first mechanism to generate load plans that are evaluated as good by the second mechanism. This leads to the design of what Zhao et al. (2016) framed as placement and improvement heuristics.

Placement and improvement heuristics

The container loading approach is designed based on the need for mechanisms to place boxes, to evaluate load plans, and to guide the placement of boxes so that good load plans are generated. Some key elements for the approach are found in Section 3.5.

We place boxes one at a time rather than using a wall building or layer building mechanism, as advised by Zhao et al. (2016). We consider potential placements as combinations of Box-Orientation-Container-Point, abbreviated as BOCP. This facilitates loading of multiple containers simultaneously. A set of BOCPs is denoted as $BOCP$ and a single BOCP within the set is denoted as $BOCP(k)$, where k is the identifier of a potential placement. In certain phases of the algorithm it is appropriate to separately consider combinations of Box-Orientation and combinations of Container-Point. These sets are denoted as BO and CP respectively, and hence $BOCP$ can be generated by combining BO and CP in a nested loop.

It is also necessary to have a notation of boxes to be placed that are not associated with a particular placement. Since boxes are assigned to batches, the notation $Batch(b)$ is used to access properties of a box b assigned to the current batch, for example volume of a box, denoted as $Batch(b).Volume$. It is beneficial to easily access certain overarching properties of the current batch as well, for example volume of all boxes, denoted as $Batch.VolumeOfBoxes$ (instead of $\sum_b Batch(b).Volume$).

We use dynamic sequencing rather than static sequencing, inspired by Zhao et al. (2016), Wang et al. (2010) and Tarantilis et al. (2009). Zhao et al. (2016) state that some kind of scoring is typically used to generate the order in which boxes are placed, and we find the term *BOCP score* appropriate for our approach, denoted as $S(k)$. The BOCP score $S(k)$ is designed to incorporate many different aspects of the current situation upon each placement. Several suggestions are listed in Section 3.5.3, including consideration of available spaces (unutilized volume), characteristics of remaining boxes, and to what extent the sides of boxes are touching other boxes or the container walls. These and other aspects incorporated in $S(k)$ are referred to as BOCP score components and denoted as $s(j,k)$, where j is the identifier of a component. Hence, each $s(j,k)$ describes some aspect of the current situation relative a potential placement $BOCP(k)$.

All BOCP score components are designed so that a higher value $s(j,k)$ indicates that the placement is anticipated as smarter, and vice versa, considering one aspect of the current situation. Hence, the BOCP score $S(k)$ indicates how smart the placement is anticipated to be considering all evaluated aspects of the current situation. This means that the $BOCP(k^*)$ with highest score among all available, i.e. $S(k^*) \geq S(k), \forall k \neq k^*$, is the most desirable and hence should be the next placement. Since components may be of different importance when generating the order in which boxes are placed, each $s(j,k)$ is assigned a coefficient $w(j)$ to weighting the score component's impact. Let $\tilde{S}(k)$ denote a simplified function for BOCP score of a potential placement $BOCP(k)$. This simplified function is further developed in Section 4.4.3:

$$\tilde{S}(k) = \sum_j w(j) \times s(j,k)$$

The work by Techanitisawad and Tangwiwatwong (2004) is very helpful in order to combine our placement heuristic introduced above with an improvement heuristic. Techanitisawad and Tangwiwatwong (2004) propose a Genetic Algorithm (GA) with chromosomes representing values of weighted coefficients in two evaluation functions within their placement heuristic. Techanitisawad and Tangwiwatwong (2004) use static sequencing, but we argue that their use of GA is appropriate for dynamic sequencing as well, as noted in Section 3.5.2.

Our scoring is therefore constructed in a similar manner, namely with GA chromosomes representing the values of $w(j), \forall j$. In other words, each $s(j,k)$ is related to a chromosome of an individual in the GA population. This means that each individual in the GA population represents a unique set of coefficients for the BOCP score. An individual in the GA population is denoted as i, and hence the extended notations $S_{pop}(i,k)$ and $w_{pop}(i,j)$ are also used, with pop to emphasize the relation to the GA population. Moreover, $W_{pop}(i)$ is used to denote $w_{pop}(i,j), \forall j$. Let $\tilde{S}_{pop}(i,k)$ denote a simplified function for BOCP score of a potential placement $BOCP(k)$, and a particular individual i in the GA population. This simplified function is further developed in Section 4.4.3:

$$\tilde{S}_{pop}(i,k) = \sum_{j} w_{pop}(i,j) \times s(j,k)$$

Based on the above, the objective for the GA is to find a set of coefficients $W_{pop}(i)$ that results in smart placements, which in turn result in a good load plan, denoted as $L_{pop}(i)$. A load plan is a set of performed placements, and $L_{pop}(i).BOCP$ is used to denote the placements that were performed. It is beneficial to easily access certain overarching properties of a load plan, for example volume of all boxes, denoted as $L_{pop}(i).VolumeOfBoxes$ (instead of $\sum_k L_{pop}(i).BOCP(k).BO.Box.Volume$). It is also beneficial to easily access properties of a given container in a load plan, for example center of gravity denoted as $L_{pop}(i,c).CG$, where c is the identifier of the container.

Since a good load plan is the overarching goal for the container loading procedure, it is appropriate to define the fitness for each individual in the GA population as the evaluation of the corresponding load plan. This is referred to as *load plan fitness* and denoted as $F_{pop}(i)$. The fitness function used by Techanitisawad and Tangwiwatwong (2004) is designed in a manner comparable to ours, even though their evaluation is less inclusive than what we need, as discussed in Section 3.5.2. We incorporate all objectives for loading listed in Section 2.3.2 in the evaluation of load plan fitness. More specifically how load plan fitness is calculated is treated in Section 4.3.

Finally, coefficient sets (chromosome values), load plans and fitness values for the entire GA population, i.e. $\forall i$, are denoted as W_{pop}, L_{pop} and F_{pop} respectively.

Mechanisms and notations to place boxes, evaluate load plans and guide the placement of boxes so that good load plans are generated have been introduced. Figure 27 is a high-level sketch of the container loading procedure based on the above, which is the second conceptual module needed to construct the schematic of the overall algorithm:

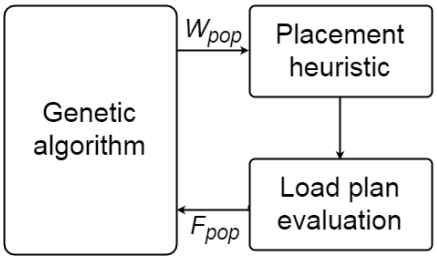

Figure 27. Container loading procedure - the second conceptual module

Figure 27 shows that the Genetic algorithm sends out coefficient sets W_{pop} and receives fitness values F_{pop} in return.

Now, recall that two different types of objective functions have been introduced. The first type was the BOCP score $S_{pop}(i,k)$ which describes how smart a placement $BOCP(k)$ is expected to be in order to generate a good load plan. The second type was the load plan fitness $F_{pop}(i)$ which describes how good a complete load plan is. $S_{pop}(i,k)$ is calculated in the placement heuristic, while $F_{pop}(i)$ is calculated in the load plan evaluation. Figure 28 is an extended version of Figure 27, combined with a downsized version of Figure 32 presented in Section 4.4.1, to highlight where the two different objective function are located within the container loading procedure:

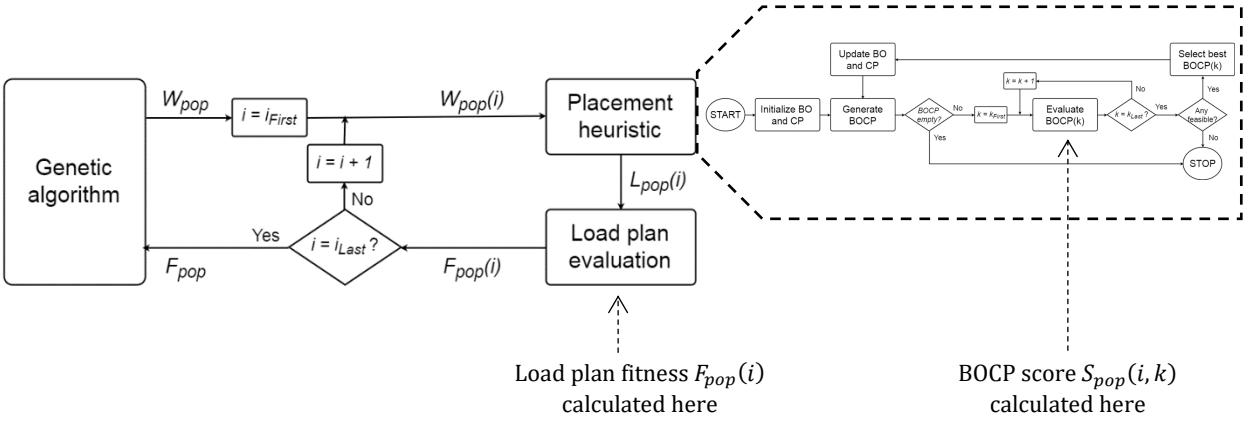

Figure 28. Where the two different objective functions are located within the container loading procedure

The GA procedure sends out BOCP score coefficients W_{pop} for the entire GA population. The placement heuristic and the load plan evaluation are then executed for each individual i in the GA population, from i_{First} to i_{Last}. In each iteration, the placement heuristic receives the coefficients $W_{pop}(i)$ for the current individual. Inside the placement heuristic, the load plan is generated by performing the placements with highest BOCP score $S_{pop}(i,k^*)$. The placement heuristic sends out a complete load plan $L_{pop}(i)$ to be evaluated. The load plan evaluation receives the load plan $L_{pop}(i)$ and calculates load plan fitness $F_{pop}(i)$. The load plan evaluation sends out the load plan fitness $F_{pop}(i)$.

If the current individual i is not the last individual in the GA population, the set of coefficients $W_{pop}(i)$ for the next individual is selected and another iteration with placement heuristic followed by load plan evaluation is executed. If instead the current individual i actually is the last individual in the GA population, the set of fitness values F_{pop} for all individuals i in the GA population are sent to the GA procedure.

Based on the fitness values F_{pop} of the individuals of the current GA population, a new GA population is produced as a combination of clones and children. The GA procedure then sends out BOCP score coefficients W_{pop} for the new generation and the cycle repeats. The load plan $L_{pop}(i^*)$ corresponding to the individual i^* with $F_{pop}(i^*) \geq F_{pop}(i), \forall i \neq i^*$ is considered to be the best load plan. This load plan will be returned from the container loading procedure when the last generation has finished, or whenever the execution is terminated by the user.

With the desire to relate Figure 27 and 28 to the term improvement heuristic used by Zhao et al. (2016) we argue that the GA and the load plan evaluation combined makes up our improvement heuristic. To clarify, it would be possible to run the placement heuristic once with a fixed set of weighted coefficients $W_{pop}(i_{fixed})$ and without load plan evaluation to calculate $F_{pop}(i_{fixed})$ afterwards. We do however believe it would be very difficult to develop a $W_{pop}(i_{fixed})$ that would work for any problem instance.

To elaborate on this statement, let us suppose that our simplified BOCP score $\tilde{S}_{pop}(i_{fixed}, k) = \sum_j w_{pop}(i_{fixed}, j) \times s(j, k)$ includes a component $s(j_{touching}, k)$ reflecting a particular aspect of the current situation, namely to what extent the box is touching other boxes and/or the container walls. Let us also suppose that we have two different problem instances at hand. Given these suppositions, we may conceive that for the first problem instance it turns out to be very beneficial to reward this aspect, in other words to have a high value on $w_{pop}(i_{fixed}, j_{touching})$, in order to generate a good load plan. At the same time, we may conceive the opposite for the second problem instance, namely that it turns out to be devastating to reward this aspect in order to generate a good load plan. Instead, for the second problem instance, it turns out to be beneficial to have a low value on $w_{pop}(i_{fixed}, j_{touching})$, and instead reward some other aspects more.

Since we have included many aspects of the current situation in our scoring, we do value the dynamic weighting which the improvement heuristic offers. By this statement we suggest that the GA over generations finds $W_{pop}(i)$ that are tuned for the particular problem instance being solved. To enclose the example, the GA would over generations recognize that it is beneficial to have a high value on $w_{pop}(i, j_{touching})$ for the first problem instance, but instead beneficial to have a low value on $w_{pop}(i, j_{touching})$ for the second problem instance. This topic is revisited in Section 6.5.

Schematic of overall algorithm

Figure 26 and 27 that have been presented earlier in this section were introduced as conceptual modules. Figure 26 is a module in which two variants of container selection procedures and four variants of container loading procedures run iteratively. Figure 27 is a module in which one variant of container loading procedure runs iteratively. To solve our total problem, the modules are integrated according to Figure 25, namely with one instance of Figure 26 and four instances of Figure 27.

The result of the integration is the schematic of the overall algorithm in Figure 29. We want to emphasize that each container loading procedure initiates entirely new coefficient sets W_{pop}, i.e. GA populations are never transferred between batches. In Figure 29 we have inserted section numbers in parenthesis to direct the reader where respective part of the overall algorithm are treated in more detail.

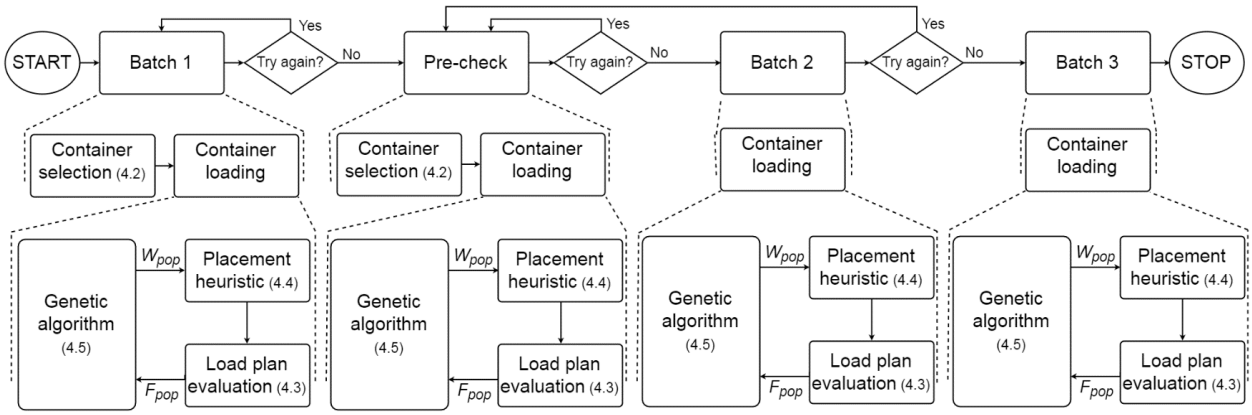

Figure 29. Schematic of overall algorithm - conceptual modules in Figures 26 and 27 integrated

Section 4.2 on the container selection provide some more insight on how the complete enumeration is performed. Section 4.3 on the load plan evaluation covers load plan fitness, the constraints applicable to a complete load plan, and how all of this is incorporated in an algorithm. We believe it is helpful to establish what a good load plan is prior to drilling into the inner workings of the placement heuristic, as the former impact the design of the latter. Section 4.4 on the placement heuristic covers BOCP score, the constraints applicable immediately upon a placement, and how all of this is incorporated in an algorithm. Section 4.5 on the GA provide some more insight on what choices we have made regarding genetic operators such as selection of parents, the crossover procedure and parameter settings.

Throughout Section 4.3-4.4 various objectives and constraints are treated in many different ways. Section 4.6 is a summary of the approach focused on objectives and constraints, aimed to highlight how the container loading approach is linked to all the objectives from Section 2.3.2 and all the constraints from Section 3.4.

We believe it is important for the reader to understand how load plan fitness and BOCP score are related in the overall algorithm, and in particular to the Genetic Algorithm, prior to undertaking Section 4.3-4.5. Figure 28 and the segments before and after this figure are intended to help the reader accurately distinguish between load plan fitness and BOCP score.

4.2 Container selection

The container selection is accomplished with a fairly simple complete enumeration algorithm. In our approach the objective is to minimize volume capacity of containers. The volume capacity could easily be replaced with cost of containers. By a *container combination* we refer to the number of containers of each available container type. For example, a combination of shipping containers in Batch 1 could be simply one 20 ft. container. An example of combination of consolidation pallets in Pre-check could be two "GREEN-45x30x38-bottom" and one "YELLOW-45X45X38-bottom". The steps of the container selection algorithm are outlined below:

Step 1. Calculate minimum volume and minimum weight based on boxes to be loaded
Step 2. Generate "relevant" container combinations based on available container types that satisfy minimum volume and minimum weight, avoid any combinations selected in previous attempts
Step 3. Sort the generated container combinations based on volume capacity
Step 4. Select the container combination with the lowest volume capacity
Step 5. Update the history of container combinations selected in previous attempts

"Relevant" in Step 2 refers to avoiding any container combinations that are redundant given the history of combinations selected in previous attempts. For example, suppose a selection of consolidation pallets in Pre-check for which "GREEN-45x30x38-bottom" and "YELLOW-45X45X38-bottom" are the two available types. Further, suppose that the combination consisting of two "GREEN-45x30x38-bottom" and one "YELLOW-45X45X38-bottom" (as in the second example above) is generated in Step 2. The latter supposition implies that this combination satisfies minimum volume and minimum volume, as well as that this combination has not been selected in a previous attempt. Now, examples of redundant, i.e. not relevant, combinations are three (or more) "GREEN-45x30x38-bottom" and one "YELLOW-45X45X38-bottom", as well as two "GREEN-45x30x38-bottom" and two (or more) "YELLOW-45X45X38-bottom". These combinations that are currently not relevant may of course become relevant in a potential later container selection attempt when the history is longer.

We want to stress that our container selection approach is based on low number of available container types as well as low total number of containers for our problem instances, as mentioned in Section 2.2.2 and 2.2.3. The algorithm outlined above scales badly, so for problems with significantly higher number of available container types and/or total number of containers our container selection approach may result in too high time complexity.

4.3 Load plan evaluation

Calculation of load plan fitness and checks of constraints applicable to a complete load plan are incorporated in an evaluation algorithm.

Recall the role of the load plan evaluation by revisiting Figure 28 in Section 4.1.2. Figure 30 is a snippet of Figure 28 which shows that the load plan evaluation receives a single load plan $L_{pop}(i)$ and calculates load plan fitness $F_{pop}(i)$ for this load plan, which is then forwarded. Since the load plan evaluation only deals with one individual i of the GA population at a time, the notation within the evaluation for load plan and load plan fitness are simplified to L and F respectively.

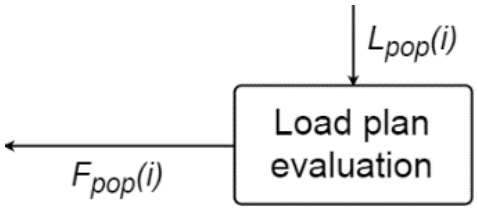

Figure 30. The role of the load plan evaluation

In Section 4.3.1 it is explained how the evaluation algorithm scans through each container in the load plan to calculate fitness and check constraints. In Section 4.3.2 it is presented in more detail how load plan fitness is calculated to describe to what extent the objectives for loading listed in Section 2.3.2 are achieved. This was on purpose left out from Section 4.1.2 as an attempt to make the presentation of the overall algorithm more comprehensible. In Section 4.3.3 it is presented in more detail how the soft load plan constraints are checked. Some constraints form Section 3.4 that are applicable to a complete load plan are merely incorporated in the load plan fitness presented in Section 4.3.2. As noted in Section 4.1.1 the batch concept facilitates design with various objectives and constraints for different stages of the loading. Tables 5 and 6 in Sections 4.3.2 and 4.3.3 respectively demonstrate how this opportunity is utilized with different configurations per batch.

4.3.1 Load plan evaluation algorithm

Load plan fitness F and constraints applicable to a complete load plan are evaluated based on the performed placements in each container of the load plan, i.e. $L(c).BOCP, \ \forall c$. A component of the load plan fitness F is denoted as $f(l)$. How $f(l), \forall l$ relates to F is presented in Section 4.3.2.

The load plan evaluation algorithm loops over the containers in the load plan and for each container c a nested loop over $L(c).BOCP$ is executed. In the inner BOCP loop, for each $L(c).BOCP(k)$ some aspects relative the other placements $L(c).BOCP(\hat{k}), \forall \hat{k} \neq k$ are evaluated, for example to what extent the other boxes are touching the particular box. At the same time, some aspects of $L(c).BOCP(k)$ are not dependent on the other $L(c).BOCP(\hat{k})$ and hence these aspects are evaluated in the outer BOCP loop, for example whether the particular box is placed on the container floor.

Since the load plan fitness F reflects the loading of all containers in the load plan, the fitness components $f(l)$ are not calculated until all containers have been scanned. However, preparing fitness calculations as well as checks of constraints do naturally take place both after the inner BOCP loop as well as after the outer BOCP loop for each container. After the last container has been scanned and all fitness components $f(l)$ have been calculated, the final calculation of load plan fitness F occurs.

4.3.2 Load plan fitness

As explained in Section 4.3.1, the calculation of load plan fitness F using the fitness components $f(l)$, $\forall l$ is the very last step in the load plan evaluation algorithm. First in this section the design of the fitness function is explained in general terms and second the different fitness function components are explained in more practical terms

Design of fitness function

An important characteristic of all fitness components $f(l)$ is that they are designed to only adopt values between 0 and 1, i.e. $0 \leq f(l) \leq 1, \forall l$, where $f(l) = 1$ always is the most desirable. For certain l it is possible to obtain $f(l) = 0$ as well as $f(l) = 1$, while for other l real problem instances will almost always result in $0 < f(l) < 1$. This design allows for an interpretation that the load plan fitness F is related to the distance to a truly optimal (but most likely practically impossible) point in the fitness component space where $f(l) = 1, \forall l$. This distance can be expressed as

$$\sum_{l} [1 - f(l)]^2$$

F is also designed to also only adopt values between 0 and 1, i.e. $0 \leq F \leq 1$, where $F = 1$ is most desirable. Hence F can be considered as the complement to this distance and expressed as

$$1 - \sum_{l} [1 - f(l)]^2$$

Based on experiments we found that it is beneficial to give certain fitness components $f(l)$ more weight in this calculation, in other words to penalize greater distance to the truly optimal point more for certain l. This weighing is however fixed, as opposite to the weighted coefficient $w(i,j)$ for the BOCP score. The fixed coefficients for weighting $f(l)$ are denoted as $c(l)$. To maintain $0 \leq F \leq 1$ the weighted distance is scaled down. Finally, the function for the load plan fitness is formulated as:

$$F = 1 - \frac{1}{\sum_l c(l)} \sum_{l} c(l)[1 - f(l)]^2$$

Overview of fitness function components

In Table 5 it is presented which fitness components $f(l)$ are included in the load plan fitness F based on which batch is being executed in the overall algorithm. In total 18 different fitness components have been developed and the number of components used for each batch varies from 7 to 15. In the leftmost column of Table 5 the associated objectives for loading from Section 2.3.2 are referred to.

Table 5. Load plan fitness components per batch with the associated objectives

Objective	Load plan fitness component $f(l)$	Batch 1	Pre-check	Batch 2	Batch 3
1.1	Leftover boxes none	✓	✓	✓	✓
1.1	Leftover boxes fewer	✓	✓	✓	✓
1.2	Leftover boxes volume lower	✓	✓	✓	✓
1.3	Leftover boxes weight lower	✓	✓	✓	✓
2.1	Promising spaces for consolidation pallets	✓			
2.1	Consolidation pallets possible height		✓		
2.1	Consolidation pallets possible weight		✓		
2.1	Consolidation pallets closer to floor		✓		✓
2.2	Horizontal stability behind boxes	✓	✓	✓	✓
2.2	Horizontal stability in front, left and right of boxes	✓	✓	✓	✓
2.2	Utilization of floor space higher	✓	✓	✓	
2.2	Soft constraints for boxes satisfied	✓	✓		✓
3.1	Center of gravity lengthwise closer to center	✓			
3.2	Center of gravity widthwise closer to center	✓			
3.3	Center of gravity heightwise closer to floor	✓			
4.1	Boxes with pallet not rotated	✓	✓		✓
4.2	Boxes with pallet reachable by forklift	✓	✓		✓
4.3	Boxes without pallet not on floor	✓			✓
All	**Total number of fitness components**	**15**	**13**	**7**	**11**

Introduction to examples

Four more detailed examples on how fitness components are calculated follows.

In the examples helper functions h are used for the preparing fitness calculations that are needed prior to calculating actual fitness components. Recall the notation $Batch$ as all boxes assigned to the batch that is being executed in the overall algorithm.

To enable a more practical, non-binary, evaluation of certain aspects sigmoid/logistic curve is used and denoted as $SCurve(x, midpoint, steepness)$ where x is the input value and $midpoint$ and $steepness$ are parameters. Our $SCurve$ function can be found in Section 9.2.1. In the examples below our values on these and other parameters are written to avoid additional notation. An exception to this is $CLOSE$, which is a constant, and more specifically a small decimal number to count almost adjacent surfaces as adjacent. Further, for Example 2 below we refer to Example 2 in Section 4.4.3 for details on "is supporting from behind" and "contact area of".

In the first example below $f(l_{LeftoverBoxesVolume}) = 1$ if all boxes in the current batch are loaded. In the second example $f(l_{HorizontalStabilityBehind}) = 1$ if all loaded boxes has adequate support from behind. In the third example $f(l_{SoftConstraintsBoxes}) = 1$ if none of the loaded boxes break a load plan constraint associated with an individual box. In Section 4.3.3 we will present the checks that precede the calculation of $f(l_{SoftConstraintsBoxes})$. In the fourth example $f(l_{CenterOfGravityWidthwise}) = 1$ if all containers in the load plan have adequate weight distribution widthwise.

Example 1: Leftover boxes volume lower

After container loop:

$$f\left(l_{LeftoverBoxesVolume}\right) = 1 - \frac{Batch.VolumeOfBoxes - L.VolumeOfBoxes}{Batch.VolumeOfBoxes}$$

Example 2: Horizontal stability behind boxes

Inside inner BOCP loop ($\forall \hat{k} \neq k$):

$$\boldsymbol{if}\ L(c).BOCP\left(\hat{k}\right)\ is\ supporting\ L(c).BOCP(k)\ from\ behind\ \boldsymbol{then}$$

$$h_{ContactAreaBehind}(k) = h_{ContactAreaBehind}(k) + contact\ area\ of\ L(c).BOCP\left(\hat{k}\right)\ and\ L(c).BOCP(k)$$

Inside outer BOCP loop ($\forall k$):

$$\boldsymbol{if}\ L(c).BOCP(k).CP.Point.X < CLOSE\ \boldsymbol{then}$$

$$h_{ContactAreaBehind}(k) = L(c).BOCP(k).BO.SideArea.Width$$

$$h_{ContactAreaBehindSum} = h_{ContactAreaBehindSum} + SCurve\left(\frac{h_{ContactAreaBehind}(k)}{L(c).BOCP(k).BO.SideArea.Width}, 0.3, 20\right)$$

After container loop:

$$f\left(l_{HorizontalStabilityBehind}\right) = \frac{h_{ContactAreaBehindSum}}{L.NbrOfBoxes}$$

Example 3: Soft constraints for boxes satisfied

After container loop:

$$f\left(l_{SoftConstraintsBoxes}\right) = 1 - \frac{h_{BreakingConstraintsSum}}{L.NbrOfBoxes}$$

Example 4: Center of gravity widthwise closer to center

Inside outer BOCP loop ($\forall k$):

$$h_{CGNumeratorY}(c) = h_{CGNumeratorY}(c) + L(c).BOCP(k).BO.Box.Weight \times \left(L(c).BOCP(k).CP.Point.Y + \frac{L(c).BOCP(k).CP.PontEnd.Y - L(c).BOCP(k).CP.Point.Y}{2}\right)$$

Outside outer BOCP loop ($\forall c$):

$$L(c).CG.Y = \frac{h_{CGNumeratorY}(c)}{L(c).WeightOfBoxes}$$

$$L(c).CGDiff.Y = \frac{Abs\left(L(c).CG.Y - \frac{L(c).DimInner.Width}{2}\right)}{\frac{L(c).DimInner.Width}{2}}$$

$$h_{CGSumY} = h_{CGSumY} + SCurve(1 - L(c).CGDiff.Y, 0.8, 50)$$

After container loop:

$$f(l_{CenterOfGravityWidthwise}) = \frac{h_{CGSumY}}{L.NbrOfConts}$$

4.3.3 Load plan constraints

As explained in Section 4.3.1, checks of constraints applicable to a complete load plan take place both after the inner BOCP loop as well as after the outer BOCP loop for each container in the load plan evaluation algorithm.

When discussing the generalization of the batch concept in Section 4.1.2 it was concluded that most constraints can be checked directly upon placement, and hence these are covered in Section 4.4 when presenting the placement heuristic. Moreover, any constraints from Section 3.4 that are merely incorporated in the load plan fitness are not covered in this section.

The practical impact of breaking soft constraints is only that the user receives a warning from the software stating that the load plan does not satisfy all soft constraints. Based on feedback from the customer service representative and DC operators the best load plan should always be provided, even though it may have some flaws. Some adjustments upon loading are acceptable, as stated in Section 1.4.

Overview of constraints

In Table 6 below it is presented which soft constraints are applicable based on which batch is being executed in the overall algorithm. In total three different soft constraints for a complete load plan have been developed and the number of constraints used for each batch varies from three to none. In the leftmost column of Table 6 the associated categories from Section 3.4 are referred to.

Table 6. Load plan constraints per batch with the associated categories

Category	Load plan constraint	Batch 1	Pre-check	Batch 2	Batch 3
2	Container stability limits	✓			✓
9	Support behind and in front for slim box not on floor	✓	✓		✓
9	Support behind for not slim box not on floor	✓	✓		✓
All	**Total number of constraints**	**3**	**2**	**0**	**3**

Constraint checks

Below checks for all three constraints are outlined using similar notation as in the fitness component examples in Section 4.3.2.

$L.BreakingConstraints$ is used to denote whether at least one constraint has been broken for the entire load plan, while $h_{BreakingConstraintsSum}$ is used to denote how many boxes have broken at least one constraint. Our values on the constraint thresholds are written to avoid additional notation. 10% is used as a threshold for support from behind. 30% and 20% are used as threshold for relative difference to the center of a container lengthwise and widthwise respectively.

Using $h_{BreakingConstraintsSum}$ to calculate the load plan fitness component $f(l_{SoftConstraintsBoxes})$ despite the existence of $f(l_{HorizontalStabilityBehind})$ is to better handle the most important horizontal stability considerations, based on feedback from the DC operators.

Inside outer BOCP loop ($\forall k$):

$$\textbf{if } L(c).BOCP(k).CP.Point.Z > 0 \textbf{ then}$$

$$\textbf{if } L(c).BOCP(k).BO.Box.Slim \textbf{ then}$$

$$\textbf{if } \frac{h_{ContactAreaBehind}(k)}{L(c).BOCP(k).BO.SideArea.Width} < 0.1$$

$$\textbf{or } \frac{h_{ContactAreaInFront}(k)}{L(c).BOCP(k).BO.SideArea.Width} < 0.1 \textbf{ then}$$

$$L.BreakingConstraints = True$$

$$\textbf{if not } L(c).BOCP(k).CountedAsBreaking \textbf{ then}$$

$$h_{BreakingConstraintsSum} = h_{BreakingConstraintsSum} + 1$$

$$\textbf{else}$$

$$\textbf{if } \frac{h_{ContactAreaBehind}(k)}{L(c).BOCP(k).BO.SideArea.Width} < 0.1 \textbf{ then}$$

$$L.BreakingConstraints = True$$

$$\textbf{if not } L(c).BOCP(k).CountedAsBreaking \textbf{ then}$$

$$h_{BreakingConstraintsSum} = h_{BreakingConstraintsSum} + 1$$

Outside outer BOCP loop ($\forall c$):

$$\textbf{if } L(c).CGDiff.X > 0.3$$

$$\textbf{or } L(c).CGDiff.Y > 0.2 \textbf{ then}$$

$$L.BreakingConstraints = True$$

4.4 Placement heuristic

A load plan is generated by placing boxes one at a time in all the selected containers simultaneously. Generating, evaluating and selecting potential placements are incorporated in a placement heuristic algorithm. The evaluation includes checks of constraints applicable immediately upon a placement as well as calculation of BOCP score.

Recall the role of the placement heuristic by revisiting Figure 28 in Section 4.1.2. Figure 31 is a snippet of Figure 28 which shows that the placement heuristic receives a set of BOCP score coefficients $W_{pop}(i)$ and generates a load plan $L_{pop}(i)$ using these cofficients, which is then forwarded. Since the placement heuristic only deals with one individual i of the GA population at a time, the notation within the heuristic for BOCP score coefficients and load plan are simplified to W and L respectively.

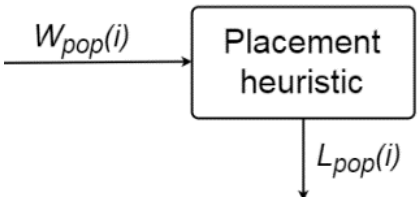

Figure 31. The role of the placement heuristic

The placement heuristic is deterministic in the sense that for a given set of BOCP score coefficients, the very same load plan will always be generated.

In Section 4.4.1 it is explained how the placement heuristic algorithm generates, evaluates and selects potential placements for all containers simultaneously.

In Section 4.4.2 it is presented in more detail how BOCP constraints are checked within the evaluation procedure. Constraints from Section 3.4 that are applicable immediately upon a placement and designed as hard are presented here, while constraints merely incorporated in the BOCP score are treated in Section 4.4.3. If any hard constraint are not satisfied for a potential placement the evaluation is immediately terminated and hence scoring is not performed for that placement.

In Section 4.4.3 it is presented in more detail how BOCP score is calculated to describe how smart the placement is anticipated to be considering the current situation, as discussed in Section 4.1.2. A placement is smart if it contributes to achieving high load plan fitness in the load plan evaluation that succeeds the placement heuristic in the overall algorithm. The function for the BOCP score of a potential placement introduced in Section 4.1.2 is further developed in Section 4.4.3.

The BOCP constraints and BOCP score are configured per batch similar to load plan fitness components and load plan constraints in Section 4.3.2 and 4.3.3 respectively. Tables 7 and 8 in Sections 4.4.2 and 4.4.3 respectively demonstrate this.

4.4.1 Placement heuristic algorithm

First in this section a schematic of the overall procedure of the heuristic algorithm is presented, second a simple illustrative example of the overall procedure is drawn and third some aspects of the sub procedures are explained in more detail. The main things we want to convey in this section is how new potential placement points are generated and where constraint checks as well as scoring occur.

Schematic of placement heuristic algorithm

In the placement heuristic sets of potential placements structured as Box-Orientation-Container-Points, denoted as $BOCP$, are generated and evaluated iteratively. The overall procedure is illustrated in Figure 32.

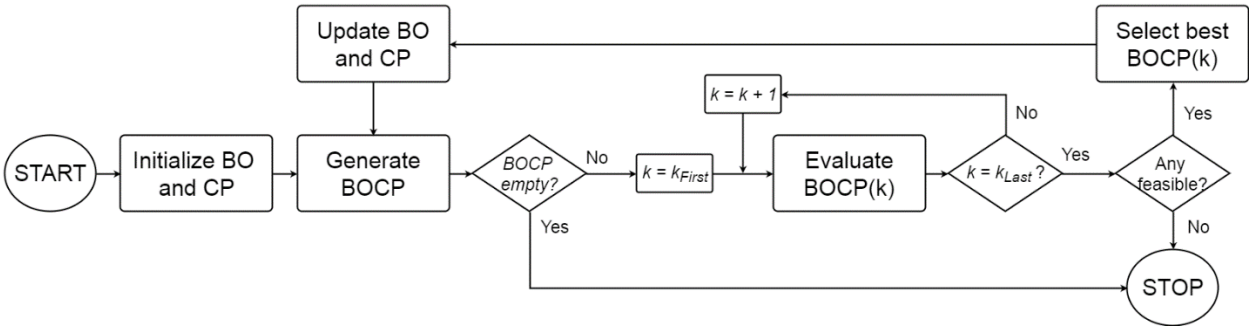

Figure 32. Schematic of placement heuristic algorithm - how a load plan is generated

The procedure illustrated in Figure 32 begins with initializing sets of Box-Orientations denoted as BO and sets of Container-Points denoted as CP. A nested loop over BO and CP is executed to generate initial $BOCP$.

Each $BOCP(k)$ is then evaluated by first checking BOCP constraints and second, if all hard constraints are satisfied, calculating BOCP score. Any $BOCP(k)$ that does not satisfy all hard constraints is marked infeasible, i.e. $BOCP(k).Feasible = False$. If a $BOCP(k)$ is infeasible and it is possible to determine that it will never become feasible it is also marked to be rejected, i.e. $BOCP(k).Rejected = True$. For example, if $BOCP(k)$ is not entirely inside the container it will be both infeasible and rejected, since this will not change throughout the loading. However, for example, if $BOCP(k)$ does not receive sufficient vertical support (e.g. not sufficient ratio of base area supported by boxes below), it will be infeasible but not rejected, since this might change as other placements are made.

When $BOCP(k_{last})$ has been evaluated, the placement with highest BOCP score is selected such that $S(k^*) \geq S(k), \forall k \neq k^*$. Thereby $BOCP(k^*)$ is added to $L.BOCP$ as a performed placement. BO and CP are then updated based on $BOCP(k^*)$, more specifically first $BOCP(k^*).BO$ is removed from BO and second CP is re-initialized with up to three new placement points.

The second iteration, as well as all subsequent iterations, begin with storing the potential placements from the previous iteration as $BOCP_{old}$ and then emptying $BOCP$. After these preparations, two steps follow to transfer old potential placements and to generate new potential placements. The first step is to transfer any "relevant" potential placements from the previous iteration, denoted as $BOCP_{old}$, to be potential placements in the current iteration, still denoted as $BOCP$. The second step is a nested loop over BO and CP, to generate new potential placements for the current iteration. What relevant implies, as well as how placements points in CP may be adjusted to generate additional new potential placements such that $BOCP(k).CP \notin CP$, are explained in more depth towards the end of this section.

Each $BOCP(k)$ is then evaluated and the cycle repeats until one of the two termination criteria is met. The first termination criteria is if $BOCP$ is empty after a generation attempt, which occur if all boxes in the batch have been loaded, but also in a couple of other scenarios. The second termination criteria is if all potential placements are infeasible, i.e. $not\ BOCP(k).Feasible, \forall k$. In either case the current $L.BOCP$ is sent out from the placement heuristic.

An illustrative example of the overall procedure

A simple illustrative example of the overall procedure is drawn for a fictive problem instance in Figure 33, consisting of only three boxes to be loaded in one container that is small relative the boxes. Two of the boxes have only one acceptable orientation, while one of boxes has two acceptable orientations. For simplicity any details related to scoring are omitted and it is just assumed that the Box-Orientation-Container-Point selected in each iteration is the one with highest score among all feasible placements. Regarding feasibility only basic constraints such as boxes being entirely inside the container and boxes not intersecting are considered. Only normal and projected placement points are used. To make the illustration cleaner, C1 is not written as the container id for each option, and point id is only written upon creation. An explanation on what happens in each iteration follows.

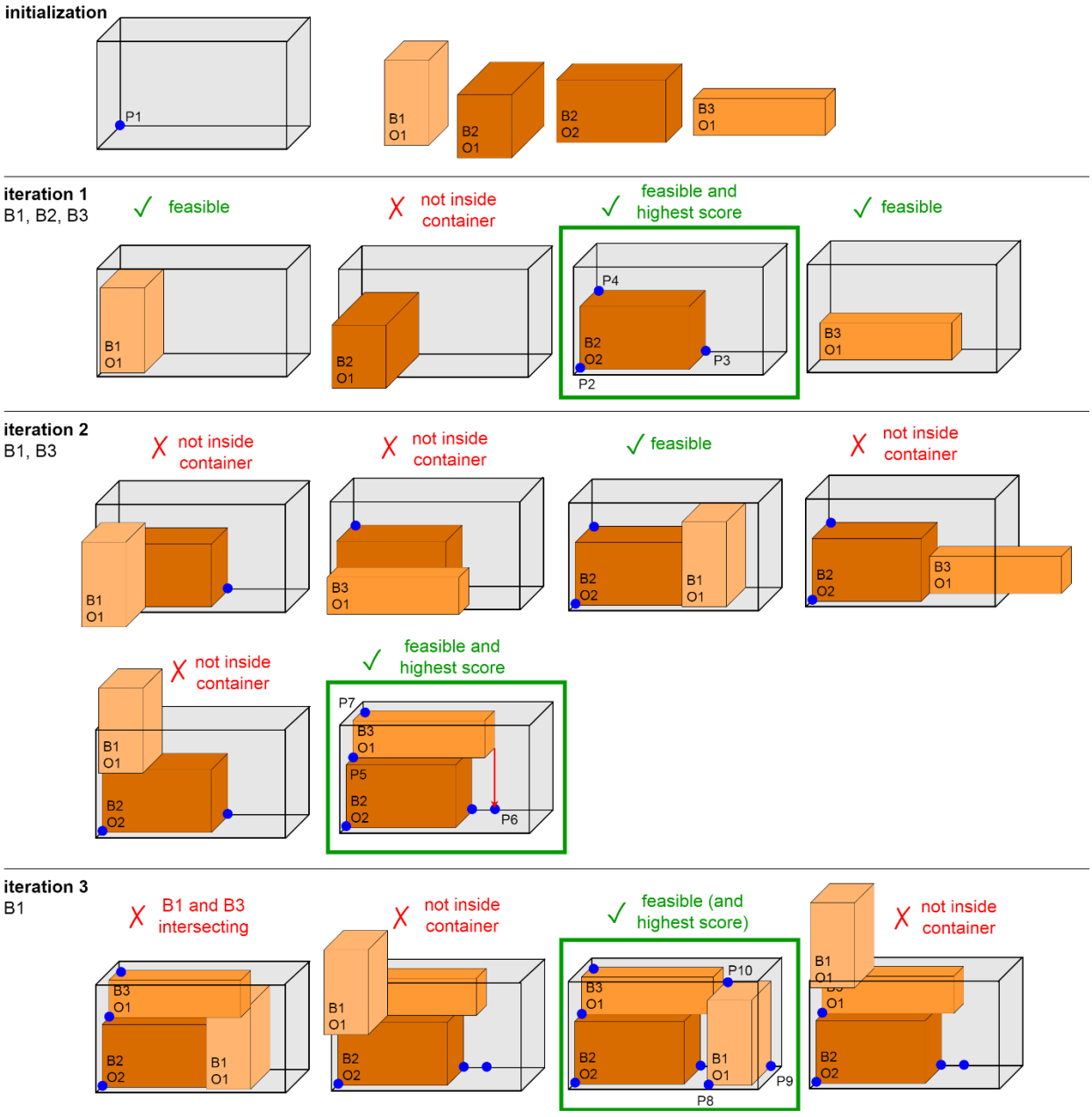

Figure 33. Example of three placement heuristic iterations to place three boxes in one container

BO is initialized with four possible combinations B1O1, B2O1, B2O2, B3O1 and CP is initialized with one possible combination C1P1. $BOCP$ is generated with four possible combinations. These are evaluated and three of four are feasible. Among the three feasible, B2O2C1P1 has the highest BOCP score and is therefore selected. All combinations involving B2 and C1P1 are removed from BO, CP and $BOCP_{Old}$. C1P2, C1P3 and C1P4 are generated as new placement points. $BOCP$ is generated with six possible combinations. These are evaluated and two of six are feasible. Among the two feasible, B3O1C1P4 has highest BOCP score and is therefore selected. All combinations involving B3 and C1P4 are removed from BO, CP and $BOCP_{Old}$. C1P5, C1P6 and C1P7 are generated as new placement points. C1P6 is a projected point. B1O1C1P2 is also removed since it was marked to be rejected as it will never be inside the container. $BOCP$ is generated with four possible combinations. These are evaluated and only B1O1C1P6 is feasible and hence selected. All three boxes have now been loaded.

Initialize BO and CP

Recall what happens in each batch from Section 4.1.1, which imply that in Batch 1 and Batch 2 the placement heuristic begins with empty containers, while in Pre-check and Batch 3 there may already be boxes in the containers when the placement heuristic begins. In other words, $L.BOCP$ is empty for Batch 1 and Batch 2 upon start, but not necessarily for Pre-check and Batch 3.

The initialization of BO is not impacted by this, since it will always be the combinations of all boxes assigned to the batch with acceptable orientations for each box. In Section 3.4.2 it was concluded that each box will have either one or two acceptable orientations, where the pallet orientation matters if the box has a pallet attached.

The initialization of CP is however very much impacted by the differences between batches. In Batch 1 and Batch 2 the initial CP will simply be the origins for each container, defined as the inner-bottom-left corner of the container where $CP.Point.X = CP.Point.Y = CP.Point.Z = 0$. In Pre-check and Batch 3, when starting with a partially loaded container, it is not possible to start with origin since there will always have been a placement made there already, unless no boxes were assigned to Batch 1. Instead, for Pre-check and Batch 3, the initial CP will be the origins of available spaces (unutilized volume) from Batch 1. These available spaces are identified via the evaluation procedure.

Generate BOCP: empty containers

If the containers are empty the generation of $BOCP$ is very straight forward, but this is only the case for the first iterations of the placement heuristic in Batch 1 and Batch 2. A nested loop over BO and CP is executed, and only new potential placements such that $BOCP(k).CP \in CP$ are generated. Based on our design of the evaluation procedure, and in particular how available spaces are identified, efficiency is gained by having the outer loop over CP and the inner loop over BO.

Evaluate BOCP(k)

BOCP score and constraints applicable immediately upon a placement are evaluated based on already performed placements of all the boxes in the particular container $L(c).BOCP$ where c corresponds to $BOCP(k).CP.Container$.

The BOCP evaluation algorithm loops over $L(c).BOCP$ to evaluate aspects of these performed placements relative to $BOCP(k)$, for example to what extent other boxes would be touching the particular box if $BOCP(k)$ would be performed. At the same time, some aspects of $BOCP(k)$ are not dependent on the already performed placements $L(c).BOCP$ and hence these aspects are evaluated separately prior to entering the loop, for example if the particular box is placed on the container floor, or if the box is placed entirely inside the container.

Given our design of constraints and scoring there are also certain aspects of $BOCP(k)$ that are too complex to capture directly within the loop over $L(c).BOCP$. These aspects require additional sub procedures that are called within the evaluation algorithm. Example of such complex aspects are stacking strength constraints and scoring based on available spaces (unutilized volume). With the intent to make the presentation more comprehensible, the evaluation algorithm is not presented at that level of detail. However, some examples are provided in Section 4.4.2 and 4.4.3 to convey the complexity of these aspects.

Constraint checks and preparing scoring calculations occur simultaneously. If a hard constraint is broken, the execution is terminated and the final calculation of BOCP score do not occur. Hard constraints are presented in Section 4.4.2. If instead all hard constraints are satisfied towards the end of the evaluation, calculation of BOCP score components occur. Recall the notation $s(j,k)$ to denote a component of the BOCP score $S(k)$. After all BOCP score components $s(j,k)$ have been calculated the final calculation of BOCP score $S(k)$ occurs. How $s(j,k), \forall j$ relates to $S(k)$ is introduced in a simplified format in Section 4.1.2 and is further developed in Section 4.4.3.

Select best BOCP(k)

Only $BOCP(k)$ that satisfy all hard constraints are further considered as the placement to perform in a given iteration. Among feasible potential placements the selection is simply to pick the placement with highest BOCP score such that $S(k^*) \geq S(k), \forall k \neq k^*$. We actually register this during the evaluation throughout k_{first} to k_{last}, so the selection is in fact already made after k_{last}, assuming at least one potential placement satisfied the hard constraints. $BOCP(k^*)$ is added to $L.BOCP$ as a performed placement.

Update BO and CP

The update of BO is simply to remove all combinations such that $BO.Box = BOCP(k^*).BO.Box$. Then, CP is first emptied, followed by up to three placement points being created and added to CP. These are based on $BOCP(k^*)$, and for certain cases also based on other performed placements $L(c).BOCP(k), \forall k \neq k^*$. Overall we distinguish three main types of placement points and we call them: *normal placement points*, *projected placement points* and *adjusted placement points*. The types normal and projected are treated here, and adjusted is covered below on the topic of generating $BOCP$ for containers that are not empty.

A *normal placement point* refers to a Container-Point that is adjacent to a corner of $BOCP(k^*)$ and created independently of any new potential Box-Orientation. A *projected placement point* refers to a Container-Point that is not adjacent to a corner of $BOCP(k^*)$, but still created independently of any new potential Box-Orientation. The three placement points added when CP is re-initialized may be either normal or projected. There are a few different cases to cover to properly handle the creation of normal and projected points. These cases are not presented in detail, but instead four normal points and one projected point (the rightmost) are illustrated in Figure 34.

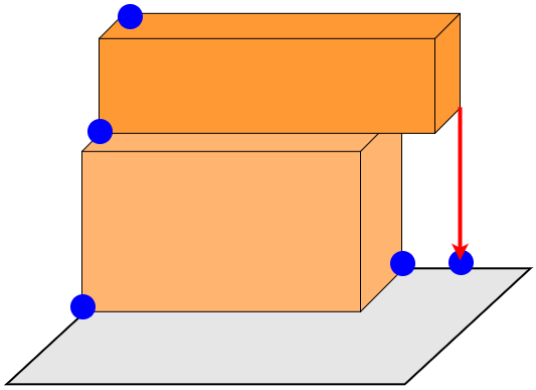

Figure 34. Example of four normal points and one projected point (the rightmost)

Generate BOCP: not empty containers

The generation of $BOCP$ if the containers are not empty is more complicated than in the empty containers case. The generation begin with storing the potential placements from the previous iteration as $BOCP_{old}$ and then emptying $BOCP$. After these preparations, two steps follow to transfer old potential placements, as well as generate new potential placements.

The first step is to transfer any "relevant" potential placements from the previous iteration, denoted as $BOCP_{old}$, to be potential placements in the current iteration, still denoted as $BOCP$. For a potential placement from the previous iteration $BOCP_{old}(k)$ to be relevant in the current iteration it must satisfy:

$$\textbf{\textit{not}}\ BOCP_{old}(k).Rejected$$

$$\textbf{\textit{and}}\ BOCP_{old}(k).BO \neq BOCP_{old}(k^*).BO$$

$$\textbf{\textit{and}}\ BOCP_{old}(k).CP \neq BOCP_{old}(k^*).CP$$

The purpose of maintaining old potential placements instead of simply re-generating all potential placements in each iteration is to enable skipping certain things in the evaluation sub procedure and thereby gaining efficiency.

The second step is a nested loop over BO and CP, to generate new potential placements for the current iteration. As mentioned in the explanation of the empty containers case, efficiency is gained by having the outer loop over CP.

When the containers are not empty the nested loop is not limited to only straight combinations of BO and CP. Additional new potential placements such that $BOCP(k).CP \notin CP$ are also generated, and the term *adjusted placement points* has been introduced for this. A normal placement point refers to a Container- Point that is created dependently of a new potential Box-Orientation, based on either a normal or a projected point. Adjusted placement points are only considered for placements that are not directly on the container floor. Any adjusted placement points will be in addition to the straight combinations of BO and CP. For the creation of adjusted points, either the Box-Orientation below the original Container-Point or the inner walls of the particular container is considered. Further, we distinguish between *forward adjusted* and *backward adjusted* placement points. The rationales behind forward and backward adjusted are different, and both cases have evolved based on testing with real problem instances at the DC.

Regarding *forward adjusted*, the rationale is to enable a design of BOCP constraints related to stacking in Example 3 in Section 4.4.2 that comply with the third factor mentioned in Section 3.4.2 under category 5 stacking constraints. The third factor stated that the way of stacking, for example how many edges (or corners) that is used by a lower box to support a box above, has a significant impact on how much weight or pressure a box may endure before bursting. From testing with real problem instances we learned that if only normal and projected points are used, potential placements may be unnecessarily infeasible (and rejected) based on exceeding stacking strength of boxes below. These placements would have been feasible if just the placement point would have been adjusted forward so that the Box-Orientation would have received support from more edges/corners of the boxes below. There are a few different cases to cover to properly handle the creation of forward adjusted points. These cases are not presented in detail, but instead two examples of forward adjusted points are illustrated in Figure 35. In the left example the point is adjusted forward along one axis, while in the right example the point is adjusted forward along two axes.

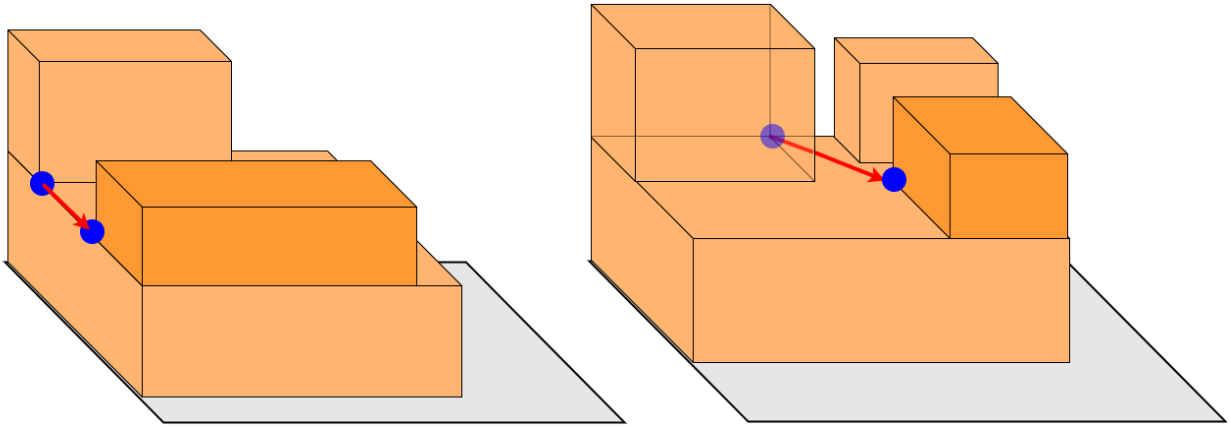

Figure 35. Examples of forward adjusted points - one axis (left) and two axes (right)

Regarding *backward adjusted*, the rationale is to enable a design of BOCP constraints related to vertical stability in Section 4.4.2 that utilize the possibility of partial support of the base area of a box as mentioned in Section 3.4.5 under category 9 stability constraints. Regarding partial support it was stated that a box placed on top of one or several other boxes may have support below 100% which then implies overhang. From testing with real problem instances we learned that if only normal and projected points are used, potential placements may be unnecessarily infeasible due to not being entirely inside the container, or intersecting with other boxes. These placements would have been feasible if just the placement point would have been adjusted backward so that the Box-Orientation would have been entirely inside the container, or would not have intersected with other boxes, respectively. There are a few different cases to cover to properly handle the creation of backward adjusted points. These cases are not presented in detail, but instead two examples of backward adjusted points are illustrated in Figure 36. In the left example the point is adjusted backward along one axis, while in the right example the point is adjusted backward along two axes.

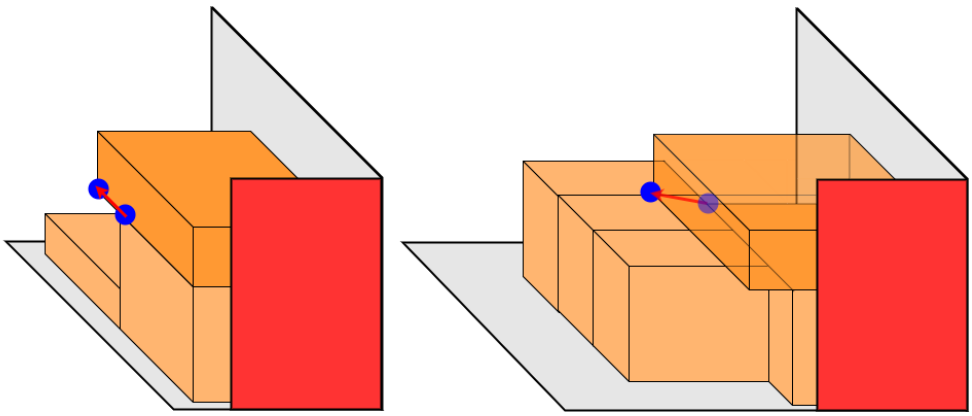

Figure 36. Examples of backward adjusted points - one axis (left) and two axes (right)

4.4.2 BOCP constraints

As explained in Section 4.4.1, checks of constraints applicable immediately upon a placement occur within the evaluation procedure of the placement heuristic algorithm. More specifically, the checks take place within a loop over already performed placements, but also as smaller separate checks prior to entering the loop, as well as via more complex sub procedures.

Constraints applicable immediately upon a placement and designed as hard are treated in this section, while any constraints from Section 3.4 that are merely incorporated in the BOCP score are covered in Section 4.4.3 instead. We want to remark that orientation constraints (category 4 in Section 3.4.2) have already been handled in the initialization of BO, but are also incorporated in the BOCP score in Section 4.4.3. If any hard constraint are not satisfied for a potential placement the evaluation is immediately terminated and hence scoring is not performed.

Overview of constraints

In Table 7 below it is presented which hard constraints are applicable based on which batch is being executed in the overall algorithm. In total 10 different hard constraints for a placement have been developed and the number of constraints used for each batch varies from 9 to 10. In the leftmost column of Table 7 the associated categories from Section 3.4 are referred to.

Table 7. BOCP constraints per batch with the associated categories

Category	BOCP constraint	Batch 1	Pre-Check	Batch 2	Batch 3
Basic	Box entirely inside container	✓	✓	✓	✓
Basic	Box not intersecting other boxes	✓	✓	✓	✓
1	Container weight capacity not exceeded	✓	✓	✓	✓
5	Stacking strength of boxes below not exceeded	✓	✓	✓	✓
5/8	Box not on top of different box if the other box is submitted as "not ok to stack a different package above"	✓	✓	✓	✓
5/8	Box not on top of similar box if the other box is submitted as "not ok to stack a similar package above"	✓	✓	✓	✓
8	Box not above floor if submitted as "must be placed on the floor"	✓	✓	✓	✓
8	Box not in front of or above hazardous box	✓	✓	✓	✓
9	Box receives sufficient base support	✓	✓	✓	✓
N/A	Consolidation pallet minimum possible height		✓		
All	**Total number of BOCP constraints**	**9**	**10**	**9**	**9**

Introduction to examples

Below three more detailed examples are outlined on how BOCP constraints are checked. The two first examples below are straight forward and therefore outlined in detail. The third example however is associated with a fairly complex sub procedure and hence an illustration with supporting notes is provided instead.

Example 1: Box not intersecting other boxes

Inside BOCP loop ($\forall \hat{k} \neq k$):

$$\textbf{\textit{if not}}\ (L(c).BOCP(\hat{k}).CP.PointEnd.X \leq\ BOCP(k).CP.Point.X$$

$$\textbf{\textit{or}}\ L(c).BOCP(\hat{k}).CP.PointEnd.Y \leq\ BOCP(k).CP.Point.Y$$

$$\textbf{\textit{or}}\ L(c).BOCP(\hat{k}).CP.PointEnd.Z \leq\ BOCP(k).CP.Point.Z$$

$$\textbf{\textit{or}}\ L(c).BOCP(\hat{k}).CP.Point.X \geq\ BOCP(k).CP.PointEnd.X$$

$$\textbf{\textit{or}}\ L(c).BOCP(\hat{k}).CP.Point.Y \geq\ BOCP(k).CP.PointEnd.Y$$

$$\textbf{\textit{or}}\ L(c).BOCP(\hat{k}).CP.Point.Z \geq\ BOCP(k).CP.PointEnd.Z)\ \textbf{\textit{then}}$$

$$BOCP(k).Feasible = False$$

$$BOCP(k).Rejected = True$$

Example 2: Box not above floor if submitted as "must be placed on the floor"

Before BOCP loop:

$$\textbf{\textit{if}}\ (BOCP(k).CP.Point.Z = 0$$

$$\textbf{\textit{and}}\ BOCP(k).BO.Box.OnlyOnFloor)\ \textbf{\textit{then}}$$

$$BOCP(k).Feasible = False$$

$$BOCP(k).Rejected = True$$

Example 3: Stacking strength of boxes below not exceeded

Separate sub procedure:

Recall the importance of the way of stacking and the potential use of McKee formula to relate ECT values to stacking strength, as discussed in Section 3.4.2, category 5. Also recall that ECT values and thickness of a package's walls are available attributes, as stated in Section 2.2.1. Consider this adaption of the simplified McKee formula for a given *sub area*:

$$MaxLoadAbove = \frac{5.87 \times ECT \times \sqrt{Thickness \times Perimeter}}{EnvirFactor}$$

An explanation of sub area as well as the various components in the expression above follows.

First, to elaborate on what a sub area is, suppose a fictive problem instance with four boxes B1-B4. In Figure 37 below B1, B2 and B3 have already been placed and the potential placement of B4 on top of B3 is being evaluated, in particular if stacking strength for any of the two sub areas SA1 and SA2 would be exceeded.

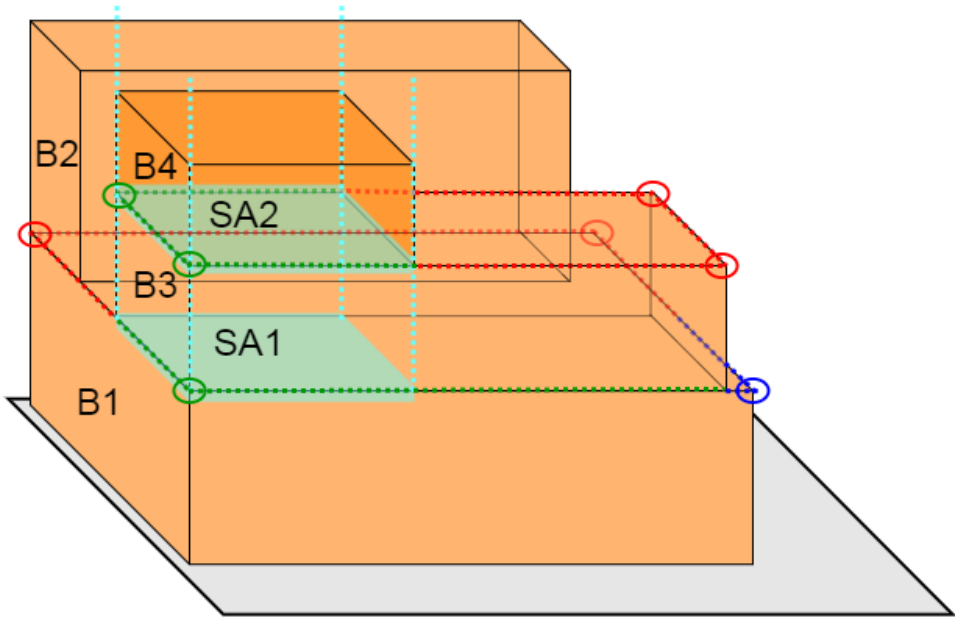

Figure 37. Example of two sub areas SA1 and SA2 to be evaluated with respect to stacking strength

The following can be understood about sub areas from the example above:

- Sub areas are generated below the potential placement being evaluated
- A sub area is located in the contact area of two boxes stacked on each other
- A sub area may be either the whole, or a portion of, the contact area of two boxes

Now the various components in our adoption of the simplified McKee formula can be discussed:

$MaxLoadAbove$ is the maximum weight (in lbs.) that is acceptable on top of a given sub area. If the sum of the weights of the boxes above any of the sub areas is exceeded, i.e. $LoadAbove > MaxLoadAbove$, the placement is not feasible and also rejected. Boxes above not entirely within the boundaries of the sub area contributes with a portion of their weight that is proportional to the ratio of base area that lies within the boundaries. Hence, $LoadAbove$ for SA1 is a portion of B3's weight plus the entire weight of B4, while $LoadAbove$ for SA2 is the weight of B4.

5.87 is a constant developed by McKee et al. (1963).

ECT is a value (in lbs. per inch) of the box immediately below the sub area related to its packaging, and more specifically the strength of the walls of the box. Typically $48 \leq ECT \leq 80$ for packages at the DC. In Section 2.2.1 it is noted that exceptions can be submitted to reduce or increase the inherited stacking strength for packages. These exceptions are fulfilled by multiplying ECT with a constant 0, 0.5, 1.5 or 2.0.

$Thickness$ is the thickness (in inch) of the walls of the box immediately below the sub area. Typically $Thickness = 0.25$ for packages at the DC.

$Perimeter$ refers to the meaning of perimeter as the distance around a two dimensional shape. The shape we consider is the sub area (in inch), but we only use $Perimeter = 2 \times Length + 2 \times Width$ if support from 4 corner is present. Since the strength

is focused in the walls, the edges of the box immediately below covered by the box immediately above is considered. SA2 results in $Perimeter = 1 \times Length + 2 \times Width$, as only one edge of B3 is covered by B4 lengthwise. The choice to consider the sub area rather than the whole contact area is not obvious, especially not for SA1, as one may argue that the weight of B4 is distributed along the wall of B1 via B3 widthwise, beyond the end of SA1. Here a somewhat pessimistic (too low) $MaxLoadAbove$ is obtained by considering SA1 rather than the full contact area of B1 and B3. However, the positive effect of B3 for SA1 is to large extent captured in $EnvirFactor$. At the same time, there are other cases when optimistic (too high) $MaxLoadAbove$ is obtained by not considering weight of boxes outside the boundaries of the sub area. For example, suppose that a fifth box would be placed on top of B3, next to B4. The weight of the fifth box would not be accounted for in $MaxLoadAbove$ for SA1, even though some if its weight likely would be distributed to SA1 via B3 widthwise.

$EnvirFactor$ refers to the environmental factor presented by Fibre Box Association (2008). Via ECT and $Thickness$ the strength of the packaging is captured. By scaling ECT based on submitted exceptions the content of the box is captured. The way of stacking is only to some extent captured via how $Perimeter$ is used. In order to better handle the way of stacking, as well as transportation aspects such as duration of the transport, humidity throughout the route, etc. the environmental factor is necessary. We choose to design the environmental factor as $EnvirFactor = EnvirFactor_{Base} \times EnvirFactor_{Support} \times EnvirFactor_{Overlap}$. $EnvirFactor_{Base}$ captures all remaining aspects except for the way of stacking. From experiments with real packages at the DC we have found $EnvirFactor_{Base} = 3.5$ to be appropriate. Consequently $EnvirFactor_{Support}$ and $EnvirFactor_{Overlap}$ are both related to the way of stacking.

More specifically regarding the environmental factor, $EnvirFactor_{Support}$ concerns the number of edges/corners of the box immediately below covered by the box immediately above. The placement of B4 exemplifies support from two corners. $EnvirFactor_{Overlap}$ concerns if the box above reach beyond the box below and hence may receive additional support from other boxes. The right example in Figure 36 in Section 4.4.1 exemplifies such scenario. There are several different cases to properly handle the various level of support and overlap. We do not present these in detail, but note that in our design $1 \leq EnvirFactor_{Support} \leq 3.4$ and $0.33 \leq EnvirFactor_{Overlap} \leq 0.67$.

Using our $SCurve$ function (see Section 9.2.1) we have also incorporated consideration to scenarios when the box immediately above the sub area almost reach out to the edges of the box immediately below. Note that B3 almost reach out to the second edge of B1 lengthwise, so instead of simply consider "support from one corner" for SA1, "support from one, but almost two, corners" is considered. In this specific case the latter will always result in a higher $MaxLoadAbove$. However, more complicated scenarios are possible, for example if no corner and no edge of box below is covered, it may in fact be appropriate with anything up to "support from almost four corners".

The design with $EnvirFactor_{Support}$ fails if the support, despite considering "support from almost…", is bad. For example if no corner and no or one edge of the box immediately below is covered by the box immediately above. These cases are not covered by $EnvirFactor_{Support} \leq 3.4$, so instead a fixed weight threshold based on the packaging option of the box below limits the weight from boxes above with bad support.

Our approach on stacking strength has evolved from a mix of theory, experiments, crafting and testing. We argue by no means that our way guarantees sufficient stacking strength and we rather

acknowledge that more dedicated effort on this topic would be required to develop a more rigorous approach. However, feedback from the DC operators indicates that it works really well in practice.

4.4.3 BOCP score

As explained in Section 4.4.1, the calculation of BOCP score $S(k)$ using the BOCP score components $s(j,k), \forall j$ is the last step in the evaluation procedure within the placement heuristic algorithm. First in this section the design of the scoring function is explained in general terms and second the different scoring function components are explained in more practical terms.

Design of scoring function

Recall the simplified scoring function that was introduced in Section 4.1.2:

$$\tilde{S}(k) = \sum_{j} w(j) \times s(j,k)$$

In the further development of this function the BOCP score components $s(j,k), \forall j$ are distinguished as two types, $j \in J_1$ and $j \in J_2$. The extended function is developed via two sub functions $S_1(k)$ and $S_2(k)$ corresponding to these two types, and the sum of the two sub function make up the extended scoring function:

$$S(k) = S_1(k) + S_2(k)$$

The score components $s(j,k), \forall j \in J_1$ are designed to only adopt values between 0 and 1, i.e. $0 \leq s(j,k) \leq 1, \forall j \in J_1$, where $s(j,k) = 1$ always is the most desirable. Typically these components adopt values greater than 0, i.e. $0 < s(j,k) \leq 1$. To maintain $0 \leq S_1(k) \leq 1$ the sub function is scaled down and hence this formulation:

$$S_1(k) = \frac{1}{\sum_{j \in J_1} w(j)} \sum_{j \in J_1} w(j) \times s(j,k)$$

The remaining score components $s(j,k), \forall j \in J_2$ are different in three major ways. First, these components much more frequently adopt values equal to 0, i.e. $s(j,k) = 0$. Second, many of these components have a binary characteristic, i.e. $s(j,k) \in \{0, s_{const}(j)\}$, where $s_{const}(j)$ is a constant for a given $j \in J_2$. Third, these components may adopt values greater than 1, i.e. $s(j,k) > 1$ is possible for at least one $j \in J_2$. To ensure that the values $S_2(k)$ are of appropriate magnitude relative $S_1(k)$ each component of the sub function is scaled down with w_{max}, which is the maximum value $w(j), \forall j$ may adopt, and hence this formulation:

$$S_2(k) = \sum_{j \in J_2} \frac{w(j)}{w_{max}} \times s(j,k)$$

Finally, the extended function for the BOCP score is formulated as the sum of $S_1(k)$ and $S_2(k)$:

$$S(k) = \frac{1}{\sum_{j \in J_1} w(j)} \sum_{j \in J_1} w(j) \times s(j,k) + \sum_{j \in J_2} \frac{w(j)}{w_{max}} \times s(j,k)$$

We view the contribution of $S_1(k)$ as the baseline score and the contribution of $S_2(k)$ mainly as being bias towards certain more difficult aspects. This may be interpreted from the following overview of scoring function components.

Overview of scoring function components

In Table 8 below it is presented which BOCP score components $s(j,k)$ are included in the BOCP score $S(k)$ based on which batch is being executed in the overall algorithm. In total 23 different score components have been developed and the number of components used for each batch varies from 11 to 20. In the leftmost column of Table 8 the associated goals and objectives for loading from Section 2.3.2 are referred to. The second leftmost column of Table 8 states whether the score component is included in $S_1(k)$ or $S_2(k)$. Recall our interpretation of type 1 as the baseline score and type 2 mainly as being bias towards more difficult aspects.

Table 8. BOCP score components per batch with the associated goals/objectives and type of component

Goal/ Objective	Type	BOCP score component	Batch 1	Pre-Check	Batch 2	Batch 3
1, 2.2, 3.3	1	Box with higher weight closer to floor	✓		✓	✓
1, 2.2	1	Box closer to floor	✓	✓	✓	✓
1, 2.2	1	Box closer to intersection of inner wall and ceiling lengthwise	✓			✓
1, 2.2	1	Box with higher stacking strength - if would provide support with edges/corners - closer to floor	✓		✓	✓
1, 2.2	1	Box with higher stacking strength - if would provide support with only lid, i.e. no edges/corners - closer to floor	✓		✓	✓
1, 2.2	1	Utilization of space length higher	✓	✓	✓	✓
1, 2.2	1	Utilization of space width higher	✓	✓	✓	✓
1, 2.2	1	Utilization of space height higher	✓	✓	✓	✓
2.1	1	Consolidation pallet possible height		✓		
2.1	1	Consolidation pallet possible weight		✓		
2.2	1	Horizontal stability behind box	✓	✓	✓	✓
2.2	1	Horizontal stability in front, left and right of box	✓	✓	✓	✓
2.1	2	Consolidation pallet at Pre-check placement point				✓
1	2	Adjusted point backward	✓	✓	✓	✓
1	2	Adjusted point forward	✓	✓	✓	✓
1	2	Box ending at same height as other adjacent box to the left	✓		✓	✓
1	2	Box with same height as more other boxes	✓		✓	✓
1	2	Box on top of similar box submitted as "not ok to stack a different package above"	✓			✓
1	2	Box submitted as "must be placed on the floor"	✓			
1	2	Box considered to be long, with the largest dimension greater than a certain threshold (1.5 m)	✓			
1	2	Hazardous box closer to intersection of door and ceiling lengthwise or box submitted as not ok to stack a package (neither similar nor different) above closer to ceiling	✓			✓
4.1	2	Box with pallet not rotated	✓	✓		✓
4.3	2	Box without pallet not on floor	✓			✓
All		**Total number of score components**	**20**	**11**	**13**	**19**

Introduction to examples

Below four more detailed examples are outlined on how BOCP score components are calculated. The three first examples below are fairly straight forward and therefore outlined in detail. The fourth example however is associated with a fairly complex sub procedure and hence an illustration with supporting notes is provided instead.

In the examples below helper functions h are used for the preparing scoring calculations that are needed prior to calculating actual score component. Recall the notation $Batch$ as all boxes assigned to the batch that is being executed in the overall algorithm.

To enable a more practical, non-binary, evaluation of certain aspects sigmoid/logistic curve is used and denoted as $SCurve(x, midpoint, steepness)$, where x is the input value and $midpoint$ and $steepness$ are parameters. Another function denoted as $Scaling(x, factor)$ is also used, where x is the input value and $factor$ is a parameter. Details on $SCurve$ and $Scaling$ can be found in Section 9.2.1 and 9.2.2 respectively. In the examples below our values on these and other parameters are written to avoid additional notation. An exception to this is $CLOSE$, which is a constant, and more specifically a small decimal number to count almost adjacent surfaces as adjacent.

In the first example below $s\left(j_{WeightCloserFloor}, k\right) \approx 1$ if $BOCP(k)$ is the heaviest box in the current batch and placed on the floor. In the second example $s\left(j_{HorizontalStabilityBehind}, k\right) = 1$ if $BOCP(k)$ has adequate support from behind. The third example is of type 2 where $s\left(j_{OnlyOnFloor}, k\right)$ adopts a constant value. In the fourth example $s\left(j_{UtilizationSpaceLength}, k\right) = 1$ if $BOCP(k)$ is a great fit in the available space (unutilized volume) lengthwise, and similarly for width and height.

Example 1: Box with higher weight closer to floor

After BOCP loop (but could be done earlier):

$$s\left(j_{WeightCloserFloor}, k\right)$$

$$= \left(1 - \frac{BOCP(k).CP.Point.Z + \frac{BOCP(k).CP.PointEnd.Z - BOCP(k).CP.Point.Z}{2}}{BOCP(k).CP.Container.DimInner.Height}\right)$$

$$\times \frac{BOCP(k).BO.Box.Weight}{\max\limits_{b \in Batch} Batch(b).Weight}$$

Example 2: Horizontal stability behind box

Inside BOCP loop ($\forall \hat{k} \neq k$):

$$\textbf{\textit{if}}\ (L(c).BOCP\left(\hat{k}\right).CP.PointEnd.X + CLOSE \geq BOCP(k).CP.Point.X$$

$$\textbf{\textit{and}}\ L(c).BOCP\left(\hat{k}\right).CP.Point.Y < BOCP(k).CP.PointEnd.Y$$

$$\textbf{\textit{and}}\ L(c).BOCP\left(\hat{k}\right).CP.PointEnd.Y > BOCP(k).CP.Point.Y$$

$$\textbf{\textit{and}}\ L(c).BOCP\left(\hat{k}\right).CP.Point.Z < BOCP(k).CP.PointEnd.Z$$

$$\textbf{\textit{and}}\ L(c).BOCP\left(\hat{k}\right).CP.PointEnd.Z > BOCP(k).CP.Point.Z)\ \textbf{\textit{then}}$$

$$h_{ContactAreaBehind}(k) = h_{ContactAreaBehind}(k)$$
$$+ \Big(\min\{L(c).BOCP(\hat{k}).CP.PointEnd.Y; BOCP(k).CP.PointEnd.Y\}$$
$$- \max\{L(c).BOCP(\hat{k}).CP.Point.Y; BOCP(k).CP.Point.Y\}\Big)$$
$$\times \Big(\min\{L(c).BOCP(\hat{k}).CP.PointEnd.Z; BOCP(k).CP.PointEnd.Z\}$$
$$- \max\{L(c).BOCP(\hat{k}).CP.Point.Z; BOCP(k).CP.Point.Z\}\Big)$$

After BOCP loop:

$$\textbf{\textit{if}}\ BOCP(k).CP.Point.X < CLOSE\ \textbf{\textit{then}}$$
$$h_{ContactAreaBehind}(k) = BOCP(k).BO.SideArea.Width$$
$$s(j_{HorizontalStabilityBehind}, k)$$
$$= SCurve(\frac{h_{ContactAreaBehind}(k)}{BOCP(k).BO.SideArea.Width}, 0.3, 20)$$

Example 3: Box submitted as "must be placed on the floor"

After BOCP loop (but could be done earlier):

$$\textbf{\textit{if}}\ BOCP(k).BO.Box.OnlyOnFloor\ \textbf{\textit{then}}$$
$$s(j_{OnlyOnFloor}, k) = 0.2$$
$$\textbf{\textit{else}}$$
$$s(j_{OnlyOnFloor}, k) = 0$$

Example 4: Utilization of space length/width/height higher

Separate sub procedure:

Prior to presenting the calculation of utilization it is necessary to explain how available spaces (unutilized volumes) are handled. Recall the statement in Section 4.4.1 that, based on how available spaces are identified, efficiency is gained by having the outer loop over CP when $BOCP$ is generated. More specifically, one new space is generated for $BOCP(1).CP$ and each time $BOCP(k).CP \neq BOCP(k-1).CP, k \geq 2$.

Figure 38 below illustrates a fictive problem instance with five placement points after two placements have been performed. For each placement point one space is generated.

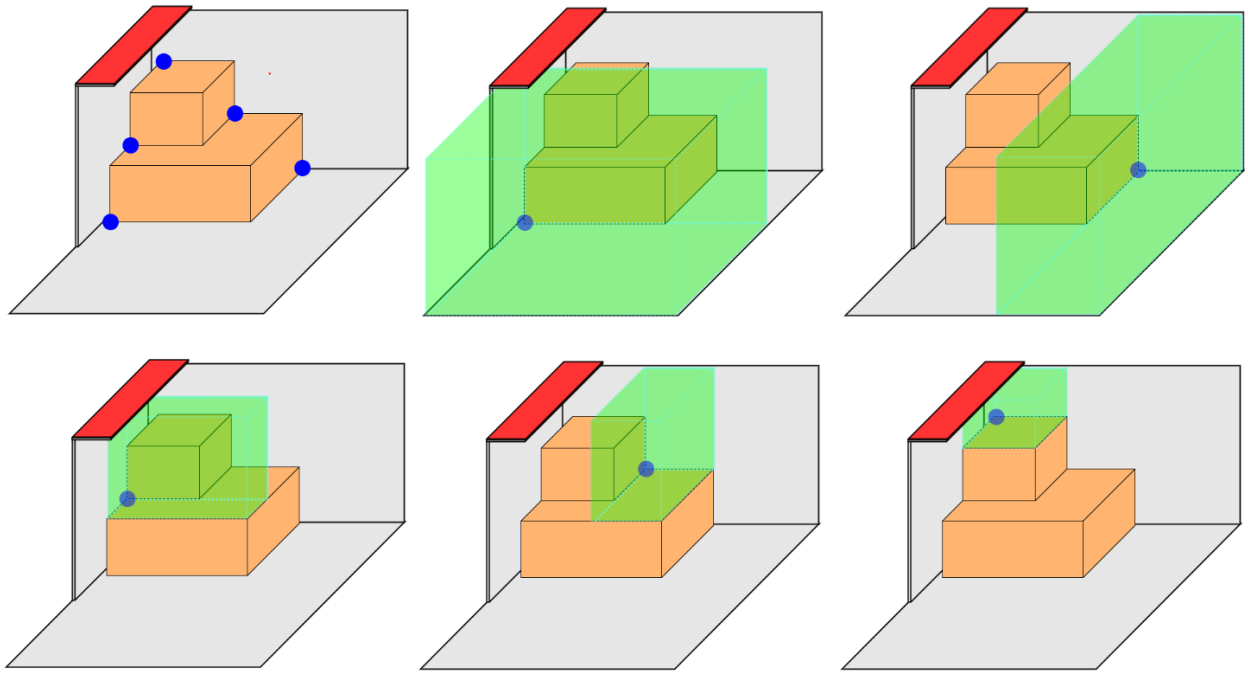

Figure 38. Examples of available spaces (unutilized volumes) for five placement points

Points on the floor are more straightforward than points above the floor. For points on the floor, the space reach from the point to the intersection with either the container walls or another box. For points above the floor, the space reach until the end of the "platform" created by one or several boxes, or the intersection with another box. There are many cases to cover to properly handle the creation of spaces. These are not presented in detail, but as a hint, consider the extended example in Figure 39 when three placements have been performed:

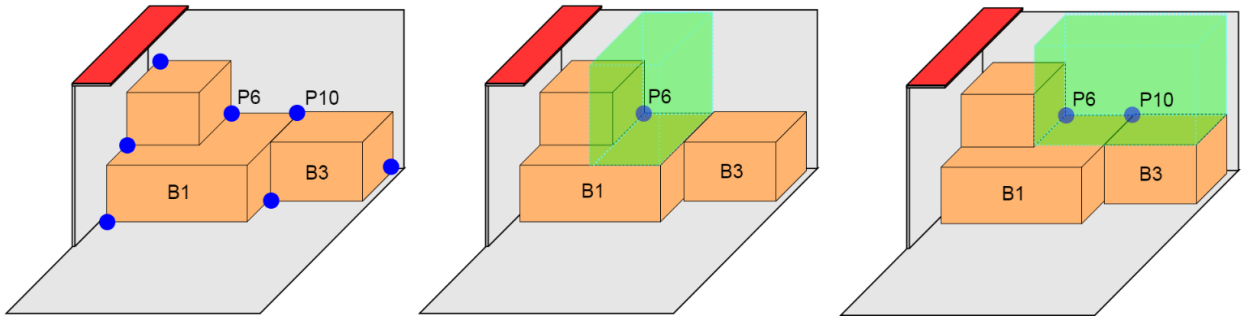

Figure 39. Extended example based on Figure 38, with available spaces for the two placement points P6 and P10

An important observation is that B1 and B3 have the same height. Now, consider the space for P6. There are two potential spaces that could be assigned P6, given that 100% base support for a platform is required. This is not to be confused with the constraint "box receives sufficient base support" outlined in Table 7 in Section 4.4.2. In fact, we do require less than 100% base support for a box (80% with a few exceptions when even less is required, to be specific). This results in the possibility for the space utilization to reach above 100%, which is handled by the min-function in the score component presented below. Spaces are distinguished based on base area such that the space with highest base area is assigned to the placement point. Further, one of the potential spaces for P6 is also applicable for P10. To only consider the base area of B3 as the platform for P1's space would

be deficient, as the space utilization would seem better than it is, and hence the limitation caused to placements at P6 would not be captured.

Given the basics for spaces explained above, consider the calculation of space utilization lengthwise. Calculation of widthwise and heightwise space utilization are similar. The purpose of our function $Scaling$ is to mainly reward quite high space utilization (see Section 9.2.2).

$$s\left(j_{UtilizationSpaceLength}, k\right)$$

$$= Scaling\left(\min\{1; \frac{BOCP(k).CP.PointEnd.X - BOCP(k).CP.Point.X}{Space.Length}\}, 0.05\right)$$

4.5 Genetic Algorithm

Coefficients for the BOCP score are tuned for the particular subproblem (batch) being solved via a Genetic Algorithm (GA).

Recall the role of the GA by revisiting Figure 28 in Section 4.1.2. Figure 40 is a snippet of Figure 28 which shows that the GA receives fitness values for the entire GA population F_{pop} and updates the population (coefficient sets) W_{pop}, which is then forwarded. $F_{pop}(i)$ and $W_{pop}(i)$ are used to access fitness and chromosomes of a specific individual in the population. $0 \leq F_{pop}(i) \leq 1, \forall i$ as indicated in Section 4.3.2 on load plan fitness. The number of chromosomes for an individual varies from 11 to 20 as indicated in Section 4.4.3 on BOCP score. The values of the chromosomes are integers between 0 and 10000, i.e. $0 \leq w_{pop}(i,j) \leq 10000\ (integer)\ \forall i,j$.

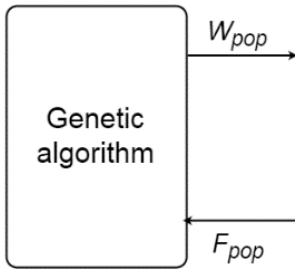

Figure 40. The role of the genetic algorithm

The GA is stochastic in the sense that for different runs of the same problem instance, different coefficients for the BOCP score will be generated, and hence possibly also different load plans.

4.5.1 Overall GA procedure

The overall GA procedure is illustrated in Figure 41, which is consistent with Figures 27, 28 and 40. The GA procedure begins with randomly initializing a population W_{pop}. This population is evaluated through the iterative sequence of placement heuristic and load plan evaluation (see Figure 28 in Section 4.1.2) which returns fitness for the population F_{pop}. A new population is produced as a combination of clones and children. More specifically, cloning is done according to elitism concept, and the following techniques are used to make children:

- Roulette wheel method to select parents
- Random crossover for inheriting chromosomes
- Mutation to maintain genetic diversity

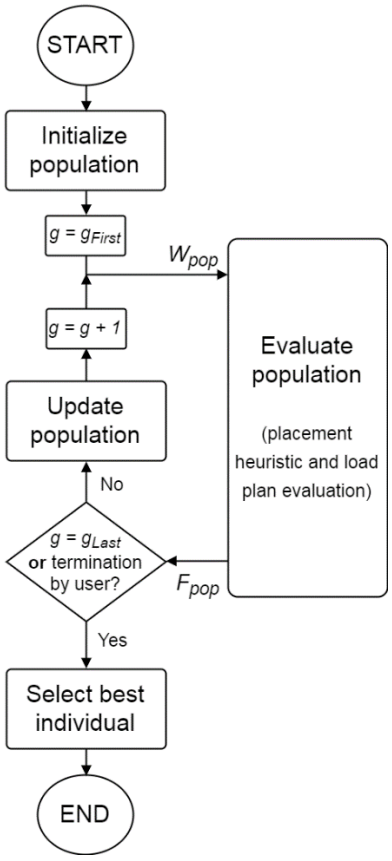

Figure 41. Schematic of genetic algorithm

In order to better distinguish fitness of individuals, $F_{pop}(i), \forall i$ are scaled based on the highest fitness $F_{pop}(i^*) \geq F_{pop}(i), \forall i \neq i^*$ using our function denoted as $Scaling(x, factor)$ where x is the input value and $factor$ is a parameter (see Section 9.2.2). More specifically, the update of the population is based on the scaled fitness values $\tilde{F}_{pop}(i), \forall i$ calculated as:

$$\tilde{F}_{pop}(i) = Scaling(\frac{F_{pop}(i)}{F_{pop}(i^*)}, 0.1)$$

With $factor = 0.1$ as above the scaled fitness $\tilde{F}_{pop}(i) = 1.00$ if $F_{pop}(i) = F_{pop}(i^*)$, while $\tilde{F}_{pop}(i) = 0.50$ if $F_{pop}(i) = 0.90 \times F_{pop}(i^*)$, etc.

When the population has been updated, the cycle is repeated for the next generation. When the last generation has been evaluated, or whenever the execution is terminated by the user, the best individual $F_{pop}(i^*)$ is selected.

4.5.2 Parameter settings

In Table 9 below the GA parameters that are unique per batch are listed. The chromosome sizes match the number of BOCP score components for each batch as listed in Table 8 in Section 4.4.3.

Table 9. GA parameters per batch

GA parameter	Batch 1	Pre-Check	Batch 2	Batch 3
Chromosome size	20	11	13	19
Generation max	500	5	20	10

In Table 10 below the GA parameters that are applicable to all batches are listed. The chromosome values are integers. Population size and mutation percentage is dependent on chromosome size, and hence these also vary per batch.

Table 10. GA parameters applicable to all batches

GA parameter	All batches
Chromosome value min	0
Chromosome value max	10000
Population size	3 × Chromosome size
Cloning percentage	0.05
Mutation percentage	3 / Chromosome size

We want to stress that very little effort has been dedicated to developing the GA parameters in Table 10. These parameters work well for us, but should preferably be considered by the reader as a potential starting point for future work.

4.6 Summary of how objectives and constraints are handled

This section summarizes how the objectives from Section 2.3.2 and the constraints from Section 3.4 have been handled throughout Section 4.3-4.4. Tables 11 and 12 below lists where in the placement heuristic and load plan evaluation the various objectives and constraints respectively are handled.

Table 11. Summary of how objectives are handled in placement heuristic and load plan evaluation

Objective number	Objective	Placement heuristic			Load plan evaluation	
		BOCP constraints	BOCP score	Other	Load plan constraints	Load plan fitness
1.1	Number of leftover boxes lower		✓			✓
1.2	Volume of leftover boxes lower		✓			✓
1.3	Weight of leftover boxes lower		✓			✓
2.1	Use consolidation pallets for smaller packages	✓	✓			✓
2.2	All sides of packages and consolidation pallets supported		✓		✓	✓
3.1	Center of gravity closer to center lengthwise				✓	✓
3.2	Center of gravity closer to center widthwise				✓	✓
3.3	Center of gravity closer to floor		✓			✓
4.1	Packages with pallet not rotated		✓			✓
4.2	Packages with pallet reachable by forklift					✓
4.3	Packages without pallet not on floor		✓			✓

Table 12. Summary of how constraints are handled in placement heuristic and load plan evaluation

Category number	Category	Placement heuristic			Load plan evaluation	
		BOCP constraints	BOCP score	Other	Load plan constraints	Load plan fitness
1	Weight limits	✓				
2	Weight distribution constraints		✓		✓	✓
4	Orientation constraints		✓	✓		✓
5	Stacking constraints	✓	✓			
8	Positioning constraints	✓	✓			✓
9	Stability constraints	✓	✓		✓	✓
10	Complexity constraints					✓

A key observation in Tables 11 and 12 is that objectives are to some extent handled via BOCP constraints and load plan constraints, while constraints are to quite significant extent handled via BOCP score and load plan fitness. This is because our objectives and constraints are very interrelated.

5 Implementation

The loading optimization software was developed using Visual Basic for Applications (VBA) in Microsoft Access. This enabled immediate access to data base tables from the enterprise software used at the DC, as well as generating load plans in Microsoft Excel, which is an environment the users are familiar with. More specifically, the loading optimization software receives dimensions and weights for packages from the enterprise software. In Section 2.2.1 it is mentioned that DC operators have the possibility to override inherited attributes and assumptions about packages by submitting *exceptions*. These exceptions are also received by the loading optimization software. The loading optimization software generates a load plan in Microsoft Excel containing visual step-by-step instructions, suggesting the placement of each package. Before a sea container departs there is a final check called "load manifest validation" in order to ensure that all packages that were included in the load plan have actually been loaded, and thereby scanned to the sea container. The entire loading process in terms of applications used is illustrated in Figure 42 and the user interface of the loading optimization software is presented in Figure 43.

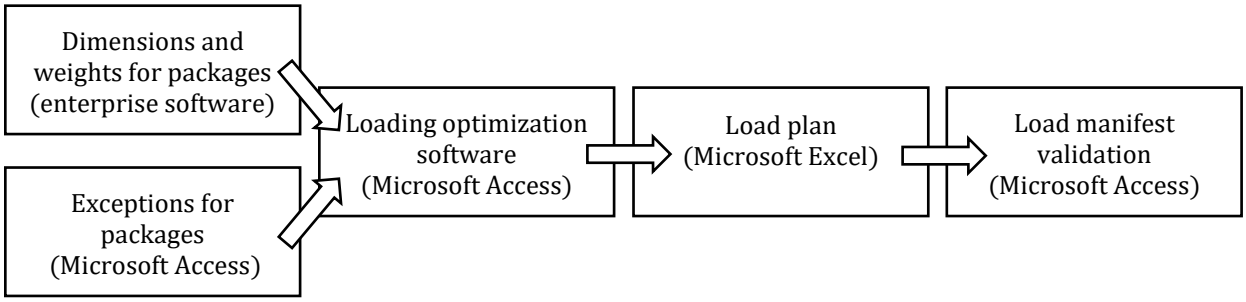

Figure 42. Applications used in loading process at the DC

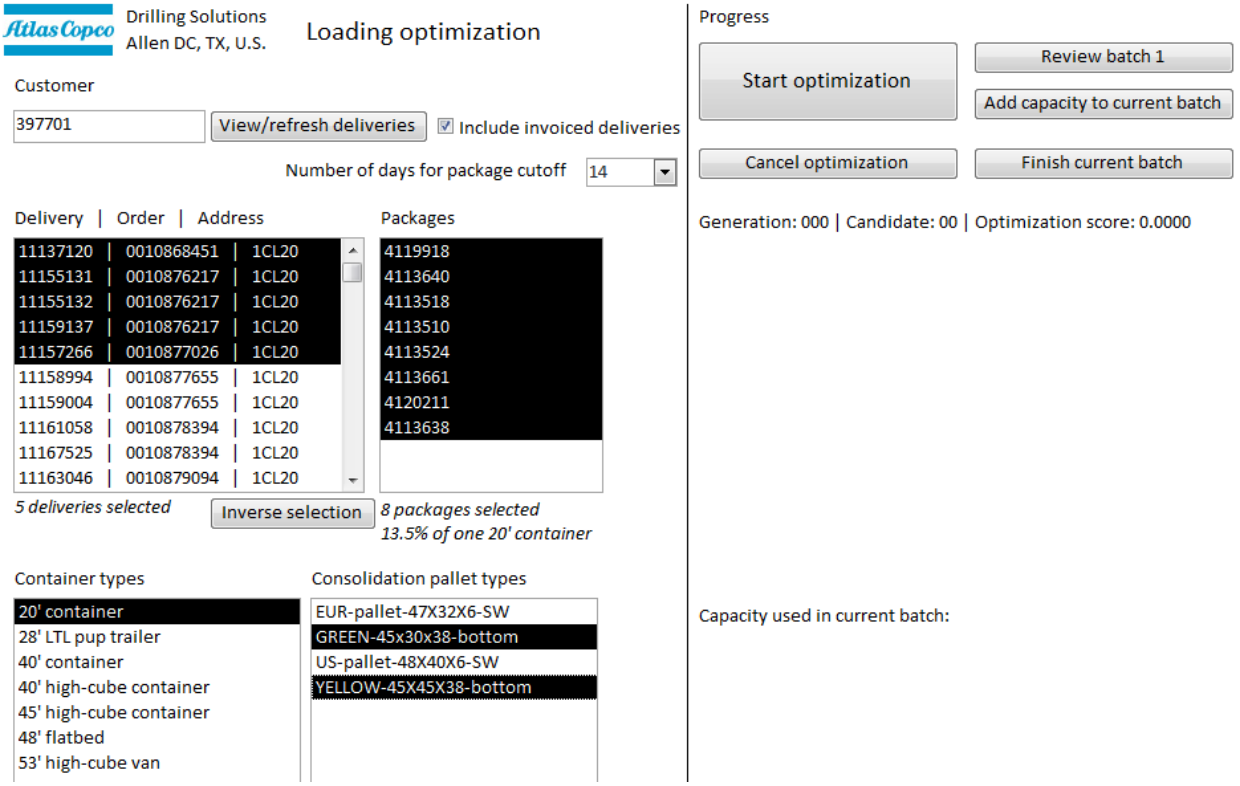

Figure 43. User interface of loading optimization software to select input and run the overall algorithm in Figure 29

As can be seen in Figure 43, the user enters a customer number and clicks to view deliveries, i.e. orders that have been released for picking. The user views which deliveries are available, with associated order number and address number (if the customer has multiple locations). The user selects which deliveries to be included, and the packages associated with these deliveries are then selected. At this point the user can see a predicted volume utilization, based on the assumption that all packages will fit in the selected shipping container type. The user also selects which consolidation pallet types that may be used. When all selections have been made, the user clicks "start optimization". While the optimization is running, the user has the possibility to review the load plan, add extra shipping containers or consolidation pallets (depending on which batch is running), as well as force the current batch to be finished. These possibilities are linked to the decision point in the genetic algorithm saying "…termination by user?", as illustrated in Figure 41 in Section 4.5.1.

Regarding the exceptions for packages, there are five exceptions that may be submitted to override the defaults, as illustrated in Figure 44 below.

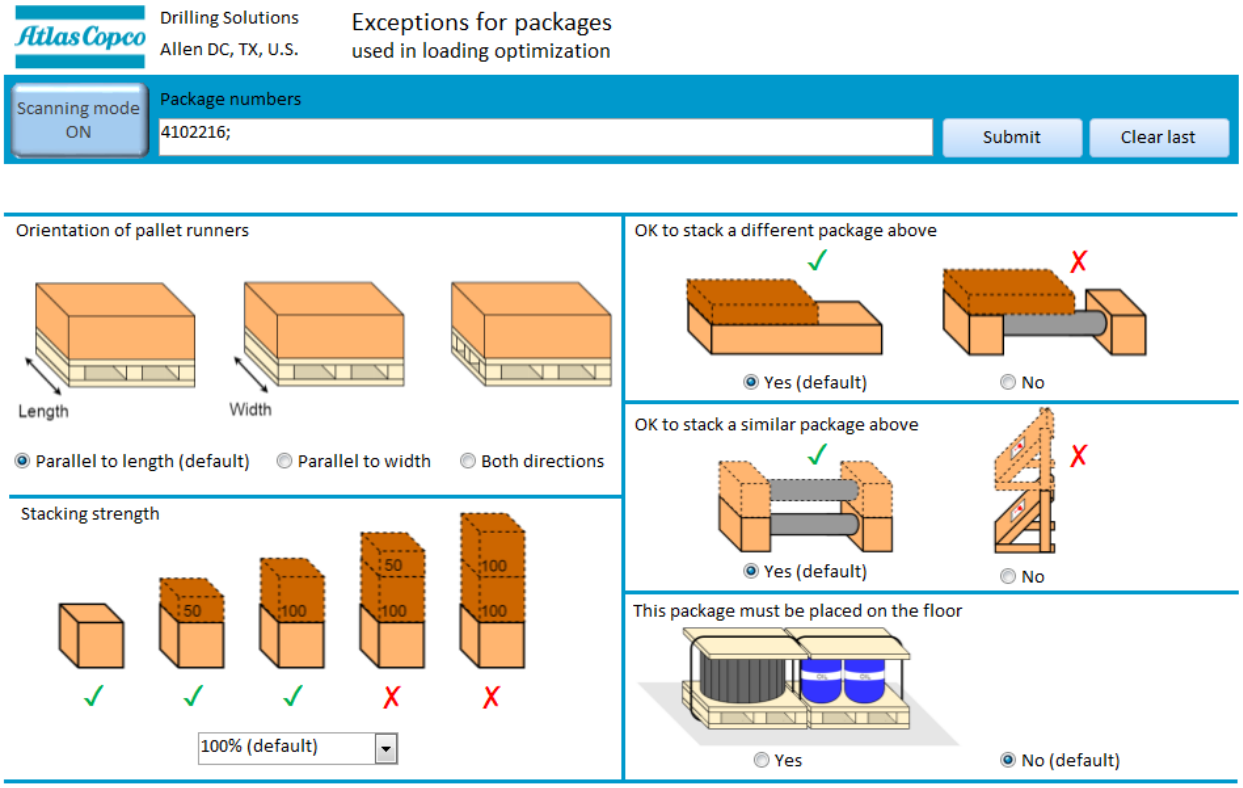

Figure 44. User interface to submit exceptions for packages

"Orientation of pallet runners" is considered upon initialization of possible Box-Orientations BO, which is treated in Section 4.4.1. "Stacking strength" results in scaling of ECT, as discussed in Example 3 in Section 4.4.2. "OK to stack a different package above", "OK to stack a similar package above", and "This package must be placed on the floor" are handled via BOCP constraints and BOCP score components, which are treated in Section 4.4.2 and 4.4.3 respectively.

6 Results and analysis

The loading optimization software has been tested and successfully implemented at the DC in Texas. The software has been proven capable of generating satisfactory load plans within acceptable computation times, which has resulted in reduced uncertainty and labor usage in the loading process. This chapter is intended to closer study the performance of the software. In Section 6.1 an overview of 20 real problem instances from the DC instances is given. In Section 6.2 two examples of load plans are presented. In Section 6.3 an introduction to the computational experiments that have been conducted using these problems is given. In Section 6.4 achievement of objectives is treated and in Section 6.5 the ability of the GA to tune the BOCP score coefficients is analyzed.

6.1 Overview of problem instances

Data for 20 loadings based upon load plans generated by the software is provided in Table 13 below:

Table 13. Problem instances used for computational experiments, with summaries of packages and shipping containers

Problem	Shipping containers	Number of packages (1)				Different sizes of packages (2)	Volume utilization (3)		Number of packages with attribute/exception (4)							
		Batch 1	Batch 2	Batch 3	Total		Batch 1	Total	Must be placed on floor	Not ok to stack different package above	Not ok to stack similar package above	Stacking strength 0% (fragile)	Stacking strength 50% (semi-fragile)	Stacking strength 150% (strong)	Stacking strength 200% (very strong)	Hazardous
1	1 20ft	22	40	2	64	29	34.1%	40.4%	0	1	0	0	0	0	0	0
2	1 20ft	16	4	0	20	14	41.9%	42.6%	0	0	0	0	0	9	0	0
3	1 20ft	18	16	0	34	18	71.4%	74.8%	0	0	0	0	0	12	0	0
4	1 40ft	29	1	0	30	12	46.7%	47.0%	0	0	0	0	0	0	3	0
5	1 40ft	22	2	0	24	15	57.5%	57.7%	5	5	5	0	0	0	0	0
6	1 40ft	37	1	0	38	18	46.7%	46.8%	0	2	0	0	0	0	0	0
7	1 40ft high-cube	24	2	0	26	19	46.4%	46.6%	0	0	0	0	0	0	0	0
8	1 20ft	26	2	0	28	16	62.6%	63.4%	3	0	0	0	0	0	1	0
9	1 40ft	38	2	0	40	19	45.6%	45.6%	0	0	0	0	0	0	3	0
10	1 40ft	24	1	0	25	12	43.9%	43.9%	0	0	0	0	0	0	0	0
11	2 20ft	29	24	0	53	26	47.7%	48.9%	2	0	3	0	0	0	3	0
12	1 20ft	17	16	0	33	17	57.8%	59.4%	1	0	1	0	0	0	2	0
13	1 20ft	14	0	0	14	9	38.3%	38.3%	0	0	0	0	0	0	0	0
14	1 40ft	33	1	0	34	19	47.2%	47.3%	4	0	0	0	0	0	13	0
15	1 20ft	20	22	2	44	20	36.7%	39.9%	1	2	2	2	0	1	3	1
16	1 20ft	35	10	1	46	30	68.8%	70.8%	0	4	4	4	1	0	2	0
17	1 20ft	22	2	0	24	10	43.7%	44.5%	0	0	0	0	0	0	0	0
18	1 40ft	9	2	0	11	9	20.4%	20.5%	4	0	3	0	0	0	4	0
19	1 20ft	24	0	0	24	11	80.0%	80.0%	1	0	0	0	0	0	1	0
20	1 20ft	23	26	0	49	25	40.3%	43.1%	3	2	0	0	2	1	0	0

Clarifications for some of the columns in Table 13 follows:

(1) Based on grouping of boxes explained in Section 4.1.1
(2) Number of unique combinations of length, width and height of packages
(3) Sum of outer volume of packages divided by sum of inner volume of shipping containers, excluding volume of any consolidation pallets suggested to be used
(4) All attributes other than hazardous are obtained by DC operators submitting exceptions for packages, see Section 2.2.1 and Chapter 5.

One observation in Table 13 is that very few packages are assigned to Batch 3. However, recall from Section 4.1.1 that packages assigned to Batch 2 are reassigned to Batch 3 upon termination of Batch 2 by the user, or if there is no relevant combination of consolidation pallets left to try. Another observation is the great variety in sizes among packages, as there are only about two packages per size on average. Also note that even in cases when the number of packages assigned to Batch 2 is relatively high, Batch 1 still makes up for the majority of the volume.

The fact that volume utilization varied from 20.5% to 80.0% is partly linked to the statement in Section 1.3 that containers may be ordered prior to reaching a high volume utilization if the customer requests shipment immediately. It is also related to stacking constraints among packages, both due to weight and cases when exceptions are applicable. To exemplify the latter, consider the partially completed loading of Problem 18 in Figure 45. By submitting exceptions "not ok to stack similar package above" the DC operator restricted the three long cylinders from being placed on top of each other, and hence they occupy much of the floor space. At the same time, consider the completed loading of Problem 3 in Figure 46. Even though the volume utilization was 74,8% excluding consolidation pallets this was a fairly easy load, as indicated in Section 6.3 and 6.4. It is therefore not sufficient to only consider the volume utilization as the measure of how difficult a loading is, and we argue that it is a common rule of thumb that should be used with care if stacking is limited.

Figure 45. Partially completed loading of Problem 18

Figure 46. Completed loading of Problem 3

In Section 3.2.2 remarks by Bortfeldt and Wäscher (2013) and Zhao et al. (2016) were outlined, stating that the same data sets, which lack practical elements, frequently have been used to develop and test new approaches. Based on these remarks we decided early on to not use benchmark data from the literature at all throughout our development efforts.

6.2 Load plan examples

The concept of load plans is first mentioned in Section 1.4 and then further discussed in Sections 2.1 and 2.4. A load plan consist of which shipping containers and consolidation pallets to use, and placements (position and orientation) of the packages within these, as stated in Section 2.4. Laptops in the forklifts (see Figure 16) were early identified as a key element to display visual step-by-step instructions, suggesting the placement of each package. By having visual instructions in the forklifts, the DC operators are able to load containers immediately without pre-staging (test loading) first. This consideration is also recognized in the literature, as mentioned in Section 3.4.5, category 10, Bortfeldt and Wäscher (2013) stress that solutions must be visualized and easy to understand for the operators doing the loading so that the loading procedure does not become too time consuming. Throughout Chapter 4, load plan is used to refer to a solution. Below follows two examples of how load plans generated by the software look like. The layout in terms of graphics and what information to show for each placement has been developed together with DC operators and customer service representatives, which has resulted in great perceived usability.

6.2.1 Example 1: loading of one 20 ft. container

The first example is the entire loading of Problem 19, which is the most difficult problem instance in terms of fitting all packages, as indicated in Section 6.3. Problem 19 has the highest volume utilization among the 20 instances, and since all packages are assigned to Batch 1, no consolidation pallet is used. Complementary pictures from the loading of Problem 19 can be found in Section 6.5 when analyzing BOCP score coefficients for this instance, more specifically Figures 68, 70 and 71. The placements of the 24 packages and center of gravity are shown in Figures 47-49 below.

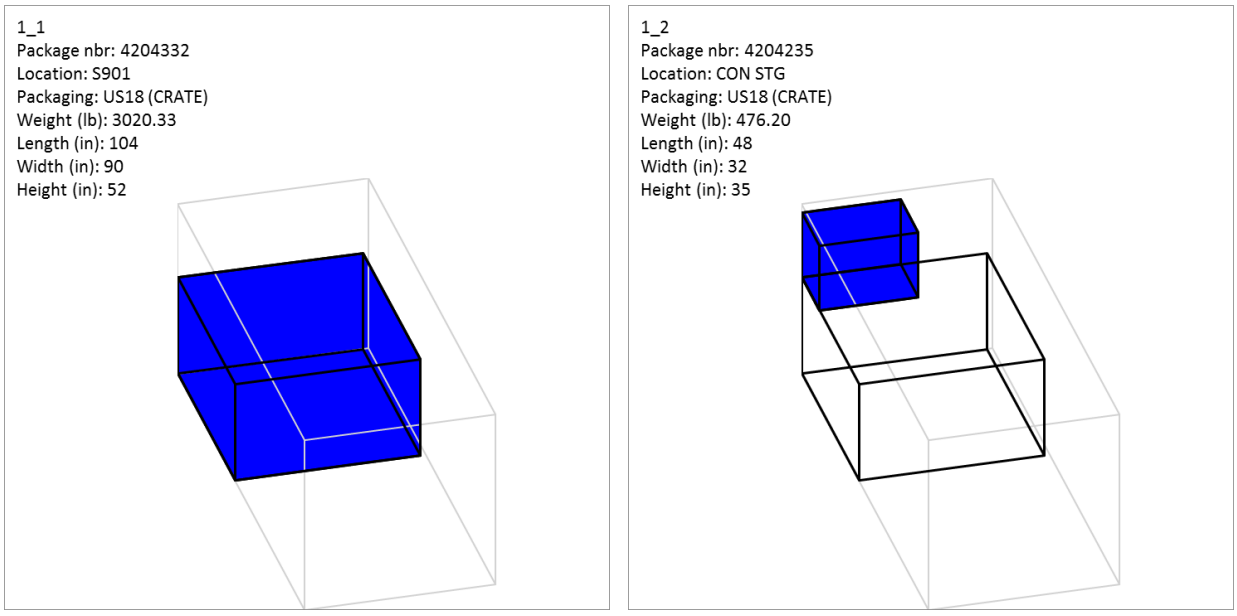

Figure 47. Placements 1-2 in Problem 19

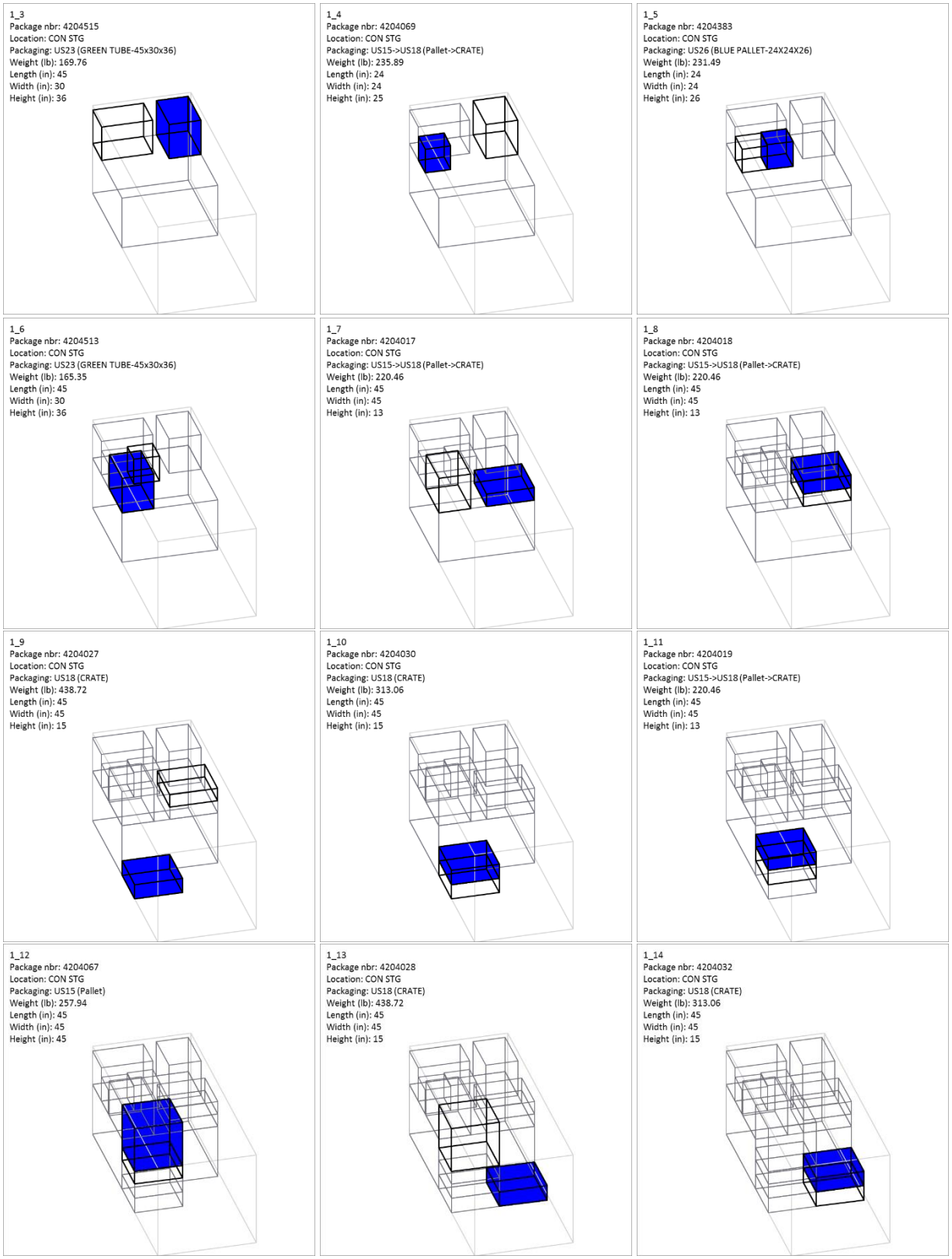

Figure 48. Placements 3-14 in Problem 19

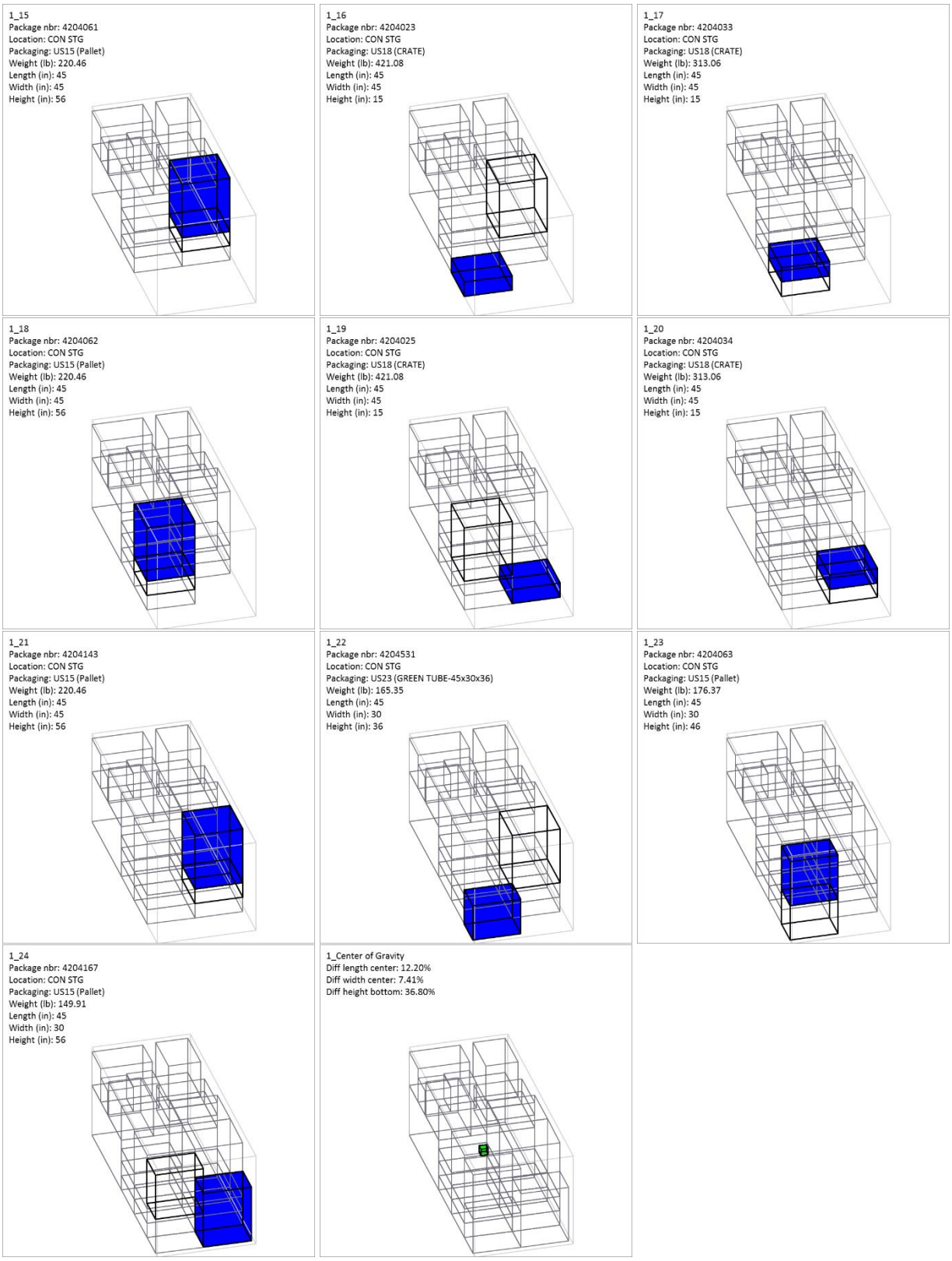

Figure 49. Placements 15-24 and center of gravity in Problem 19

6.2.2 Example 2: loading of two consolidation pallets

The second example is the two consolidation pallets that were used for Problem 20 to accommodate the 26 packages assigned to Batch 2. Recall the batch concept from Section 4.1.1, and in particular Figure 23. First, Figure 50 below shows the placements of both consolidation pallets into the shipping container, note that yellow is used for consolidation pallets instead of blue which is used for packages. The placements along the right wall of the shipping container are to accommodate three very long packages placed at later steps. Second, Figure 51 is a picture from the loading of Problem 20 showing the two consolidation pallets that have been prepared prior to loading the shipping container. Third, Figures 52-54 shows the placements of smaller packages onto these two consolidation pallets.

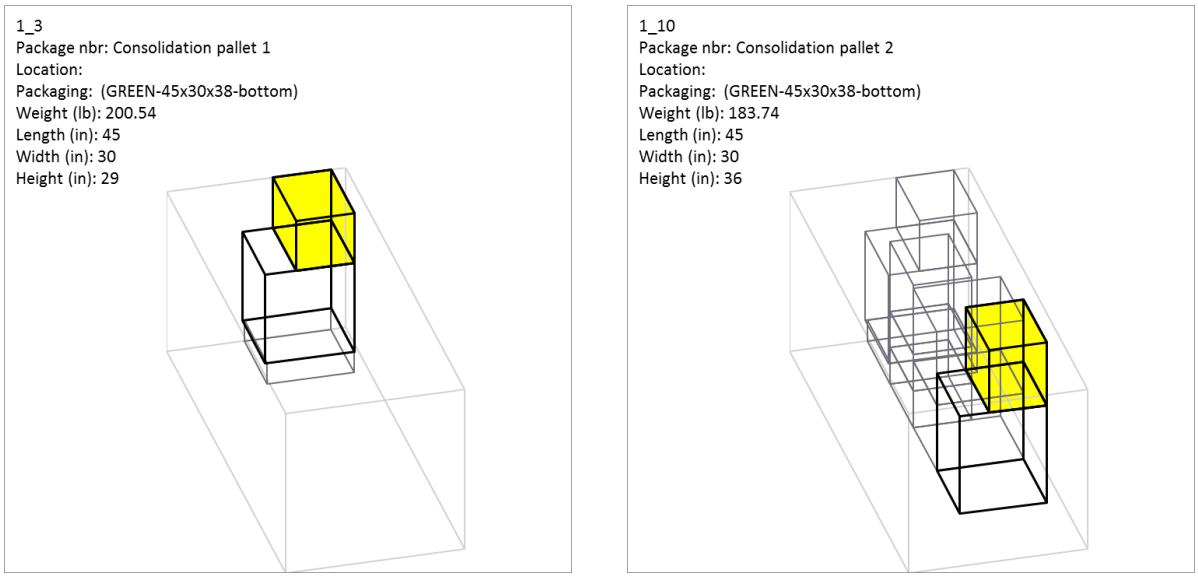

Figure 50. Placements of consolidation pallets into the shipping container in Problem 20

Figure 51. The two consolidation pallets in Problem 20 have been prepared outside the shipping container

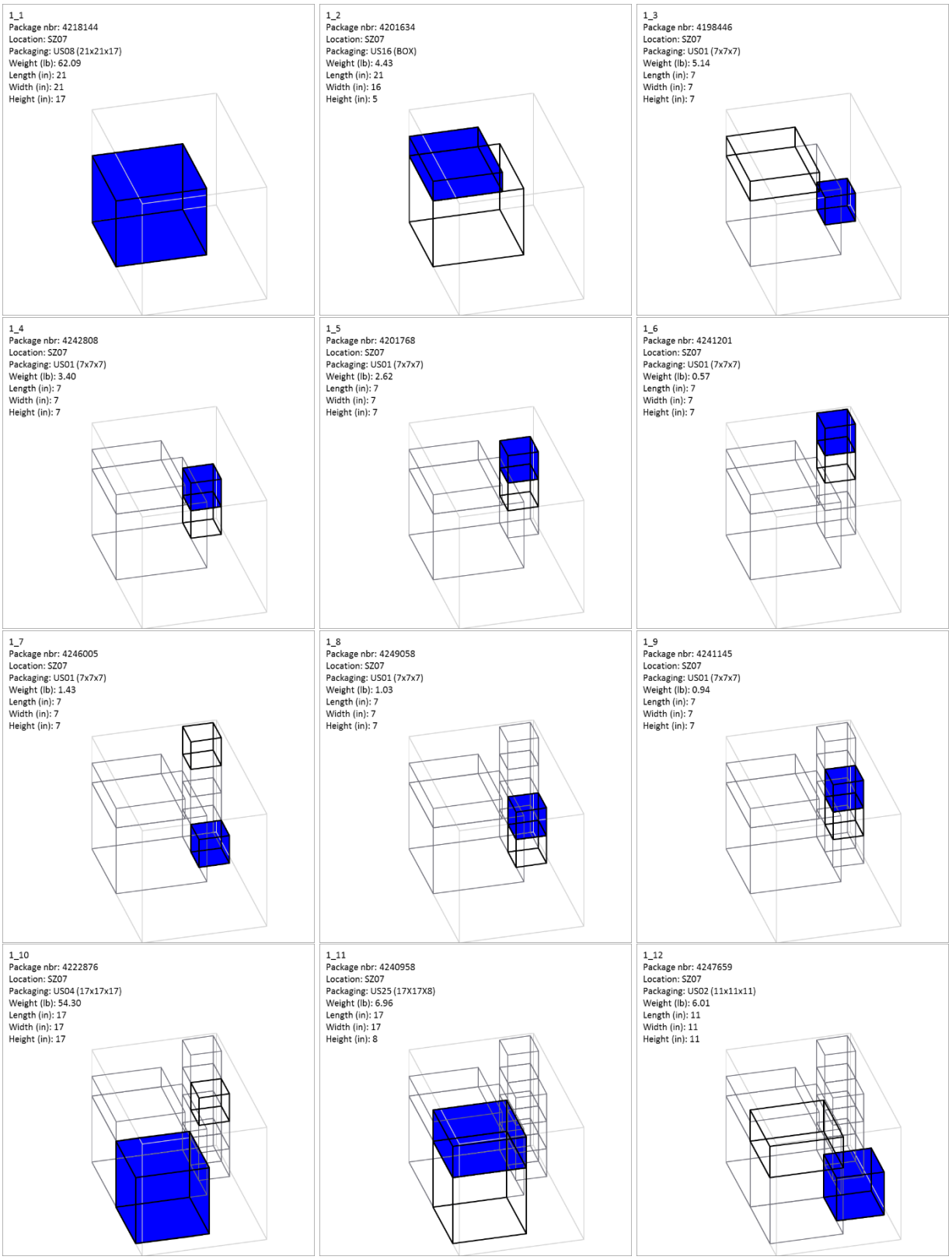

Figure 52. Placements 1-12 for consolidation pallet 1 in Problem 20

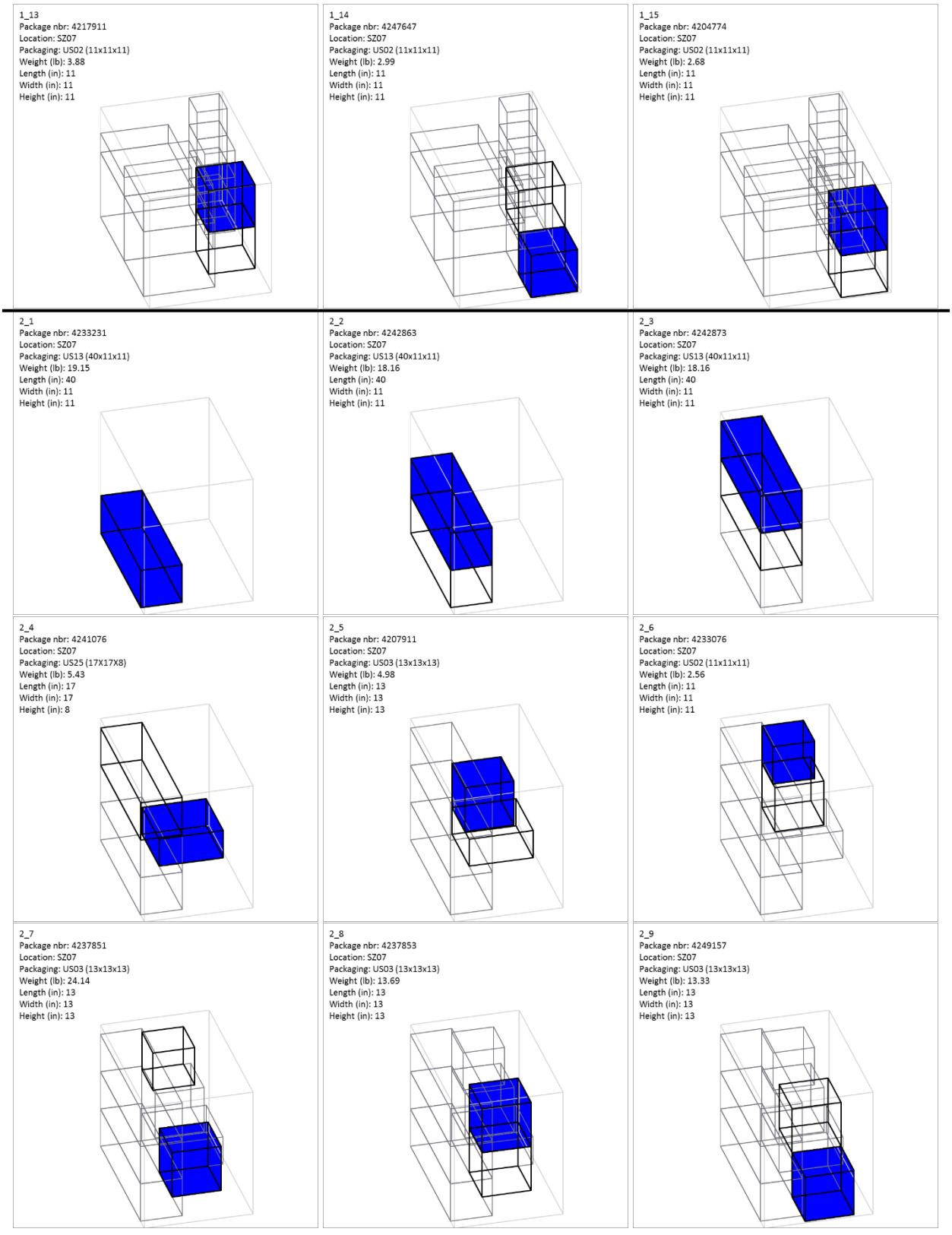

Figure 53. Placements 13-15 for consolidation pallet 1 and placements 1-9 for consolidation pallet 2 in Problem 20

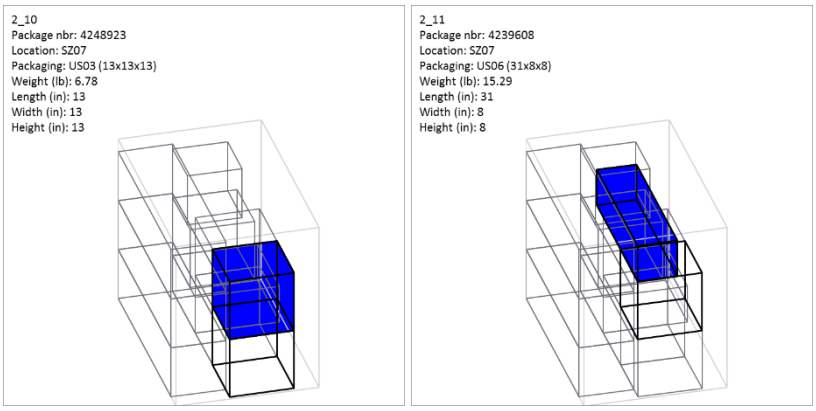

Figure 54. Placements 10-11 for consolidation pallet 2 in Problem 20

6.3 Introduction to computational experiments

Section 6.4 and 6.5 that follows are based on 10 runs to solve each of the 20 problems with the shipping containers specified in Table 13 and one consolidation pallet type. Only data for Batch 1 is considered here, as it is the most difficult batch. In each run, 200 Genetic Algorithm generations are executed. Recall from Section 4.5.2 that GA population size = 3 × Chromosome size = 60 for Batch 1, which implies 60 load plans per generation. All packages fit for every run and problem except for two runs of Problem 19. For 14 of the problems, all packages fit during the very first GA generation, while for all problems improvements are gained throughout the 200 GA generations. A Dell laptop with an Intel Core i5-2520M 2.50 GHz CPU and 4 GB RAM was used. For example, average 92 GA generations to fit all packages for Problem 19 translates to run time 9 × 92 = 828 seconds = 14 minutes.

Table 14. Summary of 10 runs to solve each of the 20 problems, with run times and numbers of GA generations

Problem	Run time for Batch 1 (seconds / GA generation)	GA generation for Batch 1 in which a load plan with no leftover packages was generated for the first time (max 200 GA generations)			GA generation for Batch 1 in which the final best load plan was generated for the first time (max 200 GA generations)		
		Min	Max	Average	Min	Max	Average
1	11	1	1	1	38	185	98
2	5	1	1	1	48	187	100
3	5	1	1	1	23	199	96
4	18	1	1	1	46	191	119
5	7	1	1	1	28	194	122
6	32	1	4	2	44	200	142
7	10	1	1	1	13	200	131
8	12	1	6	3	11	195	111
9	29	1	1	1	133	183	163
10	8	1	1	1	77	185	139
11	15	1	1	1	19	194	120
12	4	1	14	5	33	159	88
13	3	1	1	1	39	180	127
14	25	1	1	1	33	190	129
15	10	1	1	1	6	59	18
16	28	11	196	57	39	196	145
17	9	1	1	1	2	22	10
18	1	1	1	1	1	128	51
19	9	27	n/a	92	14	180	110
20	11	1	6	3	37	199	146

6.4 Achievement of objectives

In Section 4.1.2 the load plan fitness $F_{pop}(i)$ is introduced as the fitness value of an individual in the GA population and consequently a measure of how good a load plan is. In Section 4.3.2 $f(l)$ is used to denote a load plan fitness component when a load plan is being evaluated, and to capture the relation to an individual in the GA population this notation can be extended to $f_{pop}(i,l)$. Recall that $0 \leq f_{pop}(i,l) \leq 1, \forall l$ where $f_{pop}(i,l) = 1$ is the most desirable. As shown in Table 5 each component $f_{pop}(i,l)$ is related to an objective for loading from Section 2.3.2. In order to quantify the achievement of objectives, the fitness component values for the best solution, i.e. $f_{pop}(i^*,l), \forall l$, in the last generation for the 10 runs are studied for each problem instance in Table 14. The 15 fitness components for Batch 1 are listed below, with the associated objective in parenthesis:

1. Leftover boxes none (1.1)
2. Leftover boxes fewer (1.1)
3. Leftover boxes volume lower (1.2)
4. Leftover boxes weight lower (1.3)
5. Promising spaces for consolidation pallets (2.1)
6. Horizontal stability behind boxes (2.2)
7. Horizontal stability in front, left and right of boxes (2.2)
8. Utilization of floor space higher (2.2)
9. Soft constraints for boxes satisfied (2.2)
10. Center of gravity lengthwise closer to center (3.1)
11. Center of gravity widthwise closer to center (3.2)
12. Center of gravity heightwise closer to floor (3.3)
13. Boxes with pallet not rotated (4.1)
14. Boxes with pallet reachable by forklift (4.2)
15. Boxes without pallet not on floor (4.3)

The values of $f_{pop}(i^*,l), \forall l$ for Batch 1 are presented as radar charts in Figures 55-58 for the 20 problems, with each nuance representing one of the 10 runs in Table 14. When all series are not visible it is because the values are close or equal, and thereby markers and lines are on top of each other. All values equal do not imply identical load plans, but that is obviously a possibility.

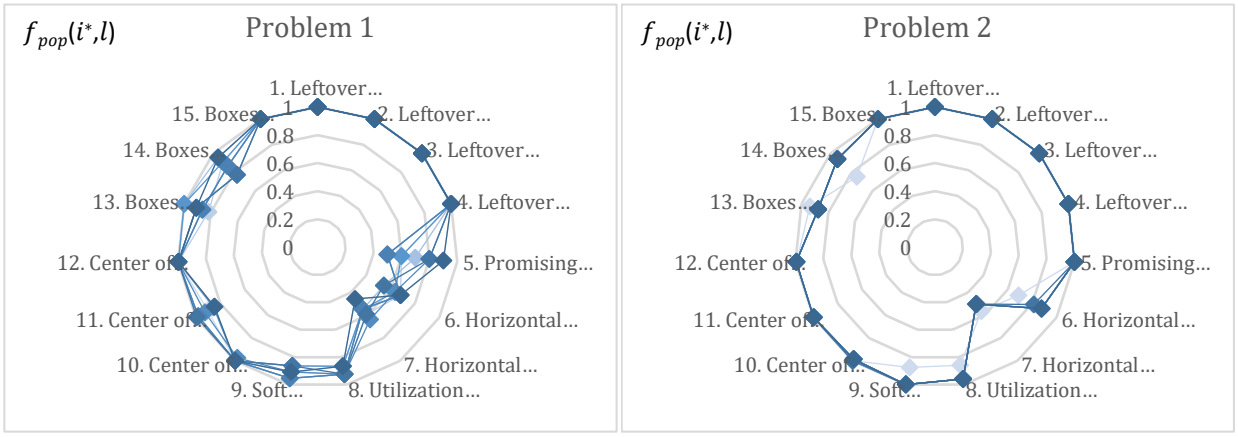

Figure 55. Load plan fitness component values for the best solutions in the last generation for Problems 1-2

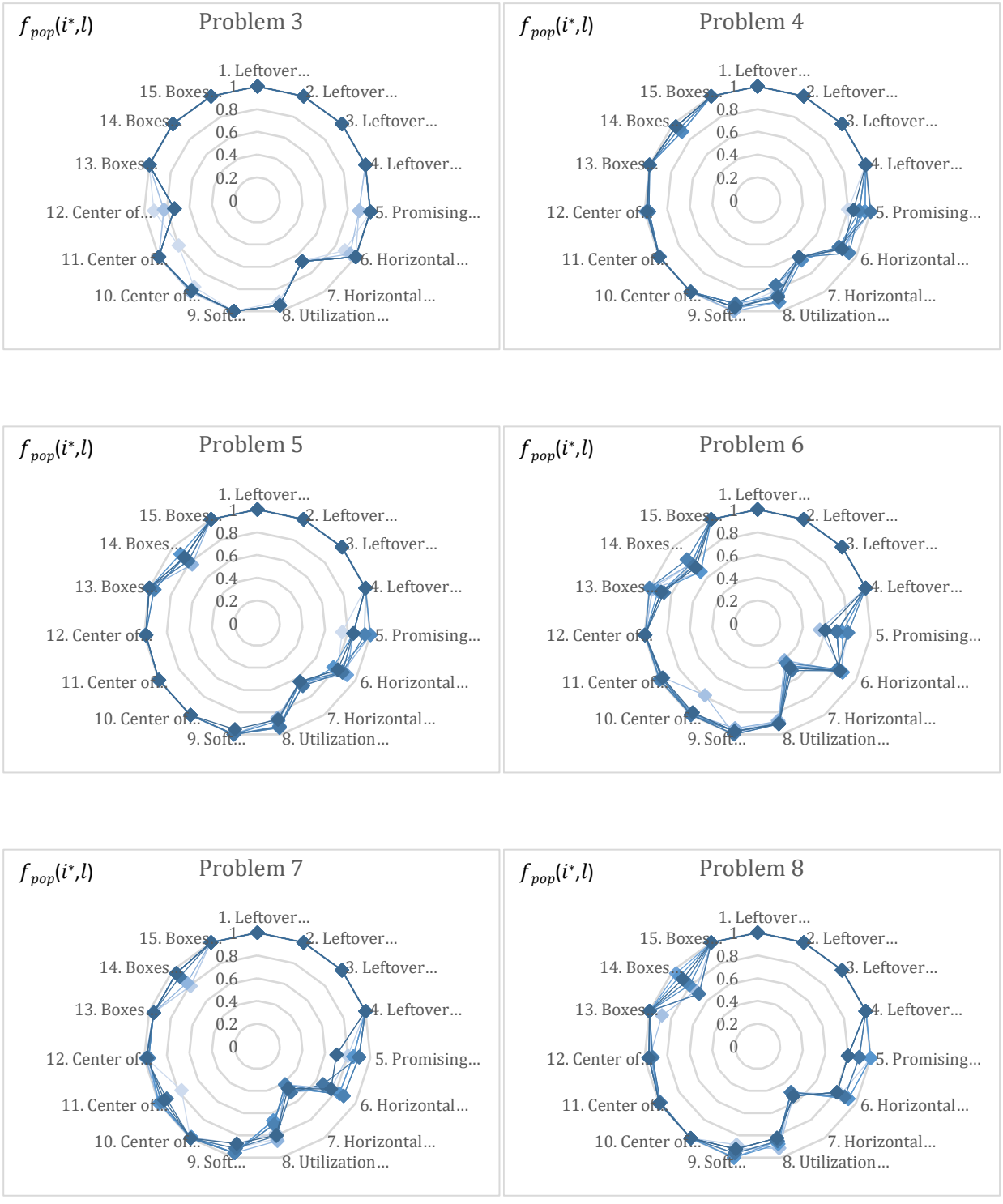

Figure 56. Load plan fitness component values for the best solutions in the last generation for Problems 3-8

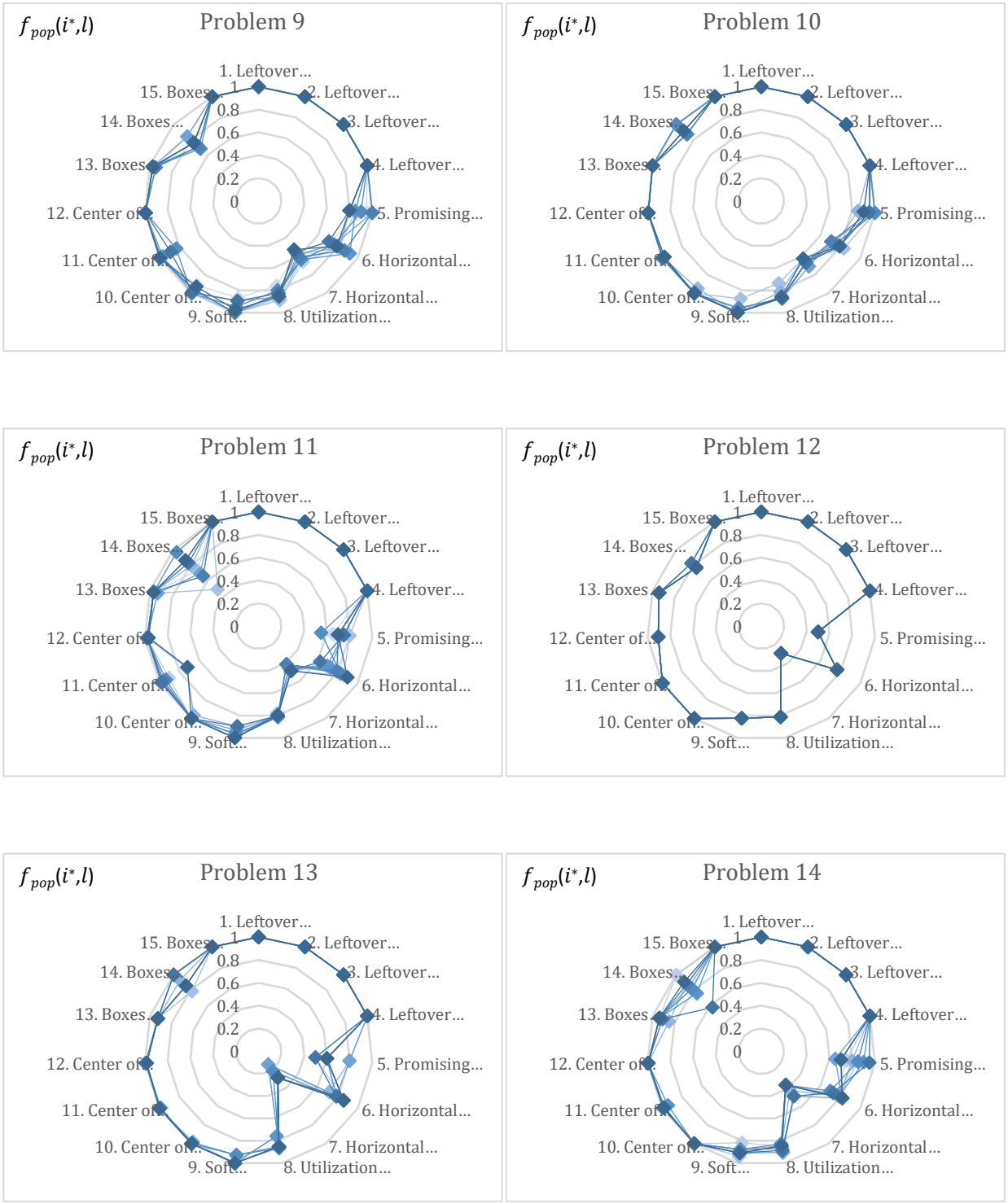

Figure 57. Load plan fitness component values for the best solutions in the last generation for Problems 9-14

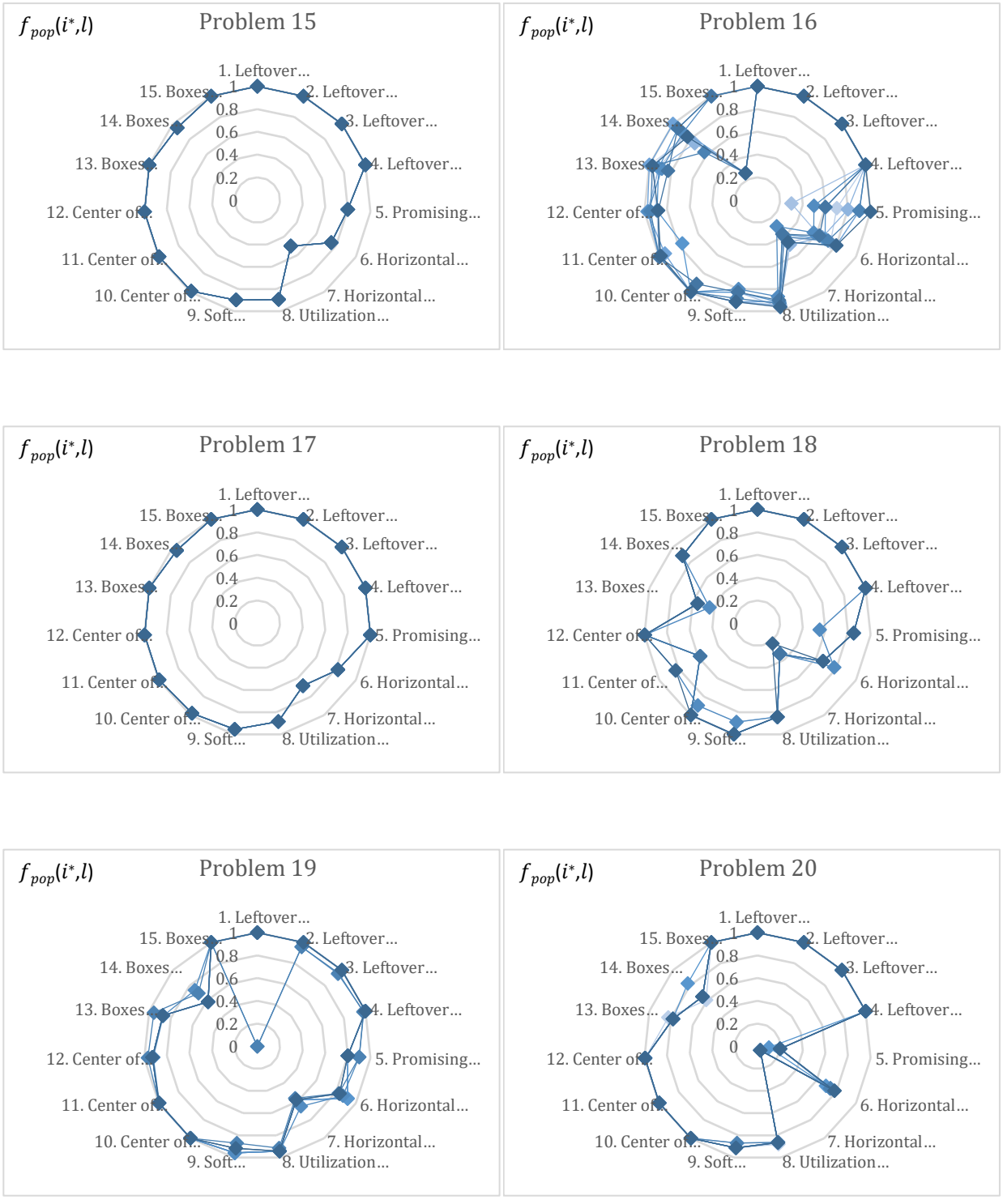

Figure 58. Load plan fitness component values for the best solutions in the last generation for Problems 15-20

When considering the fitness components, recall from Section 4.3.2 that certain components have more weight in the fitness $F_{pop}(i^*)$. This is for example true for $f_{pop}(i^*, 2)$, $f_{pop}(i^*, 3)$ and $f_{pop}(i^*, 4)$.

One general observation from Figures 55-58 is that the objectives to large extent are achieved. Another observation is that the results typically are similar over the 10 runs, which indicates stability in the proposed approach. Some more specific observations are discussed below.

$f_{pop}(i^*, 1)$ to $f_{pop}(i^*, 4)$ are always exactly 1 except for Problem 19, which is related to the statement in Section 6.3 saying that all packages fit for every run and problem except for two runs of Problem 19. Among these four fitness components $f_{pop}(i^*, 1)$ stands out as $f_{pop}(i^*, 1) = 0$ if there is any leftover box, independently of the number, volume and weight that is leftover.

$f_{pop}(i^*, 5)$ and $f_{pop}(i^*, 14)$ stands out as two of the fitness components with highest variability. Regarding $f_{pop}(i^*, 5)$ which refers to promising spaces for consolidation pallets, significant deviations from 1 may not be an issue at all given our current design of this component, where any $f_{pop}(i^*, 5) > 0$ implies at least some spaces for consolidation pallets and $f_{pop}(i^*, 5) = 1$ may in fact be a lot more space than necessary. From this we identify an opportunity for improvement in terms of redesigning our $f_{pop}(i^*, 5)$ to better reflect how much space for consolidation pallets that is actually needed, based on packages assigned to Batch 2. By not striving for more space than necessary, other objectives may be achieved to even greater extent.

Regarding $f_{pop}(i^*, 14)$ which refers to boxes with pallet reachable by forklift, deviations from 1 is expected, based on feedback from the DC operators. The scenario illustrated in Figure 22 in Section 3.4.5 is not desirable, but as described in Section 2.3.1 on Goal 4 there are several options this is dealt with in practice. One frequently used option is to place the package above on top of the package below outside the shipping container with more leeway, and then lift both packages into the shipping container together. The achievement of $f_{pop}(i^*, 14)$ could be improved by including a corresponding BOCP score component, but then other goals would suffer.

Further, $f_{pop}(i^*, 6)$ is higher than $f_{pop}(i^*, 7)$ in agreement with the feedback from DC operators that support from behind is most important, as mentioned in Section 4.3.3. Moreover, $f_{pop}(i^*, 7) = 1$ would imply support in front, left and right of all boxes, which most times is practically impossible. Some problems stands out with very low $f_{pop}(i^*, 7)$ however, for example Problem 20. As seen in Figure 59, this is partly due to three very long packages, but also due to a gap along the right wall which is wider than our constant $CLOSE$ (see examples in Section 4.3.2 and 4.4.3) currently set to 0.1 meter. $CLOSE$ may be set too low as this gap would not be considered an issue in practice, and the packages would likely be placed adjacent to the wall upon loading anyway.

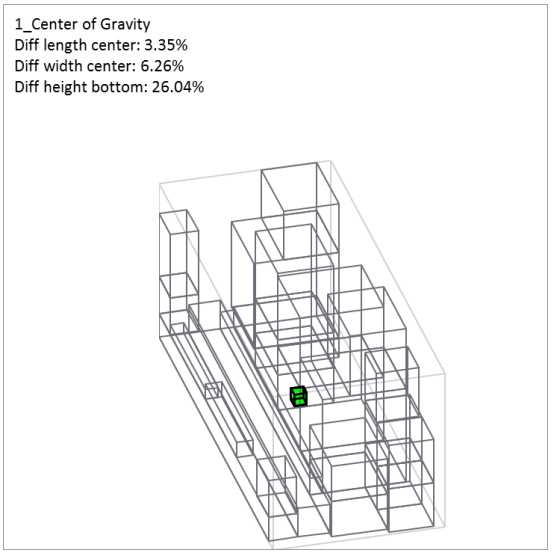

Figure 59. Completed loading of Problem 20

6.5 BOCP score coefficients

In Section 4.1.2 it is established that BOCP score coefficients $W_{pop}(i)$ is a fundamental element in our approach. It is explained that the objective for the GA is to find a set of coefficients $W_{pop}(i)$ that results in smart placements, which in turn result in a good load plan. In particular we argued that it would be very difficult to develop a fixed set of weighted coefficients $W_{pop}(i_{fixed})$ that would work for any problem instance. In this section we first demonstrate that the argument was sound, i.e. that the dynamic weighting of coefficients which the improvement heuristic offers really is necessary.

In order to conduct the analysis, the four most difficult problem instances from Table 14 in terms of fitting all packages are mainly used, namely Problems 8, 12, 16 and 19. Recall from Section 4.4.3 that each BOCP score component $s(j,k)$ is associated with a BOCP score coefficient $w_{pop}(i,j)$. The coefficient values for the best solution, i.e. $w_{pop}(i^*,j), \forall j$, in the last generation for the 10 runs are presented as box plot charts for each of these four problem instances in Figure 60. Charts for the remaining 16 problem instances are presented in Figures 74-77 in Section 9.3. Note that it is still Batch 1 that is being considered. As per convention, the lines in a box represent the first quartile, the median and the third quartile, while the extended lines represents the minimum and maximum value respectively.

Recall that for Batch 1 there are 10 BOCP score components of type 1 and 10 components type 2, as presented in Table 8 in Section 4.4.3. Also recall our interpretation of type 1 as the baseline score and type 2 mainly as being bias towards more difficult aspects. As the reader may understand from Section 2.2.1, the presence of more difficult aspects is to large extent based on the characteristics of the packages, which in turn vary greatly between problem instances. As a result of this, several type 2 components may be redundant for a given problem instance. For example, component 15 which rewards a box submitted as "must be placed on the floor", is redundant for a problem instance without any box with that exception.

From studying the box plot charts in Figures 60 and 74-77, an initial remark is that our design seems fairly robust in the sense that, for a given problem instance, all coefficients do not have to be tuned into a narrow interval in order to generate a good load plan. On the contrary, it seems like for each problem instances, several coefficients can adopt values within a fairly wide range, and this is not only true for potentially redundant type 2 components. At the same time, for all problem instances there are clear patterns that certain aspects of the current situation upon each placement should be rewarded relatively more or less. By more carefully comparing the charts in Figure 60 with each other, it is possible to observe indications which support our proposition that the dynamic weighting is necessary.

- $w_{pop}(i^*,1)$ is higher for Problem 12 than for Problem 16 (and 8)
- $w_{pop}(i^*,3)$ is higher for Problem 19 (and 16) than for Problem 12 (and 8)
- $w_{pop}(i^*,7)$ is higher for Problem 8 (and 19) than for Problem 12
- $w_{pop}(i^*,8)$ is higher for Problem 16 (and 8) than for Problem 19
- $w_{pop}(i^*,13)$ is higher for Problem 8 than for Problems 16 and 19
- $w_{pop}(i^*,14)$ is higher for Problem 12 than for Problems 16 and 19

Clear patterns can also be observed from the other 16 problem instances in Figures 74-77 in Section 9.3. Hence it seems evident that the GA is capable of tuning the coefficients for the particular problem instance being solved. Note that this was over 200 generations, with more generations and/or more runs we would expect the boxes in the plots to shrink.

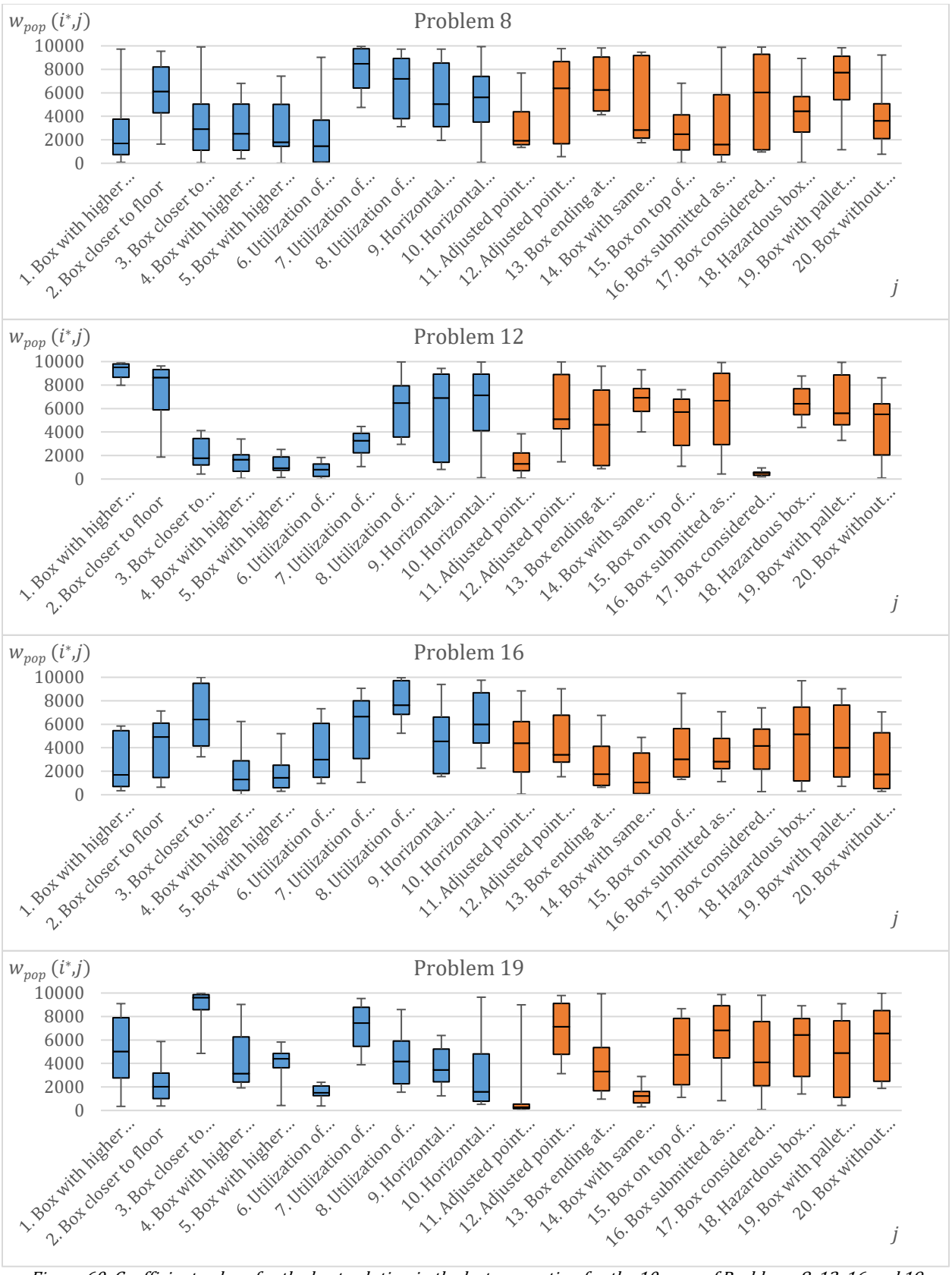

Figure 60. Coefficient values for the best solution in the last generation for the 10 runs of Problems 8, 12, 16 and 19

It is not obvious how to explain the distribution of coefficients values for a given problem instance. We assume that in general dependence between coefficients and characteristics of the problem instance occur, as well as dependence between different coefficients. With the intent to demonstrate such occurrences, some observations related to the four most difficult problem instances are made.

Regarding Problem 8, consider the partially completed loading in Figures 61-63 below:

Figure 61. Partially completed loading of Problem 8 (1/3)

Figure 62. Partially completed loading of Problem 8 (2/3)

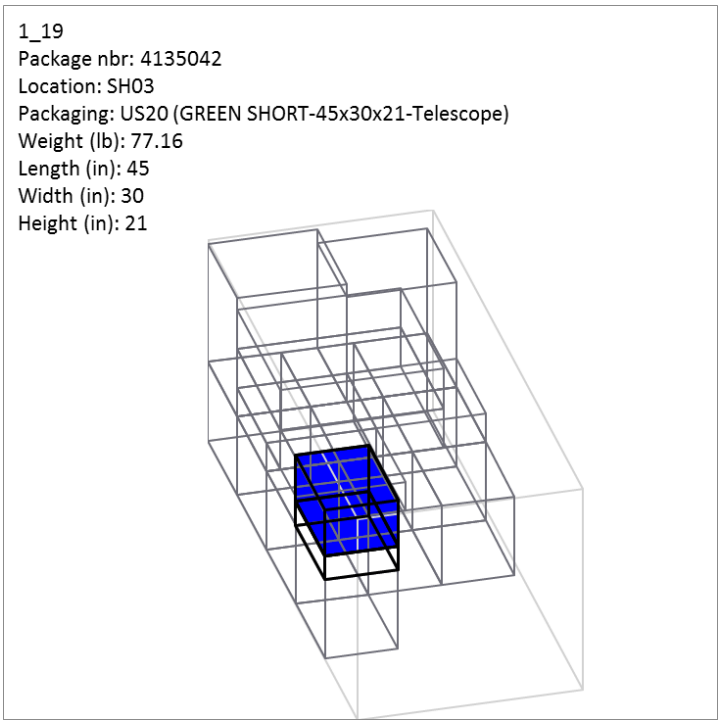

Figure 63. Partially completed loading of Problem 8 (3/3)

The packages in the back of the container fit nicely (with little wasted space) both widthwise and heightwise. In terms of BOCP score component, this is related to Batch 1 component 7 and 8, namely "utilization of space width higher" and "utilization of space height higher". In agreement with this, $w_{pop}(i^*, 7)$ and $w_{pop}(i^*, 8)$ are relatively high for Problem 8. The tight fit in the back is possible thanks to packages with the same height being placed adjacent to each other on the floor, so that packages stacked above can overlap the packages below. In terms of BOCP score component, this is related to Batch 1 component 13, namely "box ending at same height as other adjacent box to the left". In agreement with this, $w_{pop}(i^*, 13)$ is relatively high for Problem 8. Further, note that $w_{pop}(i^*, 16)$ is slightly low for Problem 8. It turns out that there are three packages, more specifically drums of oil, which are submitted as "must be placed on the floor". The bottom-left figure on the front page of this report shows the final arrangement of Problem 8, and in this figure two of the drums can be seen. Hence the slightly low value of $w_{pop}(i^*, 16)$ could be interpreted as potentially an enabler for the drums of oil to be placed closer to the door, i.e. rather later than sooner.

Regarding Problem 12, note that $w_{pop}(i^*, 1)$ and $w_{pop}(i^*, 2)$ are relatively high, indicating that it is beneficial to reward placements closer to the floor, especially for heavy boxes. At the same time we learned from experiments and the loading in practice of this particular instance that it was necessary to place package 4128780 on top of package 4151122 in order to fit all packages. This is exemplified in Figures 64 and 65 below which both show partially completed loadings. The position of the stack made up of 4151122 and 4128780 is not limited to one position within the sea container however.

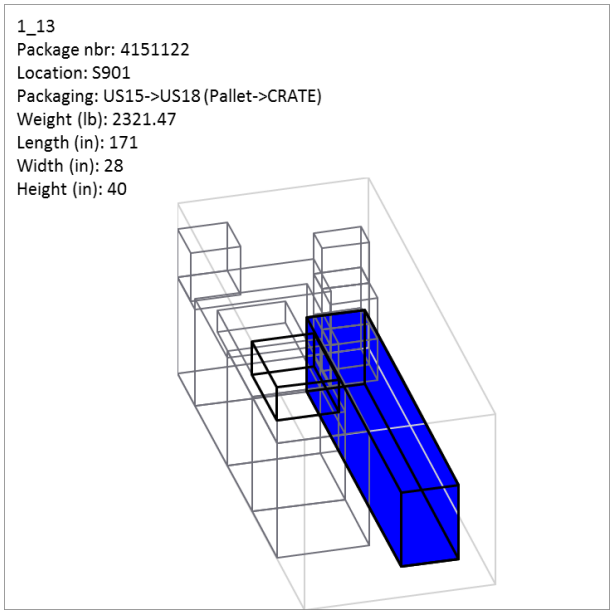

Figure 64. Partially completed loading of Problem 12 (1/2)

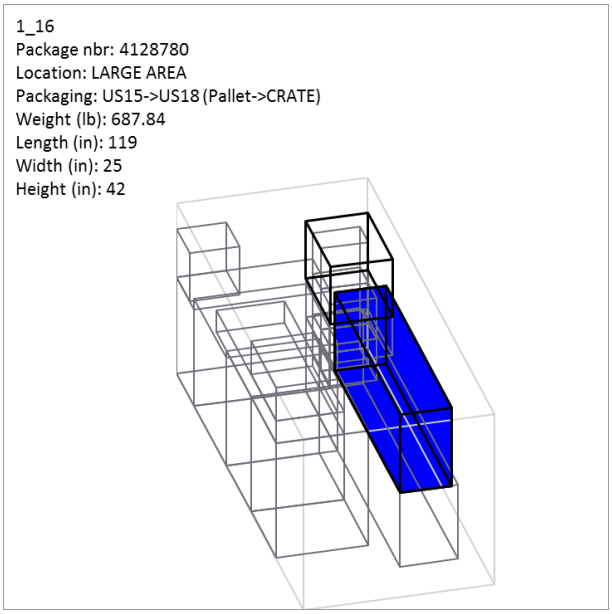

Figure 65. Partially completed loading of Problem 12 (2/2)

Now, note that $w_{pop}(i^*, 17)$ is distinctively very low for Problem 12, which is related to the BOCP score component "box considered to be long, with the largest dimension greater than a certain threshold (1.5 m)". Our criteria to consider a package as long is if the ratio of the largest versus the second largest dimension exceeds 2.5. This is true for both 4151122 and 4128780, and even though 4128780 is much lighter than 4151122, 4128780 is still fairly heavy relative the other packages in Problem 12. We may now conceive that if $w_{pop}(i^*, 17)$ was not very low, it would likely cause 4151122 and 4128780 to be placed immediately, next to each other on the floor. This would be devastating, as all packages would not be able to fit. Further, note that $w_{pop}(i^*, 18)$ is slightly high for Problem 12. It turns out that there are one package that is submitted as "not ok to stack a similar package above", which is one of the criteria for BOCP component 18. More specifically it was a piece of glass, just like in Figure 8 in Section 2.2.1.

Regarding Problem 16, note that $w_{pop}(i^*, 8)$ is relatively high. This indicates that utilization of space height is important for this problem, which is in agreement with how nicely the packages fit heightwise in the partially completed loadings in Figures 66 and 67 below, as well as the completed loading in the upper-left figure on the front page of this report.

Figure 66. Partially completed loading of Problem 16 (1/2)

Figure 67. Partially completed loading of Problem 16 (2/2)

Regarding Problem 19, the entire load plan is presented in Section 6.2.1. The gigantic package 4204332 to the left in Figure 47 almost inevitable becomes the first placement due to its weight, high space utilization widthwise and the fact that it was submitted as "must be placed on the floor". Since all packages in the instance make up to as much as 80% volume utilization of the sea container, it becomes critical to utilize the space on top of the large crate, without exceeding stacking strength constraints. To utilize the space, relatively low $w_{pop}(i^*, 2)$ and distinctively high $w_{pop}(i^*, 3)$ seems very appropriate indeed. Note that the description of component 3 is "box closer to intersection of inner wall and ceiling lengthwise". Moreover, relatively high $w_{pop}(i^*, 12)$ seems suitable to use more forward adjusted placement points, and thereby receive better support from below. The partially completed loadings in Figures 68 and 69 shows the stacking on top of 4204332.

Figure 68. Partially completed loading of Problem 19 (1/3)

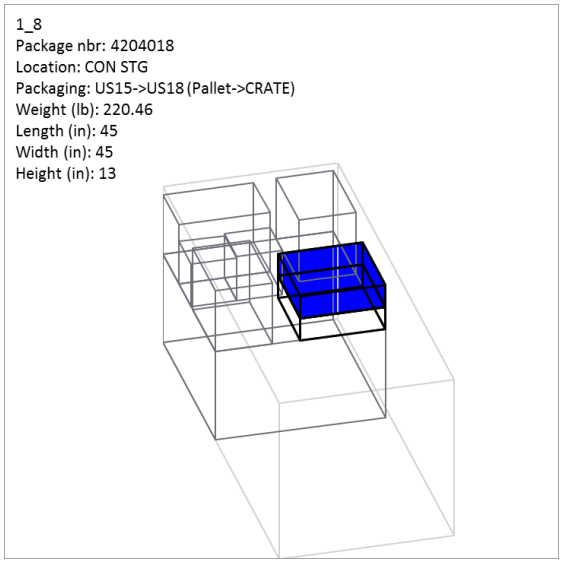

Figure 69. Partially completed loading of Problem 19 (2/3)

Further, similar to Problem 8, $w_{pop}(i^*, 7)$ is relatively high for Problem 19. The partially completed loading in Figure 70 and the completed loading in Figure 71 below show that the packages fit nicely (with little wasted space) widthwise. Figure 71 is also a great example when packages below the threshold of 200 lbs. are rotated against the pallet orientation in order to fit.

Figure 70. Partially completed loading of Problem 19 (3/3)

Figure 71. Completed loading of Problem 19

The observant reader might wonder why $w_{pop}(i^*, 8)$ is not higher for Problem 19 following the same reasoning as for Problems 8 and 16. Perhaps higher $w_{pop}(i^*, 8)$ is simply not necessary to achieve a load plan identical or similar to the one in Section 6.2.1, or perhaps higher $w_{pop}(i^*, 8)$ would result in too much bias towards space utilization heightwise. As stated previously, it is not obvious how to explain the distribution of coefficients values for a given problem instance, so that the dynamic weighting is indeed necessary.

7 Conclusions

The container loading problem solved in this thesis is highly constrained relative existing literature in the field. An optimization software was developed for the industrial company Atlas Copco to be used for sea container shipments in practice and hence all applicable practical constraints faced by the operators at the Distribution Center (DC) had to be treated properly. The purpose of the software was to reduce uncertainty as well as labor usage in the shipping process, by generating visual step-by-step instructions for how to load containers, called load plans. Promising ideas for the proposed method were obtained from the literature as well as continuous feedback from the DC operators.

The proposed method is based on three types of mechanisms:
1) To place boxes
2) To evaluate load plans
3) To guide 1) to generate load plans that are evaluated as "good" by 2)

The third mechanisms is accomplished using scoring of possible placements, with tuning of parameters using a Genetic Algorithm (GA). The placement with the highest score is anticipated as "smartest" and therefore selected to be performed. Aspects considered in the scoring are for example space utilization, horizontal support and heavier boxes closer to the floor. The *batch concept* is another fundamental design element of our approach, which refers to grouping of boxes based on their characteristics and solving subproblems in terms of partial load plans.

Our approach has been proven successful. The scoring for possible placements and load plan evaluation works well together. Most objectives that are captured by the load plan evaluation have corresponding scoring components, directly and/or indirectly, while a few objectives are merely rewarded in the load plan evaluation. A strength with the design is the inherent flexibility, with the ability to add, remove or change objectives in a modular manner. The choice of metaheuristic seems reasonable, as the GA is capable of tuning the scoring coefficients to suit the particular problem instance being solved. For a given problem instance, certain aspects seem to be more important, i.e. the coefficients should adopt values within a certain range. The objectives for a load plan are to large extent achieved for heterogeneous problem instances. Typically, equally good solutions are obtained for different runs of the same problem instance, which indicates stability in the proposed method.

The purpose of the software was fulfilled. The software is capable of generating satisfactory load plans, including loads considered difficult by the DC operators, within acceptable computation times. The laptops in the forklifts are used to display load plans when loading, suggesting the placement and orientation of each package. Thanks to this, the previous test loading activity called "pre-staging" has been eliminated. The software is being run repeatedly as new orders are received, reducing delays in ordering containers, while also being confident about fitting all packages in the number of containers orders, with as little excess capacity as possible. Additional benefits have also been identified, such as ability to provide customers with visual load manifests to identify exactly where an item is located within a container.

Future work may include improvements of the objectives for a load plan as well as the scoring of possible placements. Regarding objectives for load plans, one example is redesigning how promising spaces for consolidation pallets are rewarded, as discussed in Section 6.4. Regarding scoring of possible placements, one example is to extend the analysis conducted in Section 6.5 over more problem instances and to also include correlation of the parameters (BOCP score coefficients). Such analysis may lead to adding, removing and/or changing aspects (BOCP score components). Future work may also include evaluation of the parameter settings for the GA, as mentioned in Section 4.5.2.

8 References

Balakirsky, S., Proctor, F., Kramer, T., Kolhe, P. and Christensen, H. I. (2010), Using simulation to assess the effectiveness of pallet stacking methods, Simulation, Modeling, and Programming for Autonomous Robots **2010**, Lecture Notes in Artificial Intelligence 6472, 336-349

Bortfeldt, A. (2012), A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints, Computers and Operations Research **39** (9), 2248-2257

Bortfeldt, A. and Gehring, H. (1999), Zur behandlung von restriktionen bei der stauraumoptimierung am beispiel eines genetischen algorithmus für das containerbeladeproblem, Logistik Management – Intelligente I + K Technologien, Springer, Berlin, 83-100

Bortfeldt, A. and Gehring, H. (2001), A hybrid genetic algorithm for the container loading problem, European Journal of Operational Research **131** (1), 143-161

Bortfeldt, A. and Wäscher, G. (2013), Constraints in container loading – a state-of-the-art review, European Journal of Operational Research **229** (1), 1-20

Ceschia, S. and Schaerf, A. (2013), Local search for a multi-drop multi-container loading problem, Journal of Heuristics **19** (2), 275-294

Douglas, M. (2008), Solving the cube, Inbound Logistics, February, www.inboundlogistics.com/cms/article/solving-the-cube, 2016-10-21

Fibre Box Association (2008), Edge crush test, www.fibrebox.org/news/?EventType=1&Year=2008&Category=5, www.fibrebox.myshopify.com/products/ect-application-and-reference-guide-1, 2016-07-22

Fuellerer, G., Doerner, K. F., Hartl, R. F. and Iori, M. (2010), Metaheuristics for vehicle routing problems with three-dimensional loading constraints, European Journal of Operational Research **201** (3), 751-759

Gehring, H. and Bortfeldt, A. (1997), A genetic algorithm for solving the container loading problem, International Transactions in Operational Research **4** (5), 401-418

Gehring, H. and Bortfeldt, A. (2002), A parallel genetic algorithm for solving the container loading problem, International Transactions in Operational Research **9** (4), 497-511

Gendreau, M., Iori, M., Laporte, G. and Martello, S. (2006), A tabu search algorithm for a routing and container loading problem, Transportation Science **40** (3), 342-350

Iori, M. and Martello, S. (2010), Routing problems with loading constraints, TOP **18** (1), 4-27

Lin, J. L., Chang, C. H. and Yang, J. Y. (2006), A study of optimal system for multiple-constraint multiple-container packing problems, Advances in Applied Artificial Intelligence **2006**, Lecture Notes in Artificial Intelligence 4031, 1200-1210

MagicLogic, Cube-IQ, www.magiclogic.com, 2016-10-21

Makarem, O. C. and Haraty, R. A. (2010), Smart container loading, Journal of Computational Methods in Sciences and Engineering **10** (1/2), 231-245

McKee, R. C., Gander, J. W. and Wachuta, J. R. (1963), Compression strength formula for corrugated boxes, Paperboard Packaging **48** (8), 149-159

Mining & Construction (2013), Full marks in Svappavaara for FlexiROC T45, www.miningandconstruction.com/mining/full-marks-in-svappavaara-for-flexiroc-t45-2500, 2016-09-12

Ortec, Load Building, www.ortec.com/library-item/ORTEC-Brochure-Load-Building-EN, 2016-10-21

Talbi, E. G. (2009), Metaheuristics: From Design to Implementation, John Wiley & Sons, Hoboken

Tarantilis, C. D., Zachariadis, E. E. and Kiranoudis, C. T. (2009), A hybrid metaheuristic algorithm for the integrated vehicle routing and three-dimensional container-loading problem, IEEE Transactions on Intelligent Transportation Systems **10** (2), 255-271

Techanitisawad, A. and Tangwiwatwong, P. (2004), A GA-based heuristic for the interrelated container selection loading problems, Industrial Engineering and Management Systems **3** (1), 22-37

Wang, L., Guo, S., Chen, S., Zhu, W. and Lim, A. (2010), Two natural heuristics for 3D packing with practical loading constraints, Pacific Rim International Conference on Artificial Intelligence **2010**, Lecture Notes in Artificial Intelligence 6230, 256-267

Wikimedia Commons (2007), Container, Bahn-Güterterminal Augsburg Oberhausen, commons.wikimedia.org/wiki/File:Container_Augsburg.jpg, 2016-09-12

Wikimedia Commons: Buonasera (2014), Container ship Emma Maersk with destination Port of Hamburg in June 2014, commons.wikimedia.org/wiki/File:Container_ship_Emma_Maersk_with_Destination_Port_of_Hamburg.png, 2016-09-12

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

Zhao, X., Bennell, J.A., Bektaş, T. and Dowsland, K. (2016), A comparative review of 3D container loading algorithms, International Transactions in Operational Research **23** (1-2), 287-320

9 Appendixes

The following sections are referred to in earlier chapters.

9.1 Packaging data

Table 15. Packaging data including dimensions and other attributes discussed in Section 2.2.1

Packaging	Stand-ard	Tele-scopic	Pallet Attached	L: Outer Length [m]	W: Outer Width [m]	H: Outer Height [m]	Outer Volume [m^3]	Q = Max(L, W, H) / Min(L, W, H)
7x7x7	yes	no	no	0.180	0.180	0.180	0.006	1.0
49x5x5	yes	no	no	1.245	0.127	0.127	0.020	9.8
11x11x11	yes	no	no	0.280	0.280	0.280	0.022	1.0
31x8x8	yes	no	no	0.787	0.203	0.203	0.032	3.9
13x13x13	yes	no	no	0.330	0.330	0.330	0.036	1.0
17X17X8	yes	no	no	0.432	0.432	0.203	0.038	2.1
40x11x11	yes	no	no	1.016	0.280	0.280	0.080	3.6
17x17x17	yes	no	no	0.432	0.432	0.432	0.081	1.0
21x16x16	yes	no	no	0.533	0.406	0.406	0.088	1.3
21x21x17	yes	no	no	0.533	0.533	0.432	0.123	1.2
23x15x22 ORANGE SHORT SKID	yes	yes	yes	0.584	0.381	0.559	0.124	1.5
25x25x17	yes	no	no	0.635	0.635	0.432	0.174	1.5
ORANGE TALL PALLET-23x15x38	yes	yes	yes	0.584	0.381	0.965	0.215	2.5
BLUE PALLET-24X24X26	yes	no	yes	0.610	0.610	0.660	0.246	1.1
26x26x26	yes	no	no	0.660	0.660	0.660	0.287	1.0
GREEN SHORT-45x30x21-Telescope	yes	yes	yes	1.143	0.762	0.533	0.464	2.1
GREEN TUBE-45x30x36	yes	no	yes	1.143	0.762	0.914	0.796	1.5
GREEN TALL PALLET-45x30x38	yes	yes	yes	1.143	0.762	0.965	0.840	1.5
YELLOW -45X45X38-Telescope	yes	yes	yes	1.143	1.143	0.965	1.261	1.2
YELLOW WALK-IN-45x45x62	yes	no	yes	1.143	1.143	1.575	2.058	1.4
BOX	no	no	no	n/a	n/a	n/a	n/a	n/a
CRATE	no	no	yes	n/a	n/a	n/a	n/a	n/a
Large Tube	no	no	no	n/a	n/a	n/a	n/a	n/a
Pallet	no	no	yes	n/a	n/a	n/a	n/a	n/a
Small Tube	no	no	no	n/a	n/a	n/a	n/a	n/a

9.2 Functions

These functions are used for various objectives and constraints (with different parameter values).

9.2.1 SCurve

$$SCurve(x, midpoint, steepness) = \frac{1}{1 + e^{-steepness \times (x - midpoint)}}$$

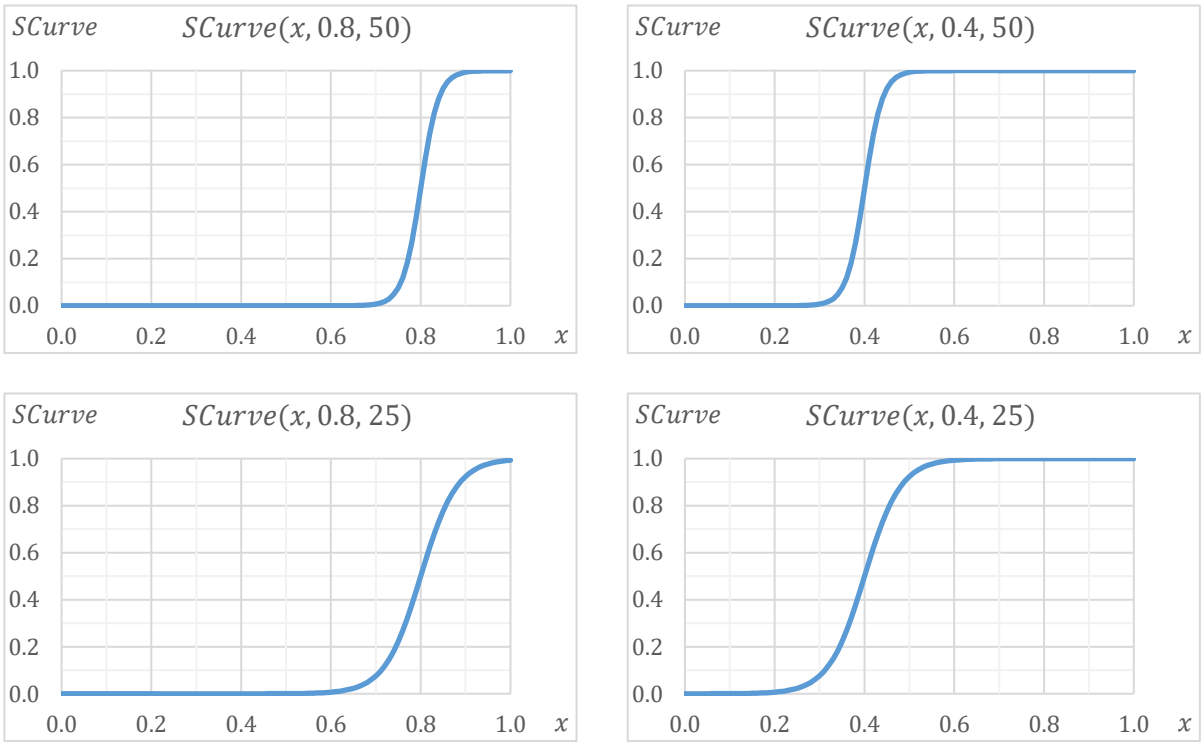

Figure 72. Examples of SCurve function for different parameter values

9.2.2 Scaling

$$Scaling(x, factor) = \frac{factor}{factor + (1 - x)}$$

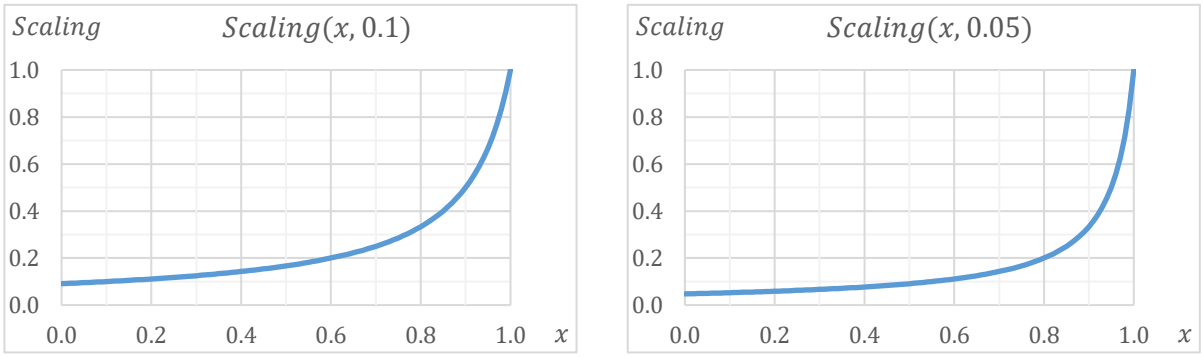

Figure 73. Examples of Scaling function for different parameter values

9.3 BOCP score coefficients: remaining problem instances

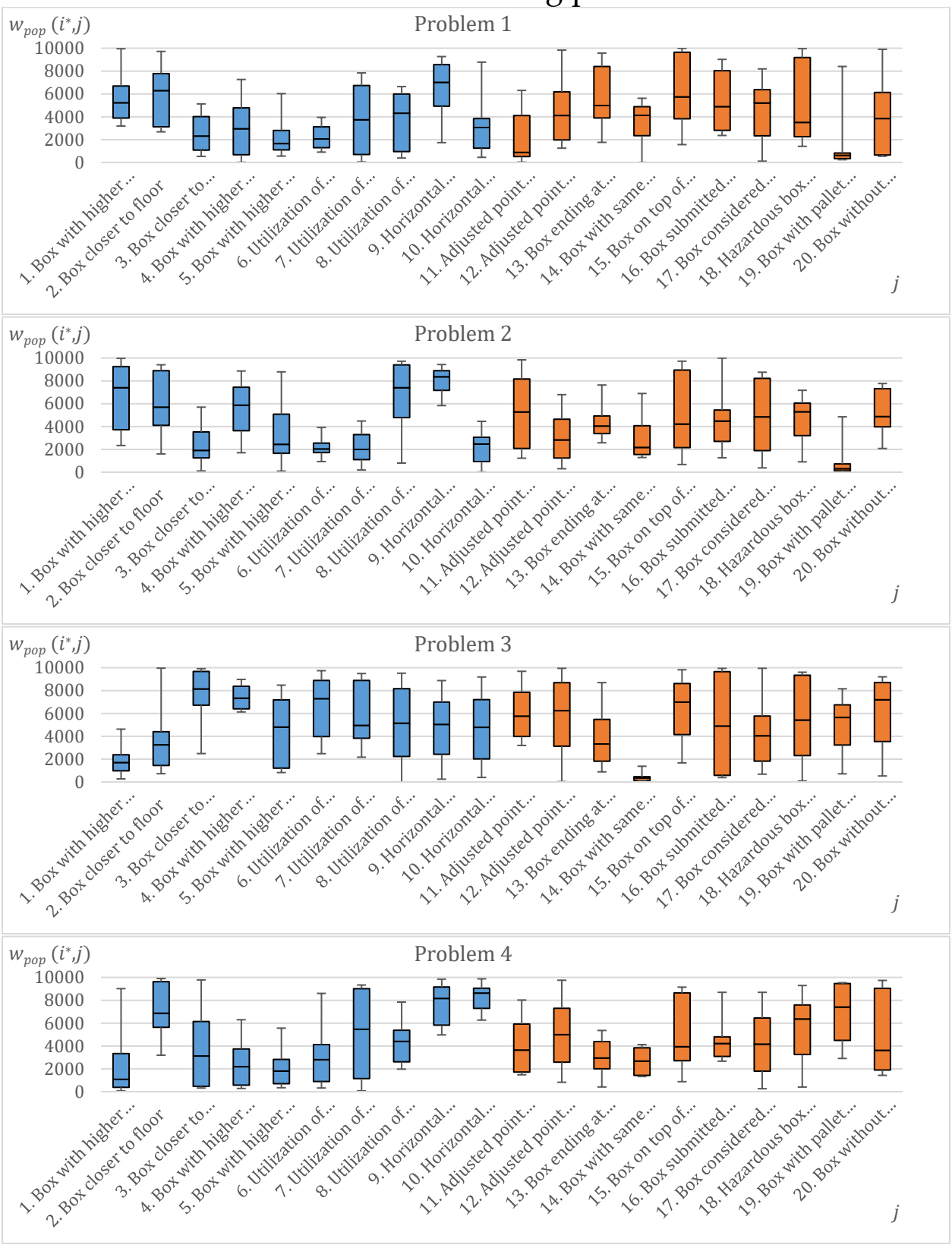

Figure 74. Coefficient values for the best solution in the last generation for the 10 runs of Problems 1, 2, 3 and 4

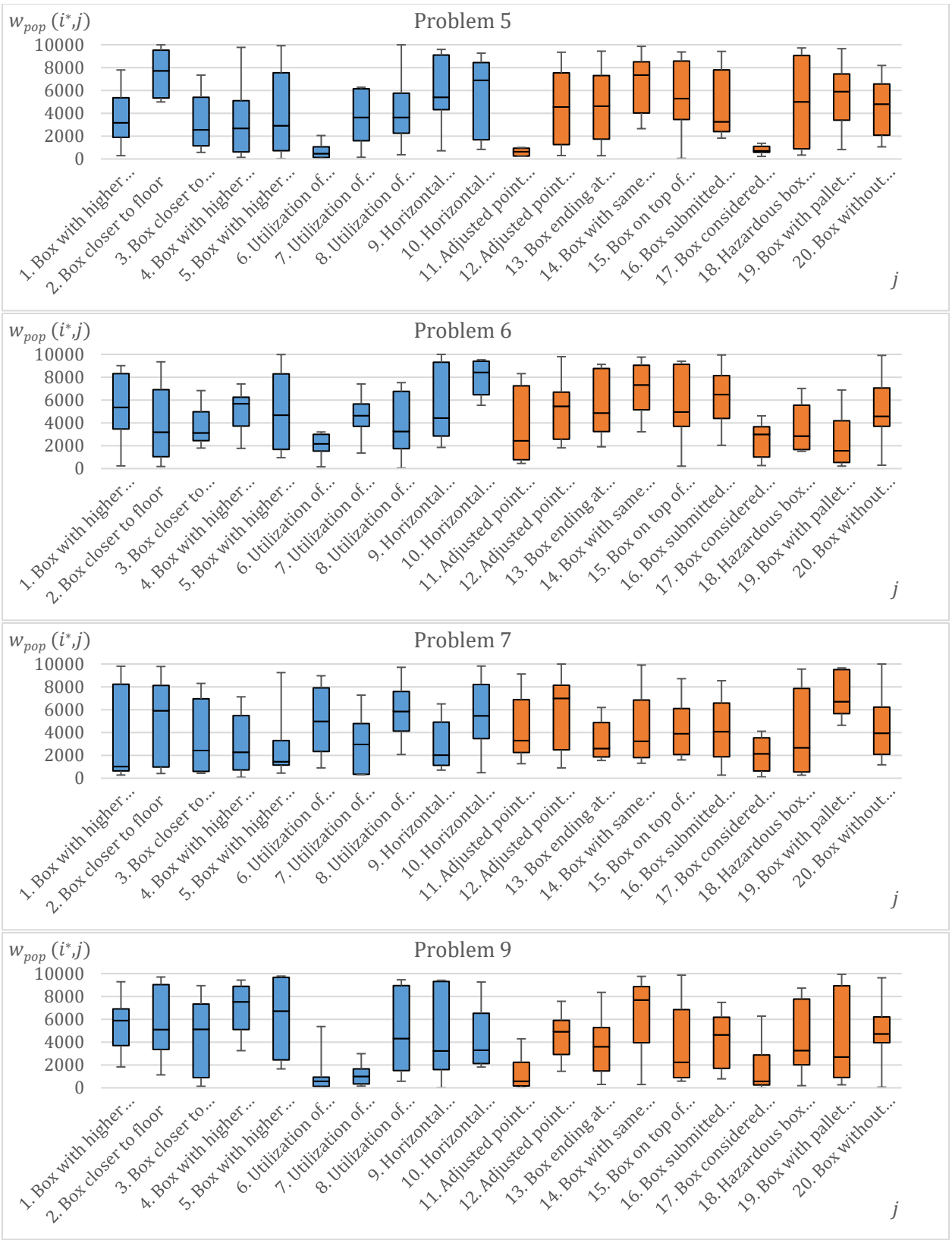

Figure 75. Coefficient values for the best solution in the last generation for the 10 runs of Problems 5, 6, 7 and 9

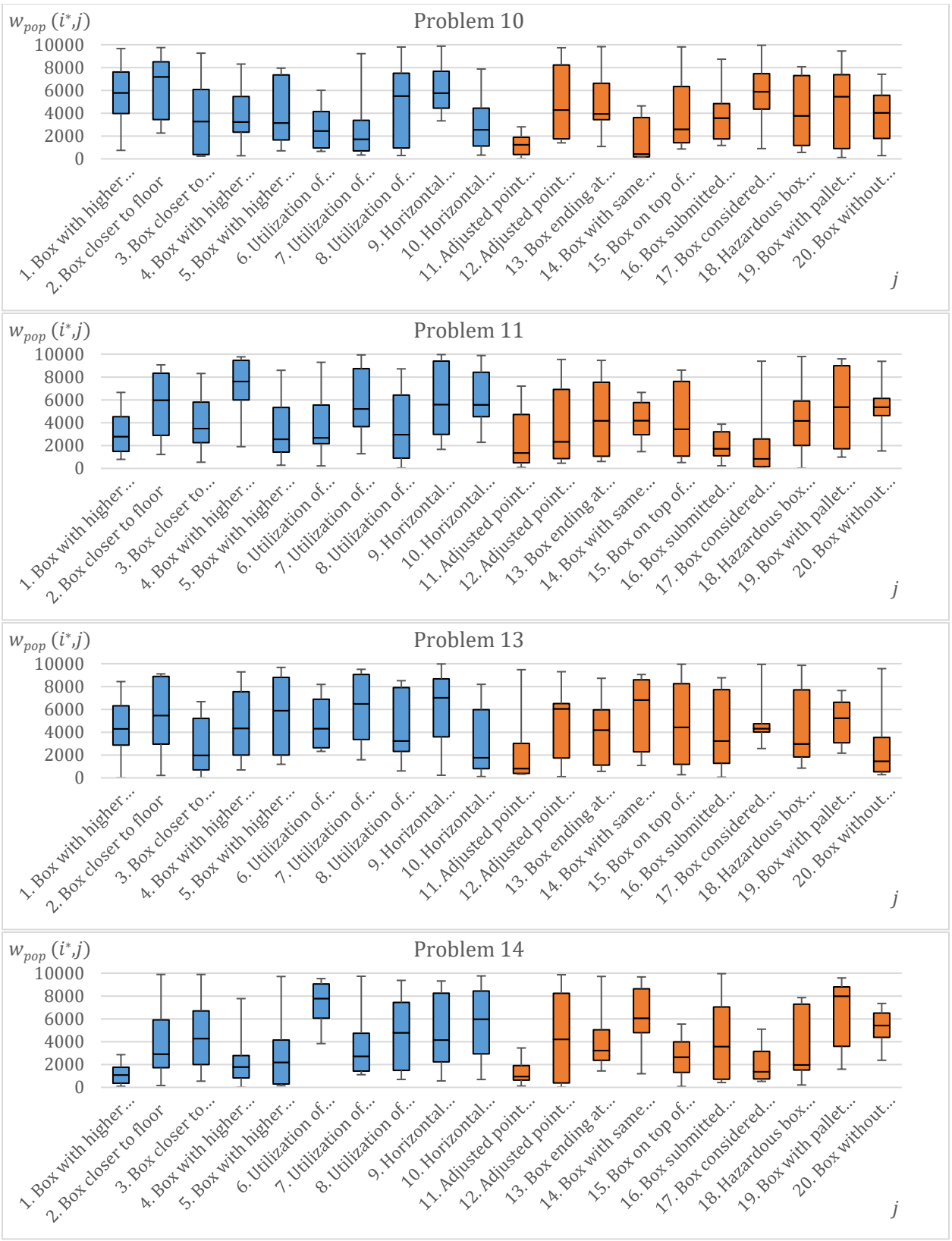

Figure 76. Coefficient values for the best solution in the last generation for the 10 runs of Problems 10, 11, 13 and 14

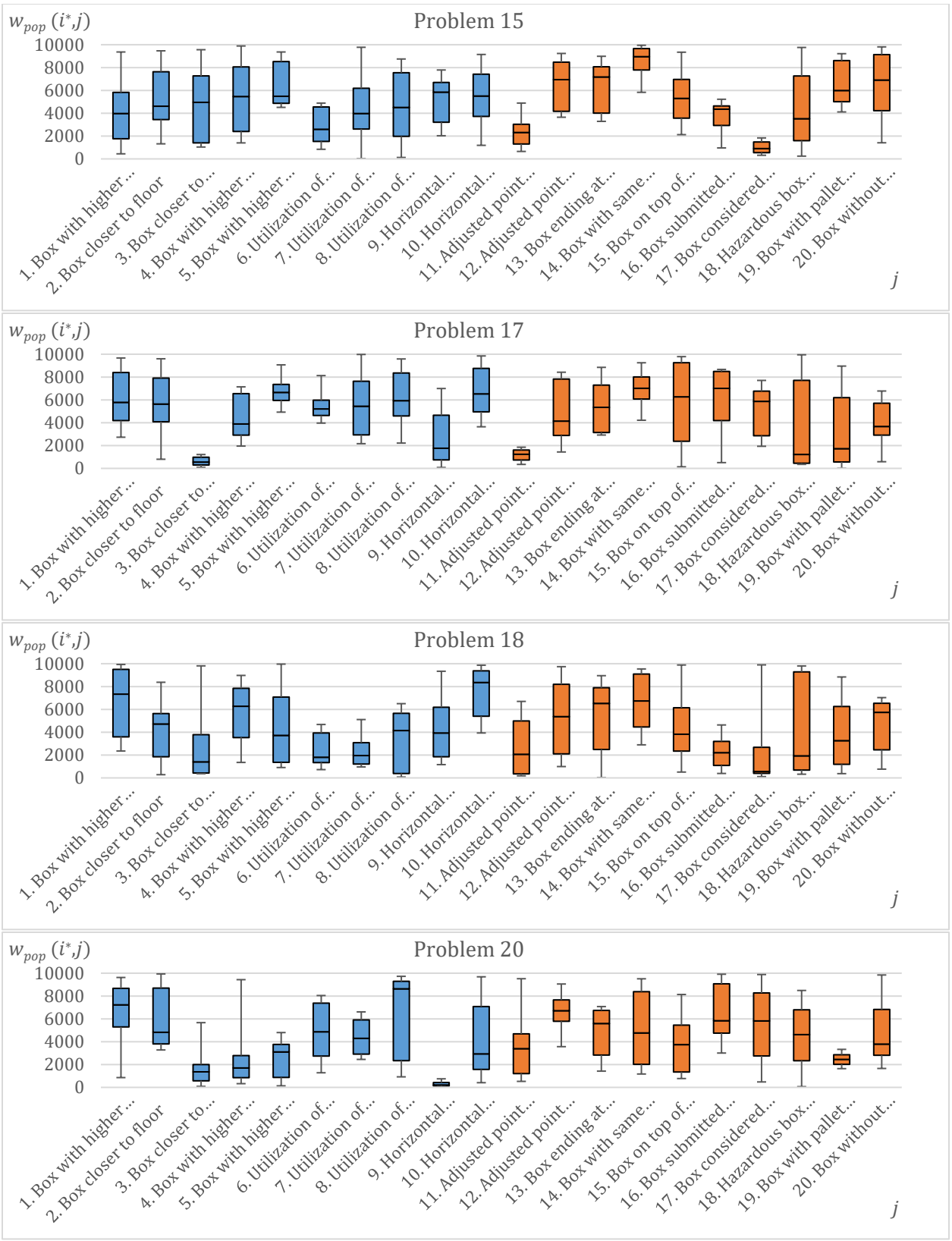

Figure 77. Coefficient values for the best solution in the last generation for the 10 runs of Problems 15, 17, 18 and 20