The Quasi-Online algorithm in a Robot Packing environment

Implementation of an improved Bin Packing algorithm

A. J. Haasdijk

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control and Department of Cognitive Robotics

The Quasi-Online algorithm in a Robot Packing environment

Implementation of an improved Bin Packing algorithm

Master of Science Thesis

For the degree of Master of Science in Systems and Control and Mechanical Engineering at Delft University of Technology

A. J. Haasdijk
4369173

Supervisors:
dr. ir. S. Grammatico
dr. ir. J. Kober
ir. T. H. R. Biemans
ir. F. B. Gorte

April 6, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The work in this thesis was supported by Picnic. Their cooperation is hereby gratefully acknowledged.

Abstract

Order picking is a central process in the supply chain of online retailers. The efficiency of the order picking process determines the capacity of an automated warehouse, and is therefore a key element in the operation thereof. To decrease the costs and the labour-intensity, and to increase the picking efficiency, a robotic order picker can be implemented with a corresponding packing algorithm. The aim of this research is to obtain a packing algorithm that is able to operate in an automated warehouse. The challenges of such a packing algorithm are threefold. The number of items packed in one bin needs to be as high as possible within reasonable time, grocery items are to be considered and the packing algorithm needs to be compatible with the environment it operates in. These challenges account for various constraints. The irregularity and the fragility of the items need to be considered, as well as their stability. The sequence of item arrival at the picking station is unknown.

The Quasi-Online algorithm by Wang and Hauser [2020] is used as a baseline for the final packing algorithm. This algorithm considers irregular items as well as a non-deterministic item sequence. First, the Quasi-Online algorithm by Wang and Hauser [2020] is recreated. To be able to decrease the computation time, the irregular shape of the items is obtained using voxelization. To reduce the instability of the item due to the voxelization process, a stable initial position of the item is obtained by using Principal Component Analysis. Furthermore, the stability measure is replaced to make sure only stable configurations of the items are considered. In the environment, the system can be used to look at least one item ahead, therefore information of one item and of two items ahead are considered. The last improvement entails the inclusion of the fragility of the item. The picking sequence is a constraint on the order of the items that can be picked without damaging the items. From simulations is shown that the Quasi-Online algorithm by Wang and Hauser [2020] can be improved by the methods described.

Contents

List of Figures

List of Tables

Acknowledgements

My journey at Picnic started with an internship in inbound. Near the end of my internship in July 2021, I had a very interesting meeting with Frank Gorte about potential thesis topics. A topic that I was immediately enthusiastic about was robot picking, especially the area of Bin Packing. The area of Bin Packing is one interesting for both Picnic, as well as the scientific world, especially regaring three-dimensional Bin Packing. The writing of this preface means finishing my thesis about this topic, which started nine months ago.

Sergio Grammatico has helped me choosing the specific thesis topic, reading and evaluating the proposals I made after which Jens Kober joined the group of supervisors. In spite of the online meetings, I am very thankful for the critical feedback and outside the box ideas. In turn, of course I want to express gratitude to professors Jamshidnejad and Ferranti for taking the time to assess my work.

Starting this thesis I was welcomed into a new team within Picnic, the automation team. Tim Biemans was to become my daily supervisor. The weekly meetings where we discussed new ideas, feedback but also personal development were very valuable to me. I am glad you could guide and help me making this project happen, from the initial brainstorm to proof-reading my final draft.

I want to thank everyone involved at Picnic. Many thanks to Tad Slaff, helping me with the implementation of the algorithm but also providing me with valuable insights concerning the application in the Picnic warehouse. To Frank, thanks for the many talks about voxels and other robot picking discussions as well as taking me to a company where robot picking is already implemented in their supply chain. I also want to thanks the rest of the automation team for all the feedback and coffee breaks during this period.

Last but not least, I want to thank my family and friends for the support during this project. Especially I want to thank Wil Haasdijk and Coen Slagers for reviewing my thesis and being available for conversations about the topic.

Completing this process has taught me a lot about professional knowledge, as well as personal development. I am looking forward to applying this in the future, where I hope to further develop these skills.

I wish you a good time reading,
Annemarijn Julia Haasdijk

Delft, University of Technology A. J. Haasdijk
April 6, 2022

Chapter 1

Introduction

The online grocery industry is rapidly expanding. To fulfill the high demand of the customers, companies need to collect and transport the orders more efficiently. It is observed that companies focus on increasing the rate of automation within their warehouses to reach the required capacity. Automated systems are able to work around the clock and therefore create a larger throughput. Furthermore, with increased automation, the labour costs are reduced [de Koster, 2018]. Processes automated within a warehouse are order picking, item storage, and put-away of items [Boysen et al., 2019]. Automated warehouses often consider a Goods-to-Person (GTP) system, where the items are transported from a storage system to the picking station with a human operator.

The orders can subsequently be picked either by a human operator or an automated order picking system. Order picking is a central process in the supply chain of online retailers and is the process of collecting articles obtained from the storage according to a customer order. The process is the most labour-intensive operation in the warehouse and is a key process for online retailers. With more efficient order picking, more articles can be picked into one bin which reduces the number of bins used and consequently reduces the transportation costs. For these reasons, automation of the order picking process will have a big impact on warehouse operations [de Koster et al., 2007]. Therefore, this thesis is focused on the application of automated order picking in the environment of an automated warehouse.

In Section 1-1, the environment of this project is discussed. From this environment, the challenges of the automated order picking are stated and summarized in Section 1-2. Finally, in Section 1-3 the problem and research question are stated.

1-1 Description of the environment

In this section, the environment where the project takes place in is discussed. This research was done in collaboration with Picnic, an app-only e-grocer. A basic understanding of the operations of Picnic is explained in Section 1-1-1. In Section 1-1-2, the warehouses that Picnic uses are explained. The specific environment and the constraints it poses on the project are discussed in Section 1-1-3.

1-1-1 Introduction to Picnic

The project environment is the online grocery retail market, within the online supermarket Picnic. As opposed to traditional brick and mortar supermarkets, Picnic is an app-only supermarket. Customers order their groceries online for a chosen time slot and Picnic delivers the groceries at their home address. Picnic uses different types of warehouses to store the products and collect the orders. The warehouses are a central part of Picnic's supply chain. Two types of warehouses currently exist. The next section elaborates on these types.

The orders are collected in special crates, called bins. From the warehouses, the products are transported with trucks to smaller locations. From these locations, called hubs, electronic vehicles transport the products to the customers.

In the vehicle, two frames are placed each containing 36 bins. There is little space between the frames and the top of the bins, therefore no items can be protruding from the bins. Furthermore, to achieve the most optimal supply chain the least amount of bins are used to transport the products. In order words, the volume utilization of the bin needs to be maximized. This reduces the transportation costs and the time required for unloading the vehicles. However, the packing itself needs to be done efficiently as well. Therefore the time required to pack the items in the bins is restricted.

1-1-2 Warehouses

There are two types of warehouses currently in use with Picnic, manual warehouses and automated warehouses. In the manual warehouses, the orders are collected by walking past supermarket-like shelves. The system where the (human) pickers walk past every article, pick the correct article from the order and put them in the customer bins is called a Person-To-Goods system. In the automated warehouse, a Goods-to-Person system is implemented. This project is situated an automated warehouse. For this reason, this section elaborates on automated warehousing. In the first section, the principles of the automated warehouse are stated. In the next sections, two important systems of the automated warehouse concerning this project will be explained.

Automated warehouse Automated warehouses are a new type of warehousing. Recently, the automated warehouse from Picnic has been put in operation. The automated warehouse is designed to serve 150.000 customers a week. Picnic's automated warehouse consists of both manual and automated processes. After the items arrive at the warehouse on pallets, the items are put into stock bins. A stock bin is filled with one type of items, for example only milk. From the decanting station, the stock bins are transported to the automated storage system using conveyor belts. When an order contains items from a specific stock bin, this bin is transported again with conveyor belts to the picking station. At the picking station, items from one order are transported from multiple stock bins to one customer bin. The customer bin is filled with the groceries of one customer.

The automated storage system, which is further explained in the next section, is a critical system in the automated warehouse. This system contains all stock available for picking in the automated warehouse and closely interacts with the picking station.

Automated Storage and Retrieval System The stock bins are stored in an Automated Storage and Retrieval System (ASRS). This system consists of racks with shuttles that are running through the aisles to retrieve the stock bins from the racks, as shown in Figure 1-1. The storage assignment of the stock bins is done randomly but can be optimized in a later stage [Roodbergen and Vis, 2009]. The route from the storage to the picking stations differs with each iteration and waiting times cannot be defined in advance. Therefore, the time it takes for the stock bin to arrive at the picking stations is not known exactly, which results in the unknown sequence of item arrival at the picking stations.

Picking station The picking station is the place where the items are picked from the stock bins and placed in the customer bins. Currently, a human picker is used to pick and place the items in automated warehouses, as shown in Figure 1-2. However, hereinafter a robotic picker will be assumed. The replacement of the picker poses the constraint that the location at which the item needs to be placed is explicitly known, as this is an input to the robot manipulator. With the arrival of the stock bin at the picking station, the items from the stock bin need to be placed directly in the customer bin.

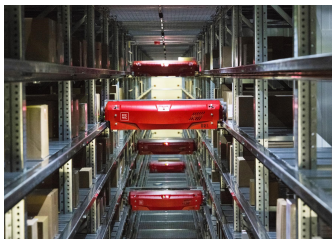


Figure 1-1: ASRS [Emerce, 2019]

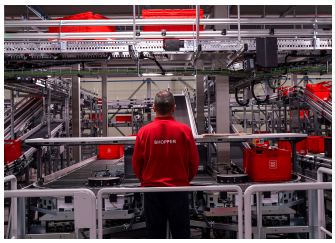


Figure 1-2: Picking station [Emerce, 2021]

1-1-3 Robot picking environment

As discussed previously, the automated warehouse consists of an automated storage system and a picking station. At the picking station, a robotic picker can be installed. The system of the automated warehouse poses constraints on the robot picking algorithm as was discussed in the previous section. In this section, additional characteristics of the robot picking system are discussed.

Stability requirement The customer bin is filled with grocery items, all comprise of different shapes. The articles are not to be damaged in the placement process, as this is a guarantee from Picnic to the customer. Therefore, the stability of the items within the bin is important. For example, when a carton of eggs is placed on top of other items resulting in an unstable pile, the eggs are likely to be damaged. As a result, the robot picking process needs to be focussed on the stability of the items.

Picking sequence An important characteristic taken into account for the order retrieval is the sequence in which the products are put into the customer bin. In order not to damage articles by putting a heavy article on a fragile one, a picking sequence is required. Within Picnic the picking sequence is determined based on the weight, volume, fragility, and contamination. Each article from the assortment is given a value between 1 and 5. In this sequence, 1 needs to be on the bottom of the customer bin and is thus picked first. 5 is placed on top. The consecutive numbers can be interchanged, but the next-to consecutive numbers can not. The articles with the same picking sequence value can interchanged as well. For an example, see Figure 1-3. This figure shows five items with different picking sequences. The items with consecutive picking sequence are interchangeable, indicated with a green mark. The items with next-to consecutive picking sequences are not interchangeable and indicated with a red cross.

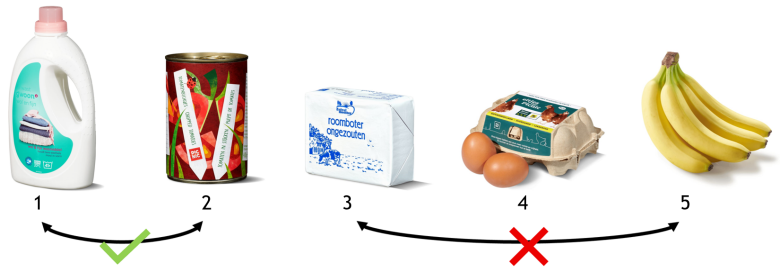


Figure 1-3: Items with picking sequence 1, 2, 3, 4, 5, respectively (left to right)

In this section, the connection between the automated warehouse and the robot picking environment was discussed. This connection results in constraints for the robot picking task. First of all, the sequence of the arriving items is not completely known. Secondly, there is no buffer space considered at the picking station resulting in direct placement

of the items. Furthermore, irregular items are placed in a bin one at the time at the picking location. The stability of the items needs to be ensured. The fragility of items is considered and no items can protrude from the bin. Besides, the bins need to be filled with the highest volume utilization achievable within reasonable time.

1-2 Challenges in automated robot picking

In the previous section, the environment of a robot-picking system is explained. This environment gives rise to various challenges. These challenges can be divided into objectives regarding the supply chain, the grocery items and the system in which the robot-picking exerts. The challenges are summarized below.

1. To account for the most efficient supply chain, the customer bins need to be filled with the most items as possible. Therefore, a goal is to reach a high volume utilization. For the same reason, the computational time to fill one bin needs to be minimized.
2. To be able to place grocery items, a couple of restrictions are considered. The articles used within Picnic have a variety of shapes, therefore the system needs to be able to handle irregular shaped items as well as the stability thereof. Furthermore, the articles have different fragility which are to be considered regarding their placement.
3. The characteristics of the system in which the robot manipulator is operating need to be respected. These entail the direct placement of the items as well as the unknown item sequence at the arrival of the items.

To account for a robot-packable system, which is a system where the item placement is defined, these challanges need to be overcome. A small number of researches focus on the application of a robot packable system in an automated warehouse with grocery items. The researches that focus on this problem result in a low utilisation rate or do not consider all the challenges as described above. One research that stands out is the research from Wang and Hauser [2020]. In this research, the majority of the constraints mentioned above are considered. However, the picking sequence is not mentioned and the algorithm is not tested on volume utilization but on success rate. Success rate is defined as the rate to place all N items within a bin. Furthermore, the success rate is low, and the computation time increases rapidly when a higher number of articles is placed.

1-3 Problem statement

In the robot picking environment, groceries are picked from stock bins and placed in customer bins. The aim is to use the smallest amount of customer bins to place all

groceries in. The environment puts constraints on the placement of grocery items, the supply chain and the system the project is operating in.

Various researches have been conducted to improve the volume utilization in the bins. However, the challenges as described in the previous sections are not or only partly respected. The research from Wang and Hauser [2020] considers the majority of constraints, but does not report the volume utilization. Therefore it is not known whether this algorithm might be practically useful in a robot packing environment.

For this thesis, the algorithm from Wang and Hauser [2020] is compared to existing benchmarks and the algorithm is improved. Therefore, the research question for the thesis is:

How can the algorithm by Wang and Hauser [2020] be used to provide an efficient solution to the three-dimensional Bin Packing Problem in a robot-packing environment?

In Chapter 2, the general Bin Packing problem is discussed as well as the methods used for solving this problem to date. Additionally, the remarks on the Quasi-Online algorithm are stated. Chapter 3 describes the improvements of the algorithm by Wang and Hauser [2020] that are proposed, resulting from the remarks discussed. In this chapter, the theory as well as preliminary results are shown of the different improvements. In Chapter 4 the different improvements are added and the algorithm is tested against a commonly used benchmark. Chapter 5 concludes on the results of the improved Quasi-Online algorithm and describes recommendations regarding future research.

Chapter 2

Bin Packing in Automated Warehouses: State of the art

This section elaborates on the Bin Packing problem. The fundamental problem is stated in Section 2-1. Before explaining and comparing the algorithms that are used to solve the Bin Packing Problems, first the main characteristics are described in Section 2-2. Subsequently, in Section 2-3 the most promising algorithms are explained and these are compared in Section 2-4. In Section 2-5, the outcomes of the evaluation of the promising algorithms are discussed. Finally, Section 2-6 contains remarks on the Quasi-Online algorithms.

2-1 The Bin Packing Problem

The problem of packing small objects into large boxes while minimizing the number of boxes used belongs to the group of Cutting and Packing problems. Wäscher et al. [2007] developed a classification scheme to identify the different Cutting and Packing problems using five criteria *Kind of assignment*, *Assortment of small objects*, *Assortment of large objects*, *Dimensionality* and *Shape of small objects*. *Kind of assignment* refers to either a minimization problem or a maximization problem. *Assortment of small objects* indicates the diversity of the shapes of items to be packed and *Assortment of large objects* the diversity of shapes of the bins the items are packed in. *Dimensionality* refers to the space the packing problem operates in and *Shape of small objects* determines which shapes are used for the items. According to this classification, the Bin Packing Problem (BPP) is a problem that minimizes the number of rectangular bins used while packing strongly heterogeneous items in bins of similar sizes. The characteristics of the BPP *Dimensionality* and *Shape of small objects* are further discussed in Section 2-2.

The classical BPP can be written as an Integer Linear Program, shown in Equation (2-2) [Delorme et al., 2016].

$$\begin{aligned} y_i &= \begin{cases} 1 & \text{if bin } i \text{ is used in the solution;} \\ 0 & \text{otherwise} \end{cases} \\ &\qquad (i = 1, ..., u) \\ x_{ij} &= \begin{cases} 1 & \text{if item } j \text{ is packed into bin } i; \\ 0 & \text{otherwise} \end{cases} \\ &\qquad (i = 1, ..., u; j = 1, ..., n), \end{aligned} \tag{2-1}$$

$$\begin{aligned} \underset{i,j}{\text{minimize}} \quad & \sum_{i=1}^{u} y_i && && \text{(2-2a)} \\ \text{subject to} \quad & \sum_{j=1}^{n} w_j x_{ij} \leq c y_i, && (i = 1, ..., u), && \text{(2-2b)} \\ & \sum_{i=1}^{u} x_{ij} = 1, && (j = 1, ..., n), && \text{(2-2c)} \\ & y_i \in \{0, 1\}, && (i = 1, ..., u), && \text{(2-2d)} \\ & x_{ij} \in \{0, 1\}, && (i = 1, ..., u; j = 1, ..., n) && \text{(2-2e)} \end{aligned}$$

As shown in Equation (2-1), y_i is the ith bin used and x_{ij} the item j packed into the ith bin. As discussed, the BPP is a minimization of the number of bins which is reflected in the objective function in Equation (2-2a). Constraint (2-2b) makes sure that the capacity of the bin is not exceeded. The capacity in the bin is defined by w_j, which is the volume of the items in the classical BPP. c is the maximum capacity the bin. Constraint (2-2c) imposes that all the items are packed in a bin exactly once. Constraint (2-2d) shows that a bin is used (1) or not used (0) in the solution. The same accounts for Constraint (2-2e): for each item j the item is either packed into bin i (1) or not (0).

The fundamental optimization problem that is described above is the base for the complete optimization problem. Extensions to the fundamental problem, such as additional constraints, are discussed in the following chapter.

2-2 Characteristics of the Bin Packing Problem

The domain of Bin Packing algorithms is substantial. Therefore it is important to categorize the differences found in BPPs. This section elaborates on four important characteristics: dimensionality, type, shapes, and stability used in BPPs.

2-2-1 Dimensionality of the algorithm

Within the BPPs, three dimensions ought to be considered. A BPP can be either one-, two- or three-dimensional. Often, in a one-dimensional problem, the dimension is the (normalized) volume or size of the item. A two-dimensional problem operates, corresponding to its name, in two dimensions: the width and the length. Three-dimensional problems use an additional dimension which is the depth. For automated order picking, the precise location of the item to be placed needs to be completely determined. Therefore, the Bin Packing algorithm in a robot picking environment should be a three-dimensional algorithm [Garey and Johnson, 1981, Lodi et al., 2002, Martello et al., 2000].

2-2-2 Type of the algorithm

Another categorization of the BPP is concerning the type of Bin Packing model. There are three model types to be considered: offline, online, and a combination between online and offline which is called quasi-online. The main difference between these three types of algorithms is the knowledge of the sequence of the items, which has a direct link with the environment the algorithms can operate in.

Offline algorithms In offline algorithms, all properties of the items that are to be placed are known beforehand, such as the sizes and volumes. Characteristics are the dimensions as well as the sequence in which the items need to be loaded in the bin. In numerous offline Bin Packing algorithms, the sequence of arrival of the items is changed to optimize the volume utilization of the items in the bin [Coffman et al., 2013].

Online algorithms On the contrary, in online algorithms, the dimensions and arrival sequences of the items are completely unknown before they arrive at the pcking location. The *worst-case performance ratio* defines the number of bins used in the heuristic relative to the number of bins used by an optimal algorithm. Three online algorithms and their corresponding offline algorithms are shown in Table 2-1. Based on this table the conclusion can be drawn that the offline algorithms use fewer bins and perform therefore better compared to the online algorithms. When all the item properties are determined beforehand, the algorithm can generate a (near-)optimal placement of the item. For the online algorithms, this is not the case.

A supplementary feature of the online algorithm is the allowance of certain operations, such as repacking, lookahead and preordering. These features are called semi-online algorithms. Repacking signifies the operation of being able to reorganize the items that are already packed in the bin. Repacking allows for a higher performance since an optimal location is pursued despite the non-deterministic sequence. The operation lookahead uncovers characteristics of the N number of items behind the first item at the picking location, resulting in a semi-deterministic sequence. Finally, preordering allows

the sorting of the items before the arrival at the packing location. For instance, the items might be ordered on the volume [Balogh et al., 2021] or the sequence might be constrained by the fragility of the items.

Table 2-1: Worst-case performance ratio (r) of three corresponding online and three offline algorithms [Baker and Coffman, 1981, Christensen et al., 2017]

Online algorithm	r	Offline algorithm	r
Next-Fit	2	Next-Fit-Decreasing	1.7
First-Fit	1.7	First-Fit-Decreasing	1.2
Best-Fit	1.7	Best-Fit-Decreasing	1.2

Quasi-online algorithms The third type of Bin Packing model, called quasi-online Bin Packing, combines the previous two methods. This type of dynamic model is initially described by Wang and Hauser [2020] and to the author's knowledge, this is the only research about the quasi-online algorithms. The algorithm by Wang and Hauser [2020] is further explained in Section 2-3.

This thesis is focused on quasi-online algorithms. The offline algorithm by itself is not suitable for solving the problem as a consequence of the environment as described in Section 1-1. The sequence is not entirely known beforehand causing the offline algorithm to fail. The online algorithm on the other side is compatible with the environment. Due to the inferior performance of this type of algorithms, a combination between the offline and the online algorithm is preferred.

2-2-3 Shape of the item

The shapes of the items are arranged into two categories: regular items and irregular items.

Regular items The class of regular items is defined by items that are geometrically simple to describe. This class consists of, among other things, spheres, cuboids, cylinders, and parallelepipeds. Examples of regular items described with point clouds are shown in Figure 2-1.

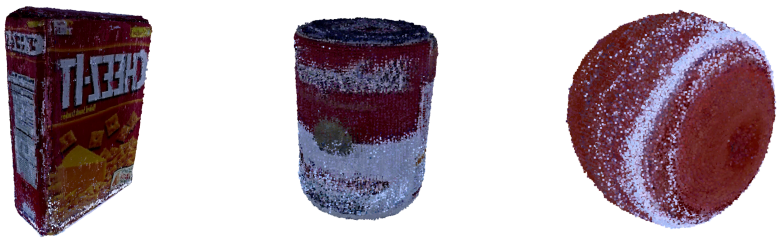


Figure 2-1: Examples of regular items

Irregular items The class of irregular items are defined by all varieties of shapes that are not contained in the regular item class. This class is very broad. In Figure 2-2 several shapes belonging to the class of irregular items are shown.

Figure 2-2: Examples of irregular items

In this thesis, both regular and irregular items are considered. As seen in Figures 2-1 and 2-2, grocery articles are part of both categories. For simplicity and to decrease computation time, various algorithms consider rectangular boxes to determine the shape of the items. However, in doing so the volume utilization of the bin is over-estimated and the number of bins used to pack items is increased compared to using irregular items. In selecting the appropriate shape determination, a trade-off is made between the computation time and volume utilization.

2-2-4 Static stability of the pile

To sustain a realistic packing of items, the stability of the pile is crucial. A distinction is made between the stability of rectangular items and other items.

Stability of rectangular items To establish the stability of rectangular items, numerous options are feasible. There are two groups of support, *partial* and *full* support. An item is *partial* supported when a fraction of the base area is supported. The base area is the area between the to-be-placed item and a readily placed item or the floor of the bin. *Full* support indicates that the base area of the item is completely supported. To which extent the base area is supported for the item to be stable, the opinions in the literature are divided. Tanaka et al. [2020] declares that the item is stable when 50 percent of the base area is supported. Eley [2002] assumes stability when all four corners are supported, as well as a percentage of the base area of each item. The percentage which suffices a stable configuration however is not given. In the research conducted by, among others, Bortfeldt and Gehring [2001], Eley [2002], Gonçalves and Resende [2012] a fully supported base area is deployed.

Stability of other items To assess the stability of items shaped other than rectangular, including irregular items, cylinders, or spheres, another approach should be considered.

The notion of partial or full support does not suffice and is more difficult to obtain. A key parameter in the evaluation of other shaped items is the notion of contact points. Contact points are defined as the points where the item is in contact with another item or the ground floor. Acquiring contact points is crucial to assess the stability of items that are not rectangular-shaped. Sujan and Dubowsky [2000] represent the stability of the item by the number of contact points. The location with the highest number of contact points has the highest value in the cost function and therefore this placement is more likely to be chosen in the final configuration. However, it does not determine whether the placement is unstable. It merely describes a probability of stability as there is not a certain number of contact points indicating stability.

Thangavelu et al. [2018] focus on the stacking of stones. They use a simulator to define whether each of the configurations is stable. The generator achieves the definition of stability by applying all forces acting on the stones, resulting in equilibrium equations. The packing algorithm obtained by Wang and Hauser [2019] identifies stability using a force and torque balance. In these static balances, the friction forces and contact forces that act upon each of the items are considered as well as the gravitational forces. However, using the equilibrium equations, not all stability issues are solved. Wang and Hauser [2019] discuss that in at least 20 % of the cases, the stability measure is not maintained. In these cases, the object does not stay put on the chosen location and might obstruct another object in the placement process.

2-3 Algorithms used for three-dimensional Bin Packing

To determine a packing plan for the items in the bin, numerous methods can be used. In this section, the algorithms are listed that are successfully applied to the three-dimensional Bin Packing problem and provide sufficiently good results. The Bin Packing problem is NP-hard, therefore obtaining an exact solution is infeasible within reasonable time. Consequently, the algorithms discussed in this section are approximation algorithms. As concluded in Section 2-2-2, a quasi online algorithm, which is a combination of an offline and an online algorithm, is preferred. For that reason, offline, online, and quasi-online algorithms are considered. In the first section, the heuristics are discussed. Heuristics are approximate algorithms that return acceptable solutions within reasonable computation time and can handle additional constraints [Faroe et al., 2003]. Subsequently, the meta-heuristics, which are heuristics that can solve a general class of problems, and Reinforcement Learning (RL) approaches are assessed. Reinforcement Learning is a learning-based approach that tries to achieve a maximum reward. Finally, the quasi-online algorithm is considered, which is explained briefly in Section 2-2-2 and more extensively in this section.

2-3-1 Heuristics

In this section, the heuristics applicable to the BPP that excel in volume utilization and computation time are considered. Heuristics are approximation algorithms based on a rule where constraints can be easily implemented. The heuristics Deepest Bottom Left Fill (DBLF), Three-dimensional Best Fit (3BF), Distance to the Front-Top-Right Corner (DFTRC), Extreme Points (EP) and Heightmap Minimization (HM) are examined as these are shown in literature to provide good results regarding the BPP or are used in combination with meta-heuristics discussed in Section 2-3-2.

DBLF As the name implies, the rule that is employed in the Deepest Bottom Left Fill heuristic is item placement on the location that is the deepest, lowest, and leftmost one. The deepest location is the placement against the backside of the bin. The algorithm tries to fill the deepest available gap or is filling as much of the gap as possible. The algorithm decides the location at item arrival considering the already assigned items, therefore belongs to the type of online algorithms. To obtain the deepest lowest location, all possible locations need to be assessed. Therefore the method is cumbersome when using a large bin for item placement [Karabulut and İnceoğlu, 2004]. Furthermore, the use of irregular items as well as the stability determination has not been applied.

3BF The Three-dimensional Best Fit algorithm receives its main characteristics from the well studied one-dimensional Best Fit (BF) algorithm. Before an item is placed, the Best Fit algorithm calculates the remaining capacity, after the object is added, of the different bins. The item is placed in the bin with the smallest remaining space. The Three-dimensional Best Fit algorithm operates similarly. Instead of selecting the appropriate bin for the item, in the Three-dimensional Best Fit algorithm, the location with the smallest gap is chosen for item placement. If multiple locations account for the smallest gap of an item, the algorithm falls back on additional rules such as the Deepest Bottom Left Fill as described before [Allen et al., 2011].

DFTRC The Distance to the Front-Top-Right Corner heuristic is developed by Gonçalves and Resende [2013] after the observation has been made that some optimal solutions could not be obtained by the Back-Bottom-Left heuristic (similar to the Deepest Bottom Left Fill heuristic). The starting point of the Distance to the Front-Top-Right Corner is are the spaces in the bins called the Empty Maximal Spaces (EMSs). These spaces represent the largest empty rectangular spaces in the bin and are represented by their minimum and maximum coordinates. For all EMSs, the distance to the front-top-right corner is calculated hence the name of the heuristic. The EMS is chosen that maximizes this distance [Gonçalves and Resende, 2013].

EP In the Extreme Points heuristic, the extreme points are used to determine the possible item locations. Analogous to the front-top-right corner points which are explained

before, the extreme points are the locations where the item can be placed. The points are determined at the location where the outline changes from vertical to horizontal and are often located at the corner points of the previously placed items [Crainic et al., 2008]. The extreme points are ordered in non-decreasing order of z, y, and x and the next item is placed. Subsequently, new extreme points are obtained and added to the list of extreme points. A disadvantage of the EP algorithm is that it is not compatible directly with irregular items. Irregular items often do not have a clear vertical and horizontal dimension. Therefore, the EP heuristic can not directly be used in an irregular Bin Packing environment. Furthermore, Crainic et al. [2008] use a non-bounded space where the items are placed in. A *1-bounded-space* algorithm, as opposed to a non-bounded algorithm, is an algorithm in which the bin is closed when the item cannot be placed in the bin. In the project environment, no buffer space is present, therefore the algorithm has to be a bounded space algorithm.

HM Comparable to the DBLF heuristic, the Heightmap Minimization heuristic searches through the grid to assess every possible location in combination with the orientation of the item. To be able to generate a packing for all items in a sequence, a score is obtained to evaluate every location-orientation combination for each of the items. A low score indicates that an item is placed close to the origin (0,0) point of the bin while maintaining a low value in the heightmap. After the score is determined for each of the placement locations and the N lowest scores are chosen for further analysis on the location, a stability check is performed. This method for obtaining the score is specifically of interest to the quasi-online algorithm due to the heightmap minimization characteristic of the heuristic, as will be explained in Section 2-3-4. The score is determined according to Equation (2-3).

$$c \cdot (X + Y) + \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} H'_c[i, j] \tag{2-3}$$

In this equation, c is a weight, X and Y are the coordinates of the placed item, and H'_c is the heightmap of the placed item with w and h the dimensions of the heightmap.

2-3-2 Meta-heuristics

Different compared to heuristics, meta-heuristics are algorithms that do not search the full search space. Consequently, meta-heuristics tend to be faster compared to heuristics. In this section, the best meta-heuristics regarding the BPP from literature research are discussed. These algorithms contain algorithms based on Genetic Algorithm (GA), Simulated Annealing (SA) and Tabu Search (TS). The results of the benchmark on these algorithms are shown in Section 2-4-2

Genetic Algorithm The GA is a meta-heuristic derived from the Darwinian evolution theory including selection, mutation, and crossover. From randomly generated solutions, the population is grown by pairing individual solutions and using crossover, and adding mutations. The individual solutions are encoded, for example by using binary strings. A selection operator determines which parents are allowed to reproduce. From two parent solutions, the offspring is generated using for example one-point crossover. One point on both solutions is chosen and the parts after are interchanged. Mutations are added by switching one of the binary values using a probability defined beforehand [Chambers, 2019]. Two researches using a GA for the BPP resulting in high volume utilization are obtained by Moon and Nguyen [2014] and Gonçalves and Resende [2013].

Moon and Nguyen [2014] propose a GA combined with the DBLF heuristic as described in Section 2-3-1. The algorithm, called Hybrid Genetic Algorithm (HGA) focuses on achieving the highest possible volume utilization through rearranging the input sequence of the items. Furthermore, stability and rectangular-shaped items are considered. In the GA by Moon and Nguyen [2014], both the sequence of the items and the rotation of the items are added to the parent solutions. From these characteristics, the offspring is generated. Different compared to the BPP considered in this thesis, the algorithm described by Moon and Nguyen [2014] can change the sequence of items and is, therefore, an offline algorithm. As the solution is based on a container loading problem, balance constraints of the bin are included. Another implication of the slightly different problem compared to the BPP is that in transportation the gaps between items in the container are filled with foam. This disregards the stability constraint and therefore this constraint is ignored. In the current project environment, no foam pieces are added and therefore the stability constraint needs to be added.

Similar to Moon and Nguyen [2014], Gonçalves and Resende [2013] use rectangular item packing in combination with a heuristic and a GA to obtain the highest volume utilization by rearranging the sequence and rotating the item. The GA is used to generate an optimal sequence and orientation of the items. Other than the Genetic Algorithm used by Moon and Nguyen [2014], Gonçalves and Resende [2013] use a Biased Random Key Genetic Algorithm (BRKGA), which uses another method how the offspring is generated. BRKGA emphasizes the solutions with high fitness, resulting in a higher chance of breeding for these individuals. The method by Gonçalves and Resende [2013] does not include stability constraints. Furthermore, the space in which the items are placed is non-bounded, similar to the EP algorithm discussed in the previous section. A non-bounded algorithm is not compatible with the project environment.

Simulated Annealing Simulated Annealing is a nature-based algorithm that uses a hill-climbing method to generate solutions. Starting with an initial solution, a hill-climbing method searches neighbors that are close to the initial solution. If the fitness of the neighbor solution is higher, the initial solution is replaced by the new solution [Glover and Laguna, 1998]. SA mimics the annealing of metal using steps parallel to annealing when a solid is heated and cooled. In metal annealing, metal is heated until the structure

melts. Thereafter in the liquid phase, the particles are distributed randomly. At the cooling phase, a certain temperature scheme is used to reach the minimum energy state of the solid [Delahaye et al., 2019]. The degree of cooling determines whether a stable local equilibrium or a crystalline structure is achieved.

At the start of the SA algorithm, a random state is chosen. When a second neighboring state is generated, this state is accepted based on an acceptance probability. With a constant low temperature, the algorithm might get stuck in a local minimum whereas with a constant high temperature the convergence of the algorithm is questioned. The temperature is gradually decreased to avoid being prematurely trapped in a solution [Eglese, 1990].

Regarding the BPP, Peng et al. [2009] developed a Hybrid Simulated Annealing (HSA) algorithm. Comparable with the HGA, the HSA algorithm combines a heuristic with SA. In the heuristic, the orientation and the fully supported stability constraints are implemented. The heuristic used is similar to the EP heuristic. The neighboring solution is applied by incrementing the solution with one position. To decrease the computation time of the algorithm, parallelization is added. With parallelization, the algorithm runs different initial states at the same time and compares the solutions thereafter. A main feature in the HSA algorithm is that it uses local arrangement to define the packing of items. In each phase of the algorithm, the best local arrangement is chosen. A local arrangement is a structure of multiple items of the same shape and same orientation that forms a rectangle, in such way that there is no unused space between the items. However, in the case of irregular items, such a local arrangement is nearly impossible to obtain.

Tabu Search Tabu Search is similar to SA a hill-climbing solution and as a consequence generates new solutions by assessing the fitness of neighboring solutions. Different compared to the meta-heuristics described thus far, TS is a deterministic method and therefore is not based on probability and randomness [Glover and Laguna, 1998]. The TS starts with an initial solution obtained by for instance a heuristic. From this initial solution, neighboring solutions are generated by using a neighboring function. When a neighboring solution is visited, it is added to a *Tabu list* which tracks the states that can not be visited in the near future. To be able to arrive at the global minimum, exceptions are made to visit the states from the Tabu list after a certain amount of iterations, called the *tabu tenure*.

Bortfeldt et al. [2003] use the Tabu Search method to obtain a Parallel Tabu Search (PTS) algorithm. Just like the HSA algorithm by Peng et al. [2009], parallelization is added in the algorithm, which is used to reduce the computation time. A greedy heuristic comparable to the 3BF heuristic is used for the initial state. In this heuristic, the items are ordered on volume, and the location where the item fits the gap is chosen. The different solutions are obtained by rearranging the item sequence or rotating the

item, similar to the HGA, BRKGA and HSA meta-heuristics. The PTS algorithm makes use of local arrangements to define the packing space. Furthermore, the PTS defines a packing sequence beforehand while this packing sequence does not have to be followed strictly. Meaning that the packing sequence can be deviated by one position. In a physical system, this results in the need for one buffer space at the picking station, which in the project environment is not possible.

2-3-3 Reinforcement Learning

Another method that has been used recently to solve the BPP is Reinforcement Learning. Due to the learning component of the RL algorithm, the computation time in operation using a RL method tends to be low compared to the methods described earlier. A RL model typically exists of four elements, a *policy*, a *reward*, a *value function* and a model of the *environment*. RL uses an agent to solve the task without being given the solutions. The agent learns from experience by obtaining rewards following a reward function. The policy defines how the agent will react with the environment. At each time step t, the reward, and the state are updated. The value function determines in the long term how the method performs and is based on a cumulative reward. To maximize the cumulative reward and to generate a global optimum, there are two types of actions that are considered. Previous actions can be *exploited* or new actions *explored*. A trade-off needs to be made between these two decisions such that the resulting cumulative reward is maximized. The last characteristic of the RL is the decision between an *on-policy* and *off-policy* framework. In an *on-policy* framework, the action values are directly learned from the near-optimal policy. The process is shown in Figure 2-3.

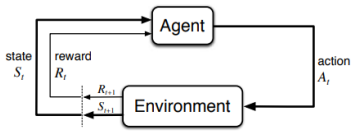


Figure 2-3: The interaction between the agent and the environment [Sutton and Barto, 2018]

Regarding the BPP, the RL is a fairly novel method used to solve the problem. Hu et al. [2017] is the first to describe the BPP problem with RL. The RL is used to increase the volume utilization considering an initial solution determined by a heuristic. The decisions that can be made are the sequence of the items, the item orientation, and the strategy to select the empty maximal space to pack the item in. The input of the RL network is the size of the items and the output is a sequence representing the packing order of items. The surface area of the bin is used to evaluate the solution. The algorithm does not consider stability.

A more recent research solving the BPP with Reinforcement Learning is conducted by Zhao et al. [2020]. Zhao et al. [2020] use a similar framework for the three-dimensional rectangular BPP as Hu et al. [2017]. However, the reward function is based on the volume utilization as well as the safety of the item placement. As can be seen in Figure 2-4, the heightmap H_n, three-dimension maps D_n and the feasibility masks M_n determine the environment state. The actor is able to act upon the orientation and the coordinates x and y of the item to achieve optimal placement. The algorithm is furthermore able to handle information about the next k lookahead items, without knowing the whole sequence [Zhao et al., 2020]. An important advantage of the research conducted by Zhao et al. [2020] is the usage of a robot picking environment. Following the use of a robot picking environment, the algorithm uses a stability constraint, unknown order sequence, and constraints imposed by the robot manipulator. However, even though the stability condition is added to the reward function, the algorithm cannot guarantee a safe placement. Furthermore, the method has been tested on a bin size of L = W = H = S centimeters, where S equals 10, 20, and 30 respectively. With increasing bin size, the performance indicated by the average volume utilization decreases. For S = 10, S = 20, and S = 30, the average volume utilization considering all benchmarks are 0.641, 0.623, and 0.551 respectively. Higher values for the bin size are not reported [Zhao et al., 2020]. Following this trend, the volume utilization when using a bin size equal to the one used by Picnic is expected to decrease even further.

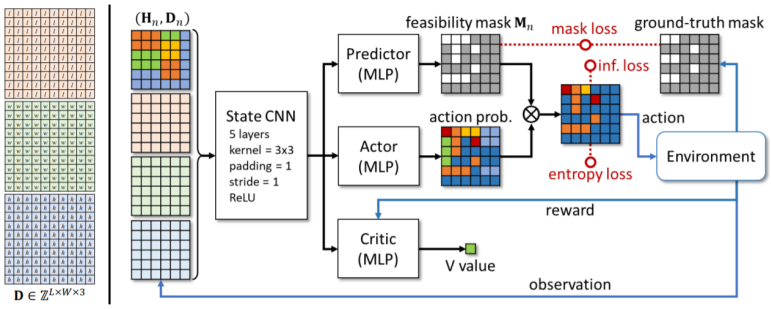


Figure 2-4: The environment state and framework of the RL algorithm [Zhao et al., 2020]

2-3-4 Quasi-Online algorithm

As described in Section 2-2-2, the Quasi-Online algorithm combines an online and an offline algorithm and is described by Wang and Hauser [2020]. The quasi-online algorithm consists of two parts, an offline *oracle* and an online *packing policy*. The offline oracle determines the location of each item given a sequence and constraints about the geometry of the item. Similar to the online model, each item is packed before the next item is revealed. Considering that the complete sequence is yet unknown, all possible sequences are examined, called a non-deterministic approach. The outcome of the offline oracle is saved and passed on to the packing policy. In the packing policy, all sets of feasible configurations are stored. As a consequence of the number of sequences checked, the

time complexity of the algorithm is expensive. According to Wang and Hauser [2020], the worst-case solution complexity is $\mathcal{O}(n!)$. Therefore the algorithm is impractical for a large number of items. The algorithm determines a packing for every possible item order that arrives and in this way generates a policy tree. In the policy tree, the location of every item is determined for every possible order sequence.

Policy tree An example of a policy tree with three items is shown in Figure 2-5. The result of every plan in the policy tree is a different packing plan. Therefore, this example results in six different packing plans. The final location for each of the items is chosen based on the highest number of locations in the tree. In Figure 2-6, the six possible packing plans are shown using heightmaps. The yellow boxes indicate the view on the bananas, the orange boxes on the oat flakes, and the green boxes on the washing detergent.

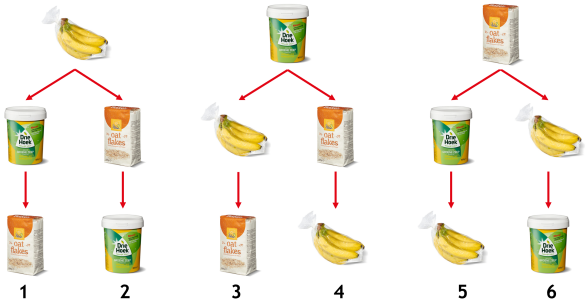


Figure 2-5: Example of a policy tree with three different items

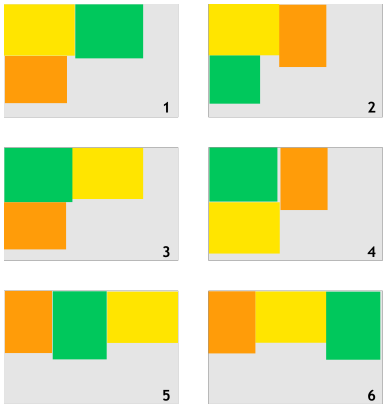


Figure 2-6: Example of packing plans with three different items

Description of the online algorithm A principal indicator for success in the Quasi-Online algorithm is the number of items that are a root in the policy tree. A root is determined as an item that is not placed on top of another item. If all items do not have edges and are therefore roots in the policy tree, all possible item sequences result in a feasible packing plan. Therefore, the packing is successful. For each item within the sequence, the location is determined. First is checked whether an already obtained plan is compatible with the next item to be placed. If there is an available plan, the location with the highest number of plans is chosen for the item. If no available plan is established, the offline planner is called to generate a renewed packing plan. If no feasible offline packing plan can be generated, the quasi-online algorithm fails. The algorithm for one item placement is shown in Appendix A-1 in Algorithm 3 [Wang and Hauser, 2020].

Offline packing plan The offline packing used in the research by Wang and Hauser [2020] is the Heightmap Minimization Heuristic from Wang and Hauser [2019], as defined briefly in Section 2-3-1. A further explanation of this algorithm is given in this section. Algorithms 4, 5, and 6 in Appendix A-1-2 describe the offline packing.

First of all, the planar-stable rolls and pitches are determined using the method of Goldberg et al. [1999]. This method determines the probability that a plane results in a stable pose, called the quasi-static probabilities. These probabilities can consequently be utilized to determine which face of the item is most likely to be a stable surface. The method is thoroughly described in Appendix B-2. For each item, a grid search over all coordinates in the bin is assessed. The item is rotated in each location over the n quasi-static orientations. Whether the item remains in the container is checked for each location-orientation pair. If this is the case, the score is obtained. In Appendix B-1 is documented exactly how the score is obtained.

The N lowest scores are determined and for these location-orientation pairs, the stability and the manipulation feasibility is checked. If there is no available location-orientation found with the quasi-static orientations, the item is rotated in every direction over the angles determined by Δr.

Stability of irregular items As defined in Section 2-2-4, the algorithm by Wang and Hauser [2019] obtains the stability by assessing a force and torque balance. First, the contact forces are determined from contact points. To reduce the number of contact points, the contact points are clustered within a grid size of 1 centimeter. The optimization problem is shown in Equation (2-4). The optimization problem searches for the minimum value of contact forces, as seen in the objective function in Equation (2-4a). The first constraint (2-4b) ensures the force equilibrium, the second constraint (2-4c) the torque equilibrium. Constraint (2-4d) ensures that in each contact point a contact force is found. Finally, Constraint (2-4e) adds the equation for the friction. If no contact forces can be found, the item is considered unstable [Wang and Hauser, 2019].

$$\begin{aligned}
\underset{x \in \mathcal{K}}{\text{minimize}} \quad & \sum_{k=0}^{n} \mathbf{c}\mathbf{x}_k && && \text{(2-4a)}\\
\text{subject to} \quad & \sum_{k=0}^{n} \mathbf{b}\mathbf{x}_k = m_i g, && , && \text{(2-4b)}\\
& \sum_{k=0}^{n} \mathbf{r}_k^{COM} \times \mathbf{x}_k = \mathbf{0}, && , && \text{(2-4c)}\\
& \mathbf{x}_k \cdot \mathbf{n}_k > 0, && (k = 0, ..., n), && \text{(2-4d)}\\
& ||f_k^{\perp}|| \leq \mu_k (\mathbf{x}_k \cdot \mathbf{n}_k), && (k = 0, ..., n) && \text{(2-4e)}
\end{aligned}$$

In this problem, $\mathbf{x}_k$ is the force vector of each contact point. The force vector is shown below, where f_x^k, f_y^k, and f_z^k are the contact forces in x, y, and z direction, respectively.

$$\mathbf{x}_k = \begin{bmatrix} f_x^k \\ f_y^k \\ f_z^k \end{bmatrix}$$

$\mathbf{c}$ and $\mathbf{b}$ are the vectors $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ and $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$ respectively, m_i is the mass of item i, g is the gravitational constant, $\mathbf{r}_k^{COM}$ the vector indicating the difference between the center of mass and the contact point, $\mathbf{n}_k$ is the normal vector of each contact point, and μ_k is the friction coefficient. A convex programming solver is used to obtain the contact forces.

2-4 Comparison of the algorithms

In this section, the algorithms from Section 2-3 are compared. A main element in assessing the different algorithms is the use of the characteristics obtained in Section 2-2. Various algorithms utilize more strict constraints, which need to be taken into account when comparing the algorithms. Furthermore, the algorithms are to be compared against a benchmark. The benchmarks used are explained in Section 2-4-1. In Section 2-4-2, the algorithms are compared and evaluated.

2-4-1 Benchmark description

Two benchmarks are used to compare the algorithms discussed. There is not a single benchmark available to compare all Bin Packing Problems. The first benchmark is originally described by Bischoff and Ratcliff [1995]. The benchmark is typically used for theContainer Loading Problem (CLP). CLPs are Cutting and Packing problems similar to the BPP, however, the container used in the CLP is bigger compared to the bin used in the BPP. As a result, the time required to place all items might increase but other than this, does not influence the ability to use the benchmark for the BPP. The benchmark by Bischoff and Ratcliff [1995] uses a container size of 587 centimeters in length, 233 centimeters in width, and 220 centimeters in length. The item dimensions are between 20 to 120 centimeters. The benchmark focuses on the number of different item sizes used. The different number of item input sizes used are 3, 5, 8, 10, 12, 15, or 20 items.

The second benchmark is typically used for Bin Packing Problems. The benchmark makes use of classes for item generation. The classes define the dimensions of the items. The benchmark is initially described by Martello et al. [2000] and uses five classes determined by Martello and Vigo [1998] and three classes determined by Berkey and Wang [1987]. For the last three classes, the bin sizes are 10, 40, and 100 centimeters respectively. The item sizes for these classes are a random value between 1 and 10, 1 and 35, and 1 and 100 centimeters. The bin sizes for the first five classes are all 100 centimeters.

The item sizes depend on a probability and item types as defined in Table 2-2. For each class k, the probability of getting item type k is 60% and 10% for the other item types. The sizes of the items are a randomized value within the bounds given in the table in columns *Width*, *Height* and *Depth*. The resulting classes are shown in Table 2-3. Due to the rotation of the items, classes 1, 2, and 3 are similar. Therefore, in researches classes 2 and 3 are often neglected.

Table 2-2: Item types for classes 1-5 [Martello et al., 2000]

Item type	Width	Height	Depth
1	[1, 50]	[66 $\frac{2}{3}$, 100]	[66 $\frac{2}{3}$, 100]
2	[66 $\frac{2}{3}$, 100]	[1, 50]	[66 $\frac{2}{3}$, 100]
3	[66 $\frac{2}{3}$, 100]	[66 $\frac{2}{3}$, 100]	[1, 50]
4	[50, 100]	[50, 100]	[50, 100]
5	[1, 50]	[1, 50]	[1, 50]

Table 2-3: Item class determination for classes 1-8 [Martello et al., 2000]

Class	Bin size	Width	Height	Depth
1	100x100x100	From Table 2-2		
2	100x100x100	From Table 2-2		
3	100x100x100	From Table 2-2		
4	100x100x100	From Table 2-2		
5	100x100x100	From Table 2-2		
6	10x10x10	[1,10]	[1,10]	[1,10]
7	40x40x40	[1,35]	[1,35]	[1,35]
8	100x100x100	[1,100]	[1,100]	[1,100]

2-4-2 Evaluation of algorithms

The algorithms described in Section 2-3 cannot be directly compared to one another due to the use of different constraints. The type, item shape, and constraints used in the algorithms are described in Table 2-4. The algorithms are consequently compared using the two benchmarks as discussed in the previous section, shown in Tables 2-5 and 2-6. Four algorithms are missing from these tables, as the results were not published in any of these two benchmarks. These algorithms are discussed first. Thereafter, the other algorithms are evaluated.

Table 2-4: Constraints and type of the algorithms discussed

Author	Algorithm	Type	Item shape	Stability	Bounded
Moon and Nguyen [2014]	DBLF	Online	Regular	x	✓
Allen et al. [2011]	3BF	Offline	Regular	x	✓
Gonçalves and Resende [2013]	DFTRC	Online	Regular	x	✓
Crainic et al. [2008]	EP	Online	Regular	x	x
Wang and Hauser [2019]	HM	Offline	Irregular	✓	✓
Moon and Nguyen [2014]	HGA	Offline	Regular	x	✓
Gonçalves and Resende [2013]	BRKGA	Offline	Regular	x	x
Peng et al. [2009]	HSA	Offline	Regular	✓	✓
Bortfeldt et al. [2003]	PTS	Offline	Regular	✓	✓
Zhao et al. [2020]	RL	Online	Regular	x	x
Wang and Hauser [2020]	QO	Online	Irregular	✓	✓

Algorithms not described by the benchmarks The four missing algorithms are the heuristics DFTRC and HM, the Quasi-Online algorithm by Wang and Hauser [2020], and the Reinforcement Learning method by Zhao et al. [2020]. The algorithm that resembles the concerned problem the most is the Quasi-Online (QO) algorithm. The QO algorithm, as well as the DFTRC heuristic, are not compared against other results.

The HM heuristic is compared to the DBLF algorithm by success rate. A packing is successful if in this case 10 items can be placed within one bin. Using 1000 item sets, the success rate is generated. The success rate of the HM heuristic is 97.1% within a time of 34.9 seconds, whereas for the DBLF heuristic the success rate is 96.3% within 50.1 seconds. No direct conclusions are drawn concerning the volume utilization.

The RL method by Zhao et al. [2020], which is described in Section 2-3-3, is somewhat similar to the fifth class of the second benchmark. However, there are important differences that make the comparison between the two methods impossible. Zhao et al. [2020] uses bin sizes $L = W = H = 10$ with item sizes $l_i \leq L/2, w_i \leq W/2$ and $h_i \leq H/2$. Even though in class 5 a bin size of $L = W = H = 100$ is used, the formula for the item sizes are equal. However, in class 5 there is a probability of placing a larger item. Therefore on average fewer items can be placed inside the bin. The number of placed items in each bin is equal to 12.2 items, resulting in a space utilization of 50.5 %. However, no comparison can be made from this result to the other algorithms.

Algorithms described by the benchmarks Apart from the four algorithms discussed in the previous section, the other algorithms can be divided into three comparable groups and compared with the two benchmarks. These groups are *offline algorithms with stability*, *offline algorithms without stability* and *online algorithms without stability*.

In Table 2-5, the rows show the number of item types considered in the first benchmark. In the columns, the different algorithms considered using this benchmark are shown. For

Table 2-5: Comparison algorithms with the first benchmark [Allen et al., 2011, Bortfeldt et al., 2003, Gonçalves and Resende, 2013, Moon and Nguyen, 2014, Peng et al., 2009]

Set	DBLF		3BF		HGA	BRKGA	HSA		PTS	
# item types	Time [s]	Util. [%]	Time [s]	Util. [%]	Util. [%]	Util. [%]	Time [s]	Util. [%]	Time [s]	Util. [%]
3	0.3	88.7	0.1	88.9	93.2	95.3	15.4	92.9	36	93.5
5	0.4	89.0	0.1	87.7	94.1	95.9	42.0	93.6	48	93.8
8	0.5	87.8	0.2	86.5	93.5	96.1	78.1	93.6	97	93.6
10	0.6	87.7	0.2	85.6	92.7	96.0	106.4	93.2	138	93.1
12	0.7	87.6	0.2	85.8	92.2	95.8	132.2	92.8	179	92.3
15	0.9	87.4	0.3	85.7	91.6	95.7	175.7	92.4	150	91.7
20	1.1	86.5	0.4	84.9	90.9	95.3	245.7	91.5	198	90.6
Avg.	0.6	**85.8**	0.2	**87.6**	**92.6**	**95.7**	113.6	**92.8**	120.9	**92.7**

some algorithms, the volume utilization is shown together with the computation time. However, for a few algorithms, the computation time is not reported. In Table 2-6, the results from algorithms using the second benchmark, as discussed in Section 2-4-1, are shown. In this table, n is the number of items placed in the bin. For each of the classes and each algorithm the number of bins used to pack n items is reported.

The offline algorithms that consider stability are meta-heuristics Hybrid Simulated Annealing and Parallel Tabu Search. Comparing HSA and PTS on computation time and volume utilization, HSA performs better on both requirements. However, the differences between the two algorithms on both the volume utilization and the computation time are minimal. The stability constraint for HSA is more strict compared to the stability constraint imposed in the PTS algorithm. For PTS 55 percent of the base area of the item is to be supported for the item to be stable whereas HSA requires full support. Therefore, the HSA has a more strict stability requirement, resulting in a smaller search space for item placement.

The offline algorithms that do not include stability, 3BF, HGA, and BRKGA, are compared on volume utilization exclusively. As identified in the first benchmark, the BRKGA has a higher volume utilization rate. Moon and Nguyen [2014] reports that the use of a more strict stability constraint reduces the volume utilization rate. Following this reasoning, the 3BF might perform worse compared to the algorithms that do include stability concerning volume utilization. The difficulty lies in comparing the HGA and the BRKGA algorithms. The BRKGA algorithm considers a non-bounded space which results in the placement of items in any available bin. However, in the HGA algorithm a bin is closed after the next one is opened, resulting in a smaller solution space. Furthermore, the BRKGA algorithm does not consider the stability constraint. Therefore, while reaching the highest volume utilization rate, this is not a guarantee for being the best algorithm considering the project environment.

Table 2-6: Comparison algorithms with the second benchmark [Allen et al., 2011, Crainic et al., 2008, Gonçalves and Resende, 2013]

Class	n	3BF	EP	BRKGA	
		# bins	# bins	Time [s]	# bins
1	50	13.6	13.8	3.5	13.4
	100	-	-	16.1	26.6
	150	-	-	41.3	36.4
	200	-	-	51.1	50.8
4	50	29.4	29.5	3.1	29.4
	100	-	-	17.1	59
	150	-	-	48.4	86.8
	200	-	-	101.5	118.8
5	50	9.0	8.4	7.1	8.3
	100	-	-	29	15
	150	-	-	68.9	20
	200	-	-	130.6	27.1
6	50	9.9	10.1	2.7	9.7
	100	-	-	11.3	18.9
	150	-	-	26.4	29
	200	-	-	48.2	37.3
7	50	8.4	7.7	5.9	7.4
	100	-	-	23.9	12.2
	150	-	-	57.1	15.3
	200	-	-	101.4	23.4
8	50	9.9	9.5	5.6	9.2
	100	-	-	23.6	18.8
	150	-	-	51.9	23.6
	200	-	-	101.1	29.3
Total bins (n = 50)		80.2	79		77.4

In the last group, with online algorithms DBLF and EP, the performance of the algorithms is tested against different benchmarks. However, compared to the 3BF algorithm, the DBLF algorithm performs worse while the EP algorithm outperforms the 3BF algorithm. Nonetheless, no conclusions can be drawn from this comparison because the EP algorithm is non-bounded and the DBLF algorithm is bounded.

2-5 Outcome of the evaluation

In conclusion, the algorithms discussed for the Bin Packing Problem are not easily comparable to one another due to the use of different constraints and missing information. From Table 2-4 is seen that the Quasi-Online algorithm is the only algorithm considering all constraints and the type according to the project environment. The results are however not comparable to the other algorithms due to not making use of one of the benchmarks.

From the algorithms that can be compared, the HGA, HSA, PTS and BRKGA algorithms outperform the other algorithms on volume utilization. However, the BRKGA algorithm is non-bounded and does not include the stability constraint, sequence or the irregular items. From these four algorithms, the HSA algorithm is the one resembling the project environment the most, even though it does not consider the irregular item and sequence constraints. The heuristic HM does include the irregular item, stability constraint and is bounded, however is missing the sequence constraint. This heuristic is used by Wang and Hauser [2020] as an offline packing of the Quasi-Online algorithm and is therefore directly compatible with the Quasi-Online algorithm. Therefore, this combination is a promising solution for the irregular Bin Packing Problem. However, more research needs to be done regarding the performance of these algorithms.

2-6 Remarks on the Quasi-Online algorithm

As described in the previous section, the Quasi-Online algorithm places irregular items directly in a bin. The bin is closed when an item cannot be placed, which ensures that the algorithm is *1-bounded*. Furthermore, the item sequence is unknown and the stability constraint is ensured. Even though the Quasi-Online algorithm includes all requirements set, the computation time of the algorithm poses a problem. As shown in Table 2-7, when using an increasing number of items to be placed the computation time increases substantially.

For the Quasi-Online algorithm to be practical in an automated warehouse, the computation time needs to be decreased. The placement of each item needs to be conducted within seconds of arrival. As is seen from Table 2-7 the item placement with ten items takes about five minutes for the placement of each item, whereas this should be obtained within a couple of seconds. Besides, Wang and Hauser [2020] note that ten items placed is the maximum they could reach within a 24-hour cutoff with 24 item sets.

On average 13.3 items are put into one Picnic bin. Therefore, regarding the project environment, this number of items should be placed inside one bin within a reasonable time.

Furthermore, the algorithm should be compatible with the automated warehouse. The algorithm by Wang and Hauser [2020] does not consider the fragility of the placed items, which is required in this project environment. Consequently, the picking sequence as explained in Section 1-1-3 should be considered in the algorithm. Furthermore, the system in which the placement takes place includes some knowledge about successive items to be placed, which the Quasi-Online algorithm does not include. In the next chapter, these shortcomings to the Quasi-Online algorithm are addressed.

Table 2-7: Results from the QOP algorithm by Wang and Hauser [2020]

Items	Success (%)	Time (mean / max, s)	# planner calls (mean / max)
2-5	99.4	22.4 / 1,520	1.4 / 43
5	97.0	65.1 / 5,800	2.1 / 46
10	43.7	2,850 / 85,478	45.8 / 5,363

Chapter 3

Improvements to the Quasi-Online algorithm

In this chapter, the improvements to the Quasi-Online algorithm are presented. First, the baseline to test the improvements for the Quasi-Online algorithm is determined in Section 3-1. Subsequently, the methods to improve this algorithm are explained. The improvements are based on the remarks of the Quasi-Online algorithm as shown in Section 2-6. The improvements are divided into two sections. The first section, Section 3-2, describes improvements to the offline part of the Quasi-Online algorithm as described in Section 2-3-4. These improvements are the rotation of the item and the replacement of the stability constraint. The improvements are required due to another method of shape determination which is used, as explained in Section 3-1-4. The second section, Section 3-3, describes improvements to the online part of the Quasi-Online algorithm. The improvements of the online algorithm reflect the knowledge of the item sequence. Finally, in Section 3-4, the modifications to the algorithm are discussed to convert the results of the Quasi-Online algorithm to a measure for volume utilization.

3-1 Obtaining the initial model

In order to be able to improve the Quasi-Online algorithm from the research by Wang and Hauser [2020], it first needs to be recreated. In this section, we will discuss what trade-offs have been made. Thereafter, the results are compared to the results by Wang and Hauser [2020].

Figure 3-1: Examples of incomplete object models

3-1-1 Inputs

The algorithm by Wang and Hauser [2020] has been developed for bin packing of irregular items. The object models used in this research are obtained from the YCB data set [Calli et al., 2015] and the APC data set [Rennie et al., 2016]. The YCB data set generates 3-D models using five PrimeSense Carmine 1.08 depth sensors, which make use of the Structured Light perception method. The Structured Light method is a commonly used method and consists of one camera and one structured light projector. A pattern is projected on the surface. The pattern is distorted when an object is placed on the surface. Using different patterns and triangulation, the depth of the object can be calculated [Geng, 2011]. A disadvantage of this method is that the Structured Light method is colour-dependent and influenced by the reflection of the surface. Therefore, the Structured Light method is not able to perceive all kinds of objects [Keerativittayanun et al., 2011, Szelag et al., 2017]. The APC data set makes use of the Structured Light method as well [Rennie et al., 2016]. Various object models are not modeled properly, as can be seen in Figure 3-1. The perception method has not been able to detect the complete object, resulting in an incomplete object model. Therefore, the data is filtered to contain only complete object models. The total data set includes 119 object models, 97 of which were complete. Wang and Hauser [2020] use a subset of the object as well. The number of object models used is given and equal to 96 object models. However, Wang and Hauser [2020] do not report which object models are used exactly. The objects that are excluded from this research are reported in Appendix D-1.

3-1-2 Parameter settings

Different parameters need to be set in the model. In Appendix A-1-2, the algorithms for the offline packing are shown. The parameters for the offline packing are the quasi-static orientations n, the number of candidates N and the angles of rotation Δr as seen

in Algorithms 4, 5 and 6. Furthermore, the value c in Equation (2-3) and the container size S are defined. In Table 3-1, the parameter settings for the Quasi-Online algorithm are shown. One parameter missing from this table is the resolution of the heightmap. This parameter will be explained in Section 3-1-4. The parameters considered are equal to the parameters used by Wang and Hauser [2020]. However, Wang and Hauser [2020] use inches for the container size which are converted to centimeters and rounded to the nearest integer.

Table 3-1: Parameter settings for the Quasi-Online algorithm by Wang and Hauser [2020]

Parameter	Value
n	4
N	100
Δr	$[0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}]$ rad
c	1
S	[50, 34, 16] cm

3-1-3 Hardware employment

The algorithm is executed on a Dell Latitude 5411 laptop with Intel Core i7-10850H (10th generation) with 32.0 RAM memory at 2.70 GHz frequency. Wang and Hauser [2020] use Amazon Web Services instance type m5.12xlarge which is not available for this project. The Amazon Web Service used by Wang and Hauser [2020] has 48 virtual cores, whereas the number of processes that can be run on the Dell laptop is equal to 12 without using multiple processes per thread. Therefore, the experiments in this project are expected to run slower compared to the experiments conducted by Wang and Hauser [2020]. The algorithm is implemented using the programming language `Python` (version 3.8) and executed using the `JetBrains PyCharm IDE` (Community Edition 2020.3.3).

3-1-4 Shape determination

Wang and Hauser [2020] use a high-resolution mesh to determine the shape of the item. However, due to the use of the high resolution mesh the time to compute the offline packing increases, as discussed by Wang and Hauser [2020]. Besides, the method used to determine overlap between the items using a high-resolution mesh is not documented by Wang and Hauser [2020]. For these reasons, another method for shape representation is considered.

Methods for shape determination Multiple methods to represent the shape are reviewed. The methods can be divided into surface-based representations and volume-based representations [Han et al., 2019]. Surface-based methods define the object shape

by extracting characteristics of the surface using image processing. The three most popular surface-based representations are segmented point clouds, meshes, and small surface patches called surfels. On the other hand, volume-based representations such as voxels often require a lot of memory. Voxels are small cubes that can together be used to describe objects. The required memory when using voxels can be minimized by increasing the size of the voxels. Furthermore, octree structures are voxel-based representations in which the size of the voxels of one object are varied.

To compute the models, surface-based methods are faster compared to volume-based representations. However, to use surface-based models to describe irregular item placement, the overlap between the objects needs to be defined. From the literature review is concluded that few methods exist that determine object overlap with irregular items using polygons [Canellidis et al., 2016, Chernov et al., 2010, Liu et al., 2015]. To the author's knowledge, there are no documented methods that use three-dimensional irregular items. The overlap between voxels is easily generated and therefore requires a low computation time. As a result, regarding the irregular three-dimensional objects, the use of the voxel-based representation is preferred. In the next sections, the voxelization method is further explained.

Voxelization Voxelization is a method to describe the geometry of a three-dimensional object using voxels. As discussed in the previous section, the method of voxelization is considered for the shape representation of the items.

Dividing the item in voxels with a low resolution will result in fast computation as well as sufficient utilization rate [Li et al., 2016]. Furthermore, due to the discretization of the items, a grid is considerably straightforward to implement, resulting in simpler stability application compared to other methods for shape representation. The implementation of the voxels demands modifications in the algorithm, which are elaborated on in the next sections.

Voxel sizes The size of the voxels influences the volume utilization in the bin. An example of an item with different voxel sizes is shown in Figure 3-2. The voxels are obtained from the point cloud, resulting in an overestimation of the item size. As seen in Figure 3-2, the smaller voxel sizes result in better approximation of the volume of the item. However, using an increasing amount of voxels, the computation time increases. Therefore, to decide about the voxel sizes, a trade-off needs to be made between the computation time and the success rate.

Table 3-2 shows the results with different voxel sizes using the settings described earlier in this section. For each experiment 100 item sets are used. The item sets are generated at random from the 97 object models. For each item set is determined whether the placement of the five items is a success. This ultimately describes the success rate. Voxel sizes smaller than 0.5 centimeters are not tested, due to the increase in the computation time with the decrease of the voxel size.

Figure 3-2: Item with voxel size of 2 centimeters (left), 1 centimeter (middle) and 0.5 centimeters (right)

Table 3-2: Results using different voxel sizes

Voxel size (cm)	Success (%)	Time (s)		# Planner calls	
	Mean	Mean	Max	Mean	Max
0.50	96	417.39	8,359	1.16	15
0.67	95	192.57	1,999	1.52	25
1.00	90	65.79	933	1.70	16
2.00	80	37.38	622	2.03	19

From these results is concluded that the voxel size of 0.67 centimeters is optimal regarding the success rate and the computation time. With the decrease of the voxel size to 0.5 centimeters, the computation time is doubled and the success rate differs with only one percent. The increase to 1.0 centimeters results in a significantly lower success rate.

3-1-5 Assumptions

To be able to generate the algorithm, a couple of assumptions are made concerning the items, placement of the items, and the environment. The assumptions are comparable to the assumptions made by Wang and Hauser [2020]. The assumptions are held throughout all experiments conducted.

1. The item is non-deformable
2. The center of gravity of the box coincides with its geometric center
3. The mass of the item is evenly distributed
4. The type of robot manipulator, movement of the manipulator, and gripper attachment point are not included
5. The initial orientation of the item is not constrained

6. The items can be placed on all sides
7. There is no buffer space at the picking station
8. One bin is open at a time. If the item does not fit into the bin, the bin is closed and a new bin is opened
9. The items in the bin are not re-arranged

The items considered in this research are grocery items. Therefore, the material of each of the products differ, from carton boxes to (unpacked) vegetables. Consequently, the material properties of each of these products are different. To incorporate the deformability of the items, the specific characteristics of each of the products need to be considered. The deformation of an item with more than a few millimeters is prohibited, since in this case the item cannot be sold any more. The resolution of the algorithm, as determined in Section 3-1-4, is bigger than a few millimeters. Therefore, the items are considered non-deformable. Similar to this reasoning, the mass is considered to be evenly distributed over the item and the center of gravity is chosen to coincide with its geometric center.

The project is focused on the packing of the items without considering the robot manipulator. The reason for this is that the dynamics of the robot manipulator do not necessarily have an influence on the packing of the items. For the same reason, the gripper attachment point is not included.

The initial orientation can be defined by using a perception method and train an algorithm to detect the orientation of the items. However, at the start of this project, the Picnic's automated warehouse was not operable yet. Therefore, the input images could not be obtained and the initial orientation of the items are therefore not included.

The picking station does not have a buffer space for the stock bin. Therefore, the bin cannot be stored at the picking station. Besides, looping the bin at the picking station is possible however requires more time and is therefore less efficient. This leads to the assumption that the packing algorithm needs to be 1-bounded, therefore one bin is open at a time. For the same reason that the supply chain needs to be as time- and space efficient as possible, the item needs to be placed as fast as possible. Therefore, re-arranging the items at the picking station is not allowed.

3-1-6 Results of the Quasi-Online algorithm

In this section, the results of the Recreated Quasi-Online algorithm are shown. The results shown below in Table 3-3 are obtained using the parameters from the previous sections, as well as the voxel size of 0.67 centimeters. 100 iterations are done for each of the different number of items. These results show that the computation time increases and the success rate decreases with increasing number of items. The planner calls increase as well. The explanation of these results is that the algorithm has increasing

Table 3-3: Results of the Recreated Quasi-Online algorithm

Items	Success (%)	Time (s)			# Planner calls	
		Mean	Max	Std	Mean	Max
2	100	59.48	210	35.13	1.00	1
3	99	90.86	317	47.25	1.00	1
4	98	109.93	364	58.20	1.04	5
5	95	192.57	1,999	229.84	1.52	25
6	87	466.37	12,316	1,548.57	3.66	91

Table 3-4: Comparison between the Recreated Quasi-Online algorithm (RQO) and the Quasi-Online algorithm by Wang and Hauser [2020] (QO)

Items	Success (%)		Time (s)		Planner calls	
	RQO	QO	RQO	QO	RQO	QO
2-5	98.0	99.4	113.21	65.1	1.14	1.4

difficulty in placing the items in the bin. With increasing number of items, the items are more likely to be placed on top of one another, causing the algorithm to re-iterate. This causes the increase in number of planner calls and therefore the increase in computation time as well.

Table 3-4 shows a comparison between the Recreated Quasi-Online algorithm (RQO) and the Quasi-Online algorithm by Wang and Hauser [2020]. As already indicated in Section 3-1-3, different hardware is used for each of the methods. Therefore, the computation time for placement of each instance is faster in the case of the Quasi-Online algorithm from Wang and Hauser [2020]. Furthermore, the planner calls and success rate are decreased in the case of the Recreated Quasi-Online algorithm compared to the Quasi-Online algorithm by Wang and Hauser [2020]. A potential reason for this is the higher accuracy for the contact forces in the Recreated Quasi-Online algorithm. The Quasi-Online algorithm by Wang and Hauser [2020] clusters the contact forces that act on the item by one centimeter. This reduces the amount of contact forces. However, due to the use of 0.67 centimeter voxels, this clustering step in the case of the Recreated Quasi-Online algorithm is infeasible. This increase in the number of contact forces in the Recreated Quasi-Online algorithm compared to the Quasi-Online algorithm by Wang and Hauser [2020] causes a more strict stability measure. Therefore, the items are less likely to be stable, decreasing the success rate and decreasing the number of planner calls.

3-2 Improvements to the offline algorithm

In this section, the improvements to the offline algorithm are considered. First, finding a stable initial position using item rotation is discussed in Section 3-2-1. An alternative stability measure is explained in Section 3-2-2.

3-2-1 Rotation of the item

For the voxelization of an item, the orientation of an item is critical. In Figure 3-3, a point cloud representing a cracker box is shown. From the rightmost view, an angle is perceived between the axes of the bin and of the point cloud. As can be seen in this image, the item is rotated with respect to the axis.

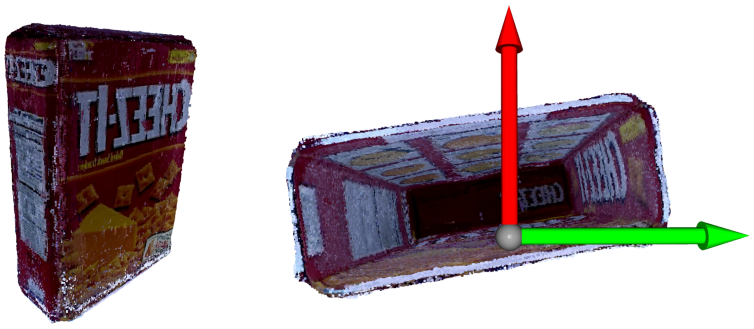


Figure 3-3: Side view (left) and bottom view (right) of a point cloud depicting a cracker box

To obtain the voxels, the point cloud is projected on a grid aligned with the axes, as shown in Figure 3-4. The image on the right shows the voxelized item. From this image can be observed that it is difficult to stabilize the voxelized item on the ground floor. This indicates that the item should be rotated before voxelization to align with the axes.

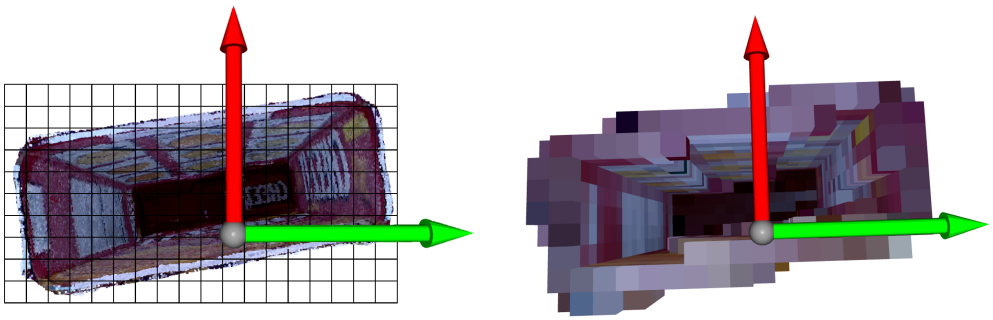


Figure 3-4: Point cloud with grid the voxels are projected on (left) and the voxelized item (right)

Theory of item rotation

A method to obtain the axes of rotation of a point cloud is by using the Principal Component Analysis (PCA) similar to the method obtained by Xiao et al. [2020]. PCA is used to obtain the main directions along which the variance is the highest, corresponding to the three axes x, y, and z of the intrinsic item frame.

To obtain the rotation axes of the item, the coordinates of every point p_i in point cloud $\mathcal{P}$ is acquired. To assess the variance in the different directions, the covariance matrix is obtained using Equation (3-1).

$$\mathbf{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{p_i} - \bar{\mathbf{p}})(\mathbf{p_i} - \bar{\mathbf{p}})^T \tag{3-1}$$

In this equation, Σ is the covariance matrix, p_i is one point in the point cloud and $\bar{p}$ is the center of the point cloud. Eigendecomposition is subsequently used to obtain the eigenvectors of the covariance matrix Σ, using the eigenvalue problem. In the following equations, $\mathbf{v}$ is the eigenvector corresponding to the eigenvalue λ.

$$\mathbf{\Sigma v} = \lambda \mathbf{v} \tag{3-2}$$

$$\det(\mathbf{\Sigma} - \lambda \mathbf{I}) = 0 \tag{3-3}$$

Three eigenvector-eigenvalue pairs are found using this decomposition, corresponding to the axes x, y, and z of the intrinsic frame. The intrinsic frame is therefore obtained by adding the three eigenvectors $\mathbf{v_1}$, $\mathbf{v_2}$ and $\mathbf{v_3}$ together to achieve rotation matrix $\mathcal{V}$. To express the point cloud in the obtained intrinsic frame the following equation is used [Xiao et al., 2020]. Xiao et al. [2020] shows subsequently that the intrinsic frame remains invariant with every rotation of the point cloud.

$$p_i' = \mathcal{V}^T(p_i - \bar{p}). \tag{3-4}$$

In Figure 3-5, the principal axes of this particular item are shown. As can be seen, the axes are roughly aligned with the expected axes of the item. Introducing the rotation of the item, the point cloud, and the voxel grid of the rotated item are shown in Figure 3-6.

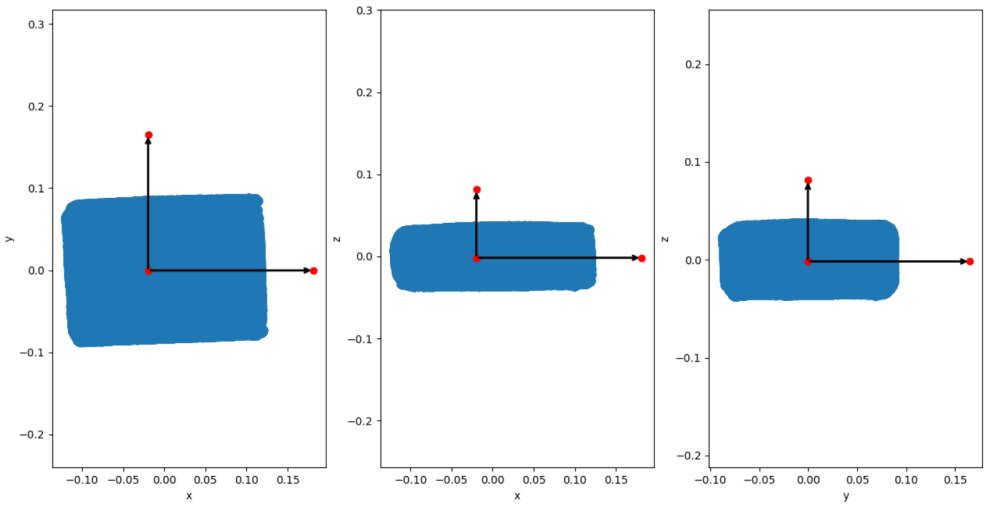


Figure 3-5: Principal axes of an item in the x, y-plane (left), x, z-plane (mid) and y, z-plane (right)

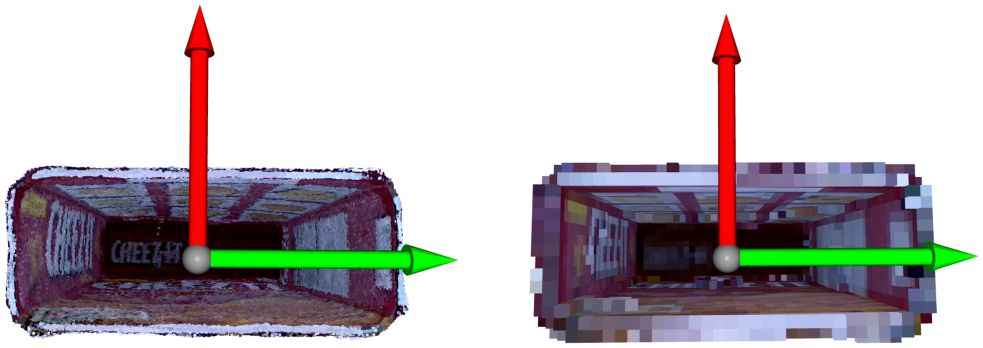


Figure 3-6: Point cloud (left) and voxelgrid (right) of the rotated item

3-2-2 Placing stability measure in grid search

As described in Section 2-3-4, a score is determined which assesses all feasible options of the location and the orientation of the item. This score is determined in the grid search. After the score is determined for each instance, a stability check is performed. However, in the case that all candidate locations are found to be unstable, the packing will be considered a failure. Therefore, to increase the success rate an extension to the stability measure is considered. This extension is the addition of the stability determination in the grid search, directly disregarding the candidate when the chosen location and/or orientation are unstable. In this section, first, the stability check obtained by Wang and Hauser [2020] is thoroughly described. Subsequently, the new location for the stability

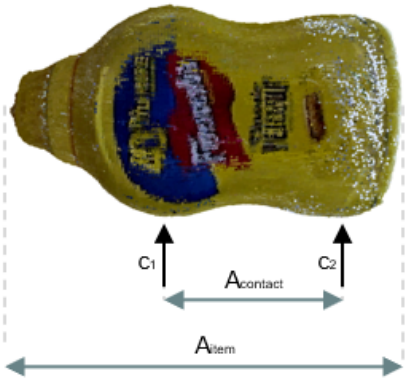


Figure 3-7: Item with two contact points c_1 and c_2.

measure is discussed. In the last section, different methods are considered for obtaining a measure of stability in the grid search.

Method for current stability check outside of grid search

For the stability check which is described by Wang and Hauser [2019], first the contact points are calculated. An example of the contact points is shown in Figure 3-7. The contact points are the points where the item is either in contact with the bin or with other already placed items. After the contact points are found, the forces applied to these contact points are calculated. This is done using a force and torque equilibrium, as described in Section 2-3-4. Similar to Wang and Hauser [2019], the convex programming solver `CVXPY` is used to solve the problem. The convex problem is shown in Equation (2-4). In case no value for the force is found, the orientation is not feasible and therefore the stability is not maintained.

Alternative location of stability measure

All new methods for the stability measure are based on implementing the stability measure in the grid search. In Algorithm 1, the method by Wang and Hauser [2019] is shown. Lines 2 until 14 determine the grid search. Lines 16 until 18 define the stability measure for the N lowest values for the score.

Algorithm 1: Stability outside grid search

input: item geometry $\mathcal{G}$, container $\mathcal{C}$, pitches and yaws $\mathcal{O} = (\phi_1, \psi_1), ..., (\phi_n, \psi_n)$, sequence of the packed items $\mathcal{S} = (s_1, ...s_i)$, transforms of the packed items $\mathcal{P} = \{P_1, ..., P_i\}$

1 **output:** Transform T or **None**
2 **for** $(\phi, \psi) \in \mathcal{O}$ **do**
3 **for** $\theta \in \{0, \Delta r, 2\Delta r, ..., 2\pi - \Delta r\}$ **do**
4 Let $R \leftarrow R_z(\theta)R_y(\phi)R_x(\psi)$;
5 Discretize legal horizontal translations for $R \cdot \mathcal{G}$ into grid $(X_1, Y_1), ..., (X_n, Y_n)$;
6 **for** $(X,\ Y) \in (X_1, Y_1), ..., (X_n, Y_n)$ **do**
7 Find the lowest collision free placement Z at translation X,Y;
8 Let T be rigid transform with rotation $\mathcal{R}$ and translation (X, Y, Z);
9 **if** $T \cdot \mathcal{G}$ *lies within* $\mathcal{C}$ **then**
10 Add T to $\mathcal{T}$
11 **end**
12 **end**
13 **end**
14 **end**
15 Score each T in $\mathcal{T}$ based on heuristic used;
16 **for** *up to N lowest values of T in* $\mathcal{T}$ **do**
17 $s \leftarrow$ isStable$(T \cdot \mathcal{G}, \mathcal{C}, P_1 \cdot \mathcal{G}_{s_1}, ..., P_i \cdot \mathcal{G}_{s_i})$
18 **end**
19 **return** None

Two adjustments are made to the stability measure as shown in Algorithm 1. First of all, lines 9 until 11 are replaced by the lines shown in Algorithm 2. In this part, the stability is determined using the *isStable* function.

Algorithm 2: Replacement lines grid search

9 **if** $T \cdot \mathcal{G}$ *lies within* $\mathcal{C}$ **then**
10 **if** $s \leftarrow$ *isStable*$(T \cdot \mathcal{G}, \mathcal{C}, P_1 \cdot \mathcal{G}_{s_1}, ..., P_i \cdot \mathcal{G}_{s_i})$ **then**
11 Add T to $\mathcal{T}$
12 **end**
13 **end**

Lines 16 until 18 from Algorithm 1 are replaced by the line shown below. As is seen, instead of adding a for loop after each of the scores is obtained to assess the stability measure, the stability is added to the grid search. When the item is considered stable, the location-orientation pair is added directly to the candidates.

15 Obtain T with the highest score;

Methods for the stability measure using alternative location

As described in the previous section, an alternative location for the stability measure can be used. The force and torque equilibrium determine the stability of the item in the research from Wang and Hauser [2019] and is further called the *equilibrium stability method*. However, due to the optimization problem that requires the calculation of many contact forces this method is computationally expensive. Therefore, two other methods are considered as well. From the literature, different methods ensuring stability are determined, as shown in Section 2-2-4. The two additional methods are based on methods found in literature.

The first alternative method, called *contact stability method*, makes use of the contact points as well. However, in contrast to obtaining the contact forces and ensuring static stability, the number of contact points is used as an indicator of stability. The number of contact points is added to the score with a weight value assigned to it. The updated score is shown in Equation (3-5). This equation is, apart from the added number of contact points C_{num} and the weight α, equal to Equation (2-3). Observing that a low score is favorable and an increased number of contact points indicate more probable stability, the weighted contact points are subtracted from the initial score.

$$c \cdot (X + Y) + \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} H_c'[i, j] - \alpha \cdot C_{num} \tag{3-5}$$

The last method considered is similar to the previous method, not an exact calculation of the stability but an indicator of a stable configuration. This method is called the *ratio stability method*. The contact points are used to determine the supported base area $A_{contact}$, as shown in Figure 3-7. The ratio between the supported base area and the total base area A_{item} area indicates a stable configuration. The ratio is determined by:

$$r = \frac{A_{contact}}{A_{item}} \tag{3-6}$$

Similar to the contact stability method, the score is updated to account for the ratio and a weight β is assigned. A high value for the ratio indicates a higher probability of a stable configuration, therefore the ratio is subtracted from the score.

$$c \cdot (X + Y) + \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} H_c'[i, j] - \beta \cdot r \tag{3-7}$$

Experiments to determine values for α and β Before the final tests can be conducted for the stability measures, the weight values for α and β from Equations (3-5) and (3-7) respectively need to be defined. Therefore, Tables 3-5 and 3-6 show results with

Table 3-5: Results with the contact stability measure for different weights α

Weight α	Success (%)	Time (s)		# Planner calls	
		Mean	Max	Mean	Max
0	19	1,027.18	9,162	5.95	85
1	18	4,442.15	70,106	10.6	160
2	18	1,255.86	19,095	3.2	44
3	18	2,049.17	18,878	8.35	77
4	18	2,244.05	17,887	11.05	77

Table 3-6: Results with the ratio stability measure for different weights β

Weight β	Succes (%)	Time (s)		# Planner calls	
		Mean	Max	Mean	Max
0	19	1,027.18	9,162	5.95	85
1	19	801.32	6,966	6.45	94
5	19	785.29	6,642	6.15	88
10	19	770.78	6,661	6.05	88
20	19	1512.53	8,189	18.7	113

placement of six items with different weight values. 20 item sets are used to decrease the run time for these experiments.

From these experiments is concluded that the contact stability method performs for all values of α worse compared to the initial settings without the contact stability measure. The success rate is decreased and the computation time is increased when using the contact stability method. For this reason, the contact stability method is not considered a good indication for the stability and is therefore omitted for the next experiments.

The ratio stability method does not seem to improve the success rate. However, for the weights in this experiment the computation time is decreased. The weight β equal to 10 results in the lowest computation time for the experiments conducted. Therefore, this weight is used in further research.

3-3 Improvements to the online algorithm

Two improvements to the online part of the Quasi-Online algorithm are considered. The first improvement is conducted by applying knowledge of the upcoming items in the sequence. This method is called *lookahead* and is further explained in Section 3-3-1. The second improvement, called the *semi-deterministic* approach, is similar to the lookahead method. It depends on the knowledge of the upcoming items in the sequence

as well. However, a difference between the lookahead method and the semi-deterministic approach is the inclusion of the picking sequence in the semi-deterministic method. This approach is explained in Section 3-3-2.

3-3-1 Considering lookahead items

By adding lookahead items, only part of the policy tree as defined in Section 2-3-4 needs to be searched. This operation is expected to decrease the computation time of the Quasi-Online algorithm. The project environment allows for a number of the lookahead items. In this section, the theory of lookahead items is shown.

Theory of lookahead items

The theory of adding lookahead items or advice to an optimization problem is broadly researched [Böckenhauer et al., 2011, Boyar et al., 2016, Renault et al., 2015]. Böckenhauer et al. [2011] define a theorem to describe the online optimisation problem with advice, which is ultimately adapted by Boyar et al. [2016]. The definition by Boyar et al. [2016] is shown in Theorem 1.

Theorem 1. *The input is a sequence of items* $\sigma =< x_1, ..., x_n >$ *revealed one-by-one. The goal is to pack these items in the minimum number of bins of unit size. At time step* t*, an online algorithm should pack item* x_t *into a bin. An algorithm* $\mathcal{A}$ *is* c*-competitive with advice complexity* $s(n)$ *if there exists a constant* c_0 *such that for all* n *and for all input sequences* σ *of length at most* n*, there exists some advice* Φ *such that* $\mathcal{A}(\sigma) \leq cOPT(\sigma) + c_0$ *and at most the first* $s(n)$ *bits of* Φ *have been accessed by the algorithm. If* $c = 1$ *and* $c_0 = 0$*, then* $\mathcal{A}$ *is optimal.*

$OPT(\sigma)$ indicates the optimal solution, which is defined by the offline algorithm. Principally, the more lookahead items (or bits of advice) included in the algorithm, the closer the solution approaches the offline solution. Boyar et al. [2016] prove that when the number of items or sizes is not restricted, as is in this proposal, the bits of advice required to achieve the optimal solution is equal to $n \log OPT(\sigma)$. However, this result is based on a classical one-dimensional bin packing problem [Boyar et al., 2016].

Zhao et al. [2020] use lookahead items in their solution to the three-dimensional Bin Packing Problem (BPP) with Reinforcement Learning. Figure 3-8 shows the volume utilization of various values for the number of lookahead items. The volume utilization increases with the number of lookahead items until a plateau zone is reached. From this research is therefore concluded that the volume utilization does not increase indefinitely with increasing number of lookahead items.

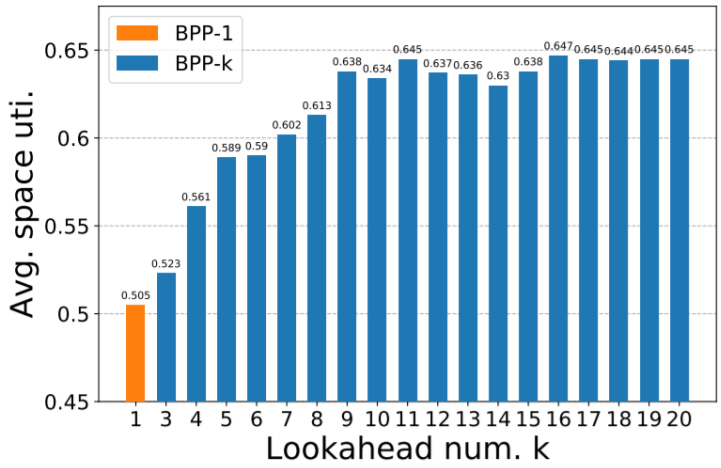


Figure 3-8: Packing performance using different values for lookahead [Zhao et al., 2020]

3-3-2 Applying semi-deterministic approach

As determined in Section 2-3-4, the Quasi-Online algorithm obtains the policy tree of all possible sequences of the given item set. By failure of one of the sequences, the packing of this item set fails. This method is called *non-deterministic*. However, when considering the picking sequence not all options in the policy tree are feasible and therefore these options do not have to be checked. In the project environment, these sequences will never occur and therefore do not have to be checked. The result is a decrease in the size of the policy tree and therefore a faster computation of the packing plans.

Theory of the semi-deterministic approach

The policy tree without any constraints, therefore a non-deterministic tree, considers $n!$ sequences where n is the total number of items to be placed. However, considering the picking sequence, which is explained in Section 1-1, numerous options are not available. The picking sequence considers that heavy items cannot be placed on top of fragile items for example. Therefore, the size of the policy tree is decreased.

In the following example, the package of oat flakes cannot be placed on top of the bananas or the soap detergent. The package of oat flakes has a picking sequence equal to 4, the soap detergent a picking sequence of 2 and the bananas a picking sequence of 1. The picking sequence of the package of oat flakes is not the same or a consecutive number of the other picking sequences, and are thus not interchangeable. Therefore, the oat flakes are always to be placed before the bananas and the soap detergent. The number of plans generated decreases from six to two plans.

No general rule can be determined from the semi-deterministic approach. This is due to the dependence on the values for the picking sequence of every item in the order.

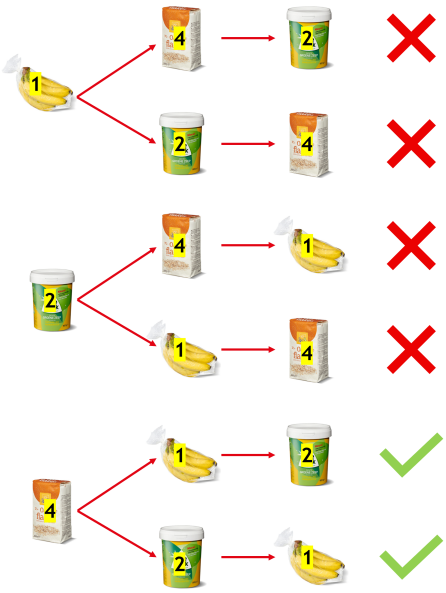


Figure 3-9: Example of a policy tree with three different items using the semi-deterministic approach

3-3-3 Obtaining the Picnic data set

The picking sequence of the items is Picnic-specific. Therefore, a data set needs to be obtained that contains models for Picnic articles. To obtain the Picnic-specific dataset, the object models from the YCB and the APC datasets are considered. The shapes of the top 100 sold Picnic items are considered and compared to the shapes in the dataset. Where they collide, the object models from the data set are scaled to align with the size of the Picnic items. In this way, the Picnic data set is obtained.

In Figure 3-10, four examples are shown to generate this data set. From each of the object models, the point clouds are changed to match the dimensions of the Picnic articles.

Figure 3-10: Four examples to generate picnic data set from object models

Using these Picnic articles, the picking sequence is tested. The picking sequence is predetermined by the fragility, size and contamination of the article. The value is obtained from a dataset and matched with the Picnic articles. Appendix D-2 shows the main characteristics of the obtained Picnic dataset.

3-4 Implementation of the benchmark

From Section 2-4-2 is concluded that the Quasi-Online algorithm can not yet be compared to other algorithms in the literature. To be able to compare the Quasi-Online algorithm with the benchmarks as defined in Section 2-4-2, the Quasi-Online algorithm needs to be rewritten to include the utilization rate.

From the two benchmarks described in Section 2-4-2, the second benchmark is chosen. The first benchmark is used specifically for container loading problems. This type of problem, although similar to the bin packing problem, uses a large size for the bin, or the container. The item dimensions are however relatively small compared to the container size, resulting in an absolute minimum of 18 items and a maximum of 3760 items to be placed. To obtain such a large policy tree is not feasible within reasonable time, resulting in the use of the second benchmark.

This benchmark places 50 items in a to-be-defined number of bins. To place this amount of items, the algorithm needs to be rewritten. As is shown in Section 2-3-4, the policy tree is determined with all the items to be placed. Therefore, with an increasing number of items to be placed the tree shows a factorial growth. This is computationally expensive and besides, not all 50 items will fit in one bin. Therefore, a maximum value for the number of items fitting in one bin is used to obtain the tree. Consequently a sliding window principle is used, fitting a fixed set of items in one bin. This is done until all the items are placed. This reduces the computation time significantly.

To summarize, the steps taken to use this benchmark are shown below.

- Obtain the maximum value for the number of items in one bin

- Fit the first set of items in one bin
- If there are items that are not fitted in one bin, add them to the next set
- Continue until all items are placed

In order to obtain the value for the maximum number of bins, a range of values is examined.

The results from this benchmark are shown in Chapter 4.

3-5 Conclusions

In this chapter, the Recreated Quasi-Online algorithm is discussed. The initial model is obtained using the data sets defined by Calli et al. [2015] and Rennie et al. [2016]. These data sets do not always provide complete object models, requiring a filtering of the data set.

The Quasi-Online algorithm is recreated with the use of voxels to define the shape of the items. The voxel size is determined using experiments. Following from these experiments, the voxel size of 0.67 centimeter results in the highest success rate as well as the lowest computation time. The results from this Recreated Quasi-Online algorithm are compared to the Quasi-Online algorithm obtained by Wang and Hauser [2020]. The main difference is in the computation time, which can be explained by the different hardware used in the two experiments.

After the intial model is obtained, several improvements are proposed. First, the improvements to the offline algorithm were explained. These improvements are required to increase the success rate of the Recreated Quasi-Online algorithm. These improvements are two-fold. First, the initial positions of each item are obtained using the Principal Component Analysis. The axes of rotation are used to align the item with the axes of the bin. The second improvement of the offline algorithm entails the replacement of the stability measure. The stability in the algorithm of Wang and Hauser [2020] is defined after the grid search. However, due to various reasons, a stability measure might be more optimal in the grid search. Three methods for this stability measure are proposed and two promising methods will be used in further experiments.

In addition to the improvements of the offline algorithm, two improvements of the online algorithm are proposed. These improvements are both based on decreasing the size of the policy tree, which is expected to result in a decrease in computation time. The first approach is by adding extra lookahead items, in which the information about the dimensions and the size of the next item are represented. Adding the lookahead items might increase the volume utilization as well. The second method to improve the online algorithm is based on the knowledge about the picking sequence. In adding the picking sequence, some sequences turn out to be infeasible. Therefore, the policy tree is reduced

as well. To include the picking sequence, a data set specific to Picnic articles is required. This data set is obtained by scaling the data sets from Calli et al. [2015] and Rennie et al. [2016] to Picnic article sizes.

Finally, a method is proposed to implement the benchmark, which is earlier described in Section 2-4-1. With the reduction of the policy tree to the number of items that fit in one bin, the benchmark can be applied to the algorithm. In the next section, these improvements are validated.

Chapter 4

Validation using numerical experiments

In this chapter, the improvements proposed in the previous section are validated using the YCP and APC data sets, and the Picnic data set obtained in Section 3-3-3. The improvements are first validated and subsequently, the results of the complete renewed Quasi-Online algorithm are shown, combining various improvements. Section 4-1 elaborates on these steps using the YCP and APC data sets. In Section 4-2 the results are shown using the Picnic data set. The last section of this chapter expands on the benchmark, as is explained in Section 3-4.

4-1 Results using the YCP and APC data sets

In this section, the results are shown using similar data sets compared to ones used by Wang and Hauser [2020]. As discussed in Section 3-1-1, the data sets are first filtered. The results are obtained using the parameters as proposed in Table 3-1. For each experiment, 100 item sets are generated and averaged. Section 4-1-1 shows the results using the implementation of the single improvements. After these improvements are implemented, an evaluation is done for each of the single improvements. In Section 4-1-2, the Principal Component Analaysis method, the ratio stability method and the implementation of one lookahead item are combined. The last improvement, the semi-deterministic approach, can be used with the Picnic data set exclusively, and is therefore not discussed in this section.

Table 4-1: Results of the Quasi-Online algorithm with PCA

	Items	Success (%)	Time (s)			# Planner calls	
			Mean	Max	Std	Mean	Max
Initial model	2	100	59.48	210	35.13	1.00	1
	3	99	90.86	317	47.25	1.00	1
	4	98	109.93	364	58.20	1.04	5
	5	95	192.57	1,999	229.84	1.52	25
PCA	2	100	85.30	337	51.15	1.01	2
	3	100	161.90	1,045	134.41	1.14	4
	4	96	359.59	2,180	377.67	1.47	7
	5	93	9,457.53	9,521	1,726	4.2	41

4-1-1 Improvement validation

In this section, the improvements proposed in Sections 3-2 and 3-3 are validated. Section 4-1-1 shows the results using the Principal Component Analaysis method, Section 4-1-1 elaborates on the two stability measures considered, in Section 4-1-1 the implementation of lookahead items is discussed. Section 4-1-1 evaluates the three implemented improvements.

Principal Component Analaysis

In Table 4-1, the results with the implementation of the Principal Component Analaysis (PCA) are shown, as well as the results of the initial model for comparison. As can be seen from this table, the success rate is similar with a small number of values when considering the PCA. Nonetheless, with a higher number of items the success rate is decreased. The reason is that the point clouds of various object models are not uniformly distributed. For example, some object models are double-walled. An example of a double-walled object model is shown in Figure 4-1. The increased amount in points at part of the item results in wrongly determined determined axes and thus do not improve the solution. For a higher number of items, however, the decrease in success rate is seen when considering the PCA method.

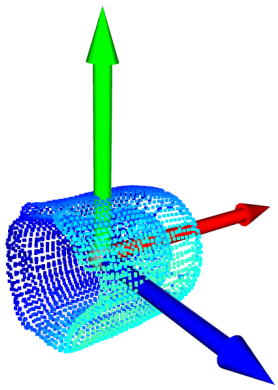


Figure 4-1: Point cloud of a double-walled item

A third interesting result from the PCA method is that it is able to place items, where the initial model gives a failure and vice versa. This can be seen from the comparison in Figure 4-2. In this experiment, six items are placed and the failure and success of each of the item sets is documented. Therefore, the two methods can be combined to result in the highest success rate. The results using the PCA method as fallback method are shown in Table 4-2. As can be seen from this table, the computation time increases when the PCA method is used as fallback method. The number of planner increases with five and six items placed. In these instances, the success rate increases as well. Therefore, the PCA method as fallback is considered a good method for improvement of the success rate.

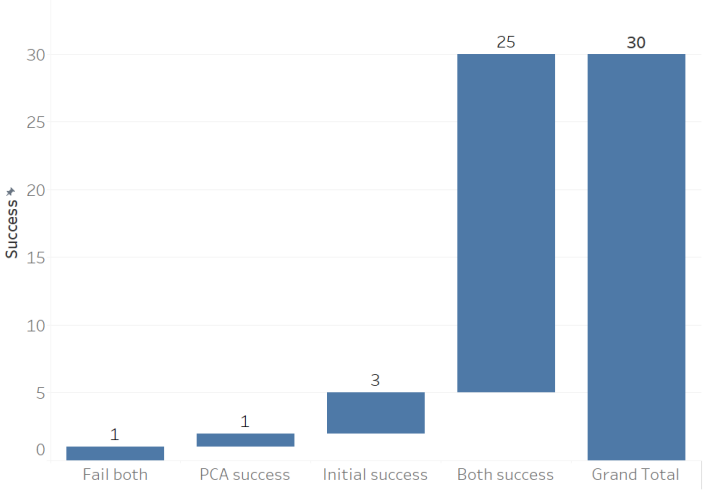


Figure 4-2: Success and failure of 30 item sets with placement of six items

Table 4-2: Results of the Quasi-Online algorithm with PCA as fallback method

	Items	Success (%)	Time (s)			# Planner calls	
			Mean	Max	Std	Mean	Max
Initial model	2	100	59.48	210	35.13	1.00	1
	3	99	90.86	317	47.25	1.00	1
	4	98	109.93	364	58.20	1.04	5
	5	95	192.57	1,999	229.84	1.52	25
	6	87	466.37	12,316	1,548.57	3.66	91
PCA as fallback	2	100	63.73	222	36.56	1.00	1
	3	99	103.73	394	48.27	1.00	1
	4	98	124.33	455	67.33	1.04	5
	5	97	229.74	3,664	456.58	1.80	25
	6	90	1,030.18	29,697	3711.39	7.31	186

Renewed stability measure

In Section 3-2-2, two methods to improve the success rate by adding the stability to the grid search are proposed. As described before, the first method, the equilibrium stability method, uses similar force and torque equilibrium compared to Wang and Hauser [2020] and places the measure in the grid search of the offline algorithm. The second method used is the ratio stability method. This method considers the ratio between the area in contact and the total area of the item as a stability measure. The results of these experiments are shown in Table 4-3, as well as the results of the initial model. As can be seen from this table, an increase in success rate is seen compared to the baseline for the equilibrium stability method. For an increasing number of items, the time to place the items increases significantly. The ratio stability method shows a smaller increase in success rate, but a decrease in computation time as well.

The results from the different stability methods are visualized in Figure 4-3, with *Initial* indicating the stability measure applied by Wang and Hauser [2019], *Equilibrium* the stability measure similar to the one of Wang and Hauser [2019] inserted in the grid search, and *Ratio* the stability indication of the area of contact in the grid search. As is expected, the equilibrium stability measure results in the highest success rate for all experiments. However, the computation time is the highest on average as well. The decrease in the computation time for the ratio stability method is apparent compared to the equilibrium stability method. The success rate of the ratio stability method is higher compared to the initial method. For these reasons, the ratio stability method is selected for the stability measure.

Table 4-3: Results of the recreated Quasi-Online algorithm with the stability methods

	Items	Success (%)	Time (s)			# Planner calls	
			Mean	Max	Std	Mean	Max
Initial model	2	100	59.48	210	35.13	1.00	1
	3	99	90.86	317	47.25	1.00	1
	4	98	109.93	364	58.20	1.04	5
	5	95	192.57	1,999	229.84	1.52	25
	6	87	466.37	12,316	1,548.57	3.66	91
Equilibrium stability	2	100	75.56	232	34.53	1.00	1
	3	99	102.77	321	42.86	1.00	1
	4	98	161.05	401	65.51	1.00	1
	5	95	225.03	1,941	134.88	1.14	15
	6	90	671.78	17,471	2,203.35	3.26	85
Ratio stability	2	100	73.42	272	38.59	1.00	1
	3	99	106.38	410	52.79	1.00	1
	4	98	134.71	466	64.48	1.04	5
	5	95	221.31	2,032	248.93	1.52	25
	6	89	326.06	8,009	1,021.03	5.31	111

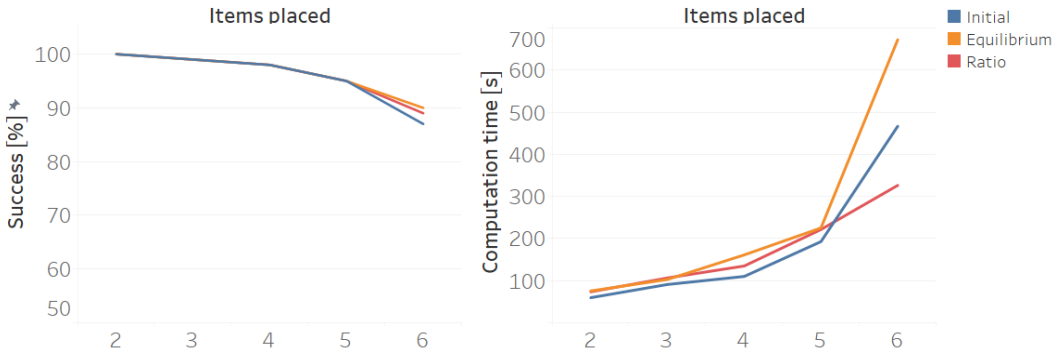


Figure 4-3: Comparison of the different stability measures

Lookahead items

In this section, the success rate and computation time of looking one or two items ahead are shown. The result is that the the dimensions and shape of the next item, and possibly of the item(s) thereafter are known. In Section 4-1-1, the results from looking one item ahead are shown. In Section 4-1-1, the results from looking two items ahead are shown.

Table 4-4: Results of the Quasi-Online algorithm with 1 item lookahead

	Items	Success (%)	Time (s)			# Planner calls	
			Mean	Max	Std	Mean	Max
Initial model	2	100	59.48	210	35.13	1.00	1
	3	99	90.86	317	47.25	1.00	1
	4	98	109.93	364	58.20	1.04	5
	5	95	192.57	1,999	229.84	1.52	25
	6	87	466.37	12,316	1,548.57	3.66	91
1 item lookahead	2	100	59.55	213	34.45	1.00	1
	3	99	90.80	318	46.46	1.00	1
	4	98	108.39	336	53.19	1.02	3
	5	95	176.88	1,106	125.00	1.22	11
	6	89	333.41	5,043	785.58	2.32	37

Implementation of one lookahead item The results from the addition of one lookahead item are shown in Table 4-4. The results are compared to the results of the baseline from Table 3-3. As can be seen, the difference is not obvious from the first few iterations with a smaller number of items. However, with an increasing number of items to be placed, the computation time per item set with the lookahead is decreased compared to the baseline. A clear difference is seen as well in the number of planner calls. With the implementation of the extra lookahead item, the number of planner calls decreases, because the policy tree is smaller and less combinations have to be checked.

Figures 4-4 and 4-5 show the time distribution in a box plot of 100 item sets for two, three, four, five and six items, respectively. As can be seen, for the experiments with two and three items placed, the whiskers overlap. The experiments with a larger number of items show a larger difference between the initial and the method with the lookahead items.

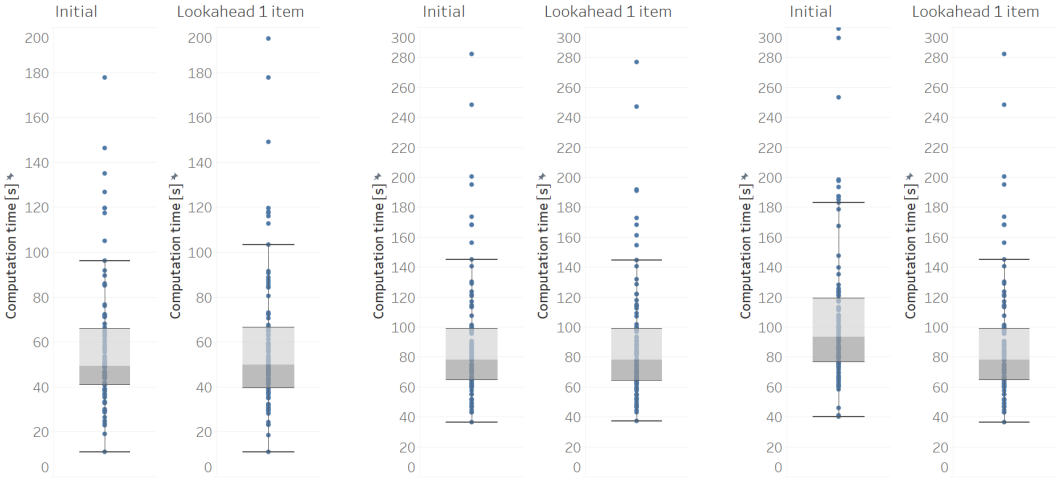


Figure 4-4: Time distribution of two, three and four items, respectively with initial (left) and 1 lookahead item (right)

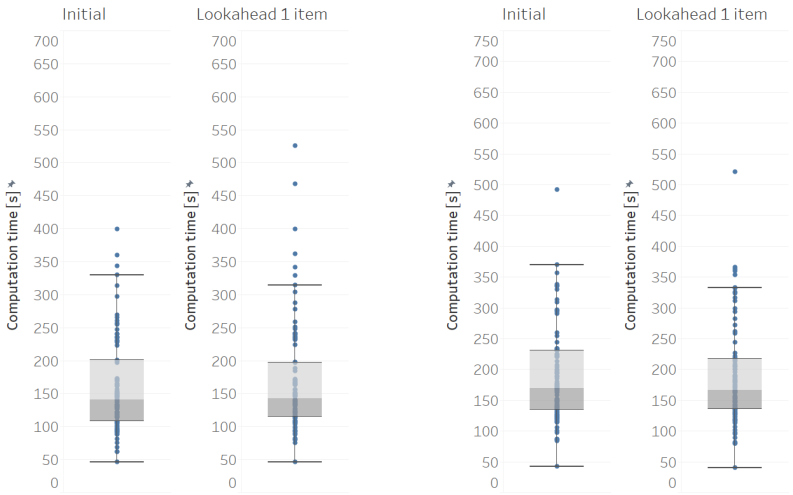


Figure 4-5: Time distribution of five and six items, respectively with initial (left) and 1 lookahead item (right)

Table 4-5: Results of the Quasi-Online algorithm with 2 items lookahead

	Items	Success (%)	Time (s)			# Planner calls	
			Mean	Max	Std	Mean	Max
Initial model	2	100	59.48	210	35.13	1.00	1
	3	99	90.86	317	47.25	1.00	1
	4	98	109.93	364	58.20	1.04	5
	5	95	192.57	1,999	229.84	1.52	25
	6	87	466.37	12,316	1,548.57	3.66	91
2 items lookahead	2	100	53.97	204	31.78	1.00	1
	3	99	93.79	362	50.101	1.00	1
	4	98	105.75	312	49.06	1.02	3
	5	95	150.05	371	70.66	1.10	5
	6	89	242.55	2,368	303.12	1.43	15

Implementation of two lookahead items Next to the addition of one lookahead item, two lookahead items are implemented. The results of this experiment are shown in Table 4-5. The computation time decreases even further with the addition of the second lookahead item, as well as the number of planner calls. The success rate stays the same compared to the initial model.

In Figure 4-6, the three methods are shown. From this figure is clearly seen that the initial model results in the highest computation time, followed by the method with one item lookahead. The experiment that considers two lookahead items features the lowest computation time. Figures 4-7 and 4-8 show the time distribution of the two, three, four, five and six placed items. From these plots the decrease in whisker size for each of the number of items placed can be seen.

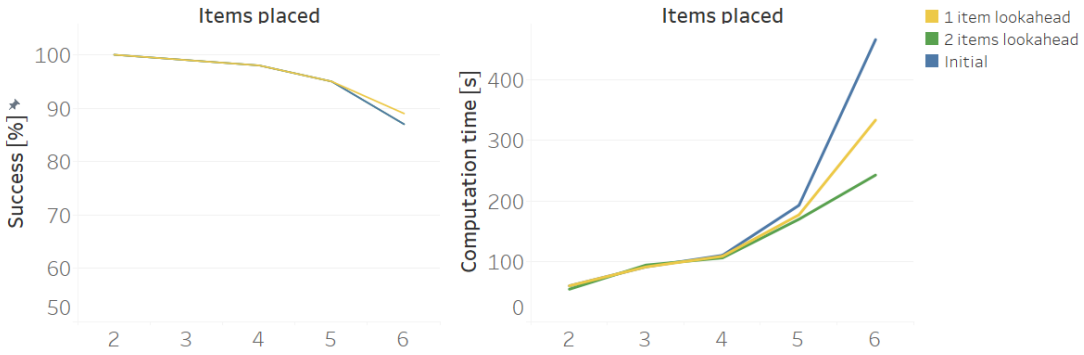


Figure 4-6: Initial, one item lookahead and two items lookahead compared on success rate (left) and computation time (right)

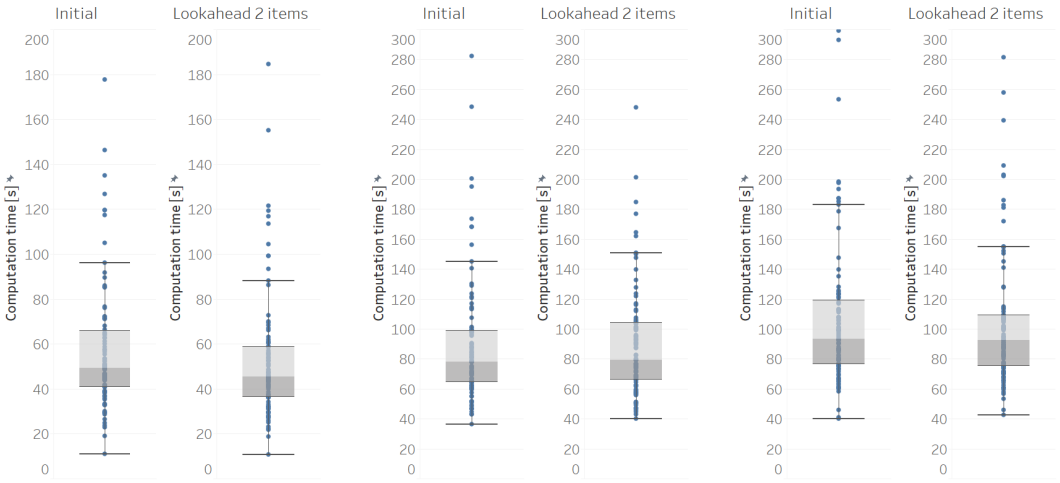


Figure 4-7: Time distribution of two, three and four items, respectively with initial (left) and 2 lookahead items (right)

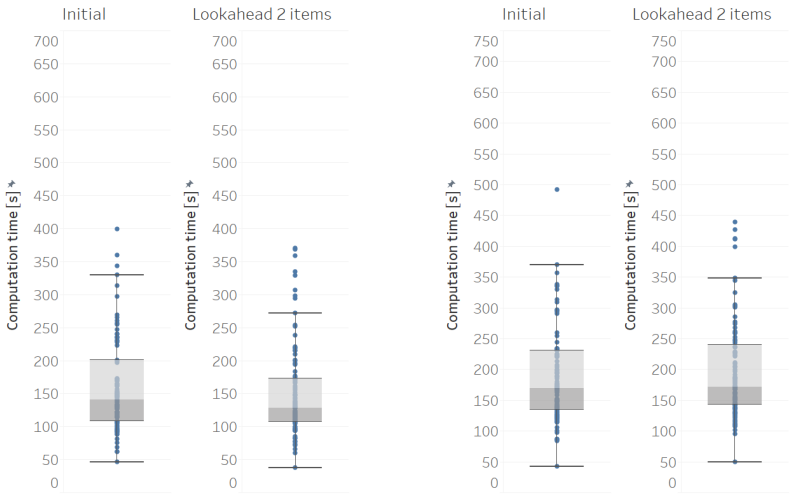


Figure 4-8: Time distribution of five and six items, respectively with initial (left) and 2 lookahead items (right)

Evaluation of the improvements

In the previous sections, three improvements to the Quasi-Online algorithm were implemented, and the results of each of the improvements were reported and compared to the baseline. From this comparison can be concluded that the PCA method as fallback procedure, the lookahead and the semi-deterministic approach result in a faster computation or a higher success rate. From the two stability methods, the equilibrium stability method results in a higher success rate on the cost of a high computation time, whereas the ratio stability method results in a lower computation time and an increase in success rate compared to the initial model as well. Due to the large increase in computation time for the equilibrium stability method, the ratio stability method is chosen for the implementation. The improvements are implemented together and tested against the baseline. The results are documented in the next section.

4-1-2 Results from Experiment I

In this section is elaborated on the experiments with the PCA method as fallback procedure, the ratio stability method and one lookahead item included, called Experiment I, using the YCP and APC data sets. First, the experimental set-up is explained. Subsequently, the results are shown.

Experimental set-up

The same parameter settings shown in Table 3-1 are used for this experiment. As concluded from Section 3-1-4, the voxel size of 0.67 centimeters is used. Furthermore, the PCA method as fallback procedure from Section 3-2-1 is chosen for the item rotation. As described in Section 3-2-2, the updated score from Equation (3-7) is used with a weight value β set equal to 10. Finally, one lookahead item is added.

Results

In Table 4-6, the results from Experiment I are reported. The computation times for the item placement are slightly higher compared to the baseline. The increase can be explained by the increase in computation time due to the PCA method as fallback procedure. An increase in success rate is seen for the placement of five items. Furthermore, the number of planner calls is decreased compared to the baseline. Appendix C-1 shows the first four packing plans with the corresponding packed items with six items placed.

4-2 Results using the Picnic dataset

This section elaborates on the implementation of the improvements using the obtained data set from Section 3-3-3. The data set is obtained by scaling similarly shaped object

Table 4-6: Results with one lookahead item, the ratio stability method and PCA method

Items	Success (%)	Time (s)			# Planner calls	
		Mean	Max	Std	Mean	Max
2	100	74.29	225	41.52	1.00	1
3	99	116.44	439	58.87	1.00	1
4	98	157.55	480	75.74	1.02	3
5	97	283.26	2,419	398.90	1.34	11

models from the YCB and APC data sets to the desired item dimensions. A data file is than called to combine the values for the picking sequence with the correct articles. The Picnic articles are chosen to be the best sold items. The characteristics of the items used in the Picnic data set are shown in Appenix D-2.

In Section 4-2-1, the four improvements shown in Chapter 3 are discussed, as well as the baseline using the Picnic data set. Similar to the previous chapter, Section 4-2-2 elaborates on the three combined improvements, the PCA method, the ratio stability method and the addition of one lookahead item, called Experiment I. In Section 4-2-3, the four improvements are implemented, including the semi-deterministic approach, called Experiment II.

4-2-1 Improvement validation

In this section, the single improvements are validated using the Picnic data set. In Section 4-2-1, the baseline is shown using the Picnic data set. Section 4-2-1 elaborates on the implementation of the PCA method, Section 4-2-1 shows the ratio stability method, Section 4-2-1 the addition of lookahead items and Section 4-2-1 the semi-deterministic approach all using the Picnic data set. Section 4-2-1 evaluates the proposed improvements using the Picnic data set.

Baseline

The results from the experiment with the Picnic data set without improvements are shown in Table 4-7. This experiment is used as baseline for the experiments in this section using the Picnic data set. Compared to the implementation without the Picnic data set, the success rate is lower. This can be explained by the increase in average size of the items used in the Picnic data set.

Principal Component Analaysis

In the previous section is decided to implement the PCA method determined in Section 3-2-1 as a fallback method. In this section, the PCA method with the Picnic data set is

Table 4-7: Results of the Quasi-Online algorithm with Picnic data set

Items	Success (%)	Time (s)			# Planner calls	
		Mean	Max	Std	Mean	Max
2	97	146.44	602	87.70	1.02	2
3	86	211.34	622	96.57	1.04	3
4	65	331.25	1,514	281.20	1.56	9
5	33	604.23	9,472	1,223.95	1.49	25

Table 4-8: Results of the Quasi-Online algorithm with Picnic data set and PCA as fallback

	Items	Success (%)	Time (s)			# Planner calls	
			Mean	Max	Std	Mean	Max
Initial model	2	97	146.44	602	87.70	1.02	2
	3	86	211.34	622	96.57	1.04	3
	4	65	331.25	1,514	281.20	1.56	9
	5	33	604.23	9,472	1,223.95	1.49	25
PCA as fallback	2	97	172.65	844	115.14	1.02	2
	3	87	231.25	782	231.25	1.04	3
	4	66	381.82	1,848	381.82	1.56	9
	5	33	659.90	9,462	1,218.90	1.50	25

implemented. Table 4-8 shows the results of this implementation. As can be seen from this table, the success rate increases with three and four items placed. The number of offline planner calls stays similar to the initial model, however the computation time increases using the PCA method as fallback procedure. On average, the same effect is seen compared to the use of the YCP and APC data sets. The success rate increases, as well as the computation time.

Renewed stability measure

In Section 4-1, a comparison has been made between the equilibrium stability method and the ratio stability method. From these experiments is concluded that the ratio stability method is the preferred method, and therefore in this section this method is tested using the Picnic data set. In Table 4-9, the results using the ratio stability method with the Picnic data set are shown. With increasing number of items placed, an increase in planner calls is seen. For the experiment with placement of five items, the computation time is reduced compared to the initial model. Therefore, the time in the offline planner is reduced significantly. Furthermore, the results show an increase in success rate.

Table 4-9: Results of the Quasi-Online algorithm with Picnic data set and ratio stability method

	Items	Success (%)	Time (s)			# Planner calls	
			Mean	Max	Std	Mean	Max
Initial model	2	97	146.44	602	87.70	1.02	2
	3	86	211.34	622	96.57	1.04	3
	4	65	331.25	1,514	281.20	1.56	9
	5	33	604.23	9,472	1,223.95	1.49	25
Ratio stability	2	97	173.37	596	89.65	1.02	2
	3	86	315.30	858	137.87	1.07	3
	4	65	450.37	2,142	373.87	1.56	9
	5	34	575.31	6,860	998.93	1.95	25

Lookahead items

The experiment for the lookahead items using the Picnic data set is conducted for both one and two lookahead items. The results are shown in Table 4-10. From this table is seen that in the first two experiments, with two and three items placed, the computation time increases using the lookahead items. A potential reason for this is the increase in success rate. When the placement of a set of items is infeasible, the iteration is immediately terminated. This results in a lower computation time when the success rate is lower. The placement of five items shows an increase in success rate as well as a substantial decrease in computation time. The offline planner calls are reduced as well, which is expected because of the reduction of the policy tree.

Semi-deterministic approach

The results with the semi-deterministic approach are shown in Table 4-11. Especially in the placement of the two items, a big difference is seen between the experiment with and without the semi-deterministic approach. The number of planner calls increases with the semi-deterministic approach. The maximum number of planner calls stays similar however, only differing by one planner call. A potential reason for the increase in average planner calls is that in some cases in might be difficult to find a stable configuration of fragile items on top of other items. Since the semi-deterministic approach includes the fragility, and cannot rearrange the products such that the fragile items are one the bottom, it might be harder to find a stable packing plan.

In Figures 4-9 and 4-10, the time distribution of 100 item sets for two, three, four and five items using the Picnic data base is shown. The difference between the initial and the semi-deterministic approach less apparent when looked at these figures. The results with the semi-deterministic approach have a lower maximum value in the placement of

Table 4-10: Results of the Quasi-Online algorithm with Picnic data set and lookahead items

	Items	Success (%)	Time (s)			# Planner calls	
			Mean	Max	Std	Mean	Max
Initial model	2	97	146.44	602	87.70	1.02	2
	3	86	211.34	622	96.57	1.04	3
	4	65	331.25	1,514	281.20	1.56	9
	5	33	604.23	9,472	1,223.95	1.49	25
1 item lookahead	2	99	173.37	596	72.24	1.02	2
	3	86	235.03	738	105.52	1.03	2
	4	66	392.29	1,234	205.80	1.30	5
	5	33	458.00	1,668	270.66	1.36	12
2 items lookahead	2	99	164.01	351	72.91	1.00	1
	3	87	233.15	737	103.44	1.01	2
	4	67	369.78	1,069	160.59	1.23	3
	5	34	436.89	1,336	207.12	1.28	7

five items. However, the spreading of the data is bigger compared to the initial model. Looking at Figure 4-10, a connection can be seen between the increase in success rate and the spread of the data. The model with an increased success rate seems to have larger whiskers, therefore iterates more times in order to obtain the higher success.

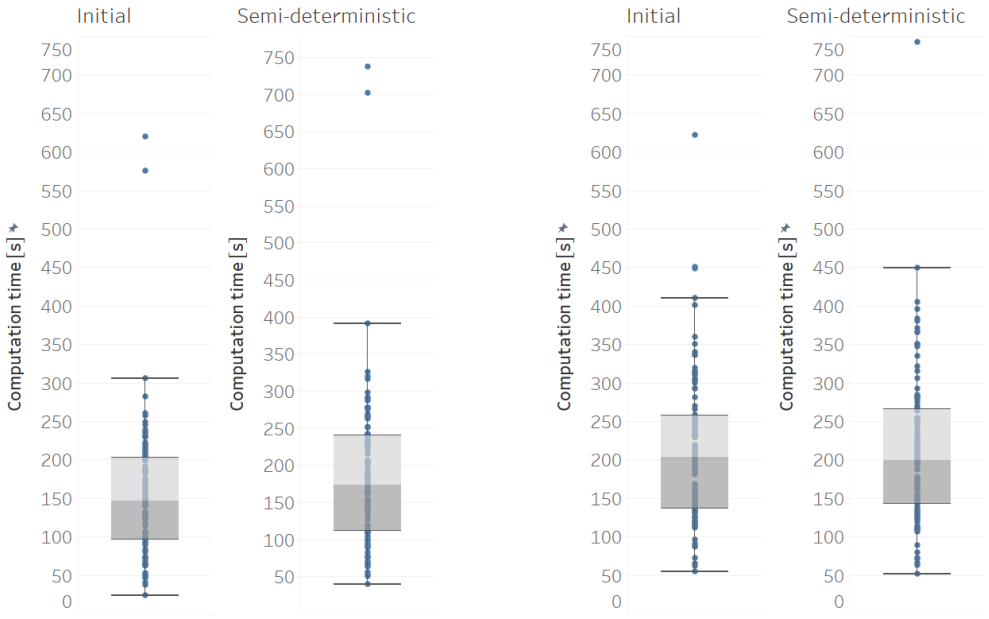


Figure 4-9: Time distribution of two and three items, respectively, initial (left), semi-deterministic (right)

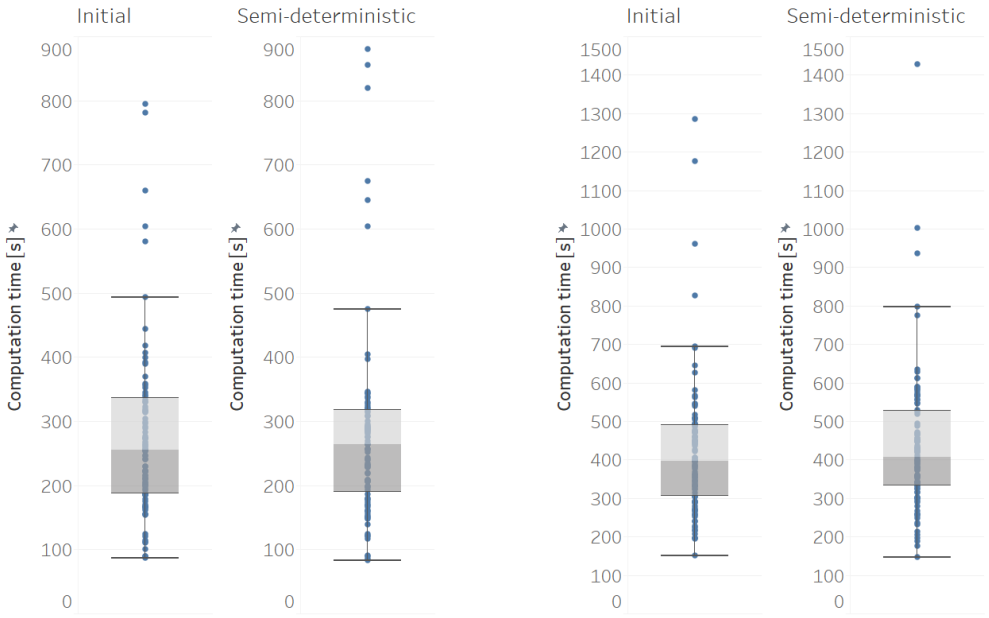


Figure 4-10: Time distribution of four and five items, respectively, initial (left), semi-deterministic (right)

Table 4-11: Results of the Quasi-Online algorithm with Picnic data set and semi-determinism

	Items	Success (%)	Time (s)			# Planner calls	
			Mean	Max	Std	Mean	Max
Initial model	2	97	146.44	602	87.70	1.02	2
	3	86	211.34	622	96.57	1.04	3
	4	65	331.25	1,514	281.20	1.56	9
	5	33	604.23	9,472	1,223.95	1.49	25
Semi-deterministic	2	95	181.09	711	94.78	1.03	2
	3	86	213.34	743	102.36	1.04	3
	4	62	337.24	1,651	305.46	1.49	8
	5	35	502.51	4,977	755.51	1.90	26

Evaluation of the improvements

Similar to the experiments of the single improvements with the YCP and APC data sets, the Picnic data set including the PCA method as fallback procedure increases the success rate as well as the computation time, the ratio stability method increases the success rate and the lookahead items increases the success rate and decreases the computation time. Using the Picnic data set, the ratio stability method decreases the computation time. The semi-deterministic approach is shown to increase the computation time with a small number of items placed, but decreases the computation time with a larger number of items placed. As a result, all improvements have important advantages and are therefore considered to be significant improvements to the algorithm by Wang and Hauser [2020].

4-2-2 Results with Experiment I

In this section, the PCA method as fallback procedure, the ratio stability method and the addition of one lookahead item are combined. The first section elaborates on the experimental set-up, and Section 4-2-2 shows the results of the implemetation.

Experimental set-up

In this Section, the Picnic data set, as described in Section 3-3-3 is used. Three of the improvements are combined: the PCA method as fallback, the ratio stability method and one lookahead item. These methods have been shown in the previous section to decrease the computation time and/or increase the success rate. The parameters used are, similar to the other experiments, shown in Table 3-1. 100 item sets of each instance are used to obtain the results.

Table 4-12: Results with the Picnic data set with PCA as fallback, one lookahead item and the ratio stability method

Items	Success (%)	Time (s)			# Planner calls	
		Mean	Max	Std	Mean	Max
2	99	147.49	286	65.22	1.00	1
3	87	268.65	1,008	129.46	1.05	2
4	67	453.30	1,522	279.46	1.30	5
5	34	640.42	3,596	548.79	1.44	9

Results

In Table 4-12, the results using the Picnic data set with Experiment 1 are shown. Compared to the results without the lookahead, ratio stability and PCA methods, as shown in Table 4-7, an increase in success rate for the placement of five items is seen. Furthermore, similar to the experiment with the YCP and the APC data sets, the computation time is increased for all experiments. The number of planner calls has decreased, which results in that the computation time is increased due to the changes in the offline planner.

4-2-3 Results with Experiment II

In this section, the semi-deterministic approach, as described in Section 3-3-2 is added to the experiment conducted in the previous section. First, the experimental set-up is discussed. Subsequently, the results from the two experiments are shown.

Experimental set-up

The experimental set-up is similar to Experiment I, as discussed in the previous section. In this experiment, next to the three implemented improvement, the semi-deterministic approach is implemented as well. This entails that the picking sequence constraint is added to the problem.

Results

Table 4-13 shows the results using the Picnic data set with the addition of the semi-deterministic approach. Appendix C-2 shows the first four iterations from this experiment using five items. Compared to the experiment using the Picnic data set without the semi-deterministic approach, as shown in Table 4-12, the decrease in computation time of the placement of five items is seen. For the same reason of the increase in offline planner calls in Section 4-2-1, in some configurations it is potentially infeasible to find a packing plan where the fragile items are placed last. Therefore, this can decrease the

Table 4-13: Results with the Picnic data set with the PCA method as fallback, the ratio stability method, one lookahead item and the semi-deterministic approach

Items	Success (%)	Time (s)			# Planner calls	
		Mean	Max	Std	Mean	Max
2	96	189.89	482	85.78	1.00	1
3	88	246.13	632	102.36	1.04	2
4	65	387.30	1,905	273.49	1.26	5
5	38	565.58	3,434	538.24	1.60	12

success rate as is seen for the placement of two items. The decrease in offline planner calls is explained by the smaller policy tree due to the lookahead item.

Figures 4-11 and 4-12 show the time distribution of the results of Experiment I using the Picnic data set and Experiment II with the Picnic data set. As can be seen, the implementation of the semi-deterministic approach results in a lower computation time for all number of items placed, except for the placement of two items. With an increase in the number of items placed, the decrease in computation time is more obvious.

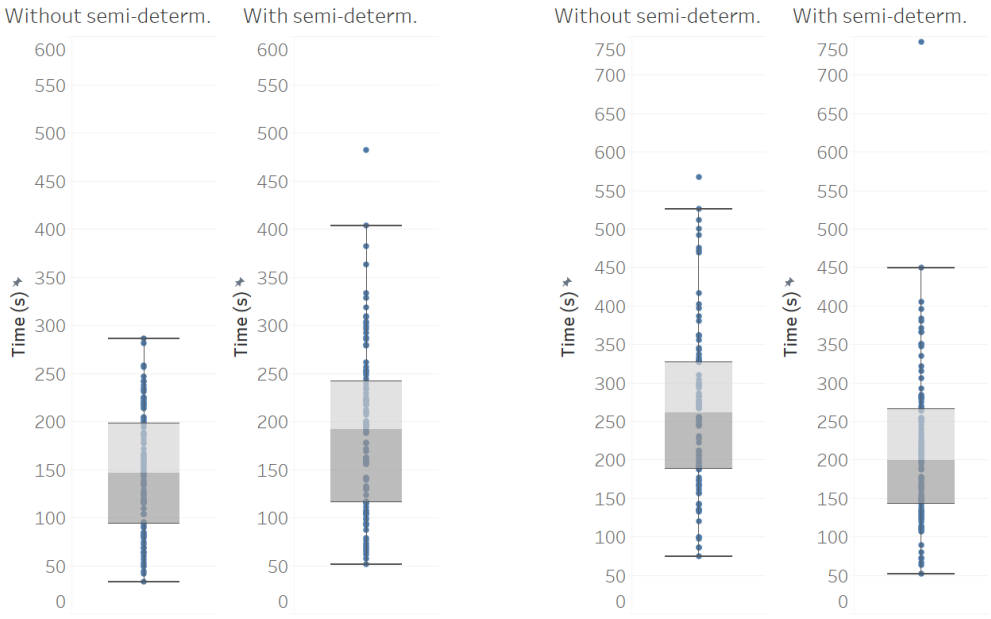


Figure 4-11: Time distribution of two and three items, respectively, initial (left), semi-deterministic (right)

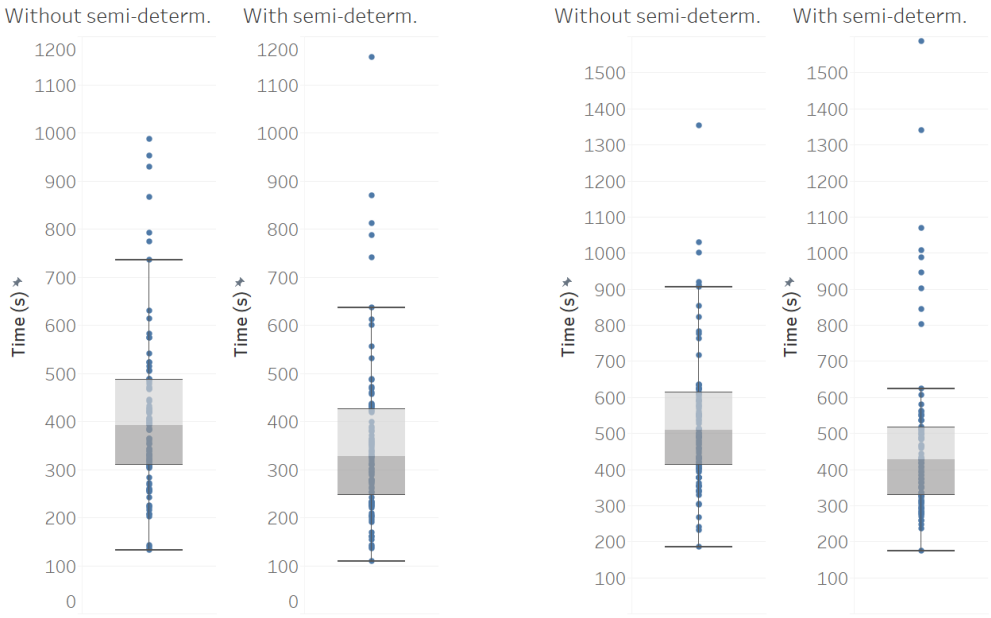


Figure 4-12: Time distribution of four and five items, respectively, initial (left), semi-deterministic (right)

Figure 4-13 shows the initial model without any improvements, the model without the semi-deterministic approach (Experiment I) and the model with the semi-deterministic approach (Experiment II). From this figure is seen the slight success rate increase at the placement of five items. Furthermore, the decrease in computation time of the method with the semi-deterministic approach is apparent.

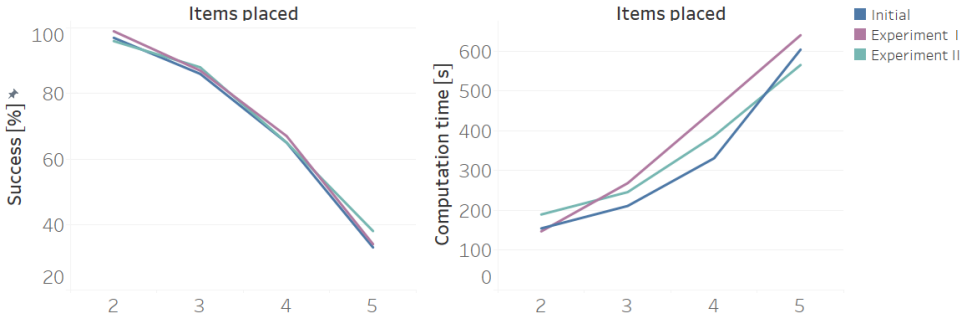


Figure 4-13: Three methods including the picnic data set, the initial, without semi-deterministic approach and with the semi-deterministic approach.

4-3 Results using benchmark by Martello et al. [2000]

In this section, the benchmark as described in Section 2-4-1 is implemented. First, the experimental set-up of the benchmark is described. Subsequently, the results of the algorithm using the experimental settings are shown.

4-3-1 Experimental set-up

In the researches using the benchmark, rectangular items are used. Furthermore, the items are placed orthogonally in the bin. For this reason when testing the improved Quasi-Online algorithm with the benchmark, rectangular items are used and the angles of rotation are changed to $\Delta r = [0, 0.5\pi]$. For each of the classes, the maximum value for the number of items in one bin needs to be defined, as explained in Section 2-4-2. This value is obtained incrementally. When the increase in the maximum value for the number of bins results in no improvements, the value is set to the first occurance of the optimal number of bins.

To obtain the item sizes used in this experiment, the item generator available at `http://hjemmesider.diku.dk/~pisinger/codes.html` is used. The item generator generates the values for the width, length and height using the classes described in Section 2-4-1. For each of the different classes, 10 instances with 50 items are generated. The total number of bins used is averaged over these 10 instances. The first instance of each class is shown in Appendix D-3.

Table 4-14: Results from the benchmark shown Quasi-Online (QO) and the lower bound (L_2) by Martello et al. [2000]

Class	Bin size (L x B x H cm)	Item size (% bin volume)	n_{bin}	n_{tot}	QO	L_2
1	100 x 100 x 100	0.444 - 50 %	9	50	17.5	12.5
4	100 x 100 x 100	12.5 - 100 %	4	50	32.9	28.7
5	100 x 100 x 100	0.0001 - 12.5%	6	50	14.5	7.3
6	10 x 10 x 10	0.0001 - 0.1 %	8	50	15.8	8.7
7	40 x 40 x 40	0.0001 - 4.3 %	7	50	11.6	6.3
8	100 x 100 x 100	0.0001 - 100 %	6	50	14.9	8.0
Total	bins				107.7	71.5

4-3-2 Results

In Table 4-14, the results from the benchmark with the different classes are shown. In the first column, the class is defined. The container dimensions and the item sizes for each class is explained in Section 2-4-1. n_{bin} indicates the maximum value for the number of items chosen in one bin and n_{tot} the total amount of items placed. The average number of bins used in the solution is shown in column QO. The last column shows the lower bound obtained by Martello et al. [2000].

4-4 Conclusions

In this section, the numerical results from all improvements described in Chapter 3 are shown. First of all, the method with the Principal Components Analysis (PCA) is implemented. The advantage of using the PCA method in the Quasi-Online algorithm is not directly apparent. However, when considering the method as a fallback method for the quasi-static orientations, the success rate increases.

From the two stability methods, the equilibrium stability and the ratio stability method, the equilibrium stability method performs slightly better regarding the success rate. However, the computation time for the equilibrium stability method is significantly higher compared to the ratio stability method. The difference in computation time between the ratio stability method and the equilibrium stability method is substantial and does therefore outweigh the increased success rate of one percent compared to the initial model. Therefore, the ratio stability method is chosen for further experiments.

The implementation of one and two lookahead items shows to reduce the time for the item placement significantly. For a low number of items placed, the decrease in time is not apparent. However, with the placement of five and six items, the computation time reduces. The decrease in computation time of the addition of two lookahead items compared to one lookahead item suggests that more lookahead items reduce the computation time even further.

The semi-deterministic approach reduces the computation time as well compared to the non-deterministic approach for a large number of items placed. However, in some cases the items are not able to be placed using the picking sequence therefore the success rate is not in all cases increased. Nonetheless, the semi-deterministic approach assures that the algorithm is able to be implemented in the environment of the automated warehouse.

The results with the implementation of the lookahead, the PCA and the ratio stability method (Experiment I) shows similar results compared to the baseline. However, the computation time is increased. The implementation of the last improvement, the semi-deterministic approach, (Experiment II with the Picnic data set) shows a clear decrease in computation time as well as a increase in the success rate for a large number of items placed. These results are valuable concerning the implementation of the Bin Packing algorithm in the automated warehouse. The proposed improvements show to have an increased performance in computation time as well as success rate, and should therefore be used for the packing algorithm of the automated warehouse.

The implementation of the benchmark was successful with the use of the sliding window method. However, to the author's knowledge there are no online 1-bounded algorithms that use this benchmark. Therefore, the results from the benchmark cannot be used to compare to existing literature.

Chapter 5

Conclusions and recommendations

5-1 Summary

In this thesis, the Quasi-Online algorithm by Wang and Hauser [2020] is recreated, improved and tested against an available benchmark from the literature. The robot picking environment where the packing algorithm is operating in provides challenges that must be tackled. The bin needs to be filled with the highest volume utilization reachable within reasonable time considering a partly unknown sequence of the items arriving at the picking station. The items packed are grocery items, therefore are irregularly shaped. The irregularly shaped items in combination with the fragility of the items must result in a stable and secure configuration such that no items are damaged.

In Chapter 2, the generic Bin Packing problem as well as the different characteristics and constraints applicable to the problem are discussed. Three types of Bin Packing algorithms are defined: offline, online, and quasi-online algorithms. Compared to online algorithms, offline algorithms result in a higher volume utilization. Offline algorithms can not be used in a robot picking environment, due to the unknown item sequence. Therefore, the third type, the quasi-online algorithm, is of interest. Until recently, this type is not elaborately researched in comparison to the other two types.

Different approximation algorithms are defined and compared against one another. Four groups of algorithms are defined: heuristics, metaheuristics, reinforcement learning and quasi-online algorithms. Only approximation algorithms are considered since the problem is NP-hard and therefore cannot be solved exactly within polynomial time. The algorithms are compared using two benchmarks, one benchmark by Martello et al. [2000] and one by Bischoff and Ratcliff [1995], in Section 2-4-2. Not all algorithms can be compared using one of these two benchmarks, as the experiments conducted do not use any commonly used benchmarks. The Quasi-Online algorithm by Wang and Hauser [2020]

in combination with the Heightmap-Minimization heuristic by Wang and Hauser [2019] use irregular items, a stability measure and are able to be applied to the system as is desired. Therefore, this algorithm resembles the project environment the most. However, no conclusions concerning the volume utilization can be made from these algorithms as the experiments used are based on the *success rate*. From the algorithms that can be compared the Hybrid Simulated Annealing algorithm by Peng et al. [2009] seems to be the one resembling the constraints and characteristics describes the most and result in the highest volume utilization.

Unlike researches conducted until recently, the Quasi-Online algorithm can be used in automated warehouses. However, various important characteristics lack from the algorithm developed by Wang and Hauser [2020]. First of all, the algorithm is tested against the success rate. In an automated warehouse, the resulting decision needs to always be a success as all items need to be packed into bins. Therefore, this metric is not generally useful for automated warehouses. Furthermore, the computation time of the algorithm by Wang and Hauser [2020] increases rapidly with the number of items used. Besides, the picking sequence is not added in the Quasi-Online algorithm.

In Chapter 3, the Quasi-Online algorithm is adapted to consider the picking sequence and to decrease the computation time while maintaining a high success rate. Before any of these adaptations are made to the algorithm, the irregular shape of the items first needs to be defined. Since Wang and Hauser [2020] do not describe how the irregular shapes of the algorithms are defined, an analysis on the different shape determination methods is conducted. This analysis demonstrates that the method of voxelization of the items is the preferred option the computation time as well as the accuracy. This is due to the slightly rotated initial orientation of the item. The axes of rotation are found using the Principal Component Analysis and the item is subsequently rotated such that a flat surface is found. To increase the success rate further, three different stability methods are added to the grid search and were tested against one another.

The second part of Chapter 3 shows the adaptations to the online part of the Quasi-Online algorithm. Considering the robot picking environment, due to a queue at the picking station the dimensions and shapes of at least the next item and the item thereafter is known. Therefore, the policy tree can be decreased using this knowledge. The last adaptation to the algorithm is the picking sequence. The Quasi-Online algorithm ensures that the item sequence is completely unknown. However, in the automated warehouse the picking sequence imposes constraints to the input item sequence. Therefore, with adding the picking sequence to the algorithm, the policy tree can be decreased as well. To be able to add the picking sequence constraints to the data set, a Picnic specific data set is obtained using the object models from the YCB and APC data sets. The resulting algorithm decreases the computation time.

Finally, a benchmark is added to be able to compare the obtained algorithm to literature. The results with the use of this benchmark show the amount of bins required for an order. The supply chain can be adapted to these results accordingly.

5-2 Conclusions

This research elaborates on the possibility and extension of the implementation of a Quasi-Online Bin packing algorithm in a robot picking environment.

First of all, different Bin Packing algorithms have been considered. The Quasi-Online algorithm is chosen as a sufficient baseline due to the compatibility with most characteristics of the system in the project environment. However, this algorithm can not be directly implemented in a robot picking environment due to various reasons. First of all, the computation time of the Quasi-Online algorithm by Wang and Hauser [2020] is substantial and increases with the number of items. This results in an undesirable high computation time when placing ten items or more. Secondly, the algorithm does not consider the fragility of the items, resulting in an incompatible algorithm in the project environment. Lastly, the algorithm by Wang and Hauser [2020] is tested on success rate exclusively. For the operation within Picnic, it is important to know and to reduce the volume utilization.

To address these problems, first the algorithm by Wang and Hauser [2020] was recreated. The explanation from Wang and Hauser [2020] on the high computation time indicated that it was caused by the high dimensional mesh models for the item sizes. Therefore, first the input to the algorithm was tackled. Voxelization was chosen as the preferred method to define the item shapes. Therefore, this method is used to implement the item shapes of the irregular grocery items. Due to the use of the different hardware the results of the voxelization cannot be directly compared to the results of the algorithm by Wang and Hauser [2020]. Therefore, additional improvements regarding the computation time were considered and compared to the recreated Quasi-Online algorithm with the added voxelization.

Four improvements were considered. The first improvement is the use of the Principal Component Analaysis (PCA) to obtain the principal axes of the item for the initial item orientation. Compared to the quasi-static orientation implementation, the quasi-static orientation results in more stable item orientations. The use of the PCA algorithm as a fallback procedure increases the success rate of the algorithm. Therefore, the PCA algorithm is further implemented as fallback method. The second improvement is regarding the stability measure. To make sure only stable placement options of the item are considered, the stability measure is moved to first assess the stability of the possible item placement. Three methods for the assessment of the stability are considered. One method is immediately disregarded due to the lower success rate. The optimal stability measure however results in a high computation time, therefore an indication for the stability using contact points is chosen. This method increases the success rate and decreases the computation time compared to the initial model.

The last two improvements are both based on reducing the policy tree which is the basis of the Quasi-Online algorithm. The reduction of the policy tree is first addressed by using the knowledge of the next item that is to be placed, called the lookahead item. This is shown to reduce the policy tree considerably, especially when using a large amount

of items to be placed. Looking two items ahead reduces the policy tree even further. The last improvement that is considered is, besides the reduction of the policy tree, an improvement that makes the algorithm compatible with the project environment. The fragility of the items is considered, giving each of the items a picking sequence value between one and five. To be able to test this method, a data set with grocery items is developed. Similar to the method where the knowledge of the next item is implemented, the addition of the picking sequence decreases the computation time as well by decreasing the policy tree. In some instances, however, a slight decrease in success rate is observed.

The improvements are thereafter tested simultaneously. For these tests, one lookahead item is implemented and the PCA method is used as fallback procedure. First, the methods are tested with a similar data set compared to Wang and Hauser [2020]. This implementation results in a higher success rate compared to the baseline, however the computation time increases as well. The last improvement, the addition of the picking sequence, can however not be implemented using this data set. For this reason, two additional experiments were conducted. The first experiment uses all improvements without the addition of the picking sequence with the Picnic data set. The second experiment includes the picking sequence as well. The result with all improvements included result in a decreased computation time and increased success rate for the placement of five items compared to the initial method. This shows that the implemented improvements, the PCA method as fallback procedure, the ratio stability method, the lookahead item and the semi-deterministic approach results in a faster and more success compared to the initial method.

The last remark of the Quasi-Online algorithm by Wang and Hauser [2020] is regarding the measure the algorithm is tested on. The success rate is used as a measure, however regarding the operation in an automated warehouse, the amount of bins used is a more useful measure. Therefore a benchmark is used to define the number of bins used when 50 items are to be placed. Obtaining the policy tree with 50 items results in a high computation time. Therefore, an alternative has been found in the form of a sliding window to assess the number of bins. This method is found to be successful, although currently the results cannot be compared to other algorithms using the same benchmark because of the boundedness and the type of the algorithm used.

In conclusion, the Quasi-Online algorithm can be used in a robot picking environment. The algorithm can be used to incorporate additional constraints fairly easily. Currently, the computation time of the algorithm is rather high to be used directly in the automated warehouse. However, the possibilities are shown to decrease the computation time by e.g. implementing lookahead items or adding the semi-deterministic approach. Furthermore, with the implementation of the algorithm in an automated warehouse, a virtual machine, similar to one used by Wang and Hauser [2020], should be used.

5-3 Discussion and recommendations

This thesis has focused on the application of the Bin Packing algorithm in an automated warehouse with grocery items.

As discussed in Section 3-1-5, the end effector of the robot manipulator is not considered in this thesis, therefore the assumption is made that the gripper can attach to the item on any surface. However, some surfaces are not compatible for the gripper, which requires a search for available surfaces for the gripper attachment point. Furthermore, the orientation of the item in the stock bin defines unavailable surfaces for grasping. Subsequently, this poses constraints on the available orientations of the item in the customer bin. The obtained placement algorithm therefore needs to be combined with a grasp surface detection method in further research. This will likely reduce the number of placement options and can therefore increase the number of bins used.

The input of this algorithm is a set of point cloud models obtained from two data sets. However, not all objects in these data sets are correctly obtained which results in the need of filtering the input data. As discussed in Section 3-1-1, both data sets make use of the Structured Light perception method. Other methods exist for object perception that can be used for this application. Further research is needed to find the best model quality produced by several perception methods.

In this thesis, the items are characterised by voxels of 0.67 centimeters. Because of this relatively large sized voxel, the stability of round items is not ensured. To be able to consider this stability as well, other approaches must be researched. For example, a various sized voxelization method, called an octree, can be used to overcome this problem. In an octree, simple structures are described by large-sized voxels and for complex structures the voxel is split into smaller subspaces [Rusu, 2013]. This alternative can be tested in further research. The method of the octree or other alternative structures are not discussed in this thesis, for the reason that the representation of the shape was thought to be sufficient using regular voxels. However, the stability and the rolling resistance of round items was not considered in this decision. The addition of various sized voxels are expected to result in more realistic packing plans, since the curvature of the items is included as well.

An important improvement to current research is the addition of different offline planning algorithms. Currently, the Heightmap Minimization (HM) algorithm by Wang and Hauser [2019] is implemented to determine the offline packing. The HM algorithm has the advantage that it focuses on placing the items on the bottom of the bin. This reduces the number of offline planner calls in the algorithm, besides ensuring a more stable packing plan. However, this might reduce the volume utilization and increase the number of bins used. Therefore, in further research different offline planning algorithms are to be considered.

In this research the assumption has been made that all the items can be picked with a robot arm. However, in reality this is not the case. For example the mass changes of for

example a net of oranges when it is picked and furthermore, surfaces that are not flat cannot be picked currently.

Due to the many different characteristics of the Bin Packing problems, no clear comparison can be made between this research and existing literature. However, the field of research of the Bin Packing problem in automated warehouses is rapidly increasing. Therefore, the expectation that the in the near future more algorithms are implemented using online 1-bounded characteristics. The implementation of the research with this benchmark can therefore be used as a reference for further research.

Lastly, the obtained algorithm can not be directly compared to the algorithm by Wang and Hauser [2020] regarding the computation time, since different hardware is used in this research. To be able to make a fair comparison regarding the computation time, the algorithm needs to be implemented in a virtual machine similar to the one used by Wang and Hauser [2020]. In this project, the means to implement the algorithm in a virtual machine were not available, therefore this step is not yet taken. For the implementation of the Bin Packing algorithm in the automated warehouse, however, this step needs to be taken in order to reduce the computation time.

Appendix A

Pseudo code

A-1 Quasi-Online algorithm

A-1-1 Online planning

Algorithm 3: QOP-Recurse(N)

input: policy tree node N at depth l

```
1  Let σ_{1:l} be the sequence of packed items in N;
2  Let P_N be the set of plans in N;
3  if for any P ∈ P_N the subgraph of P induced by {σ ∉ σ_{1:L} } has no edges  then
     return "success";
4  for All items σ_{l+1} ∉ σ_{1:l} do
5      if any plan in P_N is compatible with σ_{1:l+1} then
6          Let T_{σ_{l+1}} be the location compatible with the most plans in P_N;
7          P_C ←{P' ∈P_N ‖P' is compatible with T_{σ_{l+1}}};
8      else
9          Let P be any plan in P_N or nil if P_N = ∅;
10         P' ← Offline-Pack(σ_{1:l}, P, σ_{l+1});
11         if P' = "failure"  then return "failure";
12         P_C ← {P'};
13         For all ancestors A of N, add P' to P_A;
14     end
15     C ← add-child(N, σ_{l+1}, P_C);
16     if QOP-Recurse(C) fails return "failure"
17 end
18 return "success"
```

A-1-2 Offline packing

Algorithm 4: 3DGridSearch

input: Item geometry $\mathcal{G}$, container $\mathcal{C}$, rolls and pitches $\mathcal{O} = (\phi_1, \psi_1), ..., (\phi_n, \psi_n)$
output: All legal candidate transforms $\mathcal{T} = (\phi_1, \psi_1, \theta_1, x_1, y_1, z_1), ..., (\phi_n, \psi_n, \theta_n, x_n, y_n, z_n)$
for $(\phi, \psi) \in \mathcal{O}$ **do**
  for $\theta \in \{0, \Delta r, 2\Delta r, ..., 2\pi - \Delta r\}$ **do**
    Let $R \leftarrow R_z(\theta)R_y(\phi)R_x(\psi)$;
    Discretize legal horizontal translations for $R \cdot \mathcal{G}$ into grid $(X_1, Y_1), ..., (X_n, Y_n)$;
    for *(X, Y)* $\in (X_1, Y_1), ..., (X_n, Y_n)$ **do**
      Find the lowest collision free placement Z at translation X,Y;
      Let T be rigid transform with rotation $\mathcal{R}$ and translation (X, Y, Z);
      if $T \cdot \mathcal{G}$ *lies within* $\mathcal{C}$ **then**
        Add T to $\mathcal{T}$
      else
    end
  end
end
return $\mathcal{T}$

Algorithm 5: packOneItem

input: item geometry $\mathcal{G}$, container $\mathcal{C}$, pitches and yaws $\mathcal{O} = (\phi_1, \psi_1), ..., (\phi_n, \psi_n)$, sequence of the packed items $\mathcal{S} = (s_1, ...s_i)$, transforms of the packed items $\mathcal{P} = \{P_1, ..., P_i\}$
output: Transform T or **None**
$\mathcal{T} \leftarrow$ 3DGridSearch $(\mathcal{G}, \mathcal{C}, \mathcal{O})$;
Score each T in $\mathcal{T}$ based on heuristic used;
for *up to N lowest values of T in* $\mathcal{T}$ **do**
  $s \leftarrow$ isStable$(T \cdot \mathcal{G}, \mathcal{C}, P_1 \cdot \mathcal{G}_{s_1}, ..., P_i \cdot \mathcal{G}_{s_i})$
  if s **then**
    continue
  end
  Obtain grasp pose candidates $T_1^{\mathcal{G}}, ..., T_n^{\mathcal{G}}$ compatible with T;
  $f =$ isManipFeasible$(T \cdot \mathcal{G}, (T_1^{\mathcal{G}}, ..., T_n^{\mathcal{G}}))$;
  if f **then**
    return T
  end
end
return None

Algorithm 6: Robot-feasible packing with fall back procedures

input: item geometry $\mathcal{G}_1, ..., \mathcal{G}_N$, container $\mathcal{C}$, sequence of the packed items $\mathcal{S}0 = (s0_1, ...s0_i)$

output: Transforms $\mathcal{T} = (T_1, ..., T_N)$ and final packing sequence $\mathcal{S} = (s_1, ..., s_N)$, or **None**

Initialize $\mathcal{T}, \mathcal{S}, \mathcal{U}, \mathcal{O}$ to empty lists;
for $\mathcal{G}_i \in \{\mathcal{G}_1, ..., \mathcal{G}_N\}$ **do**
    Get planar-stable rolls and pitches for $\mathcal{G}_i$ with the top n highest quasi-static probabilities$O_i = \{(\phi_1, \psi_1$;
    Add O_i to $\mathcal{O}$;
end
for $s0_i \in \{s0_1, ..., s0_N\}$ **do**
    T = packOneItem $(G_{s0_i}, C, O_{s0_i}, \mathcal{S}, \mathcal{T})$;
    if T **then**
        Add T to $\mathcal{T}$;
        Add $s0_i$ to $\mathcal{S}$;
    else
        Add $s0_i$ to U;
    end
end
for $u_i \in \mathcal{U}$ **do**
    Let $\{(\phi_1, \psi_1), ..., (\phi_n, \psi_n)\}$ be the planar-stable orientations in O_{ui};
    for $t_r \in \{0, \Delta r, 2\Delta r, ..., 2\pi - \Delta r\}$ **do**
        for $t_p \in \{0, \Delta r, 2\Delta r, ..., 2\pi - \Delta r\}$ **do**
            $O^t = \{(\phi_1 + t_r, \psi_1 + t_p), ..., (\phi_n + t_r, \psi_n + t_p)\}$;
            T = packOneItem$(G_{ui}, C, O^t, \mathcal{S}, \mathcal{T})$;
            if T **then**
                Add T to $\mathcal{T}$;
                Add u_i to $\mathcal{S}$;
                continue with Line 15
            end
        end
    end
    return "no solution"
end
return $(\mathcal{T}, \mathcal{S})$

Appendix B

Methods

B-1 Score of the Heightmap Minimization heuristic

To determine the score of each packing arrangement, three heightmaps are considered. The first is H_c a top-down heightmap of the container , second H_t a top-down heightmap of the object to be placed and third H_b a bottom-up heightmap of the object to be placed.

The lowest collision-free value Z is computed, which is the lowest location where the item can be placed without it colliding with other items. Equation (B-1) shows the computation of the Z-value. In this equation, w and h are the dimensions of the item and x and y the pixel coordinates of location (X, Y).

$$Z = \max_{i=0}^{w-1} \max_{j=0}^{h-1} \left(H_c[x+i, y+j] - H_b[i,j] \right) \tag{B-1}$$

Consequently, the heightmap is updated to include the new placed item which results in updated heightmap H'_c. For $i \in w$ and $j \in h$, the updated heightmap is obtained according to the following equation:

$$H'_c[x+i, y+j] = \max \left(H_t[i,j] + Z, H_c[x+i, y+i] \right) \tag{B-2}$$

Using this updated heightmap, the score is obtained using Equation (2-3).

B-2 Quasi-static probabilities

Goldberg et al. [1999] determine the quasi-static estimator by projecting the item onto a sphere. Specifically, a convex hull is generated from the item in which the facets are

projected onto the sphere. The ratio of the area of the projected surface F_π of face F to the total surface area of the sphere generates the probability that the item will land on face F. In Figure B-1, the convex hull with face F and the sphere with projected surface F_π are shown.

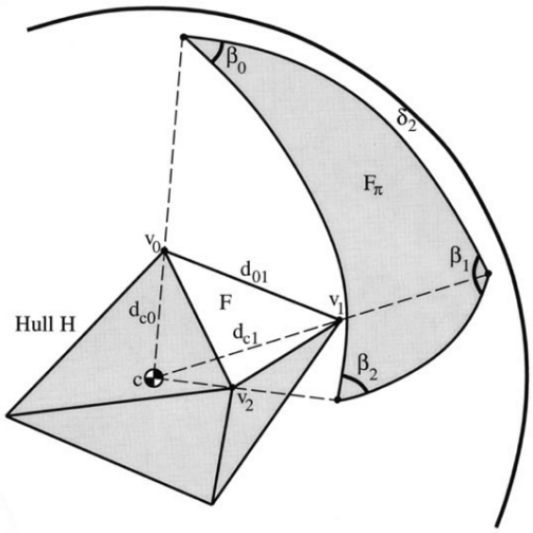


Figure B-1: Projection of convex hull on sphere [Goldberg et al., 1999]

To obtain the probabilities of all faces of the item using the method obtained by Goldberg et al. [1999], each of the faces F_i need to be triangular. Therefore, to be able to use this method from the point cloud, a mesh model is generated.

The ratio which shows the probability described in this section is given in Equation (B-3). To obtain this ratio, the angles β_0, β_1, and β_2 are to be calculated. These angles are the angles of the projected surface F_π.

$$A = \frac{\beta_0 + \beta_1 + \beta_2 - \pi}{4\pi} \tag{B-3}$$

First edges d_{c0}, d_{c1}, and d_{01} are calculated using the Pythagorean theorem. Consequently the edges are calculated as follow:

$$d_{c0} = \sqrt{c^2 + v_0^2} \tag{B-4}$$

$$d_{c1} = \sqrt{c^2 + v_1^2} \tag{B-5}$$

$$d_{01} = \sqrt{v_0^2 + v_1^2} \tag{B-6}$$

Using the law of cosines, angle δ_2 is calculated:

$$d_{01}^2 = d_{c0}^2 + d_{c1}^2 - 2d_{c0}d_{c1}\cos\delta_2 \tag{B-7}$$

Similarly, δ_0 is calculated using d_{c1}, d_{c2}, and d_{12} and δ_1 is calculated using d_{c0}, d_{c2}, and d_{01}. δ_0, δ_1, and δ_2 are the edges of the projected surface F_π. Using the spherical law of cosines, β_0, β_1, and β_2 are calculated. The equations are as follows:

$$\cos\delta_0 = \cos\delta_1\cos\delta_2 + \sin\delta_1\sin\delta_2\cos\beta_0 \tag{B-8}$$

$$\cos\delta_1 = \cos\delta_0\cos\delta_2 + \sin\delta_0\sin\delta_2\cos\beta_1 \tag{B-9}$$

$$\cos\delta_2 = \cos\delta_0\cos\delta_1 + \sin\delta_0\sin\delta_1\cos\beta_2 \tag{B-10}$$

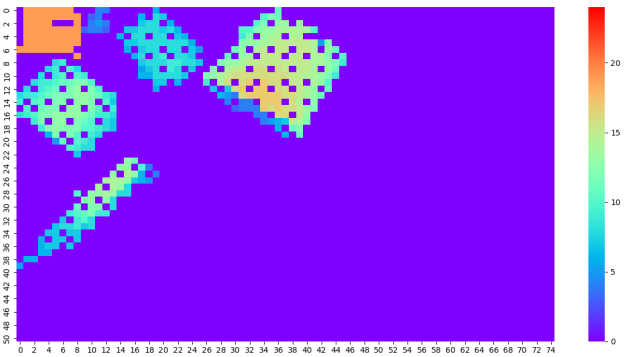
Appendix C

Packing results

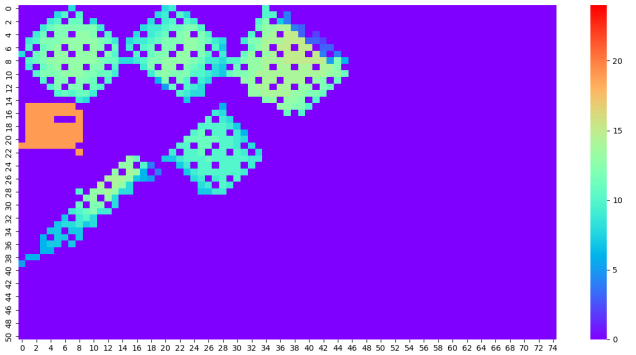
The packing results are generated using the same parameters as in the experiments from Chapter 4. The axes are labeled according to the number of voxels. The bin is therefore 75 voxels in length, 51 voxels in width and 24 voxels in height, which is equivalent to 50x34x16 centimeters.

C-1 APC and YPC data sets 5 items placed

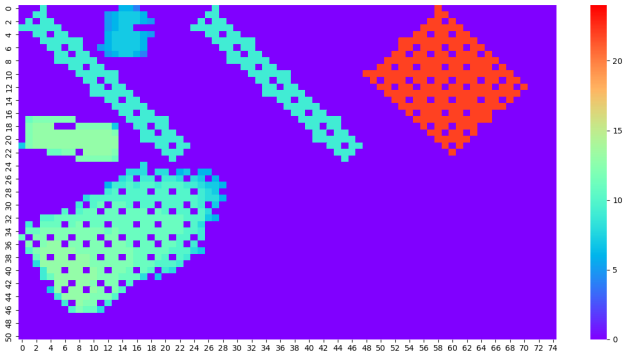
1. highland 6539 self stick notes
2. 077 rubiks cube
3. 065-a cups
4. 37 scissors
5. 65-j cups
6. 63-d marbles



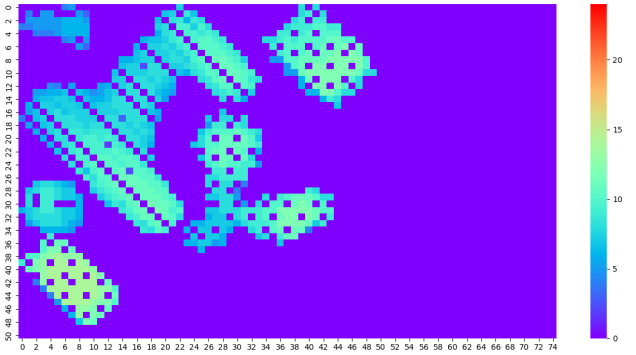
1. 013 apple
2. highland 6539 self stick notes
3. 077 rubiks cube
4. 037 scissors
5. 065-i cups
6. 65-d cups



1. paper mate 12 count mirado black warrior
2. paper mate 12 count mirado black warrior
3. 006 mustard bottle
4. 009 gelatin box
5. genuine joe plastic stir sticks
6. 73-l lego duplo

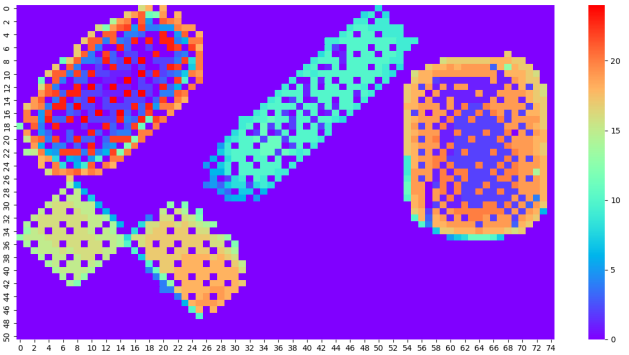


1. 035 power drill
2. 072-d toy airplane
3. 061 foam brick
4. 072-k toy airplane
5. 073-b lego duplo
6. 007 tuna fish can

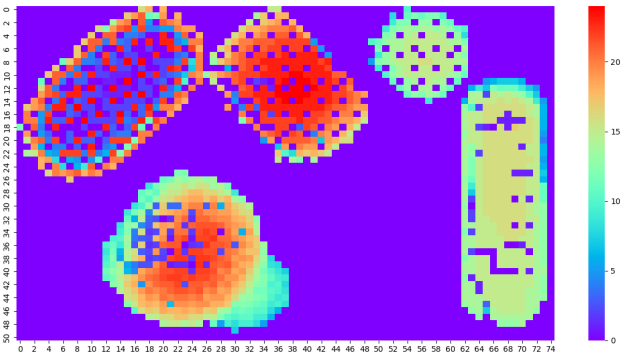


C-2 Picnic data set

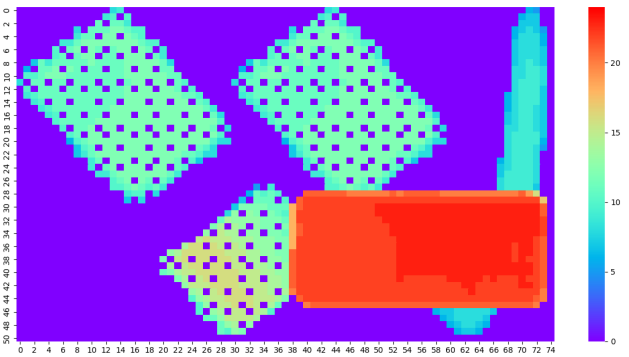
1. 86 witte puntjes
2. 79 volle melk
3. 49 peren
4. 56 mozzarella
5. 35 eitjes 6



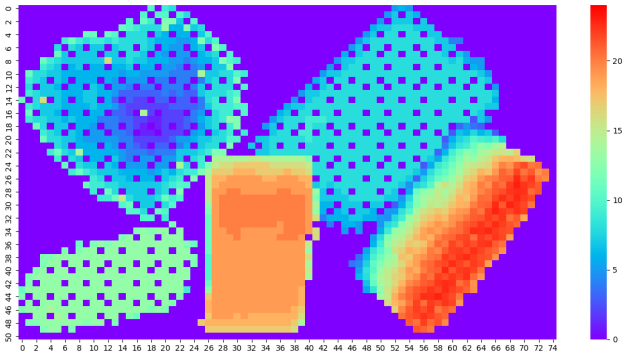
1. 86 witte puntjes
2. 33 franse kwark
3. 8 broccoli
4. 55 mango
5. 21 eitjes 10



1. 95 trostomaten
2. 95 trostomaten
3. 14 prei
4. 16 champignons
5. 85 eieren 10



1. 59 uien
2. 68 rundergehakt 500
3. 32 witlof
4. 44 druiven
5. 65 witte druiven



Appendix D

Inputs

D-1 Deleted object models

Table D-1: Deleted items from the YCB dataset [Calli et al., 2015]

Item name	Dataset
022_windex_bottle	YCB
024_bowl	YCB
029_plate	YCB
030_fork	YCB
031_spoon	YCB
032_knife	YCB
033_spatula	YCB
040_large_marker	YCB
041_small_marker	YCB
042_adjustable_wrench	YCB
043_philips_screwdriver	YCB
044_flat_screwdriver	YCB
046_plastic_bolt	YCB
047_plastic_nut	YCB
049_small_clamp	YCB
050_medium_clamp	YCB
053_mini_soccer_ball	YCB
059_chain	YCB
063-b_marbles	YCB
063-c_marbles	YCB
063-f_marbles	YCB
070-a_colored_wood_blocks	YCB
072-b_toy_airplane	YCB

Table D-2: Deleted items from the APC dataset [Rennie et al., 2016]

Item name	Dataset
dr_browns_bottle_brush	APC
first_years_take_and_toss_straw_cup	APC

D-2 Picnic data set

Table D-3: characteristics of Picnic data set

Number	Article ID	Name	L (mm)	W (mm)	H (mm)	Pick seq.
1	10573488	Komkommer 1st	313	45	46	1
2	10575111	Halfvolle melk 2L	121	92	241	1
4	90006132	Bananen 5st	180	150	80	1
5	10467898	Courgette 1st	228	55	61	1
6	10468928	Rode paprika 1st	89	89	108	1
7	90006137	Chiquita bananen 1kg	180	150	80	1
8	10074561	Broccoli 1st	163	151	146	2
9	10761954	Geraspte milde kaas 45+ 200g	192	154	36	1
10	11295128	Paprika mix 3st	142	120	88	1
11	10764031	Scharreleieren klasse M 15st	238	153	73	2
12	10844029	Halfvolle melk houdbaar 1L	91	65	213	2
13	10762927	Yoghurt Griekse stijl 1kg	130	130	133	1
14	90006011	Prei 1st(ca. 170g)	368	65	58	1
15	11454601	Toiletpapier 3 laags 8st	410	110	188	2
16	10573498	Witte champignons 250g	133	114	88	2
17	10760358	Jong belegen kaas plakken	157	104	41	1
19	90006009	Avocado eetrijp 1st	82	63	68	2
20	10636328	Witte bollen 6st	257	121	101	1
21	11293150	Eitjes 10stM/L	247	110	76	5
23	10725982	Heel fijn volkoren brood 800g	345	119	161	1
24	10074656	Mandarijnen 1kg	193	172	66	2
25	10564712	Magere Franse kwark 500g	116	116	86	2
26	10601918	Cherrytomaatjes 250g	118	77	88	1
27	10074801	Kinderbananen 6 of 7st(850g)	207	143	98	1
28	10564503	Ongezouten roomboter 250g	115	64	41	3
29	11433722	Water bruisend 1,5L	88	88	344	2
30	11433736	Water koolzuurvrij 6 x 500ml	207	138	220	1
31	10119132	Keukenpapier 3 laags 4st	285	121	238	3
32	10074490	Witlof 500g	219	184	66	3
33	11399888	Magere Franse kwark 1kg	150	150	105	1

35	11293151	Eitjes 6stM/L	149	109	71	5
36	90006122	Elstar appels 1,5kg	271	203	81	1
37	11299669	Halfvolle melk 1,5L	102	79	245	1
38	10503781	Halfvolle melk 1L	73	72	236	1
39	10911384	Halfvolle melk 1L	75	73	238	1
40	10511693	Afbak kaiserbroodjes	182	170	66	3
41	90006002	Sperziebonen 500g	299	197	48	2
42	10281918	Rundergehakt 350g	161	123	51	1
43	10468158	Bosui 1 bosje	329	67	73	1
44	10467798	Pitloze rode druiven 500g	174	111	81	3
45	10564710	CrÃ¨me fraÃ®che 200g	74	74	101	3
46	10075027	Bio fairtrade bananen 5st	217	119	151	1
47	10725992	Heel boeren waldkorn brood	324	149	143	1
48	10468008	Bloemkool 1st	220	150	150	1
49	90006051	Conference peren 1kg	185	130	128	1
50	10571223	Tomatenblokjes 400g	76	76	111	3
51	10571588	Zoete puntpaprika 2st	196	97	71	1
52	10467828	Aubergine 1st	186	87	98	1
53	10547334	Ijsthee groen 1,5L	100	78	246	2
54	90006013	Tomaten 500g	178	123	86	1
55	90006049	Mango eetrijp 1st	105	84	81	4
56	10765597	Mozzarella 125g	99	76	71	1
57	10929288	Mineraalwater 2L	106	103	253	1
58	10562310	Magere yoghurt 1L	73	73	236	1
59	10550128	Uien 1kg	190	187	98	1
60	10764022	Scharreleieren klasse M 10st	247	112	73	5
62	90006093	Snack pruim tomaten 500g	118	118	138	1
63	11454603	Toiletpapier 4 laags 6st	320	115	192	3
64	10574519	Licht gerookte spekreepjes	185	118	56	2
65	10574633	Pitloze witte druiven 500g	177	109	88	3
66	10761800	Bio eieren klasse M 10st	247	110	76	5
68	10281919	Rundergehakt 500g	211	137	51	1
69	10465898	Knoflook 2st	78	54	56	1
70	10511633	Halfvolle melk houdbaar 1L	75	73	211	2
71	10075932	Geschrapte wortel 500g	180	160	48	2
72	10567279	Volle yoghurt 1L	75	74	236	1
73	10572577	Havermout vlokken 500g	100	79	158	4
74	10075916	Rucola 75g	203	201	63	2
75	10298788	Kipdijfilet 380g	169	123	53	3
76	10969312	Trekbandzakken 60 liter 15st	236	80	76	3
77	10561650	Perssinaasappelen 2kg	261	204	78	1

78	11283567	Tomatenpuree 140g	54	54	73	4
79	11327176	Volle melk 1L	75	75	236	1
80	10761940	Milde kruidenkaas 150g	96	96	56	3
81	10511616	Volle melk houdbaar 1L	76	75	208	2
82	11328082	Kipfilet 200g	150	170	5	2
83	10636468	Rozijnen krentenbollen 6st	253	107	93	1
84	10075995	Ijsbergsla 200g	215	210	45	2
85	10764023	Scharreleieren klasse L 10st	234	114	73	5
86	10636398	Witte puntjes 6st	192	153	108	1
87	10074682	Citroenen 5st	72	72	63	2
88	10575653	Energy drink 250ml	52	52	136	4
89	11399883	Bio halfvolle melk 1,5L	83	101	248	1
90	90006067	Bio komkommer 1st	338	53	46	1
91	10119137	Tissues lotion 3 laags 90st	224	117	76	3
92	10568662	Crème fraîche 125g	72	72	68	3
93	11454602	Toiletpapier 3 laags recycled	432	95	216	2
94	10298471	Kipfilet 400g	171	122	53	3
95	90006012	Trostomaten 500g	177	128	76	1
96	10574091	Filet amèricain naturel	137	107	46	3
97	10725983	Heel boeren tijger tarwe brood	336	144	143	1
98	10965107	Cola max 1,5L	87	87	343	2
99	10547367	Cola zero 1,5L	91	91	343	2

D-3 Generated item sets for benchmark

Class 1

w	d	h
39	72	85
11	96	84
36	34	10
12	71	76
64	56	73
26	97	77
30	17	23
46	47	27
47	67	69
29	98	71
88	64	69
47	99	95
20	93	95
23	32	8
67	55	73
69	95	46
47	85	80
10	91	87
10	96	100
45	84	100
15	82	99
93	38	93
82	95	43
98	97	38
69	72	90
66	25	90
26	88	78
37	21	10
84	69	17
47	99	96
19	74	72
4	73	77
51	51	97
38	95	74
23	26	42
72	64	86
6	67	86
86	88	25
5	88	74
92	10	90
32	70	82
24	93	80
24	74	95
66	86	77
47	77	82
7	76	76
38	99	70
92	39	80
46	92	93

Class 4

w	d	h
84	70	65
11	96	84
36	34	10
68	58	61
64	56	73
96	74	78
30	17	23
46	47	27
99	70	89
73	91	81
88	64	69
47	99	95
20	93	95
23	32	8
67	55	73
69	95	46
57	55	99
52	70	79
59	79	85
45	84	100
85	78	99
93	38	93
82	95	43
98	97	38
69	72	90
66	25	90
65	95	50
37	21	10
84	69	17
56	96	88
94	50	77
63	99	73
51	51	97
38	95	74
23	26	42
72	64	86
6	67	86
86	88	25
5	88	74
92	10	90
71	65	50
24	93	80
59	92	97
66	86	77
47	77	82
50	62	53
100	81	68
92	39	80
83	99	55

Class 5

w	d	h
39	27	25
11	96	84
36	34	10
12	26	26
64	56	73
26	2	22
30	17	23
46	47	27
47	37	9
29	18	1
88	64	69
47	99	95
20	93	95
23	32	8
67	55	73
69	95	46
47	40	40
10	36	32
10	36	25
45	84	100
15	12	34
93	38	93
82	95	43
98	97	38
69	72	90
66	25	90
26	48	33
37	21	10
84	69	17
47	24	11
19	39	37
4	38	27
51	51	97
38	95	74
23	26	42
72	64	86
6	67	86
86	88	25
5	88	74
92	10	90
32	35	12
24	93	80
24	29	50
66	86	77
47	77	82
7	6	41
38	19	10
92	39	80
46	17	3

Class 6

w	d	h
6	5	9
7	1	1
4	5	1
6	4	10
8	6	2
6	5	4
7	10	3
6	2	2
5	3	10
7	6	5
7	8	7
7	7	9
10	1	9
8	10	4
6	1	4
7	9	5
1	5	10
3	3	5
8	4	2
5	3	2
2	6	4
5	7	8
10	7	10
10	6	2
9	5	10
6	5	1
5	10	4
5	2	4
3	8	8
8	7	2
3	2	10
3	7	8
4	3	1
6	6	3
5	10	5
6	8	3
5	10	7
1	9	2
7	9	9
7	4	1
9	7	9
9	4	8
7	4	8
5	6	1
1	9	8
10	3	5
2	4	6
2	2	8
1	6	6

Class 7

w	d	h
1	20	29
7	31	21
19	15	31
26	24	15
28	11	7
6	25	4
27	30	33
16	32	12
15	3	25
17	1	30
12	13	27
2	2	4
10	6	4
33	20	34
1	6	9
7	34	30
21	30	25
28	28	35
18	19	27
10	3	17
32	11	4
30	2	18
15	32	20
5	26	22
29	35	25
31	20	21
35	15	19
25	17	34
18	28	28
13	17	17
8	22	30
33	32	33
24	8	26
6	1	3
25	30	10
16	23	13
15	30	32
16	19	2
27	14	4
32	34	31
14	7	9
9	19	28
12	19	8
10	1	26
16	9	23
30	13	25
2	14	1
32	7	33
16	21	16

Class 8

w	d	h
46	75	39
27	61	21
94	35	81
86	34	60
8	26	62
76	35	4
37	50	3
76	52	22
35	23	80
67	96	75
27	8	97
47	87	59
100	1	79
18	70	64
46	91	24
97	29	65
11	55	70
23	73	5
58	24	32
35	33	42
32	46	74
45	97	78
40	7	40
60	86	32
99	75	10
36	95	51
25	50	24
65	62	84
53	8	18
88	27	42
93	32	100
13	77	88
74	43	1
26	96	13
95	80	75
76	98	83
35	10	37
71	59	52
17	19	29
97	24	61
99	37	69
39	54	38
77	94	38
55	46	71
71	69	38
70	73	35
92	4	26
82	32	68
1	66	6

Bibliography

Sam D Allen, Edmund K Burke, and Graham Kendall. A hybrid placement strategy for the three-dimensional strip packing problem. *European Journal of Operational Research*, 209(3):219–227, 2011.

Brenda S Baker and Edward G Coffman, Jr. A tight asymptotic bound for next-fit-decreasing bin-packing. *SIAM Journal on Algebraic Discrete Methods*, 2(2):147–152, 1981.

János Balogh, József Békési, György Dósa, Leah Epstein, and Asaf Levin. A new lower bound for classic online bin packing. *Algorithmica*, pages 1–16, 2021.

Judith O Berkey and Pearl Y Wang. Two-dimensional finite bin-packing algorithms. *Journal of the operational research society*, 38(5):423–429, 1987.

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

Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, and Richard Královič. On the advice complexity of the k-server problem. In *International Colloquium on Automata, Languages, and Programming*, pages 207–218. Springer, 2011.

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

Andreas Bortfeldt, Hermann Gehring, and Daniel Mack. A parallel tabu search algorithm for solving the container loading problem. *Parallel computing*, 29(5):641–662, 2003.

Joan Boyar, Shahin Kamali, Kim S Larsen, and Alejandro López-Ortiz. Online bin packing with advice. *Algorithmica*, 74(1):507–527, 2016.

Nils Boysen, René De Koster, and Felix Weidinger. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 277(2):396–411, 2019.

Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics & Automation Magazine*, 22(3):36–52, 2015.

Vassilios Canellidis, John Giannatsis, and Vassilis Dedoussis. Evolutionary computing and genetic algorithms: paradigm applications in 3d printing process optimization. In *Intelligent Computing Systems*, pages 271–298. Springer, 2016.

Lance D Chambers. *The Practical Handbook of Genetic Algorithms: New Frontiers, Volume II*, volume 2. CRC Press, 2019.

N Chernov, Yu Stoyan, and Tatiana Romanova. Mathematical model and efficient algorithms for object packing problem. *Computational Geometry*, 43(5):535–553, 2010.

Henrik I Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

Edward G Coffman, János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. Bin packing approximation algorithms: survey and classification. In *Handbook of combinatorial optimization*, pages 455–531. 2013.

Teodor Gabriel Crainic, Guido Perboli, and Roberto Tadei. Extreme point-based heuristics for three-dimensional bin packing. *Informs Journal on computing*, 20(3):368–384, 2008.

RBM de Koster. Automated and robotic warehouses: Developments and research opportunities, logistics and transport no 2 (38)/2018. *DOI*, 10:83–1734, 2018.

René de Koster, Tho Le-Duc, and Kees Jan Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2): 481–501, October 2007. ISSN 0377-2217. doi: 10.1016/j.ejor.2006.07.009.

Daniel Delahaye, Supatcha Chaimatanan, and Marcel Mongeau. Simulated annealing: From basics to applications. In *Handbook of metaheuristics*, pages 1–35. Springer, 2019.

Maxence Delorme, Manuel Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20, 2016.

Richard W Eglese. Simulated annealing: a tool for operational research. *European journal of operational research*, 46(3):271–281, 1990.

Michael Eley. Solving container loading problems by block arrangement. *European Journal of Operational Research*, 141(2):393–409, September 2002. ISSN 0377-2217. doi: 10.1016/S0377-2217(02)00133-9.

Emerce. Picnic start gerobotiseerd dc in utrecht, 2019. URL https://www.emerce.nl/nieuws/picnic-2.

Emerce. Picnic opent geautomatiseerd fulfilmentcenter in zwolle, 2021. URL https://www.emerce.nl/nieuws/picnic-opent-geautomatiseerd-fulfilmentcenter-zwolle.

Oluf Faroe, David Pisinger, and Martin Zachariasen. Guided local search for the three-dimensional bin-packing problem. *Informs journal on computing*, 15(3):267–283, 2003.

Michael R Garey and David S Johnson. Approximation algorithms for bin packing problems: A survey. In *Analysis and design of algorithms in combinatorial optimization*, pages 147–172. Springer, 1981.

Jason Geng. Structured-light 3D surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160, 2011. ISSN 1943-8206. doi: 10.1364/AOP.3.000128. Publisher: Optical Society of America.

Fred Glover and Manuel Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

Ken Goldberg, Brian V Mirtich, Yan Zhuang, John Craig, Brian R Carlisle, and John Canny. Part pose statistics: Estimators and experiments. *IEEE Transactions on Robotics and Automation*, 15(5):849–857, 1999.

José Fernando Gonçalves and Mauricio GC Resende. A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers & Operations Research*, 39(2):179–190, 2012.

José Fernando Gonçalves and Mauricio GC Resende. A biased random key genetic algorithm for 2d and 3d bin packing problems. *International Journal of Production Economics*, 145(2):500–510, 2013.

Xian-Feng Han, Hamid Laga, and Mohammed Bennamoun. Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1578–1604, 2019.

Hu, Xiaodong Zhang, Xiaowei Yan, Longfei Wang, and Yinghui Xu. Solving a new 3d bin packing problem with deep reinforcement learning method. *arXiv preprint arXiv:1708.05930*, 2017.

Korhan Karabulut and Mustafa Murat İnceoğlu. A hybrid genetic algorithm for packing in 3d with deepest bottom left with fill method. In *International Conference on Advances in Information Systems*, pages 441–450. Springer, 2004.

Suthum Keerativittayanun, Toshiaki Kondo, Panithi Sira-Uksorn, Teera Phatrapornant, and M Sato. 3d scan of a color object using a color structured light pattern. In *2011 IEEE 7th International Colloquium on Signal Processing and its Applications*, pages 460–463. IEEE, 2011.

Yangyan Li, Soeren Pirk, Hao Su, Charles R Qi, and Leonidas J Guibas. Fpnn: Field probing neural networks for 3d data. *Advances in neural information processing systems*, 29, 2016.

Xiao Liu, Jia-min Liu, An-xi Cao, and Zhuang-le Yao. Hape3d—a new constructive algorithm for the 3d irregular packing problem. *Frontiers of Information Technology & Electronic Engineering*, 16(5):380–390, 2015.

Andrea Lodi, Silvano Martello, and Daniele Vigo. Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics*, 123(1-3):379–396, 2002.

Silvano Martello and Daniele Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management science*, 44(3):388–399, 1998.

Silvano Martello, David Pisinger, and Daniele Vigo. The three-dimensional bin packing problem. *Operations research*, 48(2):256–267, 2000.

Ilkyeong Moon and Thi Viet Ly Nguyen. Container packing problem with balance constraints. *OR spectrum*, 36(4):837–878, 2014.

Yu Peng, Defu Zhang, and Francis YL Chin. A hybrid simulated annealing algorithm for container loading problem. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 919–928. 2009.

Marc P Renault, Adi Rosén, and Rob van Stee. Online algorithms with advice for bin packing and scheduling problems. *Theoretical Computer Science*, 600:155–170, 2015.

Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016.

Kees Jan Roodbergen and Iris F. A. Vis. A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research*, 194(2):343–362, 2009. ISSN 0377-2217. doi: 10.1016/j.ejor.2008.01.038.

Radu Bogdan Rusu. *Semantic 3D object maps for everyday robot manipulation*. Springer, 2013.

Vivek A Sujan and Steven Dubowsky. Application of a model-free algorithm for the packing of irregular shaped objects in semiconductor manufacture. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 1545–1550. IEEE, 2000.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

K Szelag, G Maczkowski, R Gierwialo, A Gebarska, and R Sitnik. Robust geometric, phase and colour structured light projection system calibration. *Opto-Electronics Review*, 25(4):326–336, 2017.

Tatsuya Tanaka, Toshimitsu Kaneko, Masahiro Sekine, Voot Tangkaratt, and Masashi Sugiyama. Simultaneous planning for item picking and placing by deep reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9705–9711. IEEE, 2020.

Vivek Thangavelu, Yifang Liu, Maira Saboia, and Nils Napp. Dry stacking for automated construction with irregular objects. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4782–4789. IEEE, 2018.

Fan Wang and Kris Hauser. Stable bin packing of non-convex 3d objects with a robot manipulator. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8698–8704. IEEE, 2019.

Fan Wang and Kris Hauser. Robot packing with known items and nondeterministic arrival order. *IEEE Transactions on Automation Science and Engineering*, 2020.

Gerhard Wäscher, Heike Haußner, and Holger Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3): 1109–1130, December 2007. ISSN 03772217. doi: 10.1016/j.ejor.2005.12.047.

Zelin Xiao, Hongxin Lin, Renjie Li, Lishuai Geng, Hongyang Chao, and Shengyong Ding. Endowing deep 3d models with rotation invariance based on principal component analysis. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.

Hang Zhao, Qijin She, Chenyang Zhu, Yin Yang, and Kai Xu. Online 3d bin packing with constrained deep reinforcement learning. *arXiv preprint arXiv:2006.14978*, 2020.

Glossary

List of Descriptions

Bounded space Constraint that a fixed number of bins may be open at a time

1-Bounded space Constraint that only one bin is open at a time

Contact points Locations where the item reaches another item or the container edges

Goods-To-Person Items are transported from a storage system to the picking station

Heightmap Grid map that contains the height of the obstacles

Irregular item Items that are not considered to be geometrically simple to describe

Non-deterministic approach The method where nothing is known about the item sequence, therefore the entire policy tree has to be checked

Lookahead The method where knowledge of the upcoming item(s) is applied

Offline algorithm Type of bin packing algorithm where all properties of the items are known beforehand

Online algorithm Type of bin packing algorithm where the items arrive in arbitrary order and the items need to be placed in bins directly without knowledge of the remaining items

Packing plan Plan that determines where each of the items within an order is placed

Person-To-Goods System where the pickers walk past every article to collect the items

Pick sequence Order in which the items are to be collected, determined by fragility, volume, weight and contamination

Picking station Location where the items are picked in a Goods-To-Person system

Packing policy tree Tree containing all different packing policy of an item set

Planner calls The number of times the offline planner is called

Quasi-online algorithm Type of bin packing problem which is a combination between the offline and online problem

Regular item Items that are considered to be geometrically simple to describe

Semi-deterministic approach The method where the policy tree is decreased due to the pick sequence constraints

Success rate Rate that defines the ability to place N items in a bin

Volume Utilization Percentage of the total volume in use

Voxelization Method to describe an object geometrically using cubes

Worst-case performance ratio Metric to define the worst-case performance

List of Acronyms

DBLF	Deepest Bottom Left Fill
3BF	Three-dimensional Best Fit
DFTRC	Distance to the Front-Top-Right Corner
EP	Extreme Points
HM	Heightmap Minimization
BF	Best Fit
BPP	Bin Packing Problem
SA	Simulated Annealing
GA	Genetic Algorithm
BRKGA	Biased Random Key Genetic Algorithm
TS	Tabu Search
QO	Quasi-Online
HGA	Hybrid Genetic Algorithm
HSA	Hybrid Simulated Annealing
CLP	Container Loading Problem
PTS	Parallel Tabu Search
RL	Reinforcement Learning
ASRS	Automated Storage and Retrieval System
PCA	Principal Component Analaysis

List of Symbols

Abbreviations

$\bar{p}$	Center of point cloud
δr	Angle of rotation
μ	Friction coefficient
c	Weight assigned to the score
c_k	Contact point
cm	Center of mass
f_k	Contact force
g	Gravitational constant
h	Height of the bin
m	Mass
N	Number of rotation-location candidates
n	Number of quasi-static orientations
p_i	Point in point cloud
S	Container size
w	Width of the bin