

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A heuristic approach for robotic placing synthesis

João António Agostinho Lopes Ventura Nogueira



Mestrado em Engenharia Eletrotécnica e de Computadores

Orientador: Doutor João Pedro Carvalho de Souza

Coorientador: Doutor Luís Freitas Rocha

October 21, 2024

Resumo

Esta dissertação aborda o desafio de determinar a posição de itens a serem manuseados por braços robóticos na indústria, assim como o desafio do empacotamento induzido por um operador. Detalha o desenvolvimento de um sistema de empacotamento que permite tanto o empacotamento manual quanto a geração automática de posições para itens de vários tamanhos e formas dentro de um contentor de tamanho e forma variáveis. O sistema opera em Robot Operation System (ROS), usando RViz para visualização, Flexible Collision Library (FCL) para detecção de colisão e Open Asset Import Library (Assimp) para processamento de modelos 3D. Apresenta uma heurística construtiva para o problema de carregamento de contentores em sistemas de empacotamento robótico. O algoritmo prioriza itens de maior volume e maior área de base, com o objetivo de criar soluções de empacotamento estáveis e compactas. Utiliza-se uma abordagem de construção baseada em camadas, preenchendo camadas horizontalmente antes de empilhá-las verticalmente, e incorpora verificação de colisão para garantir a viabilidade das soluções. A capacidade do algoritmo é demonstrada através da sua utilização de volume numa variedade de cenários de empacotamento com heterogeneidade de itens. A dissertação conclui discutindo potenciais trabalhos futuros, como a incorporação de interatividade manual com soluções de empacotamento automático, o uso de representações de itens irregulares diretamente para o empacotamento e a integração de heurísticas de melhoria para aprimorar o desempenho do sistema de empacotamento e a adaptabilidade às restrições do mundo real.

Palavras-chave: Problemas de Corte e Empacotamento, Posicionamento Robótico, Planeamento de Mosaico, ROS, Abordagem Heurística

Abstract

This dissertation addresses the challenge of determining the precise placement of items to be handled by robotic arms in the retail industry, as well as the challenge of operator-induced packing. It details the development of a packing system that allows both manual packing and automatic generation of placement poses for items of varying sizes and shapes within a variable size and shape container. The system operates in Robot Operation System (ROS), using RViz for visualization, the Flexible Collision Library (FCL) for collision detection, and the Open Asset Import Library (Assimp) for 3D model processing. It presents a constructive heuristic for the container loading problem in robotic packing systems. The algorithm prioritizes larger volume items and larger based area items, aiming to create stable and compact packing solutions. It utilizes a layer-based building approach, horizontally filling layers before stacking them vertically, and incorporates collision checking to ensure the feasibility of solutions. The algorithm's effectiveness is demonstrated through its volume utilization across a variety of item heterogeneity packing scenarios. The dissertation concludes by discussing potential future work, such as incorporating manual interactivity with automatic packing solutions, using irregular item representations directly for packing, and integrating improvement heuristics to further enhance the packing system's performance and adaptability to real-world constraints.

Key Words: Cutting and Packing Problem, Robotic Placing, Mosaic Planning, ROS, Heuristic Approach

Sustainable Development Goals

This work focuses on the Container Loading Problem (CLP), a subset of cutting and packing (C&P) problems. These problems are prevalent in the retail industry, where efficient packing of products into containers is crucial for storage and transportation. The research contributes to several Sustainable Development Goals (SDGs) through its innovative approach to the CLP.

SDG 8: Decent Work and Economic Growth - The automation potential of the packing algorithm can lead to increased productivity and cost savings in the retail and logistics sectors. This can create new job opportunities in the field of robotics and automation, contributing to economic growth. Additionally, the optimization of packing processes can lead to more efficient use of resources, benefiting both businesses and consumers.

SDG 9: Industry, Innovation, and Infrastructure - The development of a heuristic algorithm for the CLP directly addresses the need for innovation in industrial processes. By optimizing the arrangement of items within containers, the algorithm enhances the efficiency and sustainability of packing operations, contributing to advancements in infrastructure and logistics.

SDG 12: Responsible Consumption and Production - Efficient packing, as facilitated by the algorithm, minimizes wasted space within containers. This reduction in wasted space translates to fewer shipments, lower transportation costs, and a decreased carbon footprint, aligning with the principles of responsible production and consumption.

The research presented in this project aligns with the broader goals of sustainable development by promoting innovation, responsible resource utilization, and economic growth. The developed algorithm and tools have the potential to make a significant impact on the retail and logistics industries, ultimately contributing to a more sustainable future.

Agradecimentos

Em primeiro lugar gostava de agradecer ao meu orientador Dr. João Souza e co-orientador Luís Rocha por me terem guiado e ajudado a ultrapassar todas as dificuldades sentidas ao longo deste percurso difícil. Agradeço também à minha família, mais especificamente à minha irmã, ao meu pai e à minha mãe por estarem presentes durante os momentos de dúvida e de incerteza. Sou muito grato aos meus amigos por me acompanharem nesta jornada, especialmente ao meu amigo Rodrigo Santa Comba por toda a ajuda sentida durante esta fase. Por fim e não menos importante, um agradeciemento especial aos meus avós a quem eu dedico esta dissertação. Ao longo do meu crescimento sempre me motivaram a trabalhar muito e a ultrapassar as minhas dificuldades, levando os nossos deveres até ao fim. Sendo um dos meus principais objetivos, que ambos presenciassem o meu progresso como estudante e finalista de engenharia. Infelizmente, nenhum deles poderá presenciar o fim do meu percurso académico. Mas não deixo de acreditar que em qualquer lugar que estejam, não poderiam estar mais orgulhosos de mim. Obrigado avó Palmira e avô António, por tudo.

João Nogueira

“Estuda para seres um homem grande!”

Palmira Madalena Silva Elias Ventura

Contents

1	Introduction	1
1.1	Context	2
1.2	Motivation	2
1.3	Objectives	2
1.3.1	Manual Packing Tool	3
1.3.2	Automatic Packing Tool	3
1.4	Dissertation Structure	3
2	Technical and Theoretical Background	4
2.1	Robot Operating System (ROS)	4
2.1.1	Nodes	4
2.1.2	Topics and Messages	5
2.1.3	Publishers and Subscribers	5
2.2	ROS Vizualization (RViz)	6
2.2.1	Markers	6
2.2.2	Marker Arrays	6
2.2.3	Interactive Markers	7
2.3	Flexible Collision Library (FCL)	8
2.3.1	Collision	8
2.3.2	FCL Objects	9
2.3.3	3D Transformations in FCL	10
2.4	Open Asset Import Library (Assimp)	12
3	Related Work	13
3.1	Introduction	13
3.2	Problem Formulation and Classifications	14
3.2.1	Typology	15
3.2.2	Constraints	19
3.3	Solution Approaches	30
3.3.1	Exact Methods	30
3.3.2	Placement Heuristics	31
3.3.3	Improvement Heuristics	37
3.3.4	Tree Search Methods	40
4	Methodology	42
4.1	System Architecture	42
4.2	Loading and Processing 3D Models	43
4.2.1	Mesh Models	44

4.2.2	Bounding Boxes	44
4.3	User Inputs	45
4.4	Container Creation	46
4.5	Manual Tool	47
4.5.1	3D Piece Control	47
4.5.2	Collision Checking	49
4.5.3	Position Updating	50
4.6	Automatic Tool	50
4.6.1	Bounding Box Management	51
4.6.2	Packing Algorithm	52
4.7	Items Output	59
4.8	Visualization	60
4.8.1	Pieces	61
4.8.2	Bounding Boxes	61
5	Results	63
5.1	Manual Packing Tool	63
5.2	Automatic Packing Tool	64
5.2.1	Identical Bounding Boxes Packing	66
5.2.2	Weakly Heterogeneous Bounding Boxes Packing	69
5.2.3	Strongly Heterogeneous Bounding Boxes Packing	72
6	Conclusions	77
6.1	Future Work	77
6.2	Final Remarks	78
A	Appendix	85
A.1	Strongly Heterogeneous Packing	85

List of Figures

1.1	Robotic manipulator	1
2.1	ROS topic and message	5
2.2	ROS publisher and subscriber	5
2.3	Markers	6
	(a) Cube	6
	(b) Triangle list	6
2.4	Interactive marker	7
2.5	Interactive marker server architecture	8
2.6	FCL collision manager	9
3.1	Number of publications on "Three Dimensional Packing Problems"	13
3.2	Packing Solutions	17
	(a) Identical	17
	(b) Weakly Heterogeneous	17
	(c) Strongly Heterogeneous	17
3.3	Assortment of small items and containers	17
	(a) Items	17
	(b) Containers	17
3.4	Possible orthogonal orientations	21
3.5	Dimensions and axles positions of the truck	22
3.6	Vertical cross-sections solution	23
3.7	Stable vs unstable packing	24
3.8	Example side-view of unstable box arrangements	25
3.9	High-priority low-priority solution	27
3.10	Wall Building Solution	32
	(a) First Wall Placement	32
	(b) Second Wall Placement	32
	(c) Final Solution	32
3.11	Layer Building Solution	33
	(a) First Layer Placement	33
	(b) Second Layer Placement	33
	(c) Final Solution	33
3.12	Stack Building Solution	34
	(a) 2D View (from above)	34
	(b) 3D View	34
3.13	Types of Blocks	35
	(a) Homogeneous Blocks	35

(b) Heterogeneous Blocks	35
3.14 Block Building Solution	35
3.15 First Placed Box	36
3.16 Possible Guillotine Cuts	36
(a) First Guillotine Cut	36
(b) Second Guillotine Cut	36
(c) Third Guillotine Cut	36
4.1 System architecture	43
4.2 Mesh model creation	44
(a) Triangulation	44
(b) Knee tube item	44
4.3 3D model loading	45
(a) Knee tube	45
(b) Knee tube with bounding box	45
(c) Bounding box	45
4.4 Containers visuals	46
(a) Manual tool container	46
(b) Automatic tool container	46
4.5 Manual processing architecture	47
4.6 Translational axes	48
4.7 Orthogonal Rotational Positions	49
(a) 90° rotation around x-axis	49
(b) 90° rotation around y-axis	49
(c) 180° rotation around z-axis	49
4.8 Fully controllable interactive marker	49
4.9 3D model loading	51
(a) Files .ply directory	51
(b) File .yaml with bounding boxes characteristics	51
4.10 Bounding box prioritizing	52
(a) Volume priority $A \rightarrow B$	52
(b) Base area priority $C \rightarrow A$	52
4.11 Collision checking	53
4.12 Heuristic working flow	54
4.13 Layer determination working flow	55
4.14 Layer height determining	55
4.15 Horizontal placing working flow	57
4.16 Horizontal placement solution	58
4.17 Vertical filling (blue and yellow boxes $\rightarrow$ green boxes)	58
4.18 Vertical filling working flow	59
4.19 Items solutions	60
(a) Knee tube solution	60
(b) Reinforced bracket solution	60
4.20 Mesh model loaded pieces	61
(a) Knee tube	61
(b) Reinforced bracket	61
(c) Falange piece	61
4.21 Bounding boxes fitting pieces	62
(a) Knee Tube	62

(b)	Reinforced Bracket	62
(c)	Falange Piece	62
5.1	Manual packing solution	63
5.2	Manual packing tool solution	64
(a)	Pieces side by side	64
(b)	Piece inside of another piece	64
5.3	Relation between packing time and bounding boxes packed	67
5.4	Relation between volume utilization and bounding box packed volume	68
5.5	"knee_tube" packing solution	68
(a)	"knee_tube" bounding boxes and items	68
(b)	"knee_tube" items solution	68
5.6	IIPP packing solution	69
(a)	Bounding Boxes and items solution	69
(b)	Items solution	69
5.7	Volume utilization for different heterogeneity packing solutions	70
5.8	Bounding boxes packed for different bounding box sizes	71
5.9	SLOPP packing solution	71
(a)	Bounding Boxes and items solution	71
(b)	Items solution	71
5.10	Volume utilization for different heterogeneity packing solutions	73
5.11	Bounding boxes packed for different bounding box sizes	73
5.12	SKP packing solution	74
(a)	Bounding Boxes and items solution	74
(b)	Items solution	74
5.13	Container $500 \times 300 \times 200$ packing solution	75
(a)		75
(b)	Items solution	75
6.1	Packing using bounding boxes for irregular items limitation	78
(a)	Bounding boxes and items solution	78
(b)	Items solution	78

List of Tables

3.1	Output maximization	18
3.2	Input minimization	19
3.3	Constraints	20
3.4	Solution approaches	41
5.1	Bounding Boxes Types	65
5.2	Bounding boxes characteristics	66
5.3	Identical bounding boxes packing solutions	67
5.4	Different sized bounding boxes averages	69
5.5	Average performance for identical bounding boxes packing	69
5.6	Average performance for weakly heterogeneous bounding boxes packing	71
5.7	Three different bounding boxes packing solution	72
5.8	Four different bounding boxes packing solution	72
5.9	Five different bounding boxes packing solution	72
5.10	Average performance for strongly heterogeneous bounding boxes packing	74
5.11	Large and medium bounding boxes packing combination	75
5.12	Large and small bounding boxes packing combination	76
5.13	Medium and small bounding boxes packing combination	76
A.1	Six different bounding boxes packing solution	85
A.2	Eight different bounding boxes packing solution	85
A.3	Ten different bounding boxes packing solution	86
A.4	Twelve different bounding boxes packing solution	86
A.5	Fourteen different bounding boxes packing solution	87
A.6	Sixteen different bounding boxes packing solution	88
A.7	All different bounding boxes packing solution	89

Abbreviations and Symbols

1D	One-Dimensional
2D	Two-Dimensional
3D	Three-Dimensional
AABB	Axis-Aligned Bounding Box
BBP	Bin Packing Problem
BVH	Bounding Volume Hierarchy
C&P	Cutting and Packing
CLP	Container Loading Problem
FCL	Flexible Collision Library
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
IIPP	Identical Item Packing Problem
MBSBPP	Multiple Bin-Size Bin Packing Problem
MCLP	Multi Container Loading Problem
MHKP	Multiple Heterogeneous Knapsack Problem
MHLOPP	Multiple Heterogeneous Large Object Placement Problem
MIKP	Multiple Identical Knapsack Problem
MILOPP	Multiple Identical Large Object Placement Problem
MILP	Mixed-Integer Linear Programming
MIP	Mixed-Integer Programming
MSSCSP	Multiple Stock-Size Cutting Stock Problem
ND	N-Dimensional
NP	Nondeterministic Polynomial Time
OBRRSS	Oriented Bounding Box Rectangle-Swept Sphere
ODP	Open Dimension Problem
PLY	Polygon File Format
PP	Packing Problem
RBPP	Residual Bin Packing Problem
RCSP	Residual Cutting Stock Problem
ROS	Robot Operating System
RViz	ROS Visualization
SA	Simulated Annealing
SBSBPP	Single Bin-Size Bin Packing Problem
SKP	Single Knapsack Problem
SLOPP	Single Large Object Placement Problem
SSSCSP	Single Stock-Size Cutting Stock Problem
TS	Tabu Search
YAML	Yet Another Markup Language

Chapter 1

Introduction

The increasing importance of robotic manipulators in the retail industry is evident in their widespread adoption for object placement tasks. These robots enhance efficiency and precision, leading to improvements in productivity and process quality. Consequently, there is a growing demand for the development and refinement of robotic manipulators, particularly automatic manipulator arms. A key focus area for improvement lies in space utilization during packing processes. By packing the organizational space, these robots contribute to increased logistical efficiency, reduced storage costs, and improved safety in storage and distribution operations.

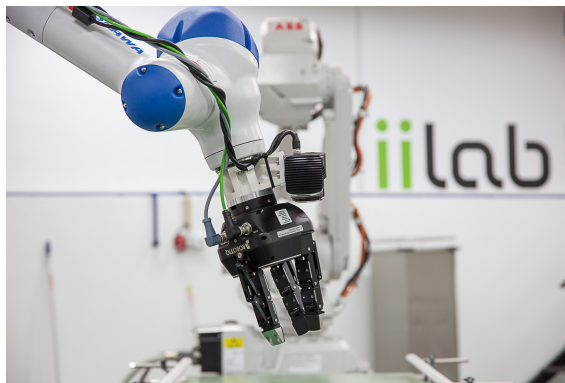


Figure 1.1: Robotic manipulator

The primary challenge in packing organizational space is efficiently packing a set of items into a container or truck. This problem, extensively documented in the literature, falls within the domain of Cutting and Packing (C&P) problems, specifically classified as Three-Dimensional Packing Problem (3D-PP). While packing and organizing items autonomously remains an ongoing area of research, the growth of the robotic and automation industry has intensified the need for effective solutions. Historically, this operational research and management issue has been addressed with the overarching goal of maximizing space utilization, minimizing resource consumption, and reducing packing costs. Automatic robotic arms have been developed to aid in retail manufacturing, programmed to operate autonomously or to execute tasks assigned by operators. However,

challenges in automating the packing process persist, making it an ongoing area of research and development.

1.1 Context

The retail industry faces ever-increasing demands for efficiency, accuracy, and cost-effectiveness in its logistics and supply chain operations. A critical component of these operations is the packing process, where products are arranged within containers for storage and transportation. Traditionally, this process has been labor-intensive and prone to errors. Moreover, the rising costs of labor and the growing complexity of product assortments necessitate innovative solutions to optimize packing processes and enhance overall productivity. In response to these challenges, the industry is turning to robotic automation, with a particular focus on the development of robotic arms that can autonomously handle and pack items. However, the successful implementation of robotic packing systems requires sophisticated algorithms and tools that can determine optimal packing configurations for a variety of items and containers, ensuring that space is utilized efficiently and that items are packed securely.

1.2 Motivation

This dissertation is motivated by the need to develop intelligent solutions for determining the precise packing positions of items to be handled by robotic arms in the retail industry. While existing research has explored aspects of robotic packing, there remains a significant gap in the development of comprehensive and user-friendly tools that can effectively determine the optimal placement of items within a container. This research aims to bridge this gap by creating a tool that empowers both human operators and autonomous systems packing processes through the accurate determination of packing positions. The aim is to contribute to the advancement of robotic automation in the retail industry by enhancing the efficiency, accuracy, and overall effectiveness of packing operations.

1.3 Objectives

This dissertation aims to develop a packing system that allows users to generate, manually or automatically, placement poses for variable-sized and shaped items within a variable-sized and shaped container, aligning with problems described in the Bin Packing Problem (BPP) literature, more specifically exploring the Container Loading problem (CLP). The algorithm prioritizes adaptability, allowing for future modifications and improvements to meet evolving needs, while considering the constraints of the developed model. This approach facilitates the autonomous operation of an automatic arm manipulator in packing scenarios.

The dissertation encompasses two distinct phases in the development of a packing solution for robotic manipulators:

1.3.1 Manual Packing Tool

The initial objective involves creating a manual packing tool. This tool allows operators to design the arrangement of items within a container. The process begins with the operator selecting the desired items and container, along with specifying the packing configuration. The operator defines the precise positions of each item within the container to achieve a collision-free solution. The resulting 3D model provides a visual representation of the final arrangement, which serves as a guide for the robotic manipulator during the physical packing process.

1.3.2 Automatic Packing Tool

The second goal is to develop an automatic packing tool based on a heuristic algorithm. This phase transitions from manual design to autonomous decision-making. A database of items is established, from which the user selects the desired items and the target container dimensions. The space organization algorithm determines the position of items within the container based on model constraints and item characteristics. This automated process eliminates the need for manual intervention, facilitating the packing process and potentially improving packing efficiency.

1.4 Dissertation Structure

The remainder of this dissertation is organized as follows. In chapter 2 the tools that are going to be used to approach the solution of the proposed problem are shown. In chapter 3, the related work is described. In chapter 4, the approach to solving the problem is presented. In chapter 5, the results are shown to compare with other benchmark proposed solutions. In chapter 6, a conclusion is drawn to reflect on the main objectives achieved during the dissertation phase, as well as the future works to be done and possible improvements.

Chapter 2

Technical and Theoretical Background

This chapter presents the technical and theoretical foundations for the robotic packing system. It begins with an overview of the Robot Operating System (ROS), outlining its components and their roles in communication and computation within robotic applications. The chapter then introduces RViz, a 3D visualization tool within ROS, and explains its functions in displaying and interacting with the simulated environment. The Flexible Collision Library (FCL) is then explored, detailing its utility in collision detection and covering concepts such as collision objects, algorithms, and geometric representations. Additionally, it explores the role of 3D transformations within FCL. Lastly, the chapter briefly discusses the Open Asset Import Library (Assimp), used for loading and processing 3D models in the Polygon File Format (PLY) file format, an essential element of the packing simulation.

2.1 Robot Operating System (ROS)

ROS is an open-source framework designed to assist in the development of robotic applications. It offers a collection of tools, libraries, and conventions that simplify the complex task of creating robust and scalable robotic software. ROS operates on a distributed computing model, where different processes (nodes) communicate with each other over a network, enabling modularity and flexibility.

2.1.1 Nodes

The fundamental units of computation in ROS. Each node is a process that performs a specific task within the robotic system. Nodes can run independently and communicate with other nodes to achieve the overall system functionality. These are used to interface with the ROS system. They initialize the node, manage parameters, and facilitate the creation of communication objects like publishers and subscribers [1].

2.1.2 Topics and Messages

Topics are named channels over which nodes exchange messages. Topics enable a many-to-many communication model, where multiple nodes can publish to and subscribe to the same topic. This is crucial for decoupling the system components and ensuring modularity. Messages Structured data units are used for communication over topics. ROS provides a wide range of predefined message types and allows for the creation of custom message types. Messages carry the data between nodes, enabling them to share information. Figure 2.1 illustrates a message being transmitted from a publisher node (depicted in orange) to a topic (represented by the blue box). This message is then received and processed by a subscriber node (shown in green).



Figure 2.1: ROS topic and message

2.1.3 Publishers and Subscribers

Publishers are objects that allow nodes to send messages to a topic. They are used to broadcast information to other nodes in the system. For instance, in the packing tools, publishers are used to send visualization data to RViz. Subscribers, on the other hand, are objects that allow nodes to receive messages from a topic. Subscribers listen to a topic and process incoming messages. Although subscribers are not explicitly shown in the provided scripts, they are essential for receiving data in other parts of a ROS-based system. Figure 2.2 illustrates a scenario where multiple nodes (depicted in orange) are publishing to a common topic (represented by the blue box), while numerous other nodes (shown in green) are subscribed to that topic, effectively receiving messages from the publishers.

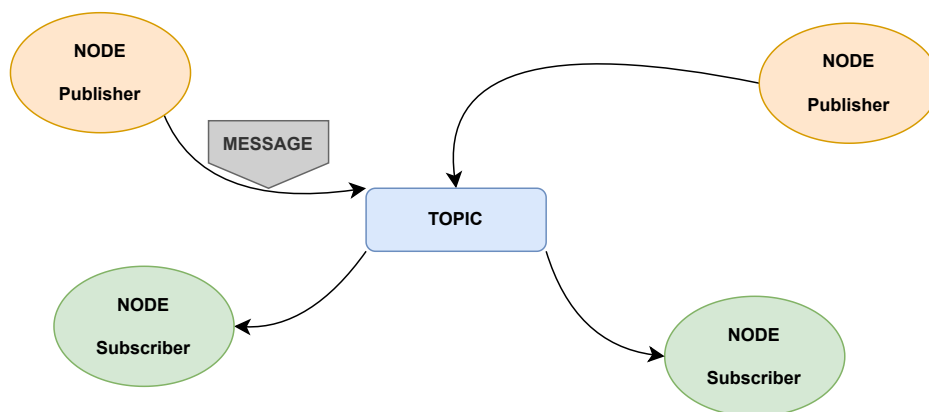


Figure 2.2: ROS publisher and subscriber

Further information in [1].

2.2 ROS Vizualization (RViz)

RViz, short for "ROS Visualization," is a 3D visualization tool for ROS. It allows users to visualize sensor data, robot models, and various markers in a three-dimensional space. RViz is an essential tool for developing and debugging robotic applications, providing real-time insights into the state and behavior of robotic systems.

- **Visualization** - It provides a graphical interface to visualize data from sensors, robot states, and other information, making it easier to understand and debug complex robotic systems.
- **Interaction** - Users can interact with the robot and its environment through interactive markers and other tools, facilitating tasks such as path planning, and environment manipulation.

RViz helps developers identify and fix issues by offering a clear view of how data flows through the system and how the robot interacts with its environment.

2.2.1 Markers

Markers in RViz are versatile visualization tools that allow users to represent various geometric shapes, lines, points, and other objects within a 3D environment. They are essential for visual debugging and interaction in robotic applications, making it easier to understand spatial relationships and the state of the system. These marker types are defined in the `visualization_msgs::Marker` message and can be used to represent different shapes and forms in the RViz environment. Figure 2.3 depicts the two types of markers used to define the objects.

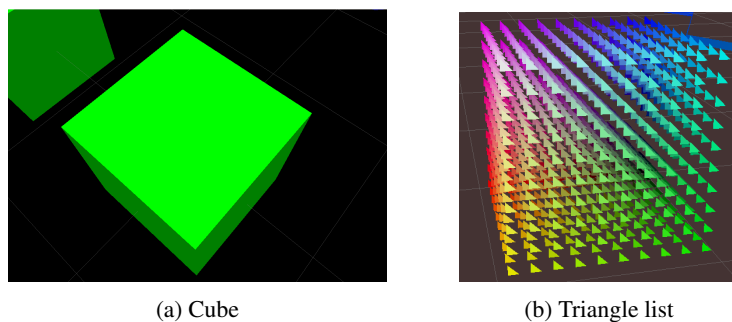


Figure 2.3: Markers. Figure from [2]

2.2.2 Marker Arrays

Marker arrays in RViz allow users to publish and manage multiple markers as a single message. This is particularly useful for visualizing complex scenes with many objects, as it simplifies the management and publication of numerous markers.

2.2.3 Interactive Markers

Interactive markers in RViz provide a way to interact with and manipulate objects in a 3D environment. They enable users to move, rotate, and modify objects using the RViz interface, making them invaluable for tasks that require user input or adjustment of objects in the scene. Some features of Interactive markers are:

- **User Interaction** - Allows users to interact with objects directly in the RViz environment.
- **Multiple Controls** - Supports various types of controls such as translation, rotation, and menus.
- **Feedback Mechanism** - Provides feedback to the application when the user interacts with the marker, enabling dynamic updates and responses.

Controls define how the marker can be interacted with. Included controls:

- **MOVE_AXIS** - Move the marker along a single axis. Interactive arrows representing the x-axis, y-axis, and z-axis translation in red, green, and blue color respectively.
- **ROTATE_AXIS** - Rotate the marker around a single axis. Interactive rings representing rotation around the x-axis, y-axis, and z-axis translation in red, green, and blue color respectively.

Figure 2.4 depicts an interactive marker with the respective **MOVE_AXIS** and **ROTATE_AXIS**.

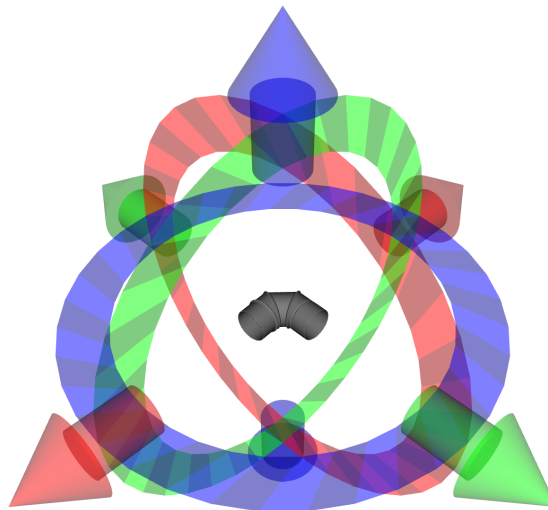


Figure 2.4: Interactive marker

2.2.3.1 Interactive Marker Server

The interactive marker server is a ROS tool that manages interactive markers and processes user interactions. It provides the infrastructure to create, update, and delete interactive markers while

handling feedback from user interactions. The server ensures that markers behave correctly in response to user actions and updates the RViz display accordingly. Figure 2.5 shows the Interactive Marker Server communication process with RViz.

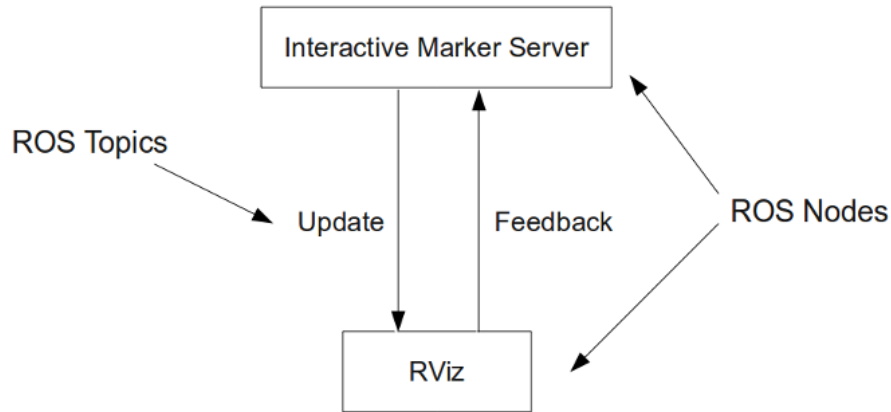


Figure 2.5: Interactive marker server architecture. Figure from [2]

The instantiation of the **interactive_markers::InteractiveMarkerServer** class allows for the creation of the interactive marker server. This server will manage all interactive markers for your application. An interactive marker is defined using the **visualization_msgs::InteractiveMarker** message. This message includes properties such as the frame of reference, name, description, scale, and pose of the marker. Further information in [2].

2.3 Flexible Collision Library (FCL)

FCL is a library designed for performing collision detection, proximity queries, and distance computations on geometric shapes. FCL is used in robotics, computer graphics, and simulation for its high performance and accuracy.

2.3.1 Collision

FCL employs algorithms to perform efficient collision detection between geometric shapes. These algorithms are designed to handle both simple and complex shapes, providing accurate results in real-time applications. In FCL, collision objects represent the shapes involved in collision detection. The **fcl::CollisionObjectd** class is used to define these objects, which can be created from different geometric primitives like boxes, spheres, and meshes. In FCL, collision objects represent the shapes involved in collision detection. The **fcl::CollisionObjectd** class is used to define these objects, which can be created from different geometric primitives like boxes, spheres, and meshes. The **fcl::collide** function is used to perform collision checks between two collision objects. It takes the collision objects, a collision request, and a collision result as inputs and returns whether a collision was detected. To perform a collision check, FCL uses **fcl::CollisionRequestd** and **fcl::CollisionResultd** classes. The request class defines the parameters for the collision query,

while the result class stores the outcome of the collision check. Figure 2.6 illustrates the collision detection mechanism employed by FCL.

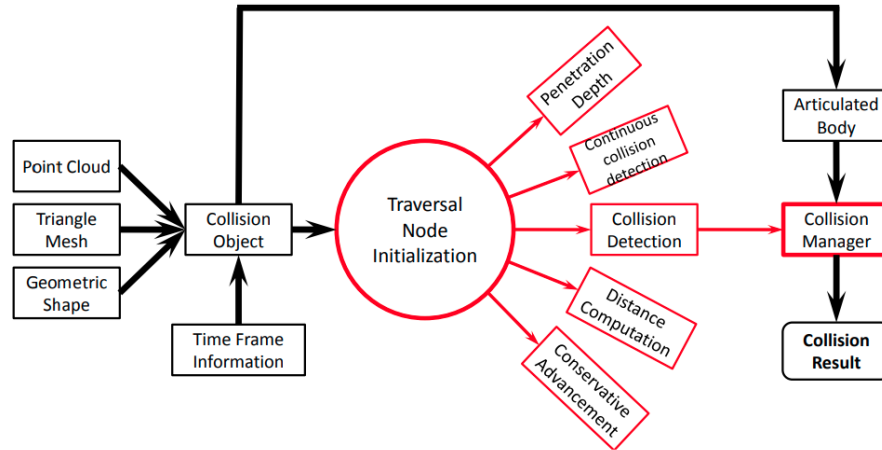


Figure 2.6: FCL collision manager. Image from [3]

2.3.2 FCL Objects

FCL supports different geometric primitives and provides efficient data structures for handling complex geometries. Here we will explain the key FCL objects like triangles and the processes involved in loading a mesh, along with important data structures like BVH, OBBRSS, and AABB.

Triangle - In FCL, triangles are the fundamental building blocks for representing complex 3D shapes, defined by three vertices in 3D space. Triangles are essential for creating mesh-based geometries, which are used for detailed and accurate collision detection.

BVH - A Bounding Volume Hierarchy (BVH) is a tree structure used to organize geometric objects hierarchically. Each node in the tree represents a bounding volume that encloses its child nodes. BVHs are used to accelerate collision detection and proximity queries by quickly eliminating pairs of objects that do not intersect. The `fcl::BVHModel` class represents a BVH of geometric primitives. It is templated on the type of bounding volume used, such as:

- **OBBRSS** - Oriented Bounding Box with RSS (OBBRSS) is a hybrid bounding volume that combines an oriented bounding box (OBB) with a rectangle-swept sphere (RSS). This combination provides a tighter fit and more efficient collision checks for arbitrary shapes compared to AABBs. The `fcl::OBBRSS` class represents an oriented bounding box combined with a rectangle-swept sphere. When creating a BVH model using OBBRSS bounding volumes, you can leverage the tighter fitting volumes for more efficient collision detection.

- **AABB** - An Axis-Aligned Bounding Box (AABB) is a simple and efficient bounding volume that is aligned with the coordinate axes. An AABB is defined by its minimum and maximum coordinates along each axis. It is used for broad-phase collision detection. The **fcl::AABB** class represents an axis-aligned bounding box.

2.3.3 3D Transformations in FCL

3D transformations are fundamental in robotics, computer graphics, and simulation. They involve changing the position and orientation of objects in three-dimensional space. In the context of FCL, transformations are essential for accurately representing and manipulating collision objects in space. In FCL, points or directions in 3D space are represented with vectors. They are used for defining the vertices of geometric shapes, the direction of transformations, and other spatial data. Vectors are often used to represent the vertices of triangles in a mesh.

Translational Transformations

Translation refers to moving an object from one position to another in 3D space. In FCL, translation is represented as a part of the **fcl::Transform3d** class. Translation is the process of moving every point of an object by the same distance in a given direction. In 3D space, translation can be represented mathematically by a vector addition. A translation in 3D space can be described by a translation vector $\mathbf{t} = (t_x, t_y, t_z)$. If a point $p = (x, y, z)$ is translated by t , the new position p' of the point is given by:

$$p' = p + t \quad (2.1)$$

In homogeneous coordinates, translation can be represented by a 4x4 matrix:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.2)$$

This matrix can then be used to transform a point $(x, y, z, 1)$:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2.3)$$

Rotational Transformations

Rotation in FCL involves changing the orientation of an object around a specified axis. This is represented using rotation matrices or quaternions. Rotation matrices are used to rotate points in

3D space. In FCL, rotation matrices are part of the **fcl::Transform3d** class and are represented by a 3x3 matrix. Rotation matrices for rotating about the x, y, and z axes by an angle θ are given by:

Rotation About the X-axis:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \quad (2.4)$$

Rotation About the Y-axis:

$$R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (2.5)$$

Rotation About the Z-axis:

$$R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

Composite rotations can be achieved by multiplying individual rotation matrices. For example, a rotation first about the x-axis, then the y-axis, and finally the z-axis can be represented by:

$$R = R_z(\theta_z)R_y(\theta_y)R_x(\theta_x) \quad (2.7)$$

Quaternions

Quaternions provide a more efficient and numerically stable way to represent rotations, avoiding gimbal lock and allowing for smooth interpolation between rotations. A quaternion $q = w + xi + yi + zk$ consists of four components: w, x, y , and z . For a unit quaternion representing rotation, the components satisfy:

$$w^2 + x^2 + y^2 + z^2 = 1 \quad (2.8)$$

In FCL, quaternions are used to define the rotation part of a transformation. The quaternion is first converted to a rotation matrix and then applied using the `linear()` method of the **fcl::Transform3d** class.

Further information in [4].

2.4 Open Asset Import Library (Assimp)

Assimp is an open-source library designed to import, export, and process 3D models. In the context of packing tools, Assimp is used for loading and processing 3D models stored in PLY files.

- **Model Loading** - Assimp's **Importer** class and its **ReadFile** method are used to read PLY files.
- **Mesh Processing** - Once a model is loaded, Assimp provides access to its mesh data, including vertices and faces. The **aiScene** and **aiMesh** classes encapsulate the model data, and methods are employed to extract vertex and face information for further processing.
- **Model Transformation and Optimization** - Assimp facilitates various post-processing steps, such as triangulation **aiProcess_Triangulate**, which ensures that the mesh is in a format suitable for collision detection. The vertices are processed to center the model, ensuring accurate placement and collision checks.

Further information in [5].

Chapter 3

Related Work

3.1 Introduction

Cutting and Packing (C&P) problems have been studied for many years, with increased attention recently. The term C&P covers a range of real-world applications, including trim loss reduction, container loading, pallet loading, and bin packing. Research in this area has evolved to address increasingly complex and diverse practical constraints. While the lack of a universally applicable solution has been a challenge, it has also driven ongoing innovation in this field. Figure 3.1 shows the number of papers on C&P from 1974 to 2021.

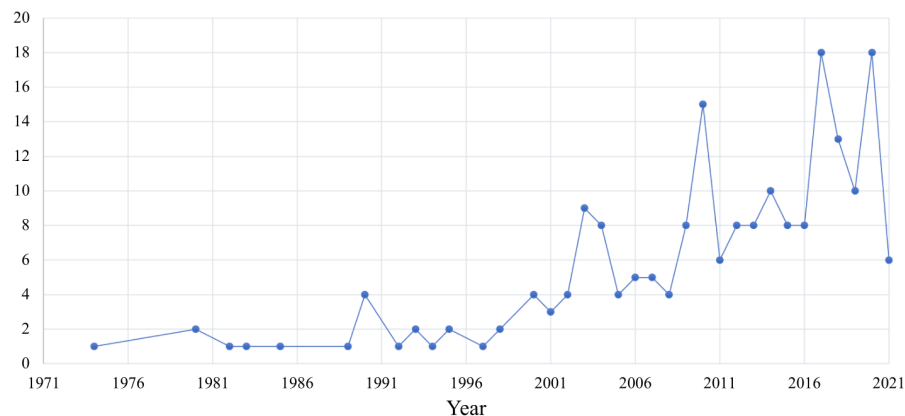


Figure 3.1: Number of publications on "Three Dimensional Packing Problems". Figure from [6]

The earliest reference to a packing problem appeared in 1980 [7], proposing a placement heuristic for filling a container with boxes. The first attempt to define and classify C&P problems was made in 1990 [8]. Recognizing the varied terminology and classifications used, the author introduced a typology to systematically categorize these problems based on dimensionality, the nature of items and objects, and the processes involved.

3.2 Problem Formulation and Classifications

Dyckhoff in [8] proposes a typology of C&P problems based on four main characteristics:

1. **Dimensionality** - This refers to the number of dimensions needed to represent the problem. It can be one-dimensional (1D), two-dimensional (2D), three-dimensional (3D), or N-dimensional (ND), where N is greater than 3.
2. **Kind of Assignment** - This distinguishes between two scenarios:
 - (B) All objects (large items like sheets or bins) are used, and only a selection of small items needs to be assigned.
 - (V) A selection of objects is used, and all small items must be assigned.
3. **Assortment of Large Objects** - This describes the variety of large objects:
 - (O) There's only one large object.
 - (I) All large objects have the same shape and size (identical figures).
 - (D) The large objects have different shapes and sizes (different figures).
4. **Assortment of Small Items** - This characterizes the variety of small items:
 - (F) There are a few small items, usually of different shapes and sizes.
 - (M) There are many small items, and most of them have different shapes and sizes.
 - (R) There are many small items, but they belong to a relatively small number of different shapes and sizes (non-congruent figures).
 - (C) All small items have the same shape and size (congruent figures).

By combining these characteristics, Dyckhoff created a system of 96 potential problem types, each denoted by a four-tuple of symbols (e.g., 1/B/O/F represents a one-dimensional problem where one large object is packed with a selection of a few small items). This system aimed to classify various C&P problems found in the literature and practice. The need for a revised typology for C&P problems, as presented by Wäscher et. al. [9], arose due to limitations in the existing typology proposed by Dyckhoff et. al. [8].

Dychoff's typology presented some problems, such as:

Ambiguity and Overlap - Dyckhoff's typology faced challenges in assigning certain C&P problems to unique types of problems. A problem could be classified under multiple categories, leading to ambiguity. Additionally, there was an overlap between problem categories F (few items of different shapes) and M (many items of different shapes), making it difficult to distinguish them clearly.

Inconsistent Categorization - The existing typology also exhibited inconsistencies in its application. The Strip Packing Problem typically involves packing rectangle-shaped items into a single rectangle container of fixed width and minimum length, which Dyckhoff [8] classified under 2/V/D/M (several different large objects), even though it involves only one large object. This misalignment created confusion and highlighted the need for a more consistent classification scheme.

Lack of Homogeneity - The existing typology could not result in homogeneous problem categories, which are essential for developing generalized models and algorithms. For example, one-dimensional cutting problems with a large number of small items of relatively few different shapes could be categorized under 1/V/D/R, regardless of whether the standard material came in a few groups of identical sizes or entirely different sizes. However, these two scenarios require distinct solutions, indicating a lack of homogeneity within the category.

3.2.1 Typology

To address the limitations of the earlier typology, Wäscher [9] proposed a refined typology aimed at providing a more comprehensive and consistent classification of C&P problems. The new typology introduced additional criteria and refined existing ones to ensure clarity and uniqueness in problem categorization. It also aimed to align with the evolving landscape of C&P research and accommodate new problem types. A typology for C&P problems is a systematic organization of these problems into homogeneous categories based on a set of characterizing criteria. The goal of a typology is to provide a structured framework for understanding and addressing real-world problems. In the context of C&P, a typology serves as a foundation for developing models, algorithms, and problem generators, as well as for categorizing relevant literature. To achieve this, problem types are defined using a combination of five key criteria: dimensionality, kind of assignment, assortment of large objects, assortment of small items, and shape of the small items.

3.2.1.1 Dimensionality

Dimensionality in C&P problems refers to the minimum number of spatial dimensions required to fully represent the geometric arrangement of the items being cut or packed. It is the minimum number of measurements (like length, width, and height) needed to describe the shapes involved in the problem. This is a key factor in classifying C&P problems, as it directly impacts the complexity of the problem and the types of solution methods that can be applied. Dyckhoff [8] identifies four categories: one-dimensional, two-dimensional, three-dimensional, and multi-dimensional (more than three dimensions). Wäscher et al. [9] adopt the same criterion but focus on the first three dimensions as they are the most common in practical applications.

3.2.1.2 Kind of Assignment

Wäscher et al. [9] explain that the "kind of assignment" criterion refers to how small items (output) are assigned to large objects (input). It is important to notice that the "packing value", in these contexts can be defined as cost, revenue, or material quantity. Wäscher based the categorization of C&P problems into two main basic types:

Output Maximization - In output maximization problems, the quantity of containers is limited. The goal is to maximize the packing value of the small items that can fit into these containers, thus maximizing the number of items (output) to be packed in the limited containers.

Input Minimization - In input minimization problems, the quantity of large objects (containers) is unlimited. The goal is to minimize the number of containers to accommodate all the small items. Since all small items must be included, the focus is on selecting the optimal set of large containers.

The "packing value" in these contexts can be defined as cost, revenue, or material quantity. Often, it's assumed to be proportional to the size (length, area, or volume) of the objects or items. In some cases, both output and input minimization can be translated into "waste minimization," which aims to minimize unused portions of large objects. While these two basic types cover many C&P problems, the authors acknowledge that real-world problems might involve selection problems for both large and small items or have multiple objectives. These more complex scenarios are considered problem variants within the typology.

3.2.1.3 Assortment of Small Items and Large Objects

Both papers [8] [9] establish a fundamental distinction between small items and large objects as a foundational concept in their typologies for C&P problems. Small items are to be packed or cut, while large objects are the containers or spaces into which the small items are placed. The categorization of sets of small items and large objects is crucial for classifying the base problem types. For small items:

Identical - All items have the same shape and size.

Weakly Heterogeneous - Items can be grouped into a relatively small number of classes with identical shapes and sizes.

Strongly Heterogeneous - Very few items have the same shape and size.

In Figure 3.2 it is observed three packing solutions with three different arrangements of small items, identical, weakly heterogeneous, and strongly heterogeneous.

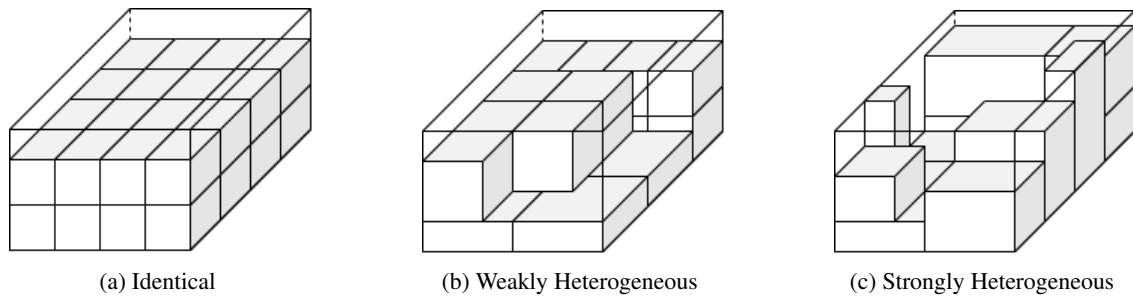


Figure 3.2: Packing Solutions

For containers, Wäscher [9] first introduced two main categories that would later be subdivided into five different types of categorization for the assortment of large objects:

One large object - This can be further divided into cases where all dimensions are fixed or where one or more dimensions are variable.

Several large objects - In this case, all dimensions are assumed to be fixed, and the objects can be identical, weakly heterogeneous, or strongly heterogeneous.

Figure 3.3 depicts the small items and large objects assortment presented in [9].

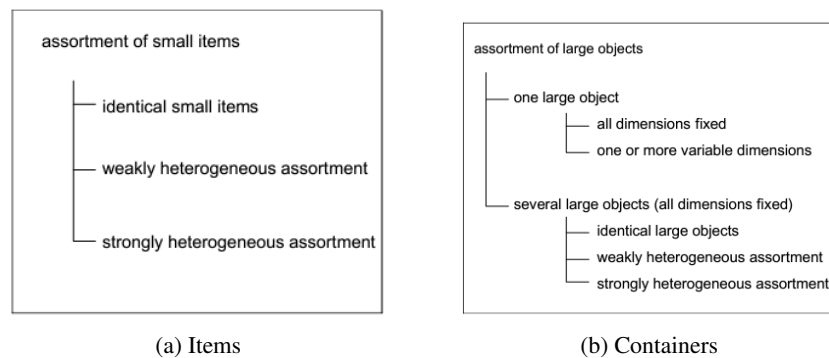


Figure 3.3: Assortment of small items and containers. Figure from [9]

The distinction between the assortment of small items and large objects is crucial to better characterizing the problem and subdividing it into different types. The typology differentiates between situations where there is only one large object, multiple identical large objects, and multiple non-identical large objects. The typology considers whether small items are few and different, many and of few types, or many and of many types. It's important to note that large objects define larger packages where small items will be packed, such as regular containers, container ships, or trucks. The small items define the items to be packed inside these large objects.

In two-dimensional and three-dimensional problems, refined problem types are defined by differentiating between regular (e.g., rectangles, circles, boxes) and irregular small items. In two-dimensional problems, regular items can be further classified as rectangular, circular, or other

shapes. The authors assume that rectangular items are placed orthogonally (at right angles) and that the set of small items consists of either only regular or irregular shapes. Problems involving non-orthogonal placement or a mix of regular and irregular shapes are considered variations. Table 3.1 demonstrates all base C&P Problems typologies for output maximization.

Table 3.1: Output maximization. Table adapted from [9]

Large objects characteristics		Small items assortment		
		Identical	Weakly Heterogeneous	Strongly Heterogeneous
All dimensions fixed	One large object	Identical Item Packing Problem (IIPP)	Single Large Object Placement Problem (SLOPP)	Single Knapsack Problem (SKP)
	Identical		Multiple Identical Large Object Placement Problem (MILOPP)	Multiple Identical Knapsack Problem (MIKP)
	Heterogeneous		Multiple Heterogeneous Large Object Placement Problem (MHLOPP)	Multiple Heterogeneous Knapsack Problem (MHKP)

IIPP - single container and the items are identical.

SLOPP - single container and the items are weakly heterogeneous.

SKP - single container and the items are strongly heterogeneous.

MILOPP - multiple identical containers and weakly heterogeneous items.

MIKP - multiple identical containers and strongly heterogeneous items.

MHLOPP - multiple containers and weakly heterogeneous items.

MHKP - multiple containers and items are strongly heterogeneous.

Table 3.2 demonstrates all base C&P Problems typologies for input minimization.

Table 3.2: Input minimization. Table adapted from [9]

Large objects characteristics		Small items assortment	
		Weakly Heterogeneous	Strongly Heterogeneous
All dimensions fixed	Identical	Single Stock-Size Cutting Stock Problems (SSSCSP)	Single Bin-Size Bin Packgin Problems (SBSBPP)
	Weakly heterogeneous	Multiple Stock-Size Cutting Stock Problems (MSSCSP)	Multiple Bin-Size Bin Packgin Problems (MBSBPP)
	Strongly heterogeneous	Residual Cutting Stock Problems (RCSP)	Residual Bin Packgin Problems (RBPP)
One large object variable dimension(s)		Open dimension problem (ODP)	

SSSCSP - containers are identical and items are weakly heterogeneous.

SBSBPP - containers are identical and the items are strongly heterogeneous.

MSSCSP - containers and the items are weakly heterogeneous.

MBSBPP - containers are weakly heterogeneous and items are strongly heterogeneous.

RCSP - containers are strongly heterogeneous and items are weakly heterogeneous.

RBPP - containers and items are strongly heterogeneous.

ODP - one container where the items are identical, weakly heterogeneous, or strongly heterogeneous.

In this Open Dimension Problem (ODP) in particular, the objective is to minimize the variable-sized dimension (2D being the length of the container, 3D being the length or height of the large container).

3.2.2 Constraints

In the context of C&P, constraints are conditions that must be met to ensure the feasibility and practicality of packing solutions. Bischoff and Ratcliff [10] were the first to identify these constraints, and subsequent research by Bortfeldt and Wäscher [11] further refined their classification. Bortfeldt and Wäscher [11] proposed a new classification that divides practical constraints into hard and soft constraints, providing a clearer understanding of the relationship between different constraints presented in C&P Problems.

- **Hard Constraints** - Mandatory conditions that must be strictly satisfied for a solution to be considered feasible. Violating a hard constraint renders the solution invalid. Examples of hard constraints include geometric constraints (e.g., items cannot overlap) and practical constraints like weight limits (the total weight of packed items cannot exceed the container's capacity).
- **Soft Constraints** - Desirable but not mandatory. While adhering to soft constraints is preferred, slight violations may be tolerated within certain limits. These constraints often involve trade-offs with other objectives, such as maximizing container space utilization. Examples of soft constraints include weight distribution (ensuring an even spread of weight) and positioning constraints (placing items of the same customer together).

The table 3.3 summarizes all the main constraints found in the literature:

Table 3.3: Constraints [6]

Hard Constraints	Soft Constraints
Orientation	Loading Priority
Weight Limit	Complete Shipment
Weight Distribution	Allocation
Stability	Pattern Complexity
Stacking	Customer Positioning
Physical Positioning	

Orientation

In 3D-PP, there are six possible orthogonal orientations for a rectangular item [11]. This is because each of the three dimensions of the item (length, width, and height) can be aligned with any of the three axes of the container (x, y, and z), and each of these alignments can be flipped 180°. However, not all of these orientations may be allowed due to practical considerations. Orientation constraints are typically implemented as hard constraints, meaning that any solution that violates these constraints is considered infeasible [11]. Bortfeldt and Wäscher categorize these constraints into five cases:

1. **Single Orientation** - Only one specific orientation is allowed for each item, both vertically and horizontally. This means the items cannot be rotated at all. This constraint is often used to simplify the problem and reduce computational complexity
2. **Single Vertical Orientation** - Each item can only be oriented in one way vertically, but it can be freely rotated horizontally in 90-degree increments.
3. **Two Prohibited Vertical Orientations** - Up to two vertical orientations may be prohibited for each item, while the horizontal orientation remains free. This case includes case 2 but also allows for items that can be freely rotated.

4. **Five Prohibited Orientations** - Up to five orientations may be prohibited for each item. This is the most general case and includes all the previous cases.
5. **No Restrictions** - There are no constraints on the orientation of the items. They can be freely rotated in both vertical and horizontal directions.

When items are not placed orthogonally within a container (i.e., the surfaces of the items are not aligned parallel to the floor and walls of the container), the problem no longer falls under the standard category of orthogonal 3D C&P problems [11]. In Figure 3.4 there are represented the six possible orthogonal positions for a regular item.

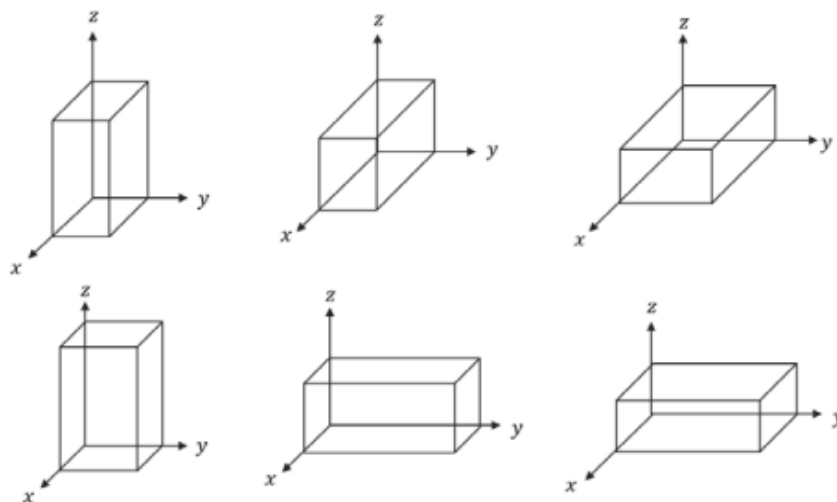


Figure 3.4: Possible orthogonal orientations. Figure from [12]

Weight Limit

The weight limit constraint in CLPs is a safety constraint that ensures the total weight of the packed items does not exceed the maximum weight capacity of the container [6]. This constraint is crucial for preventing damage to the container, and the cargo, and ensuring the safety of those involved in the transportation process [11]. In the literature, weight limits are typically addressed as hard constraints, meaning that any solution violating this limit is considered infeasible [11]. Exceeding the weight limit can have serious consequences, such as structural damage to the container, instability during transport, and potential accidents.

In addressing weight limit constraints, researchers have employed approaches to ensure feasible and efficient container loading solutions. Terno [13] ensures this constraint using a branch-and-bound approach that incorporates a layer-wise loading strategy with optimal two-dimensional loading patterns. This method prioritizes the creation of complete layers of items, which aligns with practical loading practices and often leads to efficient space utilization.

Egeblad [14] addresses weight constraints in the context of furniture loading, where items have varying weights and fragilities. Their approach involves a multi-stage heuristic that combines geometric analysis, template-based packing, and local search optimization. By considering

weight distribution and stability alongside profit maximization, their method achieves high loading utilization while ensuring the safe transportation of furniture items.

Bortfeldt [15] extends the weight constraint consideration to the Capacitated Vehicle Routing Problem, where the weight of loaded items directly impacts vehicle capacity and fuel consumption. The author proposes a hybrid algorithm combining tabu search for routing and tree search for packing, effectively addressing both the routing and loading aspects of the problem while adhering to weight restrictions.

Weight Distribution

Bortfeldt and Wäscher [11] define the weight distribution constraint, also known as the load balance constraint, as a requirement to distribute the weight of the cargo evenly across the container floor. This ensures that the center of gravity of the items is close to the geometric center of the container floor, which helps prevent the cargo from shifting during transport.

Bischoff and Ratcliff [10] explain that weight distribution is an important consideration in CLPs for two main reasons:

1. **Handling Operations** - Uneven weight distribution can make certain handling operations, such as lifting the container onto a ship, difficult or even impossible.
2. **Axle Loading** - When a container is transported by truck, uneven weight distribution can result in unacceptable axle weight distributions, potentially leading to safety hazards and road damage.

In the context of the Multi Container Loading Problem (MCLP), Alonso et. al. [16, 17] formulate the weight distribution constraint to ensure that the weight on each axle of the truck does not exceed a predetermined limit. The weight borne by each axle is a function of the total weight loaded onto the truck and the position of the center of gravity of the load. Figure 3.5 shows a truck with the positions of the axles.

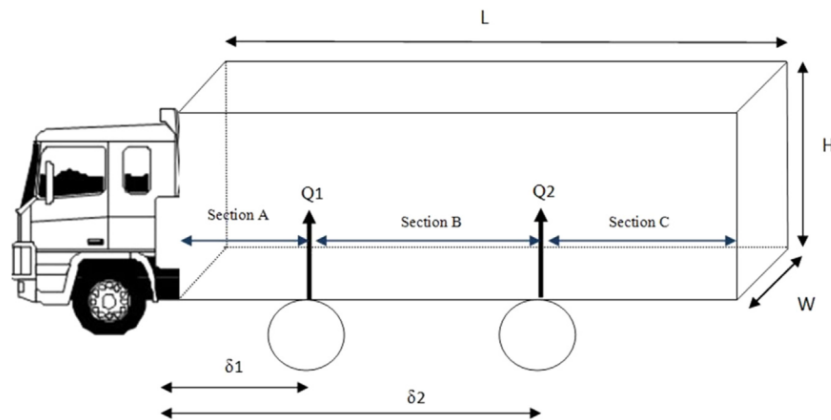


Figure 3.5: Dimensions and axles positions of the truck. Figure from [17]

The weight distribution constraint ensures that the load's center of gravity is located between the two axles, δ_1 and δ_2 . This helps to prevent the truck from tipping over.

Alonso et. al. [18] extend the weight distribution constraint to consider the weight distribution along the height of the container. This is important to prevent the cargo from shifting during transport. The stability of the load is affected by the weight distribution, as well as the shape and size of the items. There are several different ways to improve the stability of the load, such as using interlocked stacking or adding additional support. Davies and Bischoff [19] address the weight distribution problem in container loading, focusing on the need for an even distribution of weight within a container to ensure stability and safe handling during transportation. The authors propose a new heuristic approach to achieve high space utilization while maintaining an even weight distribution. The approach involves dividing the container into vertical cross-sections, walls, and packing boxes within these sections according to specific rules as depicted in figure 3.6. The authors evaluate the performance of their proposed approach against existing methods and demonstrate its effectiveness in producing loading arrangements that combine high space utilization with an uniform weight distribution. They also discuss the potential for further improvements by incorporating additional considerations such as the height of the center of gravity and the fragility of items.

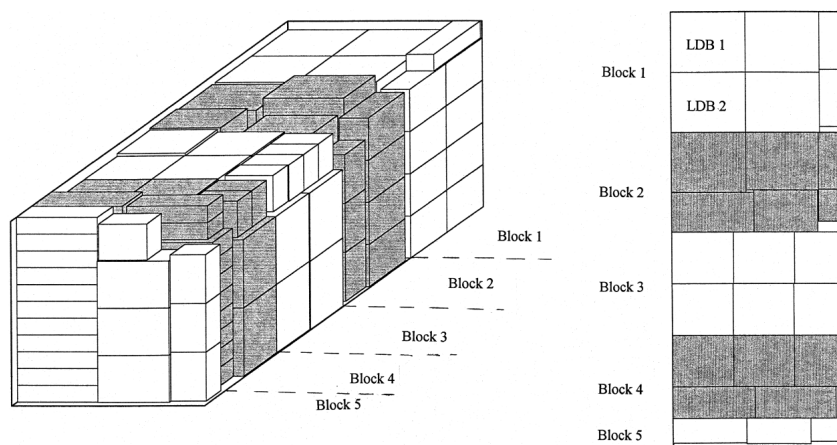


Figure 3.6: Vertical cross-sections solution. Figure from [19]

Stability

The stability constraint in 3D packing problems is a crucial factor to consider, as it ensures the integrity of the cargo and the safety of the goods and personnel involved in the loading and transportation process. Unstable loads can lead to damage or injury.

Bortfeldt and Wäscher [11] highlight that load stability is often prioritized alongside container space utilization. They differentiate between two types of stability:

- **Vertical Stability** - This prevents items from falling due to gravity when the container is stationary. It is typically addressed by requiring that the base of each box be fully or partially

supported by the container floor or other boxes. Some approaches even allow for partial support, where a minimum percentage of the box's base needs support.

- **Horizontal Stability** - This ensures items do not shift when the container is in motion. It is often evaluated by the percentage of items not in direct contact with other items or container walls. Interlocking box layers and compacting procedures are common strategies to enhance horizontal stability.

Bortfeldt and Wäscher [11] emphasize that achieving high container space utilization, especially in output maximization problems, can inherently lead to load stability. They also note that additional measures, like fillers or supports, can further enhance stability. However, they caution that relying solely on space utilization or a single stability measure may not guarantee a completely stable load.

Ali et. al. [6] state the importance of stability constraints, particularly in online packing problems. They categorize stability into static and dynamic stability, aligning with Bortfeldt and Wäscher [11] vertical and horizontal stability concepts.

- **Static Stability** - This focuses on preventing items from falling during loading operations and is often ensured by requiring full or partial support of the box's base.
- **Dynamic Stability** - This addresses the stability of the load during transportation and is further divided into longitudinal and lateral stability. Longitudinal stability prevents forward and backward movement during acceleration and braking, while lateral stability prevents sideways movement during turns.

Ali et. al. [6] highlight that most online packing studies have overlooked practical constraints like stability. They emphasize the need for more research, especially considering the real-world applications of online 3D packing problems. Figure 3.7 is a 3D view of some examples of stable and unstable packing.

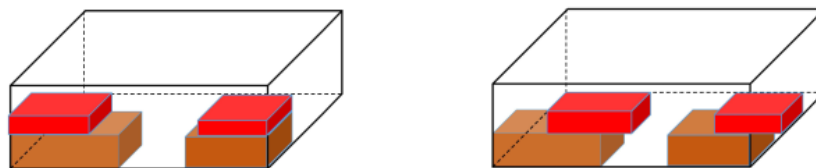


Figure 3.7: Stable vs unstable packing. Figure from [20]

Bischoff [21] introduces a new heuristic approach to address CLPs where the cargo has varying load-bearing strengths. The method focuses on ensuring that the weight of each item doesn't exceed its maximum capacity to avoid damage. The heuristic is integrated into a search algorithm that optimizes parameter settings, aiming for quick and efficient solutions with minimal user expertise. It outperforms existing methods in problems with load-bearing constraints and performs well even when these constraints aren't a concern.

Carpenter and Dowsland [22] emphasize that maximizing the number of boxes isn't the sole concern. They propose criteria for assessing the stability and clampability of pallet stacks, considering factors like box contact, base support, and the presence of guillotine cuts. These criteria aim to ensure that pallet loads are suitable for transportation and storage, addressing real-world constraints often overlooked in theoretical models. De Castro Silva et al. [23] present a greedy search heuristic for the three-dimensional bin packing problem with a focus on packing stability. The algorithm prioritizes stability by considering the center of gravity and equilibrium conditions of packed items. It assumes homogeneous boxes with equal density, simplifying the analysis of forces and moments. The heuristic aims to minimize the number of bins used while ensuring that the resulting packing is physically stable. Figure 3.8 is a side view of some examples of unstable box arrangements.

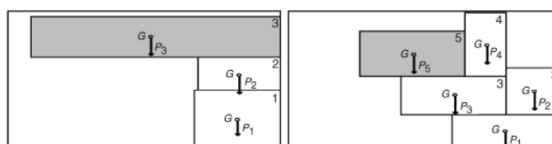


Figure 3.8: Example side-view of unstable box arrangements. Figure From [23]

Oliveira et al. [24] investigate the relationship between support factors, mechanical equilibrium, and grids in cargo stability for the CLP. They propose approaches based on mechanical equilibrium to ensure stability and formulate the problem as an integer programming model solved with a branch-and-cut algorithm. The study compares different grid configurations and demonstrates that their approaches can achieve higher packed volumes than traditional support factor methods. Techanitisawad and Tangwiwatwong [25] propose a genetic algorithm-based heuristic for container selection and loading problems. The algorithm considers practical constraints like box orientation, stack priority, stack stability, and container stability. It aims to minimize transportation costs by selecting an optimal set of containers and maximizing volume utilization through efficient loading patterns. The genetic algorithm approach allows for flexibility and adaptability in solving these complex problems.

Stacking

Stacking constraints also referred to as load-bearing constraints, are crucial in the CLP to ensure the safe transportation of goods. These constraints consider the limitations in how boxes can be stacked on top of each other due to their load-bearing capacity.

Bortfeldt and Wäscher (2013) provide a comprehensive explanation of stacking constraints, emphasizing the factors that influence a box's load-bearing strength. These factors include:

- **Material of the boxes** - The material used to construct the box significantly impacts its strength. For example, a wooden box is generally stronger than a cardboard box.

- **Content of the boxes** - The type and density of the contents play a role. A box filled with books can withstand more weight than a box filled with delicate glassware.
- **Strength of the side walls** - The structural integrity of the box's side walls, often measured by an edge crush test, is crucial in determining its load-bearing capacity.
- **Transportation conditions** - Environmental factors during transportation, such as humidity and the duration of the journey, can weaken a box's structure and reduce its load-bearing strength.

Ali et. al. [6] also discuss stacking constraints, focusing on how they prevent damage by restricting stacking configurations. They mention setting limits on weight and pressure and prohibiting specific stacking configurations. However, their explanation lacks the depth provided by Bortfeldt and Wäscher, who underlie factors influencing load-bearing strength and the nuances of different approaches. In the literature, various approaches have been used to address stacking constraints:

- **Fragility** - Fragile items are designated as unable to bear any weight and thus cannot have other items stacked on top of them.
- **Stacking Restrictions** - Certain box types are restricted from being stacked on top of other types. For instance, heavier boxes should not be placed on top of lighter ones.
- **Stacking Limits** - A maximum limit is set on the number of boxes that can be stacked vertically to prevent the bottom boxes from being crushed.
- **Weight and Pressure Limits** - Explicit limits are defined for the maximum weight or pressure a box can withstand per unit area. This can vary depending on the box's orientation.

Physical Positioning

The physical positioning constraint in the context of 3D-PPs refers to restrictions on the placement of boxes within a container. These constraints are typically determined by the attributes of the box, such as its size, weight, or contents, and may necessitate placing a specific item in a designated location or area within the container.

In the paper by Bortfeldt and Gehring [26], the container is divided into zones (e.g., door, bottom), and certain box types are restricted to specific zones. This could be due to the size or weight of the box, where heavier items might need to be placed at the bottom for stability, or larger items might need to be placed near the door for easier access. In another example from Haessler and Talbot [27], the constraint arises from the limited maneuverability of a clamp truck used for loading railcars. Large and heavy items should not be assigned to the doorway due to the truck's difficulty maneuvering.

In Moura and Bortfeldt [28], the authors discuss a two-stage packing problem where the first stage involves packing boxes onto pallets and the second stage involves loading these pallets into

trucks. In this context, physical positioning constraints could arise due to the need to place heavier boxes at the bottom of the pallet for stability or to arrange boxes in a way that facilitates unloading at different customer locations. Terno et al. [13] propose an efficient approach for the multi-pallet loading problem, which involves loading a set of distinct products onto pallets while minimizing the number of pallets needed. While not explicitly stated, physical positioning constraints could be incorporated into their approach to ensure the stability of the loaded pallets, for example, by placing heavier items at the bottom or distributing weight evenly.

The physical positioning constraint adds an extra layer of complexity to the 3D-PPs by introducing real-world considerations into the problem. It ensures that the packing solution is not only geometrically feasible but also practical and safe for loading, transportation, and unloading processes.

Loading Priority

Bortfeldt and Wäscher [11] define loading priority as the degree of importance assigned to items to be loaded, based on factors like delivery deadlines or product freshness. It arises when there isn't enough space to load all items, necessitating a choice. These priorities can be either:

- **Hard** - A subset of items must be loaded
- **Soft** - Other items should be loaded, potentially with varying priority levels.

Ali et. al. [6] define loading priority as assigning a degree of importance to items based on delivery deadlines and requirements, such as product freshness. It is mainly considered in output maximization problems, where, due to the limited number of available containers, only a set of small items can be packed.

Ren et. al. [29] introduce an algorithm for the SLOPP and SKP that is also utilized for multiple CLPs. Items are assigned either low or high priority. However, this algorithm focuses on hard constraints, ensuring that all high-priority items are loaded while maximizing container volume utilization. In Figure 3.9 a solution obtained by [29] separates red boxes (high-priority ones) from white boxes (low-priority ones).

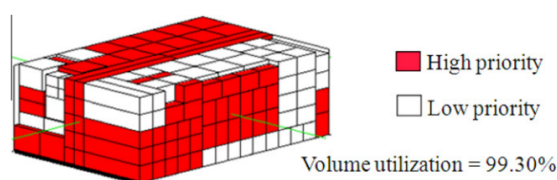


Figure 3.9: High-priority low-priority solution. Figure From [29]

Sheng et. al. [30] studied a single CLP with expiring orders, where all items within an order must be packed together or left behind. The authors develop a simulated annealing algorithm to select a collection of orders and a tree search method to pack the selected orders into the container,

considering loading priority as one of the constraints. However, the paper does not elaborate on how the loading priority is specifically defined or implemented within their algorithm.

Complete Shipment

Bortfeldt and Wäscher [11] define complete shipment constraints as a constraint that arises when a customer's order must either be entirely filled or not filled at all. This constraint is most applicable to output (value) maximization problems, where there is limited container space and some items will inevitably be left behind. The authors give the example of a kitchen unit being packed separately and needing to be assembled at the customer's house; incomplete shipments would not be permitted in this case. Ali et. al. [6] recall this definition, stating that the complete shipment constraint ensures that "the entirety of each order's items must be contained within a single shipment." They further elaborate that this constraint has been considered in several studies and provide the example of a company that receives daily orders where all items in an order must be placed in the same container.

Eley et. al. [31] addresses the complete shipment constraint in the context of MHLOPP. The author identifies four variations of this constraint, where cargo subsets must either be loaded completely or left behind entirely. These variations range from a single cargo subset with identical boxes to multiple cargo subsets with varying box types and combinations. Eley et. al. [31] propose a heuristic solution method to address these different scenarios, ensuring that complete shipments are prioritized while maximizing container space utilization.

Allocation

Bortfeldt and Wäscher [11] describe allocation constraints as those that arise in multiple CLPs. There are two types of allocation constraints:

- **Connectivity constraints** - Items of a particular subset must be loaded into the same container. This could be because they are shipped to the same destination or because a customer wants their entire order in one shipment.
- **Separation constraints** - Certain items or classes of items must not be loaded into the same container. The authors give the example of food and perfumery articles not being allowed to be shipped together.

Ali et. al. [6] do not explicitly mention allocation constraints but do discuss a similar concept in their description of customer positioning constraints. They state that "*some items such as foodstuff and chemicals should not be placed next to each other*". This aligns with the separation constraint described by Bortfeldt and Wäscher [11]

Terno et al. [13] examine allocation constraints within the SSSCSP encountered in manual pallet loading. To minimize loading/unloading time, their algorithm prioritizes loading pallets with items of a single type when possible. If there are not enough items of one type to fill a pallet,

the algorithm aims to distribute these items across as few pallets as possible, striking a balance between minimizing pallet usage and maintaining efficient loading/unloading operations.

Pattern Complexity

Bortfeldt and Wäscher [11] explain that complex loading patterns may not be suitable for manual loading, as they can be difficult for personnel to understand and implement, leading to increased packing time. Additionally, automated packing technologies may not always be able to handle complex arrangements, potentially requiring more labor and higher costs. The authors mention the guillotine-cutting constraint as a commonly considered complexity constraint. Guillotine patterns are created by a series of cuts parallel to the container faces, making them easier to describe and pack. Ali et. al. [6] provide a similar definition, stating that the pattern complexity constraint "is related to the degree of difficulty in understanding and implementing the loading pattern by robots or staff." They also mention the guillotine cutting pattern and note that it is not always appropriate, as it can lead to unstable loads during transportation.

Martello et al. [32] introduce the concept of robot-packable patterns in the context of the 3D-PP. These patterns are designed to be easily implemented by robots that can lift and release boxes at designated positions. The authors propose an algorithm to generate robot-packable patterns, focusing on optimizing container space utilization while ensuring that the resulting patterns are feasible for robotic packing systems.

Customer Positioning

Bortfeldt and Wäscher [11] refer to customer positioning constraints as grouping constraints. These constraints require that certain subsets of items be placed close together or within a certain distance of each other. This is often done to make it easier to check if an order is complete during loading and unloading, reducing the number of errors. Ali et. al. [6] uses the term "customer positioning constraint" and provides a similar definition. They state that this constraint requires a specific subset of items to be packed (or not packed) within a predefined distance of each other. The authors note that packing items belonging to the same customer close together can facilitate the unloading process.

Christensen and Rousøe [33] investigate the customer positioning constraint in the context of container loading with multi-drop constraints. They address the challenge of delivering items to multiple customers in a specific sequence, aiming to minimize unnecessary unloading and reloading operations. Their approach involves loading items into the container based on the delivery sequence, ensuring that items for earlier stops are accessible without having to move items intended for later stops. This strategy optimizes the unloading process and improves overall efficiency in multi-drop scenarios.

3.3 Solution Approaches

Packing problems are a subset of C&P Problems, with the distinction arising primarily from their real-world applications. Packing problems focus on efficiently placing items into containers for logistics and transportation purposes. The complexity of packing problems increases significantly when dealing with 3D objects, such as boxes being loaded into containers. These problems are classified as NP-hard, meaning there's no known efficient algorithm to find optimal solutions for large-scale instances [9, 6].

3.3.1 Exact Methods

Exact methods for 3D packing problems aim to find the optimal solution, guaranteeing the best possible arrangement of items within a three-dimensional space. These methods are typically grounded in mathematical optimization, ensuring a mathematically proven best outcome. However, the trade-off for this guarantee is computational complexity, as the time to find the optimal solution can grow exponentially with the problem's size and complexity. This limitation makes exact methods suitable for small to medium-sized problems but often impractical for large-scale problems [6].

A prominent exact method is Mixed-Integer Linear Programming (MILP), which formulates the packing problem as a mathematical model. The objective is often to minimize the number of containers used or maximize the packed items' value. Constraints within the model serve several crucial purposes:

- **Non-overlap** - Ensuring items don't occupy the same space.
- **Containment** - Ensuring items fit entirely within the container's dimensions.
- **Practical Constraints** - Addressing real-world considerations like weight distribution, load-bearing capacity, and stability.

Early research by Chen et. al. [34] introduced a MILP model considering multiple container and item sizes without orientation restrictions. Subsequent studies have refined this approach, incorporating practical constraints like weight distribution [35], stability, fragility, and load priorities [20]. Hifi et. al. [36] also contributed a MILP formulation aimed at minimizing the number of bins used.

Another exact method, the Branch-and-Bound Algorithm, systematically explores the solution space by making sequential decisions about item placement. It employs upper and lower bounds to prune branches that cannot lead to optimal solutions, enhancing efficiency. Martello et. al. [37] pioneered this approach for the 3D-PP, although without initially considering practical constraints. Fasano [38] also discusses a branch-and-bound algorithm to solve a Mixed-Integer Programming (MIP) model for the three-dimensional bin packing problem. The paper explains that the non-intersection constraints make it difficult to find optimal solutions using standard MIP algorithms.

Therefore, the author proposes a MIP-based heuristic approach that uses a branch-and-bound algorithm to find satisfactory (sub-optimal) solutions.

Constraint Programming is a method that represents the problem as a set of constraints that the solution must satisfy, such as non-overlap and containment. Constraint Programming solvers then employ various techniques to find solutions that meet all these constraints [39]. This method is beneficial when dealing with complex, real-world constraints that are difficult to express in linear equations.

Relative placement formulations define the relative positions of boxes (e.g., left or right, above or below) using decision variables to prevent overlaps and ensure that the packing is feasible. Both Chen et. al. [34] and Paquay et. al. [40] use relative placement formulations in their MILP models. Paquay et. al. [40] extended this concept by using two reference positions for each box, the front-left-bottom corner and the rear-top-right corner. Space-indexed formulations divide the container space into discrete units and use decision variables to indicate whether each unit is occupied. Fekete et. al. [41] discuss this method and highlights its effectiveness for certain types of packing problems. Junqueira et. al. [35] employs a similar concept called "normal patterns", which restricts the possible box positions to multiples of box dimensions, reducing the model's size. Graph-theoretic characterization, as introduced by Fekete et. al. [41], uses graph theory to represent feasible packing. Nodes represent boxes, and edges indicate overlaps in projections. This approach helps develop efficient tree search algorithms to find optimal solutions.

In addition to these methods, techniques can be used in conjunction with exact methods to improve their efficiency and scalability. Fekete et. al. [41] and Elhedhli et. al. [42] both propose hybrid approaches that combine exact methods with heuristics or decomposition techniques to solve larger instances of 3D C&P problems. Fekete et. al. [41] introduces a two-level tree search algorithm. The first level uses heuristics and bounds to determine a promising subset of boxes to pack. The second level then employs a graph-theoretic characterization and a tree search to find a feasible packing for the selected subset. This approach breaks down the complex problem into more manageable subproblems, improving the algorithm's efficiency. Elhedhli et. al. [42] presents a layered stacking heuristic that packs boxes in layers, prioritizing the density of the bottom layers for stability. While the overall approach is a heuristic, each layer is solved optimally using an exact method based on a mixed-integer programming (MIP) formulation. This iterative framework allows the application of exact methods to larger instances that would be intractable for a direct exact approach.

3.3.2 Placement Heuristics

Placement heuristics, also known as construction heuristics, are approximation algorithms designed to generate initial feasible solutions for the NP-hard 3D-PP [6, 43]. Due to the computational complexity of exact methods in solving large-scale, real-world packing problems, placement

heuristics are essential as they provide a starting point for further optimization using more sophisticated techniques. The effectiveness of these heuristics depends on how they sequence or categorize items, manage the available space within the container, and adhere to practical constraints. Common types of placement heuristics include wall building, layer building, stack building, block building, and guillotine cutting. These methods are valued for their ability to produce good quality solutions within reasonable computational times, making them suitable for real-world applications where finding optimal solutions might be computationally infeasible [6].

Wall Building

This heuristic involves constructing vertical layers or "walls" across the depth of the container. Each layer's depth is usually determined by the first box placed in that layer. Figure 3.10 depicts a simplified representation of the process of achieving a wall-building solution.

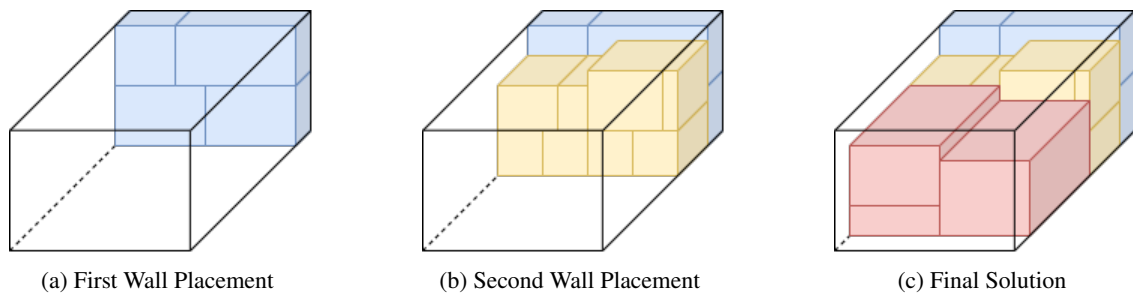


Figure 3.10: Wall Building Solution

This method tries to maximize the use of the space by adjusting the layers' width and height as more items are added. This approach was first introduced by George and Robinson [7], who proposed a heuristic for packing boxes into a container by building layers across the container's width and maintaining an even workforce. The algorithm focuses on selecting the next box type and position to optimize space utilization. Gehring et. al. [44] extended this concept by developing a computer-based heuristic for packing pooled shipment containers using a wall-building approach. Their heuristic generates alternative stowage plans, allowing the user to select the most suitable one based on various criteria. Below is an image explaining how a wall or vertical layer is defined by the first box placed.

Bischoff and Ratcliff [10] further explored the wall-building approach, highlighting its advantages in terms of stability and even weight distribution within the container. They proposed a method that constructs the loading pattern from vertical layers, ensuring that the load is secure during transportation. Bortfeldt et. al. [45] incorporated the wall-building approach into a parallel tabu search algorithm for container loading. Their heuristic aims to maximize the stowed box volume while meeting loading constraints, such as stability and weight distribution. Chien and Deng [46] also utilized the wall-building concept in their container packing support system. Their

algorithm generates vertical strips, combines them into lateral walls, and then combines the walls to determine efficient packing patterns. The system also includes a graphical user interface for visualizing the packing process and results. More recently, Dereli and Das [47] proposed a hybrid simulated annealing algorithm for solving multi-objective CLPs, incorporating a wall-building approach. Their algorithm loads the container layer by layer, determining the dimensions of each layer before filling, and aims to optimize multiple objectives simultaneously, such as maximizing weight and volume utilization.

Layer Building

This approach packs items into bins using a horizontal layer-by-layer procedure. It involves placing boxes on the floor of the bin to create a base layer and then stacking subsequent layers on top until no more can fit within the height of the container. Figure 3.10 demonstrates a simplified representation of the process of achieving a layer-building solution.

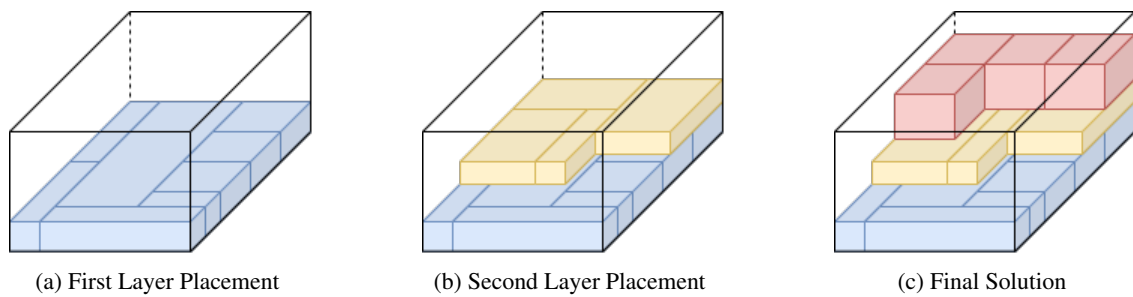


Figure 3.11: Layer Building Solution

Notably, Bischoff and Ratcliff [10] highlight its ability to produce stable and evenly distributed packing patterns, particularly relevant for pallet-loading problems where stability is crucial. They construct layers using up to two different box types, prioritizing the utilization of the loading surface. Lodi et al. [48] incorporate this heuristic within a Tabu Search framework for the 3D-PP. Their approach focuses on optimizing both vertical filling (items with similar heights) and horizontal filling (efficiently packing items with their bases on layer floors). In addition to these foundational works, recent research has expanded the application of layer building heuristics. Lim et al. [49] introduce dynamic prioritization, adjusting box-type priorities based on iterative solution analysis to enhance packing efficiency. Hasan et al. [50] employ a layer-based algorithm for a multi-objective 3DBPP, aiming to minimize the number of bins and achieve balanced bin weights. Júnior et al. [51] adapt the heuristic for real-world scenarios involving multi-compartment containers in beverage distribution, demonstrating its practicality and effectiveness in addressing complex, real-world constraints.

Stack Building

Stack building is a loading pattern where items are arranged in vertical columns (stacks) and then packed onto the container floor, effectively converting the 3D-PP into a 2D problem. This approach simplifies the three-dimensional packing problem by breaking it down into two steps:

- **Building Stacks** - Items are arranged vertically into stacks, considering factors like item dimensions, weight distribution, and stability.
- **Packing Stacks into Containers** - The pre-built stacks are then treated as two-dimensional objects and packed onto the container floor, often using algorithms designed for two-dimensional packing problems.

Figure 3.12 shows the 2D construction of stacks and the resulting packed solution generated from a stack building approach.

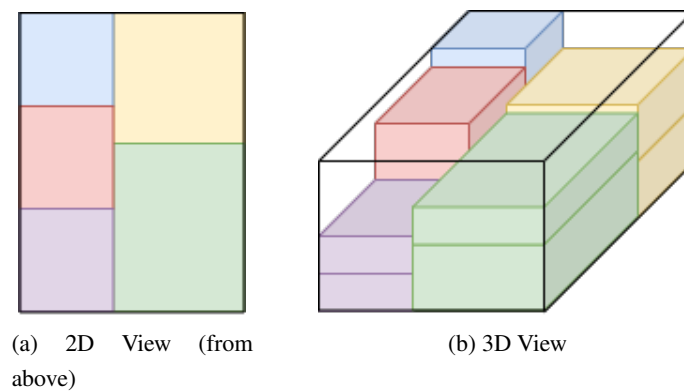


Figure 3.12: Stack Building Solution

Gehring and Bortfeldt [52] proposed a two-step approach for the SKP. First, a greedy algorithm is employed to generate a set of stacks, aiming to maximize the volume utilization of each stack. Then, a genetic algorithm is used to arrange these stacks optimally on the container floor, maximizing the total value of packed boxes while adhering to constraints. Toffolo et al. [53] presented a heuristic decomposition approach for a multiple CLP. The method involves creating regular and relaxed layers, which are then combined into stacks. These stacks are treated as 2D objects and packed into containers using a 2D bin packing problem solver. This approach simplifies the complex 3D problem by addressing layer building and stack placement separately. Olsson et al. [54] focused on automating container loading for Atlas Copco, incorporating real-life constraints. They used a greedy heuristic to pack boxes into stacks, prioritizing stability and stacking constraints to prevent damage. A genetic algorithm is then employed to optimize the parameters of the greedy heuristic, ensuring the generation of high-quality load plans. This approach highlights the importance of considering practical constraints and using optimization techniques to improve the heuristic's performance.

Block Building

In this method, one or more types of boxes are packed into blocks, and then the container is filled with these blocks rather than single boxes. These blocks can be homogeneous, containing items of the same type and orientation, or heterogeneous, containing items of different types, as seen in Figure 3.13.

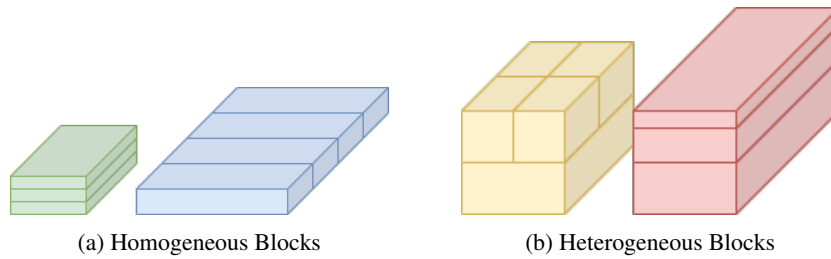


Figure 3.13: Types of Blocks

In Figure 3.14 we can see the final solution containing the junction of all these homogeneous blocks and heterogeneous blocks.

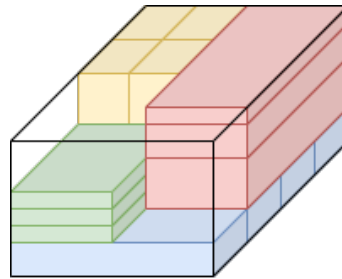


Figure 3.14: Block Building Solution

Eley [55] expanded on the block-building approach for the CLP by developing a two-step method. First, a greedy heuristic generates an initial arrangement of blocks. Then, a tree search algorithm is applied to refine the arrangement and improve space utilization. In a parallel advancement, Bortfeldt et al. [45] introduced a parallel tabu search algorithm, enabling the exploration of larger solution space and potentially leading to the discovery of superior solutions for the CLP. Fanslau and Bortfeldt [56] further generalized the block building approach to include heterogeneous blocks, which can contain boxes of different types with minimal gaps between them. This increased the flexibility of the method, allowing it to handle a wider range of problem instances. Zhu and Lim [57] also contributed to the field by proposing an iterative-doubling greedy look-ahead algorithm that efficiently identifies high-quality solutions within short time frames, utilizing the block building approach.

Guillotine Cutting

It involves a sequence of cuts made parallel to the container's faces (i.e., the sides of the container) to divide the container into smaller rectangular sections. Each resulting section is then filled with one or more boxes. The cuts are called "guillotine" cuts because they are similar to the action of a guillotine, completely separating a piece into two smaller pieces. This heuristic is often used in combination with other optimization techniques, such as tree search algorithms or genetic algorithms, to find efficient packing solutions. Figures 3.15 and 3.16 illustrate the concept of guillotine cuts in packing. They show a placed block and the potential guillotine cuts that can be made to divide the remaining space within the container.

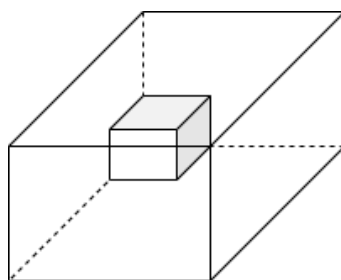


Figure 3.15: First Placed Box

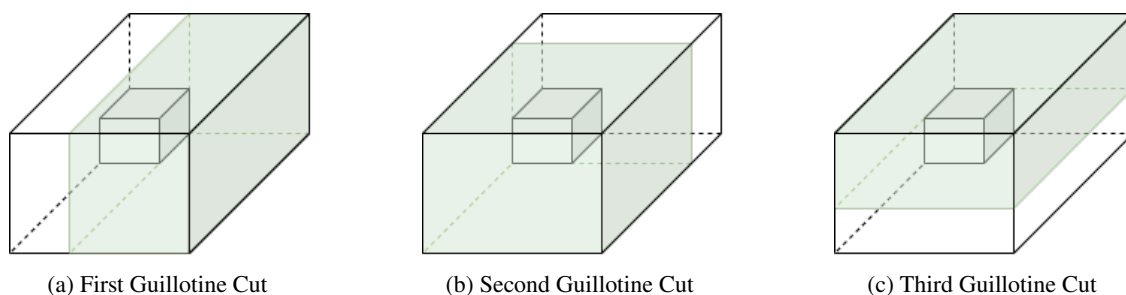


Figure 3.16: Possible Guillotine Cuts

In the context of the reviewed articles, several authors mention the use of guillotine cutting heuristics in their research. For instance, Sheng et al. [30] propose a tree search method that incorporates guillotine cutting constraints to pack items into containers efficiently. Liu et al. [58] also present a binary tree search algorithm that includes guillotine cutting constraints to find optimal packing patterns. Fanslau and Bortfeldt [56] utilize a block-building strategy and attempt to cut the blocks using guillotine cuts to achieve high space utilization. De Queiroz et al. [59] employ guillotine cutting to address various packing problems, including the knapsack problem, cutting stock problem, and strip packing problem. Morabito and Arenales [60] combine a graph search method with a guillotine cutting approach to solve the CLP.

3.3.3 Improvement Heuristics

Improvement heuristics are algorithms that build on existing solutions to a problem to find better solutions. In the context of 3D-PPs, these heuristics are used to optimize the placement of boxes within containers. Zhao et al. [43] explain that improvement heuristics typically operate on one or more complete solutions, making incremental changes to explore the solution space and identify superior packing arrangements. These heuristics vary in complexity, ranging from simple neighborhood searches that accept only improving moves to sophisticated meta heuristics capable of escaping local optima. Ali et. al. [6] note that while improvement heuristics are common in offline 3D-PPs (where all box dimensions are known beforehand), they are less frequently used in online 3D-PPs due to the real-time nature of decision-making in these scenarios, which often precludes the use of time-consuming improvement procedures.

Genetic Algorithms

Genetic Algorithms (GAs) are adaptive methods used to solve search and optimization problems. In the context of packing problems, GAs are used to evolve and optimize solutions over multiple generations. A solution is typically represented as a chromosome, which encodes the arrangement of items within a container. The fitness of each chromosome is evaluated based on how well it satisfies the packing objectives (e.g., maximizing space utilization, and minimizing the number of bins). Through operations like selection, crossover, and mutation, GAs iteratively refine the population of chromosomes, favoring those with higher fitness scores. This process continues until a termination criterion is met, such as reaching a maximum number of generations or finding a satisfactory solution [6, 43].

Gehring and Bortfeldt [52] were the first to introduce the application of genetic algorithms to the CLP. Their approach involved generating a set of disjunctive box towers and arranging them on the container floor to optimize a given criterion. This seminal work laid the foundation for using genetic algorithms in this domain. Bortfeldt and Gehring [26] expanded on this by proposing a hybrid genetic algorithm that combined the strengths of genetic algorithms with other optimization techniques for the CLP. This demonstrated the potential of hybrid approaches to enhance the performance of genetic algorithms in packing problems. Gonçalves and Resende [61] introduced a parallel multi-population biased random-key genetic algorithm for the single CLP. This approach used a maximal-space representation to manage free spaces in the container and hybridized a novel placement procedure with a multi-population genetic algorithm based on random keys. The genetic algorithm evolved the order in which box types were loaded and the corresponding layer types used in placement. In 2013, Gonçalves and Resende [62] extended their research to address both 2D and 3D PP using a biased random-key genetic algorithm. This highlighted the versatility of genetic algorithms in handling different dimensions and complexities of packing scenarios. Kang et al. [63] also investigated a hybrid genetic algorithm but with a new packing strategy for the three-dimensional bin packing problem. Their approach aimed to minimize the number of cuboid spaces generated during packing by matching the object's dimensions with those of the

packing space. This study demonstrated the potential for innovation within the genetic algorithm framework to tackle specific challenges in packing problems.

Tabu Search

Tabu Search (TS) is a meta heuristic algorithm that guides a local search procedure to explore the solution space beyond local optimality. It achieves this by using a memory structure, known as the tabu list, to discourage revisiting recently explored solutions, thereby preventing cycling and promoting diversification of the search. In the context of packing problems, TS has been effectively employed to optimize the arrangement of items within containers, aiming to minimize the number of bins used or maximize the utilization of container space [6].

Bortfeldt and Gehring [64] pioneered the application of Tabu Search (TS) to CLPs, focusing on scenarios with few box types but many items per type (weakly heterogeneous). Their innovative approach employed a two-stage process. Initially, a greedy heuristic generated potential stowing plans. Subsequently, a Tabu Search mechanism intelligently guided the selection and refinement of these plans, enhancing the overall solution quality. This work stands as a seminal contribution, marking one of the earliest successful integrations of TS into the container-loading domain. Building upon this foundation, Bortfeldt et al. [45] introduced a parallel TS algorithm for container loading. By distributing the search across multiple processors within a local area network, they aimed to accelerate the solution process and potentially discover even better packing configurations. This parallel approach represented a significant advancement in tackling the computational challenges associated with larger problem instances. In a broader context, Lodi et al. [48, 65] developed TSpack, a unified TS code designed to address a variety of two- and three-dimensional bin packing problems. The key innovation of TSpack was its flexible neighborhood structure, which could be adapted to accommodate various problem-specific constraints, such as item rotation restrictions or the requirement for guillotine cuts. This adaptability made TSpack a valuable tool for practitioners and researchers dealing with diverse packing scenarios. Crainic et al. [66] further advanced the field with TS2PACK, a sophisticated two-level TS heuristic for the three-dimensional bin packing problem. The algorithm's first level focused on the strategic assignment of items to bins to minimize the total number of bins required. The second level then delved into the tactical arrangement of items within each bin, employing an Interval Graph representation to streamline the search for optimal packing configurations. This hierarchical approach proved to be highly effective, consistently outperforming existing methods in benchmark tests.

Simulated Annealing

Simulated annealing (SA) is a stochastic optimization technique inspired by the annealing process in metallurgy, where a material is heated and slowly cooled to reduce defects and reach a low-energy state. In the context of packing problems, SA is employed to find near-optimal solutions by iteratively exploring the solution space and allowing occasional moves that worsen the objective

function. This helps the algorithm escape local optima and potentially discover better solutions [6].

Lai and Chan [67] delve into the application of simulated annealing (SA) within the context of the two-dimensional cutting stock problem, a prevalent form of packing problem encountered in industries like paper, glass, and textiles. The authors meticulously detail the algorithm's design and the calibration of its parameters, aiming to minimize material wastage (trim loss) during the cutting process. This study serves as a practical guide for implementing SA in scenarios where rectangular pieces need to be efficiently cut from larger sheets. Faina [68] extends the application of SA to the more complex realm of three-dimensional packing problems. The author elucidates the modifications necessary to adapt SA to this context and presents experimental results that underscore the algorithm's efficacy. Notably, Faina [68] emphasizes that their approach does not rely on predetermined ranking rules for item placement and employs a comprehensive 3D strategy, unlike methods confined to packing items in layers, walls, or columns. Dereli and Das [47] introduce a hybrid SA algorithm designed to tackle multi-objective CLPs. This hybrid approach integrates SA with a heuristic filling procedure to address the competing objectives of maximizing container weight utilization and minimizing unused volume. The authors' work demonstrates the versatility of SA in handling optimization problems with multiple, potentially conflicting, goals.

Greedy Randomized Adaptive Search Procedure

The Greedy Randomized Adaptive Search Procedure (GRASP) is a meta heuristic approach that has been effectively applied to various combinatorial optimization problems, including packing problems and container loading. GRASP is an iterative process that combines a constructive phase, where a feasible solution is built, with a local search phase, where the solution is iteratively improved.

In the context of packing problems, the constructive phase of GRASP typically involves placing items into containers or onto pallets according to a set of rules. These rules can be based on various criteria, such as item size, weight, or fragility. The constructive phase is randomized to ensure that different solutions are generated in each iteration, promoting exploration of the solution space [69]. The local search phase aims to improve the solution generated in the constructive phase by making small changes to it. These changes can involve swapping the positions of items, changing the orientation of items, or moving items between containers. The local search continues until no further improvement can be found Alonso et al. [70].

Several articles have explored the application of GRASP to packing problems and container loading. Moura and Oliveira [69] presented a GRASP approach for the CLP, focusing on maximizing container space utilization while ensuring cargo stability. Their algorithm, GRMODGRASP, is based on an improved version of the George and Robinson heuristic [7], a wall-building heuristic that packs boxes in a container from back to front. Alonso et al. [70] extended the application of GRASP to MCLP, considering practical constraints such as weight distribution, stability, and delivery dates. Their algorithm includes improvement methods tailored to the problem, such as

an adaptation of ejection chains, and has been shown to find high-quality solutions in short computing times. Moura and Bortfeldt [28] proposed a two-stage packing procedure that could be integrated with a GRASP framework. In the first stage, boxes are loaded onto pallets using a container loading algorithm. In the second stage, the loaded pallets are placed into trucks using a tree search algorithm. Özdemir and Yiğit [71] presented a block building based GRASP method for the CLP. Their method focuses on constructing solutions by placing blocks of boxes into the container and uses GRASP as the search strategy. The proposed method achieved high loading factors on benchmark datasets.

3.3.4 Tree Search Methods

Tree search algorithms offer a systematic approach to solving packing problems by exploring a tree-like structure of potential solutions. Starting from an empty container (root node), the algorithm expands promising branches, each representing a partial solution, until a complete configuration is found [6].

Fanslau and Bortfeldt [56] introduce a tree search algorithm for the CLP, utilizing a generalized block-building approach where cuboid blocks containing various box types fill the container. Their partition-controlled tree search ensures both efficiency and diversity in the solution exploration process. Lim et al. [49] propose an iterated construction algorithm with dynamic prioritization for solving single and multiple CLPs. This algorithm combines a greedy heuristic with a tree-search procedure, updating priorities based on the analysis of the best solution found by the heuristic.

Liu et al. [58] present a binary tree search algorithm for the CLP, grouping boxes into strips and layers while adhering to constraints such as full support, orientation, and guillotine cutting. Each node in the binary tree represents a container loading plan, with the solution being the layer set with the highest total box volume. Ren et al. [29] introduce a tree search method for the CLP with shipment priority, based on a greedy heuristic that selects blocks of identical items with the same orientation for packing. Their approach incorporates five evaluation functions for block selection, contributing to the search tree's branches.

Sheng et al. [30] propose a heuristic algorithm for the CLP with multiple constraints, prioritizing expiring orders and using a simulated annealing strategy for order selection. A tree-graph search procedure is then employed to pack the selected orders into the container. Zhu and Lim [57] present a novel block-building approach that constructs packings by sequentially placing blocks of boxes into residual space cuboids within the container. A key aspect of their approach is the selection of the optimal block-residual space pair at each step, guided by a new heuristic for evaluating the fitness of residual spaces.

Table 3.4 summarizes all the solution approaches present in this literary review.

Table 3.4: Solution approaches

Solution Approach		Type	Description
Exact Methods		Constructive /Improvement	Typically grounded in mathematical optimization, ensuring a mathematically proven best outcome. However, the trade-off for this guarantee is computational complexity, as the time to find the optimal solution can grow exponentially with the problem's size and complexity.
Placement Heuristics	Wall Building	Constructive	Create vertical layers of boxes, with layer depths determined by the first box placed in each layer. Packing consequential vertical layers until the container is filled.
	Layer Building	Constructive	Create horizontal layers of boxes, with layer heights determined by the first box placed in each layer. Packing consequential horizontal layers until the container is filled.
	Stack Building	Constructive	Loading pattern where items are arranged in vertical columns (stacks) and then packed onto the container floor, effectively converting the 3D-PP into a 2D problem.
	Block Building	Constructive	Group boxes into blocks (homogeneous or heterogeneous) and then pack the container with these blocks.
	Guillotine Cutting	Constructive	Guillotine cutting methods involve dividing the container into smaller rectangular sections using parallel cuts (guillotine cuts) and then packing each section with boxes.
Improvement Heuristics	Genetic Algorithm	Improvement	Genetic algorithms (GAs) are optimization methods that use a population-based approach to find solutions to packing problems. Solutions are represented as chromosomes, and their fitness is evaluated based on packing objectives. GAs use operations like selection, crossover, and mutation to evolve the population towards better solutions.
	Tabu Search	Improvement	Tabu Search (TS) is a metaheuristic that uses a tabu list to avoid revisiting recent solutions and promote diversification in local search.
	Simulated Annealing	Improvement	Simulated annealing (SA) is a stochastic optimization technique inspired by metallurgy. It allows for occasional moves that worsen the objective function to avoid getting stuck in local optima and find near-optimal solutions for packing problems.
	Greedy Randomized Adaptive Search Procedure	Constructive /Improvement	GRASP is a metaheuristic that combines a constructive phase and a local search phase to solve combinatorial optimization problems like packing and container loading.
Tree Search Methods		Constructive /Improvement	Tree search algorithms explore potential solutions in a tree-like structure, starting from an empty container and expanding promising branches to find complete configurations.

Chapter 4

Methodology

This chapter presents the operational framework of the 3D packing tool, designed to address the CLP. The tool offers two modalities: manual packing and automatic packing. Distinct system architectures for each modality are outlined, including input, processing, and output stages.

User inputs for both tools are examined, emphasizing their influence on simulation behavior and outcomes. Methods for loading and processing 3D models are explained, detailing the representation of objects and spatial relationships within the virtual environment. Container creation is addressed, covering modeling for visual representation and collision detection. The manual tool is analyzed, exploring the user interface, 3D control mechanisms, and collision checking procedures. The automatic tool's algorithm approach is described, focusing on bounding box creation and management, placement algorithm logic and heuristics, and packing solution generation. Distinct visualization strategies for each tool are presented, balancing the need for accurate representation with computational efficiency and user-friendliness.

By the chapter's conclusion, readers will understand the technical foundations and operational principles of the 3D packing simulation tool, which addresses diverse packing scenarios, from manual to automated placement.

4.1 System Architecture

This chapter presents the technical foundations of the 3D packing tool. The tool is engineered to accommodate two distinct user modalities: a manual, user-driven approach, and an automatic, algorithm-guided approach. Each modality necessitates a unique architectural design adapted to its specific requirements. Figure 4.1 depicts the system architecture of the packing tool.

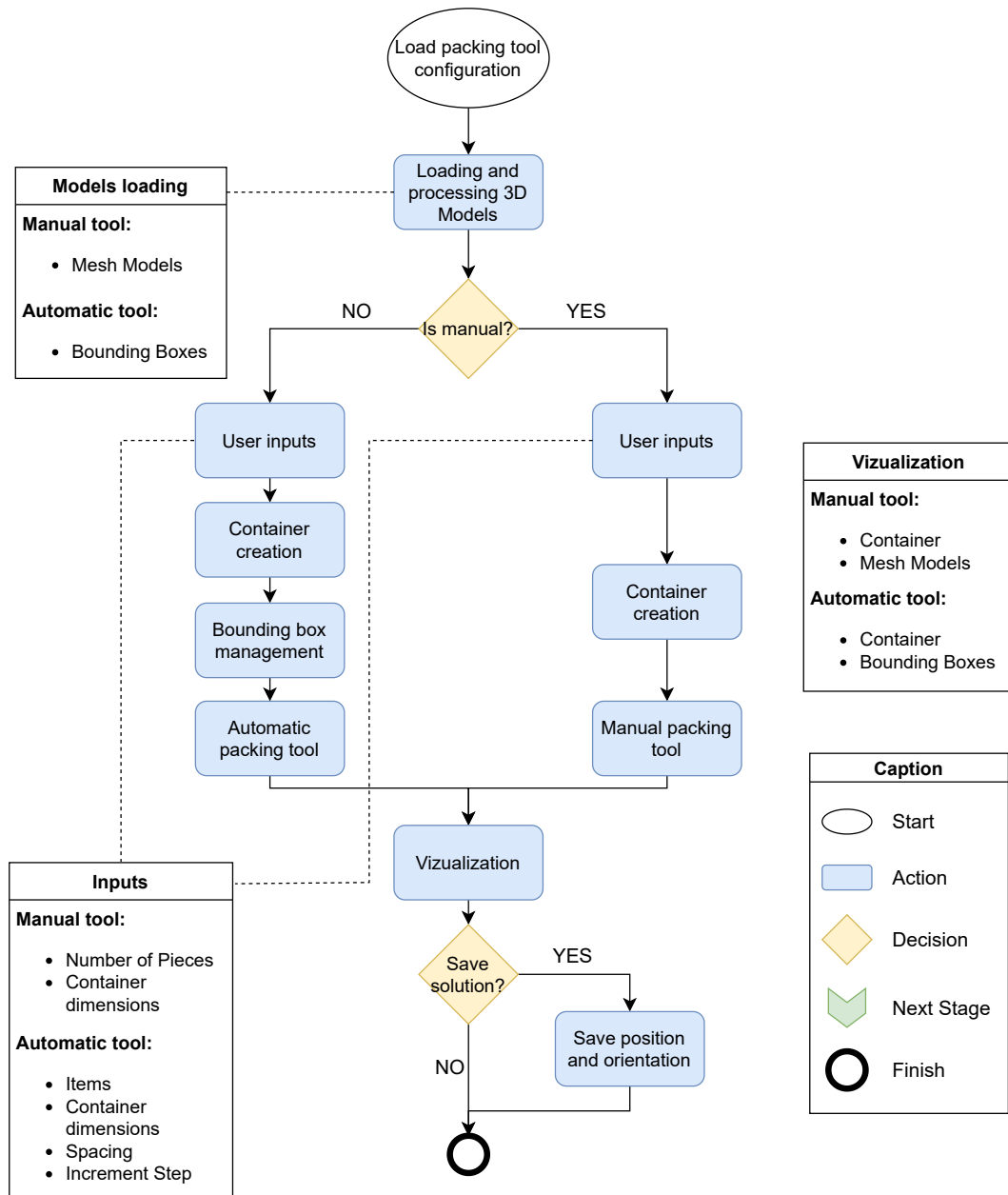


Figure 4.1: System architecture

4.2 Loading and Processing 3D Models

This section presents an overview of the preliminary stages of 3D packing. It elucidates the processes involved in obtaining user input, loading 3D models in the PLY format, and generating corresponding bounding boxes. These initial steps are fundamental for establishing the loading of the items upon which subsequent packing algorithms operate and for verifying the solutions derived from the placement algorithm, which is detailed in Section 4.6.2.

4.2.1 Mesh Models

In the manual 3D packing, pieces are represented using high-definition meshes loaded from PLY files via the Assimp (Open Asset Import Library). Assimp enables the importation of 3D model formats and their conversion into a format suitable for the application. The loading of the model acts as a bridge between the FCL and the raw PLY format. It receives a PLY file, ensures all geometric shapes are represented as triangles (triangulation), and constructs a bounding box that tightly encapsulates the model.

The triangulation method (implemented using the Assimp library explained in Section 2.4) involves transforming all geometric shapes into triangle lists, the simplest form of polygons. This ensures the representation and processing of the 3D model by FCL for collision detection purposes. The resulting triangles are then utilized to create a BVH (Bounding Volume Hierarchy) model. BVH is a tree-like structure that organizes the triangles for collision detection by rapidly eliminating non-intersecting object pairs (Section 2.3). Figure 4.2 illustrates the outcome of the triangulation process, revealing the individual triangles, and the final loaded mesh.

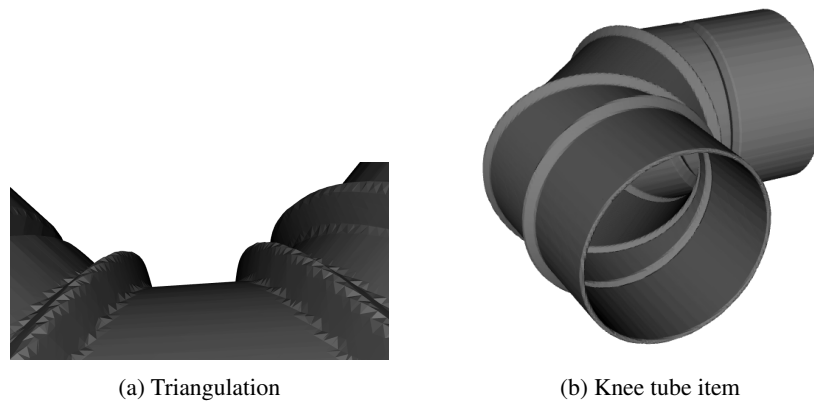


Figure 4.2: Mesh model creation

4.2.2 Bounding Boxes

In the automatic tool, bounding boxes are employed to simplify collision detection, enhancing the efficiency of the packing algorithm. Within the FCL framework, these bounding boxes are represented using bounding volumes, notably OBBRSS (Oriented Bounding Box with Rectangle Swept Sphere) and AABB (Axis-Aligned Bounding Box), as discussed in Section 2.3.2. OBBRSS enables the creation of precise bounding volumes that tightly enclose objects, thereby improving the accuracy of collision detection. Converting these to AABBs allows for simple collision checking, particularly in broad-phase detection, ultimately optimizing the overall performance of the packing algorithm.

Figure 4.3 showcases an example of an item's raw model representation of an item alongside its corresponding bounding box after 3D loading. This illustrates how the bounding box simplifies collision detection, enabling efficient packing algorithms.

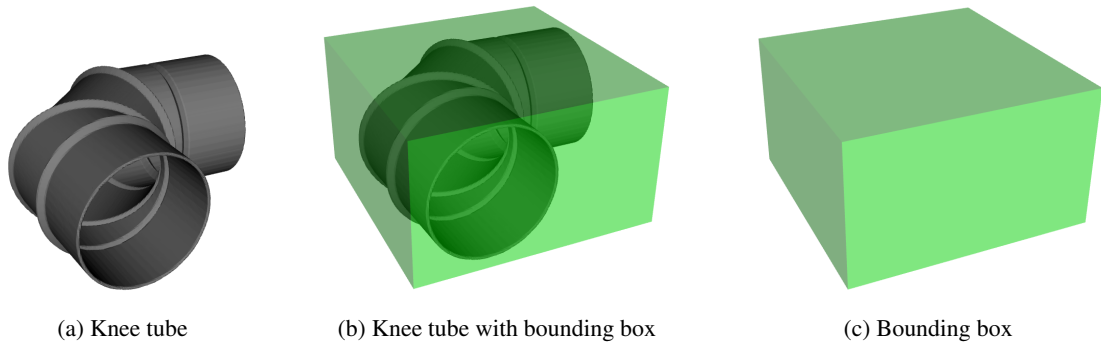


Figure 4.3: 3D model loading

The container creation process of loading and processing 3D models ensures the representation of pieces with high-definition meshes, triangulating these meshes, and generating efficient bounding volumes such as OBBRSS and AABB. These steps are fundamental to establishing a foundation for collision detection and packing algorithms within the 3D packing simulation environment.

4.3 User Inputs

The user input acts as the interactive interface between the user and the packing simulation. It systematically guides the user through a series of prompts to gather essential input parameters, thereby facilitating customization and adaptability in the packing process. The following input parameters are requested from the user:

Number of Pieces - For each distinct 3D model file identified within the specified directory, the user is prompted to indicate the desired quantity of instances to include in the simulation. This allows for the creation of scenarios with varying numbers of different object types.

Container Dimensions - To delineate the virtual packing environment, the user is prompted to input the length, width, and height of the container. These dimensions define the available space and directly influence the resulting packing solution.

Spacing - To account for real-world constraints where objects cannot be placed in direct contact, allowing the user to specify a spacing value for the automatic packing tool. This value represents the minimum permissible distance between any two objects, ensuring a physically feasible packing arrangement.

Increment Step - The user defines the size of the incremental steps taken during the heuristic search for suitable placement positions for bounding boxes in the automatic packing tool.

By collecting this user-defined input users adapt the packing simulation to their specific needs. The resulting data (quantities, container dimensions, and spacing) are then used to initialize the

simulation and drive the packing algorithm. The **Spacing**, **Increment Step**, and **Container Dimensions** are inputs exclusive to the automatic packing tool.

4.4 Container Creation

The packing simulation relies on creating container representations that serve both visualization and collision detection purposes. Two distinct container models are generated, designed to the specific requirements of each tool:

Manual Tool Container Model - For the manual tool, the container is represented visually using a mesh model marker within RViz. This open-top design provides an intuitive and uncluttered view of the container's interior, facilitating the manual placement of objects. The container collision is created using a FCL mesh model. This open-top collision model allows for realistic interactions during manual packing, where objects can be freely placed from above.

Automatic Tool Container Model - The automatic tool, on the other hand, employs a more streamlined approach. A single yellow RViz CUBE marker outlines the entire packing volume, encompassing all six sides of the container. This simplified visualization focuses on the overall shape and dimensions of the container, aiding the user in assessing the spatial constraints for automatic packing. To ensure accurate collision detection during the automatic packing process, the container is modeled with six thin cuboids, each one for each side of the container. This closed container model provides the necessary constraints for the algorithm to optimize packing efficiency within a fully enclosed space.

By crafting visual representations and collision models, the packing simulation environment ensures high realism and accuracy. This approach provides the unique requirements of both manual and automatic packing processes, providing users with the necessary tools to visualize and interact with both packing strategies effectively. Figure 4.4 depicts the visuals of the container for both the manual and automatic packing tools.

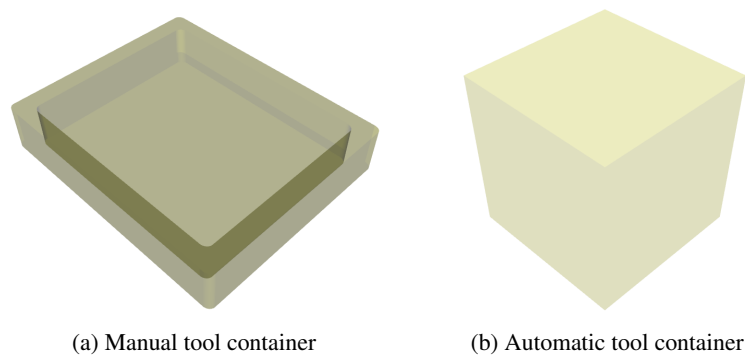


Figure 4.4: Containers visuals

4.5 Manual Tool

The manual packing tool offers a user-centric approach to the 3D bin packing problem. It is designed to combine human spatial reasoning and decision-making abilities within a virtual environment. By providing an interactive interface in RViz, the tool allows users to directly manipulate and position 3D models representing the items to be packed. This hands-on approach is valuable when dealing with complex or irregularly shaped objects where intuitive placement is more precise than algorithmic solutions.

The tool incorporates features to facilitate accurate packing. Collision detection mechanisms, implemented using FCL, prevent objects from intersecting with each other or the container's boundaries. This ensures that the resulting packing arrangements are physically feasible. Additionally, the tool offers 3D controls for precise translation and rotation of objects, with a snapping functionality to help alignment of orthogonally placed objects. The ability to visualize the packing process in real-time allows users to iteratively refine their solutions, making this tool a valuable asset for scenarios where human judgment and adaptability are needed.

As illustrated in Figure 4.5, the manual packing tool is awaiting user input to initiate object manipulation. Upon receiving input, the model assesses the feasibility of the requested action. If the action is deemed possible, the object's position is updated accordingly. However, if the action results in a collision or violates any constraints, the model reverts to its previous position and awaits further user input.

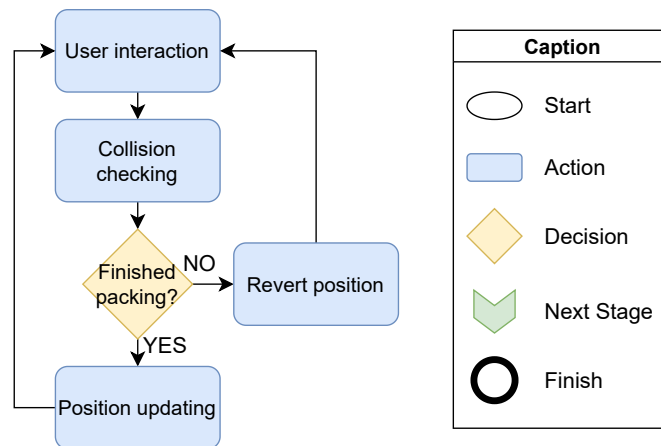


Figure 4.5: Manual processing architecture

4.5.1 3D Piece Control

The 3D control functionality of the manual packing tool is implemented through interactive markers in RViz, enabling users to manipulate the position and orientation of 3D models within the virtual environment. This task facilitates this interaction by augmenting an interactive marker with 3D control capabilities, allowing users to adjust both the position and orientation of models through translational and rotational controls.

Translational Control

Translational control enables users to manipulate the position of 3D models along the x, y, and z axes within the virtual environment. This functionality is used for the adjustment of the placement of models within the container. The user achieves this by interacting with the red, green, and blue arrows, which respectively represent translational movement along the x, y, and z axes, as depicted in Figure 4.6.

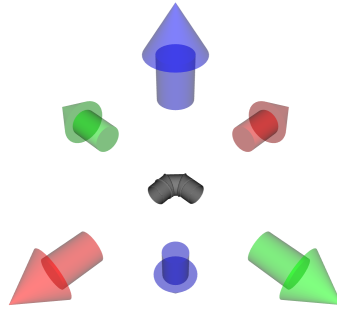


Figure 4.6: Translational axes

Rotational Control

Rotational control enables users to adjust the orientation of 3D models around the x, y, and z axes within the container. This functionality is facilitated by interacting with the red, green, and blue rings representing rotation around the x, y, and z axes, respectively, as shown in Figure 4.7. To simplify the packing process and ensure proper model alignment, the tool incorporates a feature that snaps the model's orientation to the nearest 90 degrees. This is accomplished through the following steps:

- **Roll, Pitch, and Yaw** - The current orientation, represented as a quaternion, is converted into roll, pitch, and yaw angles.
- **Snapping Angles to Nearest 90 Degrees** - Each of these angles is rounded to the closest multiple of 90 degrees.
- **Finding Closest Orientation** - The tool calculates the most proximate orientation from a set of predefined orientations comprising combinations of 0° , $\pm 90^\circ$, and 180° .

The operator can select and manipulate the position of the piece by utilizing the rotation axes depicted in Figure 4.7. The red, green, and blue rings respectively correspond to rotation around the x, y, and z axes.

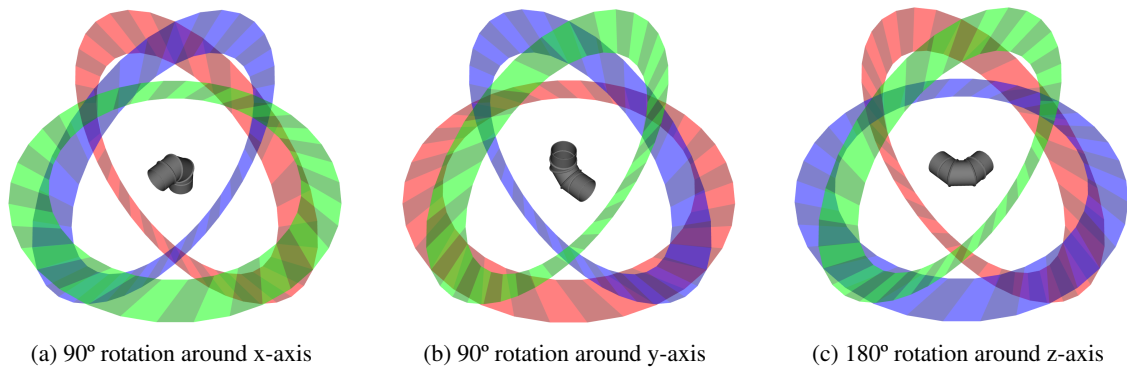


Figure 4.7: Orthogonal Rotational Positions

Figure 4.8 shows the loaded piece as a controllable Interactive Marker.

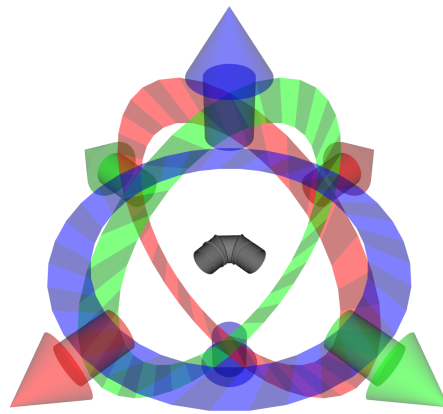


Figure 4.8: Fully controllable interactive marker

4.5.2 Collision Checking

The procedure maintains the integrity of the virtual environment by preventing 3D models from overlapping or intersecting with each other or the container walls. This process utilizes FCL to perform collision detection through two mechanisms:

- **Pieces Collision Checking** - This verification process iterates through each previously placed piece, assessing whether it intersects with the piece currently being positioned.
- **Container Boundaries Collision Checking** - This process iterates through each FCL object representing the container's walls and bottom, verifying whether the piece being positioned intersects with any of them.

If no collisions are detected with either the container walls or other pieces the new position of the piece is collision-free, the piece is placed and its position updated, later explained in Section 4.5.3.

4.5.3 Position Updating

In the event of either translational or rotational movement, this position updating ensures the application of the new position and orientation through an FCL transformation matrix. This process captures feedback from the interactive marker, converting the pose to FCL format, performing collision detection, and updating the pose if no collisions are found. The procedure is outlined below:

- **Saving the original position** - Before the user-initiated movement, recording the current position and orientation of the model.
- **3D transformations** - A new collision object is generated, transforming the model's mesh based on the updated position and orientation.
- **Collision checking** - Utilizing the transformed collision object, a collision check is conducted to determine if the new pose results in any collisions with the container walls or other pieces within the virtual environment.
- **Position updating** - If the collision check is nonexistent, the piece's pose is updated to reflect the new position and orientation. Otherwise, if a collision is detected, the model reverts to its original, collision-free position.

The 3D control functionality of the manual packing tool empowers users with intuitive mechanisms for translating and rotating 3D models within a virtual container. By snapping rotations to the nearest 90 degrees, the tool facilitates the packing process.

4.6 Automatic Tool

The automatic packing tool aims to solve the CLP. It is designed for scenarios where a large number of irregular items or boxes need to be packed, using computational power to investigate a large solution space that would be too difficult for manual packing tool exploration. Consequently, the tool is designed to address problems such as the Irregular Item Packing Problem (IIPP), Single Large Object Placement Problem (SLOPP), and Single Knapsack Problem (SKP), which are detailed in Section 3.2.1.

The tool employs a layer-based building heuristic detailed in Section 3.3.2, guided by a priority system that considers both the volume and base area of the bounding boxes. This heuristic aims to create stable and compact packing arrangements by placing larger and more stable bounding boxes first, followed by a systematic filling of horizontal layers and subsequent vertical stacking. The tool utilizes FCL for collision detection, ensuring that the resulting packing solution is physically valid. A feature of the automatic tool is its use of an Yet Another Markup Language (YAML) file to store pre-calculated bounding box information. This approach accelerates the packing process by avoiding redundant calculations, making it suitable for repetitive packing tasks or scenarios with a large number of bounding boxes. The visualization of the final packing arrangement is

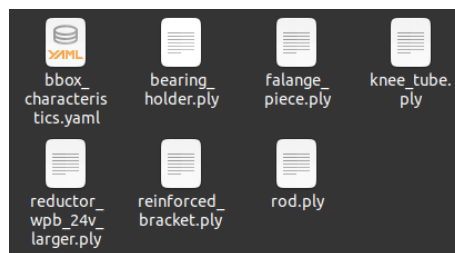
simplified using bounding boxes, focusing on the overall spatial utilization rather than the details of each item.

4.6.1 Bounding Box Management

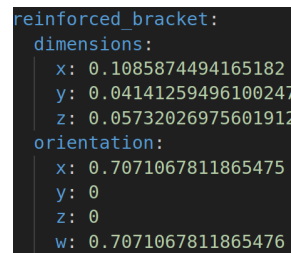
The objective of the bounding box management process is to establish and maintain a centralized repository of bounding box dimensions (length, width, height) for 3D models stored in **PLY** format within a specified directory. This approach accelerates the future loading of these models and preserves bounding box characteristics for subsequent solutions involving the same components.

1. **Loading existing data** - Retrieves bounding box information from a pre-existing **YAML** file, if available, to preclude redundant calculations for known values.
2. **Identifying new models** - Identifies **PLY** files within the directory that are not yet referenced in the **YAML** file.
3. **Calculating bounding boxes** - Employs the FCL library to load these newly identified **PLY** models and compute their corresponding bounding boxes.
4. **Orienting for stability** - Reorients each bounding box to establish its largest side as the base, thereby enhancing stability within the placement algorithm.
5. **Saving updates** - Writes the updated bounding box information back to the **YAML** file for future reference and processing.

The approach ensures that bounding box calculations are only repeated for **PLY** files that have been modified since the last update, thus automating the maintenance of accurate and up-to-date bounding box data. Figure 4.9 illustrates the directory containing the bounding box characteristics associated with the pieces to be loaded. This information is stored for future use and faster implementation in subsequent packing processes.



(a) Files **.ply** directory



(b) File **.yaml** with bounding boxes characteristics

Figure 4.9: 3D model loading

4.6.2 Packing Algorithm

This section details the automatic packing algorithm. Initially, a priority order is established for placing bounding boxes based on both volume and base area, prioritizing larger, more stable boxes for optimal packing efficiency. A collision checking mechanism is then implemented to prevent overlap between placed boxes and container boundaries. Finally, a layer-based building heuristic is outlined, strategically placing boxes in layers, and prioritizing horizontal placement for stability while incorporating vertical placement for better space utilization. This heuristic guides the algorithm in constructing a compact and feasible packing solution.

Bounding Box Priority List

The initial step in the automatic packing algorithm involves establishing a priority order for placing bounding boxes to optimize packing. This prioritization is guided by two main criteria. First, larger boxes are given precedence over smaller ones, recognizing that larger boxes become increasingly challenging to fit as available space diminishes throughout the packing process. Second, in scenarios where boxes share the same volume, the box with the larger base area is prioritized. This criterion is pertinent for the layer-based building heuristic, later explained in Section 4.10 employed by the algorithm, where boxes are placed in successive layers. Prioritizing boxes with larger base areas in the early stages establishes a stable foundation for subsequent layers, ultimately contributing to a more overall compact and stable packing solution. This dual-criterion approach ensures a balanced consideration of both volume and base area, leading to a more efficient and effective packing outcome. Figure 4.10 illustrates this concept: bounding box A is prioritized over B due to its larger volume, while box C is prioritized over A due to its larger base area if their volumes are equal.

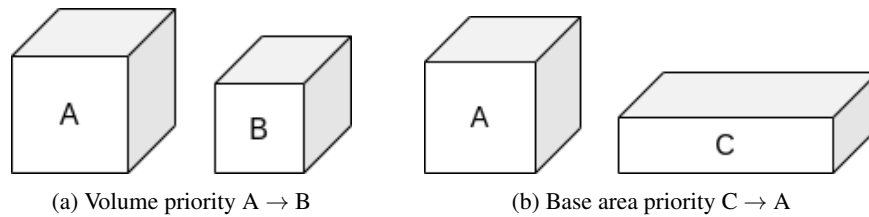


Figure 4.10: Bounding box prioritizing

The advantage of utilizing meshes is their integration with interactive markers in RViz (Section 2.2.3), enabling users to manipulate the pieces' position and orientation. This direct approach interaction allows for precise adjustments, facilitating the exploration of possible packing configurations. However, it's important to note that rendering and interacting with meshes is computationally more demanding than simpler representations like bounding boxes, particularly for large or complex models. Nonetheless, the raw 3D models in the manual tool provide accuracy and detail, consequently achieving tightly packed arrangements. This prioritization establishes a sequence for placing bounding boxes within the container. By generating this ordered list (e.g.,

$box_C \rightarrow box_A \rightarrow box_B$), this facilitates subsequent placement algorithms in making decisions regarding the optimal order of box placement, thereby enhancing the packing outcome.

Collision Checking

A fundamental requirement of any packing algorithm is to generate a solution in which the packed bounding boxes do not intersect each other and remain within the container's boundaries. To achieve this, a recurrent collision checking mechanism is implemented, comprising two distinct objectives.

1. **Bounding boxes collision checking** - Verified by iterating through every bounding box placed and checking if it intersects the bounding box of the item pretended to be placed.
2. **Container walls collision checking** - Verified by iterating through every container FCL object (walls, bottom, and top objects) and checking if the bounding box of the item pretended to be placed intersects them.

In Figure 4.11, it can be seen examples of collision between two bounding boxes and also the collision of a bounding box with a container wall.

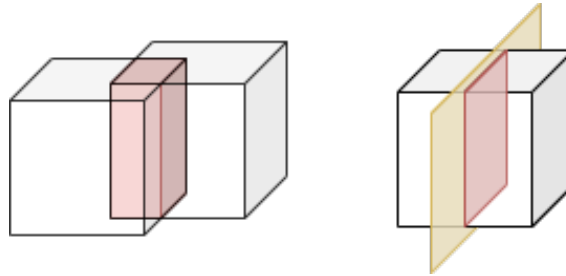


Figure 4.11: Collision checking

Collision checking is implemented to ensure the feasibility and real-world applicability of packing solutions. Playing an important role in the packing simulation by verifying the physical plausibility of the resulting arrangement. If no collisions are detected with existing objects in the solution, the position is deemed valid, and the box is placed accordingly.

Layer Based Building Heuristic

The proposed heuristic, as detailed in Section 3.3.2, employs a layer-based approach to achieve a single orientation packable solution (also detailed in Section 3.2.2). This decision aligns with the intended functionality of the packing tool, which is designed for automatic packing. The heuristic aims to pack bounding boxes horizontally, incorporating a variable threshold between them to facilitate real-world robotic placement, thus forming a layer of bounding boxes arranged side by side. Furthermore, it incorporates vertical packing with the objective of container space utilization and achieving a more compact solution. Progression to the next layer occurs only when

the current layer can no longer accommodate additional boxes. Figure 4.12 demonstrates the three main stages of this layer-based building heuristic, the layer creation, the horizontal placing and the vertical filling.

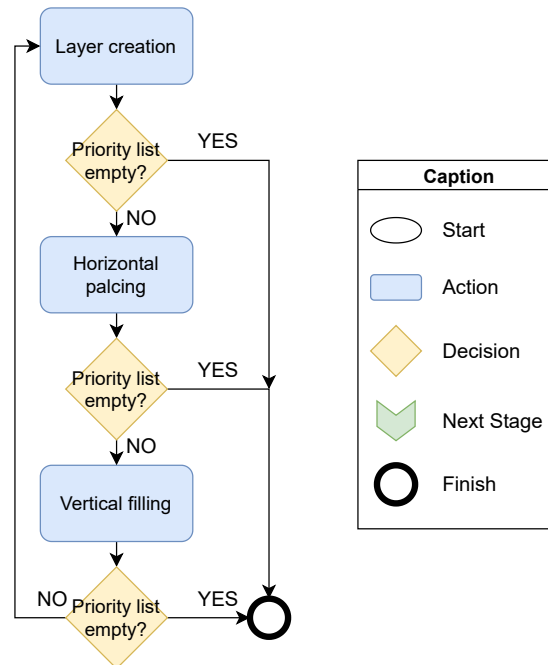


Figure 4.12: Heuristic working flow

Layer Creation

Layer creation is based on the principle that the maximum height of each layer is determined by the first box placed within it. This characteristic ensures that no other boxes within the layer exceed this established height. If a box height exceeds the container maximum height, the height of the next type of box in the priority list will determine the new layer height. If no boxes can be created, the packing ends. Figure 4.13 represents the layer creation process of the heuristic.

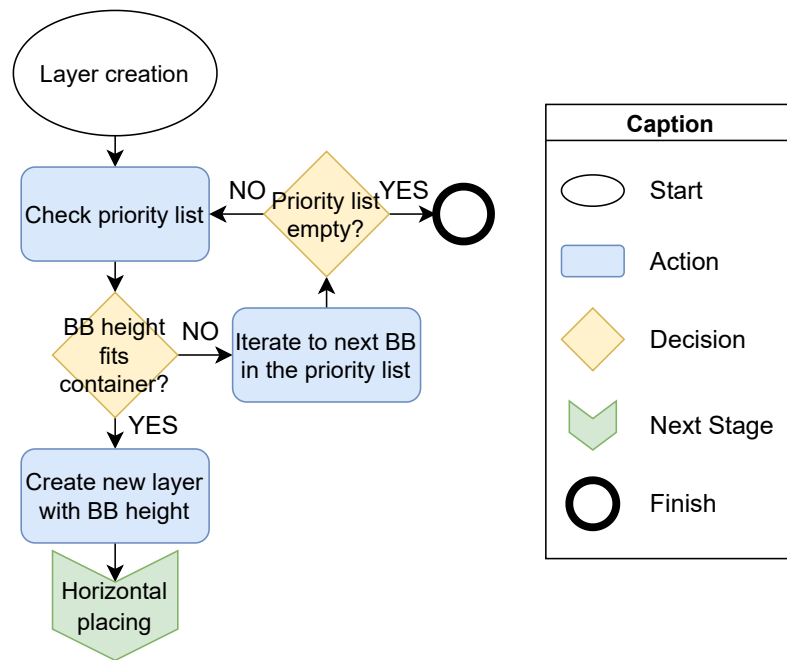


Figure 4.13: Layer determination working flow

By delineating the packing solution into distinct layers, the heuristic enforces the filling of one layer before progressing to the next. This approach searches for a more compact and stable foundation for subsequent packing. Figure 4.14 demonstrates an example of layer creation. Showing the first box to be placed determines the layer height.

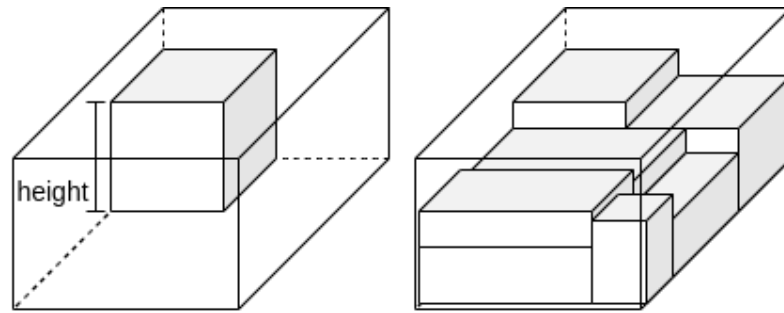


Figure 4.14: Layer height determining

Horizontal Placing

Horizontal placement is achieved by iterating through each box in the established priority order (Chapter 4.6.2), scanning the container space in two directions to identify a suitable location for each bounding box.

- **Y-axis iteration** - Iteration proceeds incrementally, with each step determined by the value of the **increment** variable. A collision check is performed at each iteration to ensure the possibility of the placement:

- **Collision Detected** - Continuing of the iteration process.
- **No Collision Detected** - The placement of each bounding box is followed by iterating to the next bounding box in the established priority list, ensuring a systematic packing process.
- **X-axis iteration** - Iteration proceeds by one small step (determined by the **increment** variable) if all attempted placements along the current y-axis alignment result in collisions.

Figure 4.15 represents the horizontal placing process of the developed heuristic.

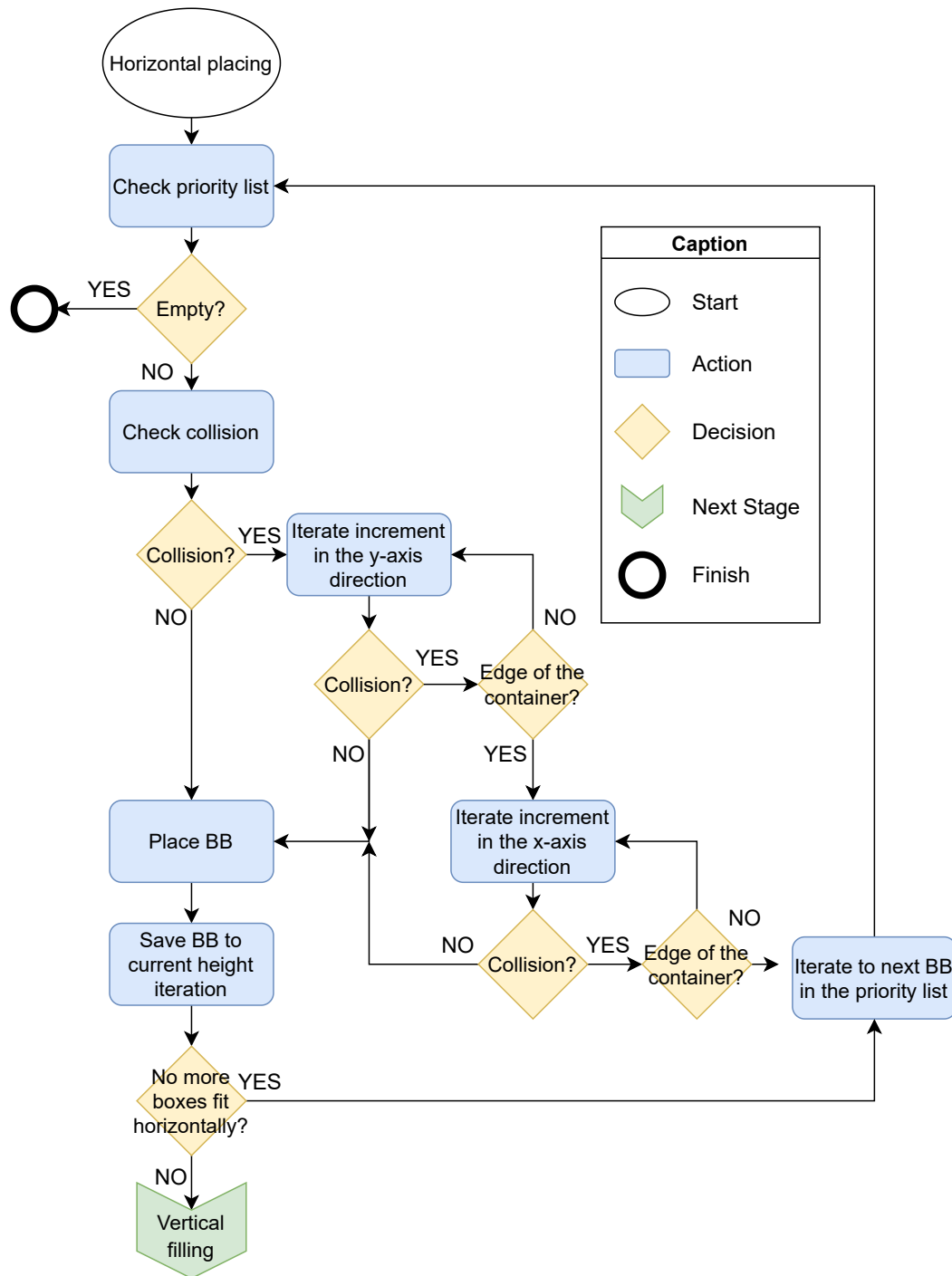


Figure 4.15: Horizontal placing working flow

Figure 4.16 illustrates this process, with boxes placed in the following sequence: *blue box* → *yellow box* → *red box* → *green box*. The x, y, and z axes are represented in the figure by red, green, and blue arrows, respectively.

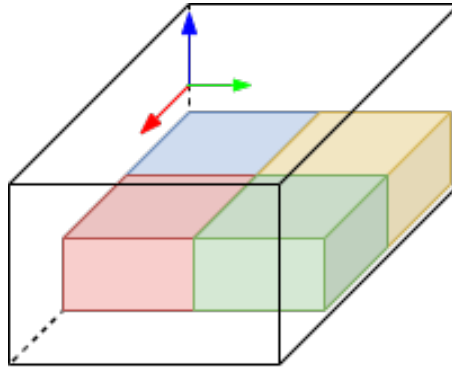


Figure 4.16: Horizontal placement solution

This approach ensures a tight placement of each box, contingent on the value of the increment variable. Notably, the increment variable is defined as the minimum value between the user-specified input and half the length of the smallest edge among all bounding boxes.

Vertical Filling

In a vertical filling approach, each layer is assessed for potential upward filling, maintaining a horizontal orientation. The heuristic prioritizes placing boxes at the lowest possible z-axis coordinates on top of earlier placed bounding boxes within each layer to optimize stability. A vertical filling approach within a layer, similar to stack building (Section 3.3.2), may compromise stability due to the lack of horizontal support. For every placed box in horizontal placement, the algorithm tries to place a box on top of it without crossing the layer height limitation. This process occurs subsequently until no more boxes can be placed on top of each other. Figure 4.17 illustrates this concept: blue boxes establish layer height, followed by yellow boxes; green boxes, while filling vertically, are placed horizontally to ensure stability and a compact arrangement.

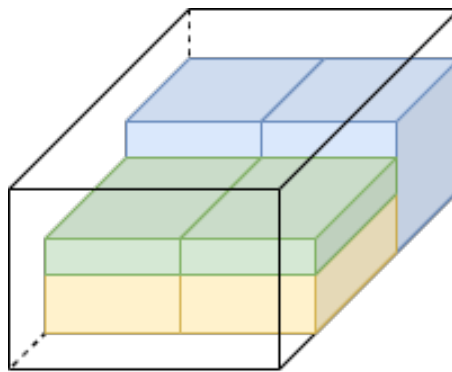


Figure 4.17: Vertical filling (blue and yellow boxes → green boxes)

Figure 4.18 represents the vertical filling process of the developed algorithm.

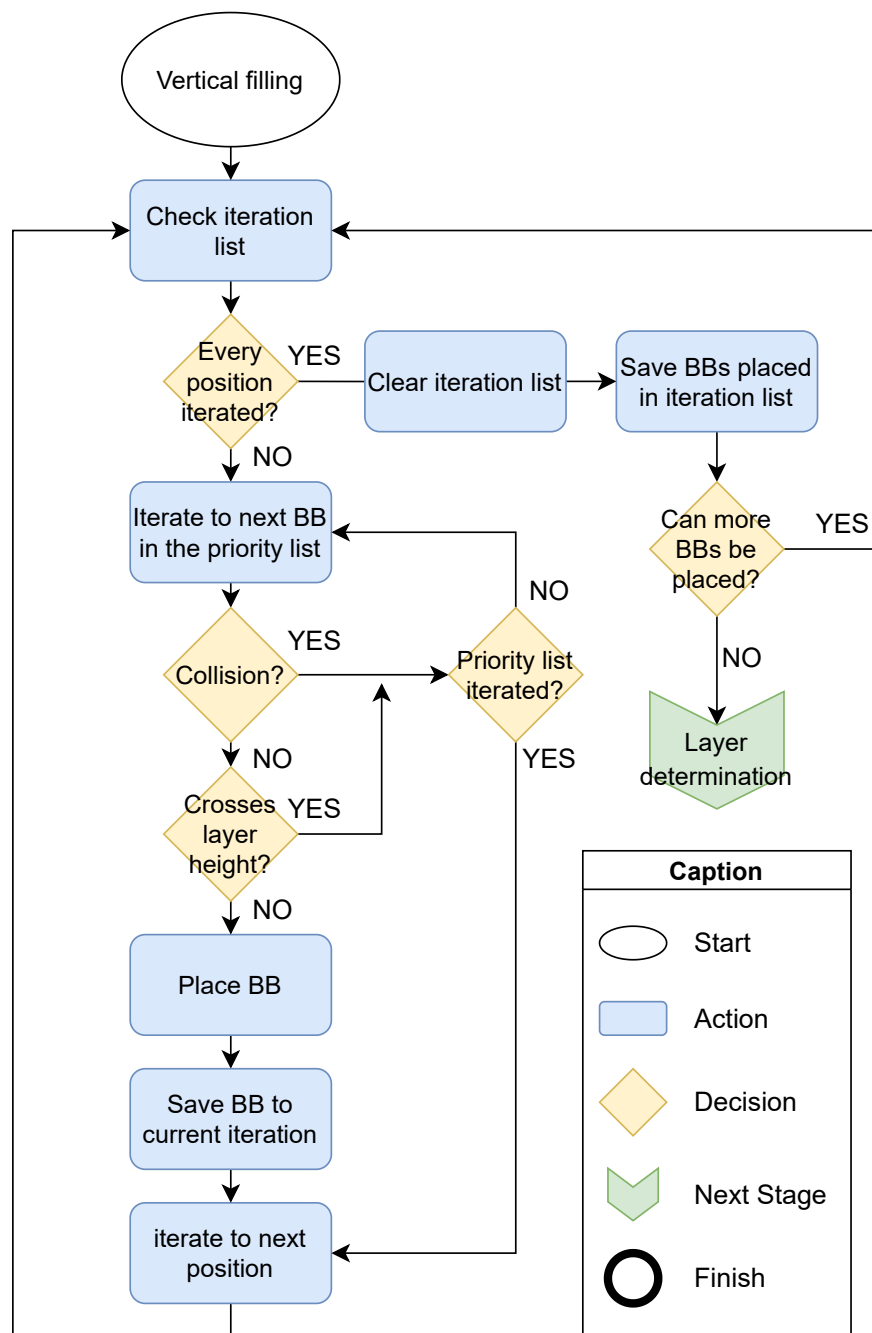


Figure 4.18: Vertical filling working flow

4.7 Items Output

The procedure serves as a mechanism to save the results of the automatic and manual packing process. Specifically, it captures the final arrangement of items within the container by recording their final positions and orientations.

- **User prompt** - After the automatic packing algorithm finishes, the user decides whether they want to save the packing solution.
- **File naming** - If the user chooses to save, they are asked to provide a filename for the solution file.
- **Packed items solution** - The operation then formats the relevant data into the YAML format. For an item in the solution:
 1. The item's unique identifier (name) is recorded
 2. Its position within the container is stored as x, y, and z coordinates.
 3. Its orientation is represented as a quaternion (a mathematical way to describe rotations in 3D space).
- **File saving** - The formatted YAML data is written to a file with the specified filename in a designated directory. This process ensures the file is successfully created and provides feedback to the user.

By saving the final arrangement of items in a YAML format, it enhances the reproducibility, analysis, and communication of packing solutions, contributing to the overall utility of the packing tool. Figure 4.19 illustrates the approach of saving the corresponding items solutions for each piece. Notably, the Knee Tube remains unrotated, as its original pose already presents the largest base area of its corresponding bounding box. Conversely, the Reinforced Bracket's (and bounding box) is rotated to achieve this optimal orientation.

<pre>bb_knee_tube_1: position: x: -0.1430925016850233 y: -0.1439641839917749 z: 0.0568059886097908 orientation: x: 0 y: 0 z: 0 w: 1</pre>	<pre>bb_reinforced_bracket_1: position: x: -0.1857062752917409 y: -0.2113398651219904 z: 0.02170629748050124 orientation: x: 0.7071067811865475 y: 0 z: 0 w: 0.7071067811865476</pre>
(a) Knee tube solution	(b) Reinforced bracket solution

Figure 4.19: Items solutions

4.8 Visualization

Visualization is essential for both understanding the packing process and evaluating the quality of packing solutions. This section details the visual representations employed within the manual and automatic packing tools, emphasizing the approaches taken to balance visual fidelity and user interaction. By specifying the visualization strategies to the specific requirements of each tool,

this approach ensures a user-friendly experience for manual packing and facilitates the solution generation for automated packing. The following subsections will develop into the specifics of container and piece or bounding box visualization for both tools.

4.8.1 Pieces

In the manual tool, each piece is visually represented by its complete 3D model, or mesh, constructed from triangles loaded from PLY files. These files contain the vertex coordinates and connectivity information that define the 3D object's shape. The mesh representation offers a high degree of visual precision, allowing users to perceive the actual shape of the piece. This level of detail is valuable for pieces with complex geometries, where the overall shape significantly influences packing decisions.

The advantage of utilizing meshes is their integration with interactive markers in RViz (Section 2.2.3), enabling users to manipulate the pieces' position and orientation. This direct interaction allows for precise adjustments, facilitating the exploration of possible packing configurations. However, it's important to note that rendering and interacting with meshes is computationally more demanding than simpler representations like bounding boxes, particularly for large or complex models. Nonetheless, the raw 3D models in the manual tool provide accuracy and detail, consequently achieving tightly packed arrangements. Figure 4.20 displays the loaded mesh models, which represent the irregular items to be packed.

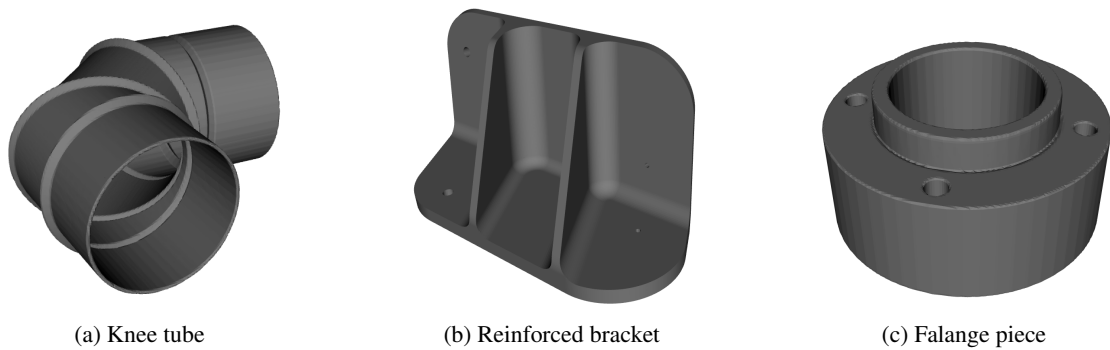


Figure 4.20: Mesh model loaded pieces

4.8.2 Bounding Boxes

In the automatic tool, pieces are visualized using simple bounding boxes. These are parallelepipeds that enclose the entire piece, calculated as the smallest possible parallelepiped that can contain the piece's geometry. Bounding boxes offer a simplified representation, conveying the piece's size and general orientation. Fine details and specific shape features are not visible and considered.

There is no direct user interaction with the bounding boxes in the automatic tool. The packing is performed by the packing algorithm developed 4.6.2, and the bounding boxes serve as a visualization method of the resulting arrangement. Bounding boxes are computationally more efficient

than raw 3D models. They require less processing power to render and are used in scenarios where the focus is on the overall packing efficiency rather than the details of each piece. The bounding boxes in the automatic tool sacrifice accuracy for computational efficiency, making them suitable for large-scale packing problems or quick visualizations. Figure 4.21 displays the loaded mesh models, which depict the irregular items to be packed, along with their corresponding bounding boxes used in the packing process.

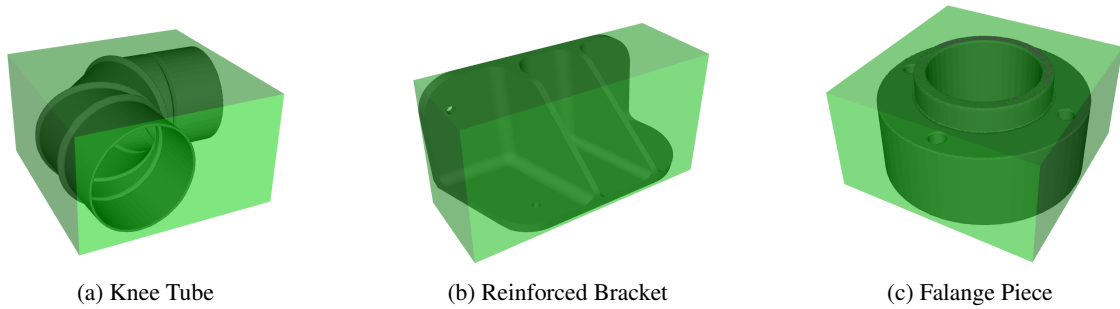


Figure 4.21: Bounding boxes fitting pieces

Chapter 5

Results

This chapter presents the results obtained from both the manual and automatic packing tools. It showcases the manual tool's flexibility and customizability in generating diverse packing configurations. The automatic tool's performance is then assessed across various scenarios within the context of the CLP. The evaluation consists of testing three packing problems (IIPP, SLOPP, and SKP) with varying degrees of item heterogeneity (identical, weakly heterogeneous, and strongly heterogeneous) and different item sizes (small, medium, and large). The analysis of these results aims to evaluate the effectiveness and efficiency of the proposed automatic packing algorithm under different conditions.

5.1 Manual Packing Tool

The manual packing approach provides a flexible packing solution, allowing the operator to create configurations with precise placement of objects. The pieces are loaded as interactive markers and the container as a yellow marker in RViz. The container is stationary and the pieces are manipulated for customized packing solutions. Figure 5.1 demonstrates the manual packing solution of two pieces, the pieces are not colliding with each other or the container thus showcasing a feasible solution in the real-world.

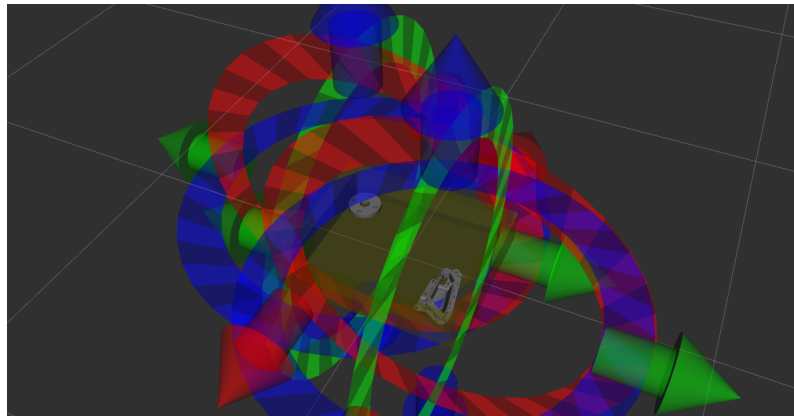


Figure 5.1: Manual packing solution

Figure 5.2 illustrates this capability by showcasing the "rod" positioned inside the "knee_tube". This demonstrates that the algorithm accommodates any feasible position, where the primary constraint lies with the robot's capabilities rather than the limitations of the packing solution itself.

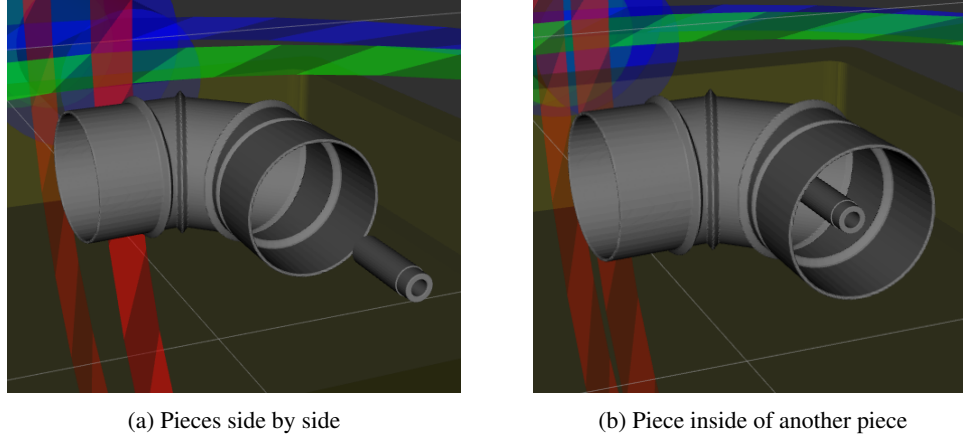


Figure 5.2: Manual packing tool solution

5.2 Automatic Packing Tool

The automatic packing algorithm is designed to address the CLP due to the presence of a single container to be packed. Its suitability is assessed across three packing problems (IIPP, SLOPP, SKP) featuring identical, weakly heterogeneous, and strongly heterogeneous assortments of items bounding boxes, respectively. Heterogeneity is classified as weakly heterogeneous with 2-5 different types of bounding boxes and strongly heterogeneous with 5 or more types, as per Dias et al. [72]. Three testing phases are done to evaluate the heuristic packing performance for each problem typology. Algorithm testing utilizes a 500 cm x 500 cm x 500 cm container and eighteen distinct irregular bounding boxes. In addition to varying heterogeneity, the results are further categorized by small, medium, and large item's bounding box sizes, as detailed in Table 5.1. Small boxes have volumes between 0-200 cm³, medium boxes between 200-400 cm³, and large boxes exceed 400 cm³. An incrementing size of 0.5 cm ensures tight packing, while spacing is set to 0 to maximize volume utilization and accurately assess packing efficiency. From this point forward, any mention of a bounding box will refer to the corresponding irregular item listed in Table 5.1.

Table 5.1: Bounding Boxes Types

Small bounding boxes	Medium bounding boxes	Large bounding boxes
rod	support_bracket	reductor_wpb_larger
cover_plate	puller_right	carcass_left
bearing_holder	falange_larger	triangle_bracket
double_side_bracket	t_bracket	knee_tube
falange_smaller	reinforced_bracket	
single_side_bracket	falange_piece	
	multi_side_bracket	
	reductor_wpb_smaller	

To evaluate the proposed automatic system packing volume utilization and packing time, it is necessary to ensure the container cannot be fully packed regardless of the types of boxes used. This allows for the determination of accurate volume utilization results by generating scenarios where not all boxes can fit within the container. To achieve this, the following formulation was employed:

$$A_i = \frac{V_{container}}{V_i} \quad (5.1)$$

Let A_i represent the number of bounding boxes of type i required to volumetrically fill the container, $V_{container}$ denote the container's volume, and V_i represent the volume of bounding box type i . This ensures the calculation of the precise number of bounding boxes needed to achieve 100% volume utilization if they were to fit perfectly within the container. Table 5.2 provides an overview of the different irregular bounding box types, their respective dimensions (in centimeters), volumes (in cubic centimeters), and the calculated number of boxes needed to volumetrically fill the container. To accurately assess the packing algorithm's efficiency, two functionalities were developed to measure key performance metrics:

Volume utilization - This metric quantifies the proportion of the container's volume occupied by the packed bounding boxes, providing insight into the algorithm's space utilization efficiency.

Packing time - This metric records the duration required to pack all feasible bounding boxes within the container.

Table 5.2: Bounding boxes characteristics

Bounding box type	Length (cm)	Width (cm)	Height (cm)	Volume (cm ³)	Number of bounding boxes to pack
rod	2.00	2.00	6.20	24.80	5040
cover_plate	11.68	6.50	1.10	83.21	1502
bearing_holder	8.49	8.50	1.60	115.41	1083
double_side_bracket	8.75	3.60	5.56	175.28	713
falange_smaller	10.49	10.50	1.60	176.11	710
single_side_bracket	6.67	8.49	3.20	181.11	690
support_bracket	14.15	3.85	3.73	203.55	614
puller_right	13.15	4.50	3.53	209.18	598
falange_larger	11.49	11.50	1.60	211.33	591
t_bracket	6.15	8.67	4.19	223.44	559
reinforced_bracket	10.86	4.14	5.73	257.76	485
falange_piece	8.50	8.50	4.50	325.12	384
multi_side_bracket	14.92	8.77	2.76	361.10	346
reductor_wpb_smaller	6.89	6.89	7.65	363.34	344
reductor_wpb_larger	8.50	8.49	9.00	648.99	193
carcass_left	22.46	11.88	3.46	923.71	135
triangle_bracket	4.40	12.02	24.74	1308.12	96
knee_tube	19.38	19.21	11.16	4154.91	30

5.2.1 Identical Bounding Boxes Packing

The IIPP involves packing uniform bounding boxes into a single container. Consequently, there is no heterogeneity among the boxes, and all solutions are derived from packing a single type of box while ensuring, through the formula in Equation 5.1, that the container is filled. Table 5.3 presents the number of packed bounding boxes, their corresponding volume utilization, and the packing time for each scenario.

Table 5.3: Identical bounding boxes packing solutions

Bounding box type	Packed bounding boxes	Volume utilization (%)	Packing time (s)
rod	4025	79.85	237.585
cover_plate	1148	76.42	27.616
bearing_holder	725	66.94	16.107
double_side_bracket	520	72.92	13.601
falange_smaller	464	65.37	9.535
single_side_bracket	525	76.07	12.504
support_bracket	468	76.21	8.770
puller_right	390	65.27	6.707
falange_larger	464	78.44	9.789
t_bracket	385	68.82	8.497
reinforced_bracket	352	72.59	7.041
falange_piece	250	65.02	5.673
multi_side_bracket	255	73.66	4.145
reductor_wpb_smaller	294	85.46	7.474
reductor_wpb_larger	125	64.90	2.388
carcass_left	112	82.76	1.913
triangle_bracket	88	92.09	1.214
knee_tube	16	53.18	0.309

Packing time increases with smaller bounding boxes due to the higher frequency of collision checks required between placed boxes within each layer and the box being positioned. Figure 5.3 illustrates the relationship between packing time and the number of packed bounding boxes, demonstrating the increase in time as the number of smaller boxes rises.

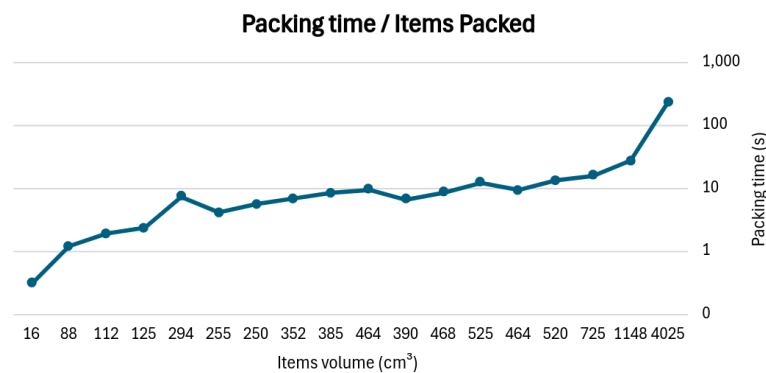


Figure 5.3: Relation between packing time and bounding boxes packed

Figure 5.4 illustrates the relationship between volume utilization and the volume of each packed bounding box. It is observed that as the volume of the packed boxes increases, the variation in volume utilization also increases.

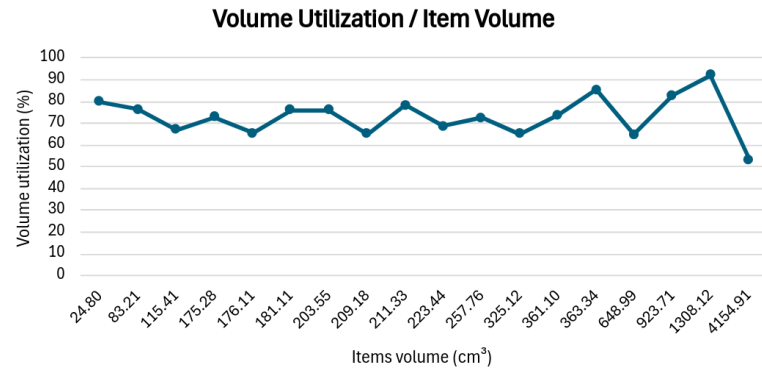


Figure 5.4: Relation between volume utilization and bounding box packed volume

This is attributed to the amplified effect of volume on the tightness of a solution, which is dependent on the specific shape of the bounding box. As illustrated in Table 5.3, the "knee_tube" box type, due to its large volume and specific shape, does not fit perfectly within the container. Amplified by the box's volume, this imperfect fit results in a lower volume utilization than other box types. Figure 5.5 depicts a packing solution where the "knee_tube" piece is repeatedly placed until the container reaches its maximum capacity.

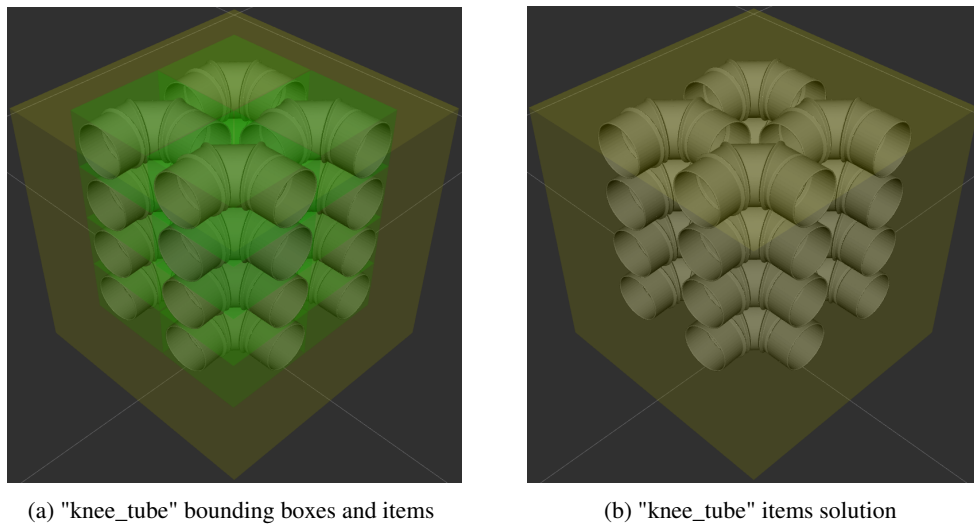


Figure 5.5: "knee_tube" packing solution

Table 5.4 presents the average volume utilization and packing time. Notably, the volume utilization demonstrates minimal variation across large, medium, and small bounding boxes, reinforcing the observations in the volume utilization per packed bounding box graph (Figure 5.4). This further supports the assertion that the size of the bounding boxes impacts packing time due to the frequency of collision checks.

Table 5.4: Different sized bounding boxes averages

Bounding boxes size	Average volume utilization (%)	Average packing time (s)
Small	72.93	52.825
Medium	73.18	7.262
Large	73.23	1.456

The automatic packing tool achieved an average volume utilization of 72.48% with an average packing time of 7.229 seconds.

Table 5.5: Average performance for identical bounding boxes packing

Average volume utilization (%)	Average packing time (s)
72.48	7.229

Figure 5.6 demonstrates a packing solution for the IIPP.

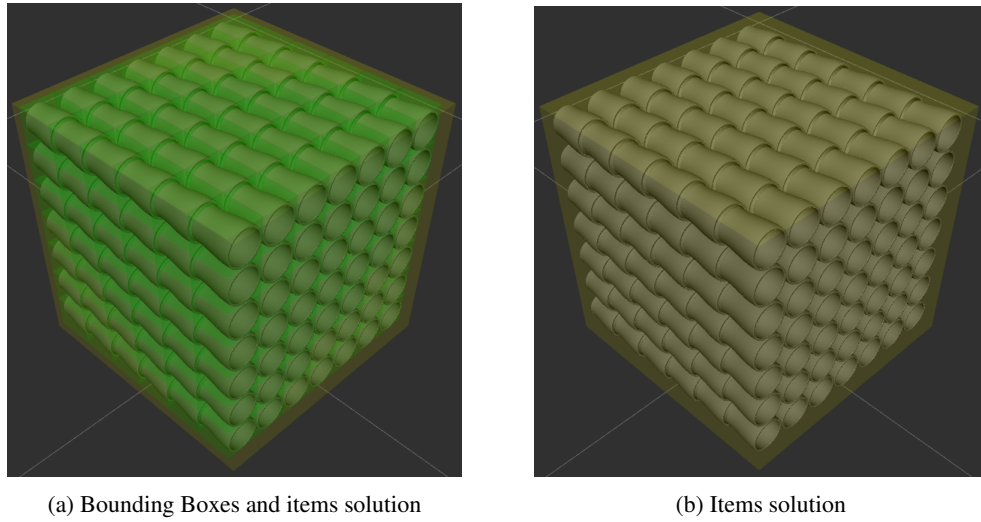


Figure 5.6: IIPP packing solution

5.2.2 Weakly Heterogeneous Bounding Boxes Packing

In the following two sections, to ensure diverse solutions while maintaining a consistent number of packed bounding boxes to fill the container, the quantity of each bounding box type is determined by the total number of distinct types to be packed. Specifically, the number of bounding boxes for each type is proportional to the ratio of that type to the total number of different bounding box types:

$$N_i = \frac{A_i}{B_i} \quad (5.2)$$

Let N_i represent the number of bounding boxes of type i to be packed, A_i denote the number of bounding boxes of type i required to volumetrically fill the container, and B_i represent the number of different bounding box types intended for packing. The values for A_i for each bounding box type are provided in Table 5.2.

The SLOPP involves packing weakly heterogeneous bounding boxes into a single container. This section considers scenarios with weak heterogeneity, ranging from two to five distinct types of bounding boxes per packing solution.

A slight increase in volume utilization is observed with increased bounding box heterogeneity for weakly heterogeneous bounding boxes. The types of bounding boxes to be packed are selected randomly, with the constraint that at least one bounding box of each size category (small, medium, and large) is included in each packing solution. Packing solutions with 3, 4, and 5 different bounding box types in each solution are presented in Tables 5.7, 5.8, and 5.9, respectively. Figure 5.7 depicts the relationship between volume utilization and packed bounding box heterogeneity. While the change is minor, the number of different bounding box types influences volume utilization due to weak heterogeneity, leading to improved packing solutions. This is attributed to the increased variety of bounding box types, which enables better fitting of boxes within the container.

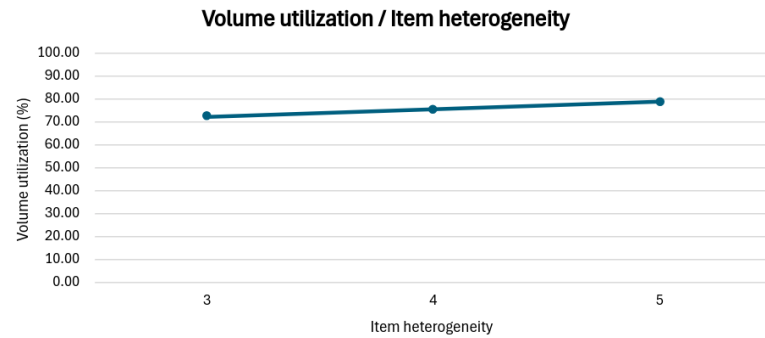


Figure 5.7: Volume utilization for different heterogeneity packing solutions

As illustrated in Figure 5.8, a direct consequence of the priority-based packing approach, where larger boxes are placed first, is that smaller boxes may be excluded from the solution if they cannot completely fill the remaining container space.

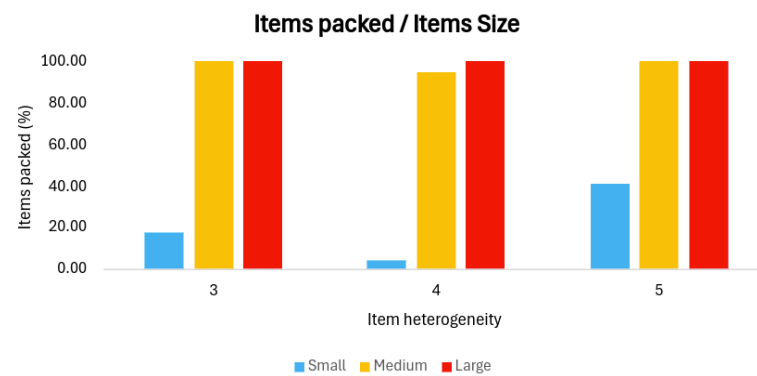


Figure 5.8: Bounding boxes packed for different bounding box sizes

Overall, the packing algorithm demonstrated an average volume utilization of 75.76% and an average packing time of 8.560 seconds for weakly heterogeneous assortments of bounding boxes.

Table 5.6: Average performance for weakly heterogeneous bounding boxes packing

Average volume utilization (%)	Average packing time (s)
75.76	8.560

Figure 5.9 demonstrates a packing solution for the SLOPP.

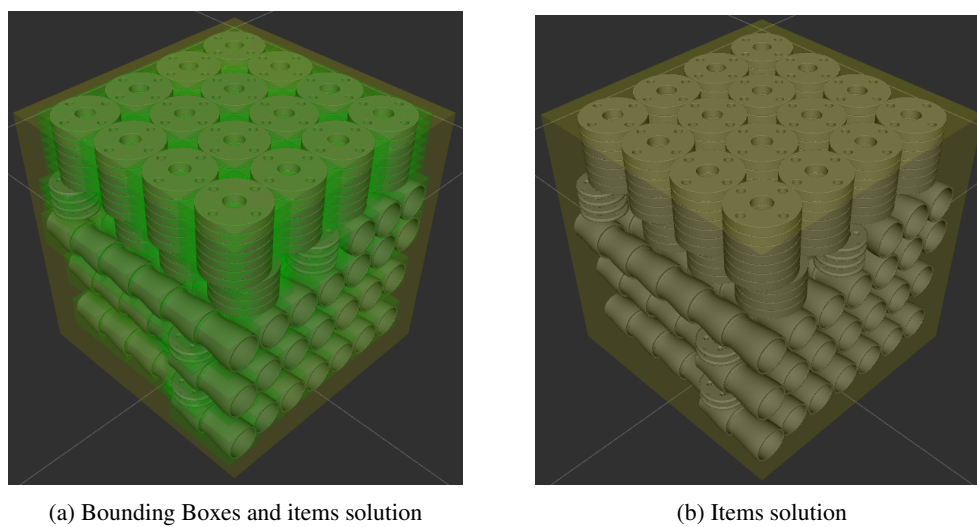


Figure 5.9: SLOPP packing solution

Table 5.7: Three different bounding boxes packing solution

Bounding box type	Bounding boxes to pack	Packed bounding boxes
double_side_bracket	238	42
reinforced_bracket	162	162
reductor_wpb_larger	64	64
Volume utilization (%)		Packing time (s)
72.52		7.093

Table 5.8: Four different bounding boxes packing solution

Bounding box type	Bounding boxes to pack	Packed bounding boxes
bearing_holder	271	12
falange_larger	148	136
reductor_wpb_smaller	86	86
knee_tube	8	8
Volume utilization (%)		Packing time (s)
75.689		6.874

Table 5.9: Five different bounding boxes packing solution

Bounding box type	Bounding boxes to pack	Packed bounding boxes
cover_plate	300	96
single_side_bracket	138	86
support_bracket	123	123
multi_side_bracket	69	69
reductor_wpb_larger	39	39
Volume utilization (%)		Packing time (s)
79.06		11.712

5.2.3 Strongly Heterogeneous Bounding Boxes Packing

The SKP involves packing strongly heterogeneous bounding boxes into a single container. This section considers scenarios with strong heterogeneity, ranging from six to eighteen distinct types of bounding boxes per packing solution.

For strongly heterogeneous bounding boxes, it is observed that volume utilization remains consistent across varying combinations of large, medium, and small bounding boxes. This is confirmed by combining large and medium boxes (Table 5.11), large and small boxes (Table 5.12), and medium and small boxes (Table 5.13). However, a notable difference lies in packing time, which is influenced by the size of the chosen bounding boxes, similar to the observation in Section 5.2.1. Larger bounding boxes require fewer units to fill the container, resulting in fewer collision

checks and faster packing. Results were obtained using diverse object types, with varying heterogeneity (six, eight, ten, twelve fourteen sixteen and all of the different bounding boxes types). Figure 5.10 demonstrates that despite variations in the number of bounding box types, volume utilization suffers from small variations ranging between (73.93% and 82.88% volume utilization) within the container.

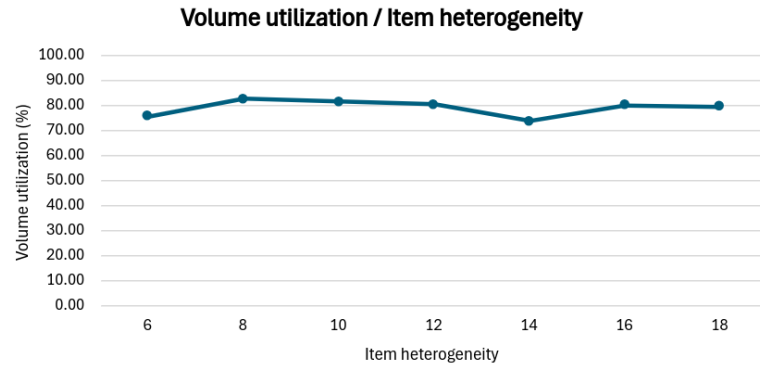


Figure 5.10: Volume utilization for different heterogeneity packing solutions

The characteristics of the packed solutions are detailed in Appendix A.1. Bounding boxes were selected randomly, ensuring the inclusion of at least one type from each size class (small, medium, and large). Notably, the number of packed bounding boxes decreases with decreasing box size. This is due to the algorithm's prioritization of larger bounding boxes, ensuring better volume utilization by prioritizing those with greater volume, as already outlined in Section 5.2.2. For instance, Table A.2 demonstrates that the four smallest bounding box types are not fully packed due to the size-based prioritization outlined in Section 4.6.2. Figure 5.11 further supports this observation, displaying the proportion of packed bounding boxes by size for each heterogeneity level. Across all levels, all large bounding boxes are consistently packed.



Figure 5.11: Bounding boxes packed for different bounding box sizes

Across all solutions with varying numbers of bounding box types, the packing algorithm

achieved an average volume utilization of 79.29% and an average packing time of 14.069 seconds, as shown in Table 5.10. This higher volume utilization is attributed to the greater packing density facilitated by the strong heterogeneity of bounding boxes. However, the increased packing density necessitates more collision checks and computations, resulting in a longer average packing time compared to previous scenarios. Figure 5.12 demonstrates a packing solution for the SKP.

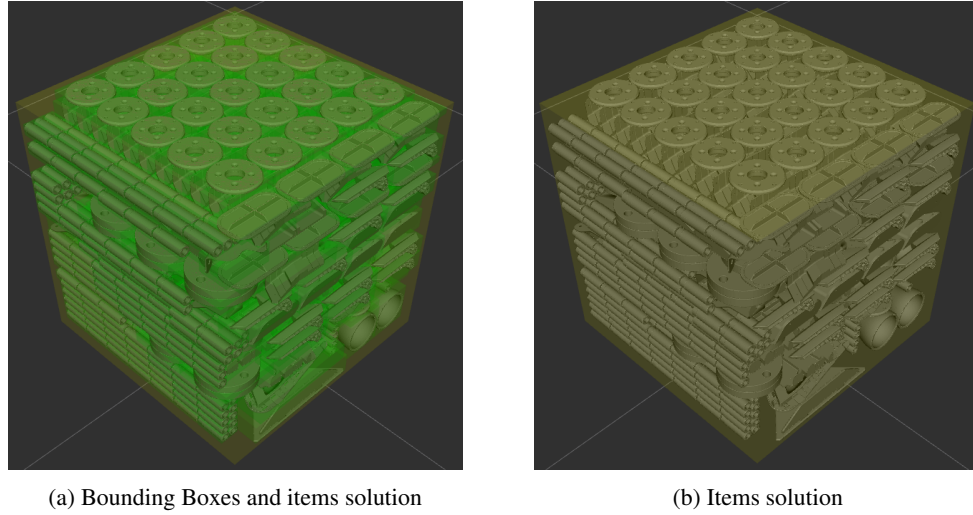


Figure 5.12: SKP packing solution

Table 5.10: Average performance for strongly heterogeneous bounding boxes packing

Average volume utilization (%)	Average packing time (s)
79.28	14.069

A pattern emerges during testing, revealing a correlation between bounding box heterogeneity and packing performance. Comparing the averages across the three heterogeneity packing solutions types, we observe an increase in average volume utilization corresponding to an increase in bounding box heterogeneity, alongside an increase in packing time. This suggests that greater bounding box heterogeneity leads to improved packing solutions, but at the cost of increased packing time due to the heightened complexity of such solutions. Figure 5.13 illustrates a packed container of different dimensions, demonstrating the versatility of this tool in packing various object sizes and container dimensions.

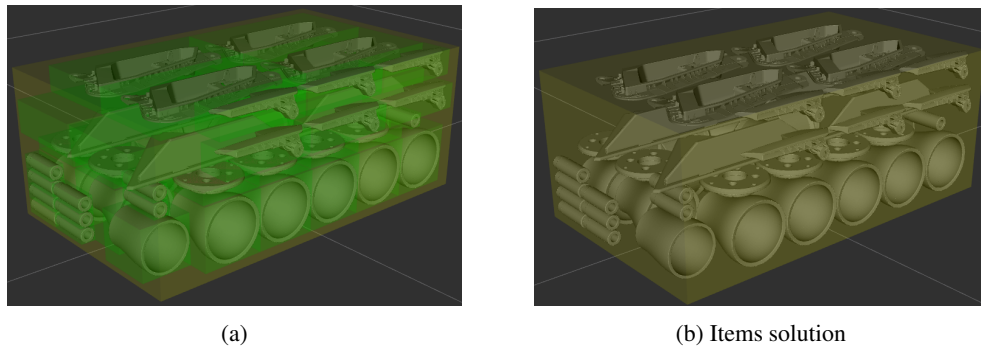
Figure 5.13: Container $500 \times 300 \times 200$ packing solution

Table 5.11: Large and medium bounding boxes packing combination

Bounding box type	Bounding boxes to pack	Packed bounding boxes
support_bracket	51	0
puller_right	50	10
falange_larger	49	28
t_bracket	47	5
reinforced_bracket	40	35
falange_piece	32	32
multi_side_bracket	29	29
reductor_wpb_smaller	29	29
reductor_wpb_larger	16	16
carcass_left	11	11
triangle_bracket	8	8
knee_tube	3	3
Volume utilization (%)	Packing time (s)	
74.43	8.049	

Table 5.12: Large and small bounding boxes packing combination

Bounding box type	Bounding boxes to pack	Packed bounding boxes
rod	504	180
cover_plate	150	31
bearing_holder	108	33
double_side_bracket	71	71
falange_smaller	71	71
single_side_bracket	69	69
reductor_wpb_larger	19	19
carcass_left	14	14
triangle_bracket	10	10
knee_tube	3	3
Volume utilization (%)	Packing time (s)	
79.28	18.305	

Table 5.13: Medium and small bounding boxes packing combination

Bounding box type	Bounding boxes to pack	Packed bounding boxes
rod	360	142
cover_plate	107	18
bearing_holder	77	25
double_side_bracket	51	7
falange_smaller	51	51
single_side_bracket	49	49
support_bracket	44	44
puller_right	43	43
falange_larger	42	42
t_bracket	40	40
reinforced_bracket	35	35
falange_piece	27	27
multi_side_bracket	25	25
reductor_wpb_smaller	25	25
Volume utilization (%)	Packing time (s)	
78.93	22.855	

Chapter 6

Conclusions

This dissertation introduces a heuristic-based automatic algorithm for the CLP and a manual packing tool within the context of robotic packing systems. The algorithm employs a layer-based building approach, prioritizing larger bounding boxes and then systematically filling horizontal layers followed by the vertical stacking of layers. The dissertation successfully developed both a manual and automatic packing tool for arranging items within a container. The manual tool supports human operators in designing packing configurations, while the automatic tool utilizes a heuristic algorithm to generate efficient solutions, particularly for large-scale or complex scenarios. Both tools contribute to improving packing processes in the retail industry.

The algorithm's performance was evaluated across three packing problems: IIPP, SLOPP, and SKP, each representing different levels of bounding box heterogeneity. Results indicate the algorithm's volume utilization, consistently exceeds 72% across multiple scenarios. However, there is a trade-off between packing efficiency and computational time, with increased heterogeneity or increased number of items, leading to longer packing times due to the complexity of finding suitable placements.

6.1 Future Work

Future work could focus on enhancing the automatic packing tool by incorporating manual interactivity, allowing users to refine and customize the algorithm-generated solutions. This would provide a more user-centric approach, combining the efficiency of automation with the flexibility of human decision-making. Additionally, exploring the use of irregular item representations instead of bounding boxes could lead to tighter and more precise packing configurations, maximizing space utilization within the container. Figure 6.1 demonstrates a limitation of packing using bounding boxes, where the "triangle_bracket" pieces are not supported due to their irregular shape.

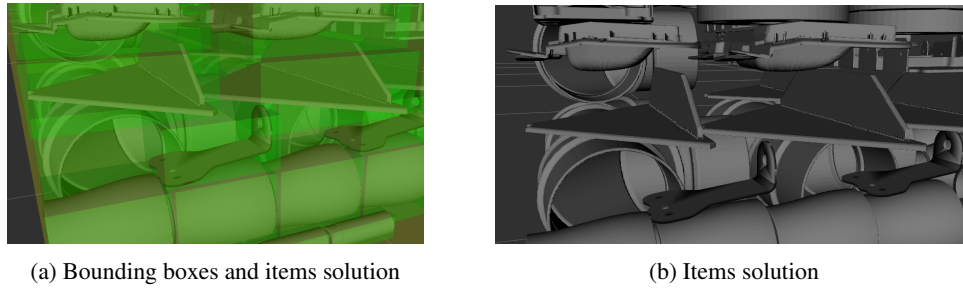


Figure 6.1: Packing using bounding boxes for irregular items limitation

To further refine the packing solutions, investigating the integration of improvement heuristics, such as genetic algorithms or tabu search, could enhance the results obtained by the initial constructive heuristic. These heuristics could explore a wider range of potential solutions, potentially leading to even higher volume utilization. Finally, incorporating a wider range of practical constraints, including weight limits, weight distribution, stability, and pattern complexity, could enhance the real-world applicability of the packing tool, ensuring that the generated solutions are not only efficient but also feasible and safe for transportation and handling. Additionally, considering an online bin packing approach per layer could be beneficial in scenarios where items are added while the next layer solutions are being developed. This would allow for more dynamic and adaptive packing solutions, as items can be placed based on their availability and the current state of the container.

6.2 Final Remarks

This dissertation contributes to robotic packing by introducing a heuristic algorithm that effectively addresses the CLP. The algorithm's layer-based approach, prioritization of larger bounding boxes, and adaptability to various heterogeneity levels make it a valuable tool for various packing processes. While the algorithm demonstrates good volume utilization, future research could focus on improving its computational efficiency, especially for identical item packing problems. Integrating this algorithm with robotic control systems could lead to fully automated packing solutions, increasing productivity and cost savings in the retail and logistics industries.

Bibliography

- [1] Open Source Robotics Foundation Willow Garage, Stanford Artificial Intelligence Laboratory. Robot operating system. (Accessed 2024-07-18).
- [2] Open Source Robotics Foundation Willow Garage, Stanford Artificial Intelligence Laboratory. Ros vizualization. (Accessed 2024-07-18).
- [3] Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In *2012 IEEE International Conference on Robotics and Automation*, pages 3859–3866. IEEE, 2012.
- [4] Open Source Robotics Foundation Willow Garage, Stanford Artificial Intelligence Laboratory. Flexible collision library. (Accessed 2024-07-18).
- [5] Assimp Team. Open asset import library. (Accessed 2024-07-18).
- [6] Sara Ali, António Galvão Ramos, Maria Antónia Carravilla, and José Fernando Oliveira. On-line three-dimensional packing problems: A review of off-line and on-line solution approaches. *Computers & Industrial Engineering*, 168:108122, 2022.
- [7] John A George and David F Robinson. A heuristic for packing boxes into a container. *Computers & Operations Research*, 7(3):147–156, 1980.
- [8] Harald Dyckhoff. A typology of cutting and packing problems. *European journal of operational research*, 44(2):145–159, 1990.
- [9] Gerhard Wäscher, Heike Haußner, and Holger Schumann. An improved typology of cutting and packing problems. *European journal of operational research*, 183(3):1109–1130, 2007.
- [10] Eberhard E Bischoff and MSW Ratcliff. Issues in the development of approaches to container loading. *Omega*, 23(4):377–390, 1995.
- [11] Andreas Bortfeldt and Gerhard Wäscher. Constraints in container loading—a state-of-the-art review. *European Journal of Operational Research*, 229(1):1–20, 2013.
- [12] . *Container Packing Problem with Guillotine Cutting and Complete-Shipments Constraints*. PhD thesis, , 2016.

- [13] Johannes Terno, Guntram Scheithauer, Uta Sommerweiß, and Jan Riehme. An efficient approach for the multi-pallet loading problem. *European Journal of Operational Research*, 123(2):372–381, 2000.
- [14] Jens Egeblad, Claudio Garavelli, Stefano Lisi, and David Pisinger. Heuristics for container loading of furniture. *European Journal of Operational Research*, 200(3):881–892, 2010.
- [15] Andreas Bortfeldt. A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. *Computers & Operations Research*, 39(9):2248–2257, 2012.
- [16] Maria Teresa Alonso, Ramon Alvarez-Valdes, Francisco Parreño, and Jose Manuel Tamarit. Algorithms for pallet building and truck loading in an interdepot transportation problem. *Mathematical Problems in Engineering*, 2016(1):3264214, 2016.
- [17] MT Alonso, R Alvarez-Valdes, Manuel Iori, F Parreño, and JM Tamarit. Mathematical models for multicontainer loading problems. *Omega*, 66:106–117, 2017.
- [18] Maria Teresa Alonso, Ramón Alvarez-Valdés, Manuel Iori, and Francisco Parreño. Mathematical models for multi container loading problems with practical constraints. *Computers & Industrial Engineering*, 127:722–733, 2019.
- [19] A Paul Davies and Eberhard E Bischoff. Weight distribution considerations in container loading. *European Journal of Operational Research*, 114(3):509–527, 1999.
- [20] Valentina E Ocloo, Armin R F"ugenschuh, and Olivier M Pamen. A new mathematical model for a 3d container packing problem. *Brandenburgische Technische Universit"at Cottbus-Senftenberg*, 2020.
- [21] EE Bischoff. Three-dimensional packing of items with limited load bearing strength. *European Journal of Operational Research*, 168(3):952–966, 2006.
- [22] H Carpenter and WB Dowsland. Practical considerations of the pallet-loading problem. *Journal of the Operational Research Society*, 36(6):489–497, 1985.
- [23] JL de Castro Silva, NY Soma, and N Maculan. A greedy search for the three-dimensional bin packing problem: the packing static stability case. *International Transactions in Operational Research*, 10(2):141–153, 2003.
- [24] Liliane de Azevedo Oliveira, Vinícius Loti de Lima, Thiago Alves de Queiroz, and Flávio Keidi Miyazawa. The container loading problem with cargo stability: a study on support factors, mechanical equilibrium and grids. *Engineering optimization*, 53(7):1192–1211, 2021.
- [25] Anulark Techanitisawad and Paisitt Tangwiwatwong. A ga-based heuristic for the interrelated container selection loading problems. *Industrial Engineering and Management Systems*, 3(1):22–37, 2004.

- [26] Andreas Bortfeldt and Hermann Gehring. A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131(1):143–161, 2001.
- [27] Robert W Haessler and F Brian Talbot. Load planning for shipments of low density products. *European Journal of Operational Research*, 44(2):289–299, 1990.
- [28] Ana Moura and Andreas Bortfeldt. A two-stage packing problem procedure. *International Transactions in Operational Research*, 24(1-2):43–58, 2017.
- [29] Jidong Ren, Yajie Tian, and Tetsuo Sawaragi. A tree search method for the container loading problem with shipment priority. *European Journal of Operational Research*, 214(3):526–535, 2011.
- [30] Liu Sheng, Shang Xiuqin, Cheng Changjian, Zhao Hongxia, Shen Dayong, and Wang Feiyue. Heuristic algorithm for the container loading problem with multiple constraints. *Computers & Industrial Engineering*, 108:149–164, 2017.
- [31] Michael Eley. A bottleneck assignment approach to the multiple container loading problem. *OR Spectrum*, 25:45–60, 2003.
- [32] Silvano Martello, David Pisinger, Daniele Vigo, Edgar Den Boef, and Jan Korst. Algorithm 864: General and robot-packable variants of the three-dimensional bin packing problem. *ACM Transactions on Mathematical Software (TOMS)*, 33(1):7–es, 2007.
- [33] Søren Gram Christensen and David Magid Rousøe. Container loading with multi-drop constraints. *International Transactions in Operational Research*, 16(6):727–743, 2009.
- [34] C.S. Chen, S.M. Lee, and Q.S. Shen. An analytical model for the container loading problem. *European Journal of Operational Research*, 80(1):68–76, 1995.
- [35] Leonardo Junqueira, Reinaldo Morabito, and Denise Sato Yamashita. Three-dimensional container loading models with cargo stability and load bearing constraints. *Computers amp; Operations Research*, 39(1):74–85, 2012.
- [36] Mhand Hifi, Imed Kacem, St’ephane N’egre, and Lei Wu. A linear programming approach for the three-dimensional bin-packing problem. *Electronic Notes in Discrete Mathematics*, 36:993–1000, 2010.
- [37] Silvano Martello, David Pisinger, and Daniele Vigo. The three-dimensional bin packing problem. *Operations Research*, 48(2):256–267, 2000.
- [38] Giorgio Fasano. A mip approach for some practical packing problems: Balancing constraints and tetris-like items. *4OR*, 2(3):161–174, 2004.
- [39] Oliviana Xavier do Nascimento, Thiago Alves de Queiroz, and Leonardo Junqueira. Practical constraints in the container loading problem: Comprehensive formulations and exact algorithm. *Computers amp; Operations Research*, 128:105186, 2021.

- [40] C'elia Paquay, Michael Schyns, and Sabine Limbourg. A mixed integer programming formulation for the three-dimensional bin packing problem deriving from an air cargo application. *International Transactions in Operational Research*, 23(1-2):187–213, 2016.
- [41] S'andor P Fekete, J"org Schepers, and Jan C van der Veen. An exact algorithm for higher-dimensional orthogonal packing. *Operations Research*, 55(3):569–587, 2007.
- [42] Samir Elhedhli, Fatma Gzara, and Yi Feng Yan. A mip-based slicing heuristic for three-dimensional bin packing. *Optimization Letters*, 11(8):1547–1563, 2017.
- [43] Xiaozhou Zhao, Julia A Bennell, Tolga Bektaş, and Kath Dowsland. A comparative review of 3d container loading algorithms. *International Transactions in Operational Research*, 23(1-2):287–320, 2016.
- [44] Menschner K Gehring, Hermann and M Meyer. A computer-based heuristic for packing pooled shipment containers. *European Journal of Operational Research*, 44(2):277–288, 1990.
- [45] Gehring Hermann Bortfeldt, Andreas and Daniel Mack. A parallel tabu search algorithm for solving the container loading problem. *Parallel computing*, 29(5):641–662, 2003.
- [46] Chen-Fu Chien and Jing-Feng Deng. A container packing support system for determining and visualizing container packing patterns. *Decision Support Systems*, 37(1):23–34, 2004.
- [47] T'urkay Dereli and G'ulesin Sena Das. A hybrid simulated annealing algorithm for solving multi-objective container-loading problems. *Applied Artificial Intelligence*, 24(5):463–486, 2010.
- [48] Andrea Lodi, Silvano Martello, and Daniele Vigo. Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research*, 141(2):410–420, 2002.
- [49] Andrew Lim, Hong Ma, Jing Xu, and Xingwen Zhang. An iterated construction approach with dynamic prioritization for solving the container loading problems. *Expert Systems with Applications*, 39(4):4292–4305, 2012.
- [50] Jasim Hasan, Jihene Kaabi, and Youssef Harrath. Multi-objective 3d bin-packing problem. In *2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*, pages 1–5. IEEE, 2019.
- [51] Rodolfo Ranck J'unior, Horacio Hideki Yanasse, Reinaldo Morabito, and Leonardo Junqueira. A hybrid approach for a multi-compartment container loading problem. *Expert Systems with Applications*, 137:471–492, 2019.
- [52] H Gehring and A Bortfeldt. A genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, 4(5/6):401–418, 1997.

- [53] T'ulio AM Toffolo, Eline Esprit, Tony Wauters, and Greet Vanden Berghe. A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem. *European Journal of Operational Research*, 257(2):526–538, 2017.
- [54] Jonas Olsson, Torbj"orn Larsson, and Nils-Hassan Quttineh. Automating the planning of container loading for atlas copco: Coping with real-life stacking and stability constraints. *European Journal of Operational Research*, 280(3):1018–1034, 2020.
- [55] Michael Eley. Solving container loading problems by block arrangement. *European Journal of Operational Research*, 141(2):393–409, 2002.
- [56] Tobias Fanslau and Andreas Bortfeldt. A tree search algorithm for solving the container loading problem. *INFORMS Journal on Computing*, 22(2):222–235, 2010.
- [57] Wenbin Zhu and Andrew Lim. A new iterative-doubling greedy–lookahead algorithm for the single container loading problem. *European Journal of Operational Research*, 222(3):408–417, 2012.
- [58] Sheng Liu, Wei Tan, Zhiyuan Xu, and Xiwei Liu. A tree search algorithm for the container loading problem. *Computers amp; Industrial Engineering*, 75:20–30, 2014.
- [59] Thiago A De Queiroz, Fl'avio K Miyazawa, Yoshiko Wakabayashi, and Eduardo C Xavier. Algorithms for 3d guillotine cutting problems: Unbounded knapsack, cutting stock and strip packing. *Computers amp; Operations Research*, 39(2):200–212, 2012.
- [60] Reinaldo Morabito and Marcos Arenales. An and/or-graph approach to the container loading problem. *International Transactions in Operational Research*, 1(1):59–73, 1994.
- [61] Jos'e Fernando Gonalves and Mauricio GC Resende. A parallel multi-population biased random-key genetic algorithm for a container loading problem. *Computers amp; Operations Research*, 39(2):179–190, 2012.
- [62] Jos'e Fernando Gonalves 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.
- [63] Kyungdaw Kang, Ilkyeong Moon, and Hongfeng Wang. A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. *Applied Mathematics and Computation*, 219(3):1287–1299, 2012.
- [64] Andreas Bortfeldt and Hermann Gehring. Applying tabu search to container loading problems. In *Operations Research Proceedings 1997*, pages 533–538. Springer, 1998.
- [65] Andrea Lodi, Silvano Martello, and Daniele Vigo. Tspack: A unified tabu search code for multi-dimensional bin packing problems. *Annals of Operations Research*, 131(1-4):203–213, 2004.

- [66] Teodor Gabriel Crainic, Guido Perboli, and Roberto Tadei. Ts2pack: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research*, 195(3):744–760, 2009.
- [67] KK Lai and Jimmy WM Chan. Developing a simulated annealing algorithm for the cutting stock problem. *Computers & industrial engineering*, 32(1):115–127, 1997.
- [68] Loris Faina. A global optimization algorithm for the three-dimensional packing problem. *European Journal of Operational Research*, 126(2):340–354, 2000.
- [69] Ana Moura and Jos'e Fernando Oliveira. A grasp approach to the container-loading problem. *IEEE Intelligent Systems*, 20(4):50–57, 2005.
- [70] Mar'ia T Alonso, Ram'f3n Alvarez-Vald'0e9s, and Francisco Parre no. A grasp algorithm for multi-container loading problems with practical constraints. *4OR*, 18(1):49–72, 2020.
- [71] Merve "Ozdemir and Tuncay Yiğit. A block-building based grasp method for solving container loading problem. *Advances in Artificial Intelligence Research (AAIR)*, 3(1):12–18, 2023.
- [72] Rommel Dias Saraiva, Napoleão Nepomuceno, and Plácido Rogério Pinheiro. A two-phase approach for single container loading with weakly heterogeneous boxes. *Algorithms*, 12(4):67, 2019.

Appendix A

Appendix

A.1 Strongly Heterogeneous Packing

Table A.1: Six different bounding boxes packing solution

Bounding box type	Bounding boxes to Pack	Packed bounding boxes
double_side_bracket	119	8
puller_right	100	48
reinforced_bracket	81	81
falange_piece	64	64
triangle_bracket	16	16
knee_tube	5	5
Volume utilization (%)		Packing time (s)
75.87		6.913

Table A.2: Eight different bounding boxes packing solution

Bounding box type	Bounding boxes to pack	Packed bounding boxes
rod	630	139
cover_plate	188	37
single_side_bracket	86	22
support_bracket	77	73
puller_right	75	75
t_bracket	70	70
reductor_wpb_smaller	43	43
reductor_wpb_larger	24	24
triangle_bracket	12	12
Volume utilization (%)		Packing time (s)
82.88		20.080

Table A.3: Ten different bounding boxes packing solution

Bounding box type	Bounding boxes to pack	Packed bounding boxes
rod	504	200
bearing_holder	108	34
falange_smaller	71	31
support_bracket	61	61
falange_larger	59	59
reinforced_bracket	48	48
multi_side_bracket	35	35
reductor_wpb_larger	19	19
triangle_bracket	10	10
knee_tube	3	3
Volume utilization (%)	Packing time (s)	
81.69	17.329	

Table A.4: Twelve different bounding boxes packing solution

Bounding box type	Bounding boxes to pack	Packed bounding boxes
rod	420	162
cover_plate	125	5
double_side_bracket	59	9
falange_smaller	59	52
support_bracket	51	51
puller_right	50	50
t_bracket	47	47
falange_piece	32	32
multi_side_bracket	29	29
reductor_wpb_larger	16	16
triangle_bracket	8	8
knee_tube	3	3
Volume utilization (%)	Packing time (s)	
80.56	14.234	

Table A.5: Fourteen different bounding boxes packing solution

Bounding box type	Bounding boxes to pack	Packed bounding boxes
cover_plate	107	14
bearing_holder	77	10
falange_smaller	51	3
single_side_bracket	49	49
support_bracket	44	0
puller_right	43	43
reinforced_bracket	35	35
falange_piece	27	27
multi_side_bracket	25	25
reductor_wpb_smaller	25	25
reductor_wpb_larger	14	14
carcass_left	10	10
triangle_bracket	7	7
knee_tube	2	2
Volume utilization (%)	Packing time (s)	
73.93	8.539	

Table A.6: Sixteen different bounding boxes packing solution

Bounding box type	Bounding boxes to pack	Packed bounding boxes
rod	315	197
cover_plate	94	18
bearing_holder	68	55
double_side_bracket	45	1
single_side_bracket	43	9
support_bracket	38	37
puller_right	37	37
falange_larger	37	37
t_bracket	35	35
reinforced_bracket	30	30
falange_piece	24	24
multi_side_bracket	22	22
reductor_wpb_larger	12	12
carcass_left	8	8
triangle_bracket	6	6
knee_tube	2	2
Volume utilization (%)	Packing time (s)	
80.21	15.579	

Table A.7: All different bounding boxes packing solution

Bounding box type	Bounding boxes to pack	Packed bounding boxes
rod	280	172
cover_plate	83	36
bearing_holder	60	8
double_side_bracket	40	2
falange_smaller	39	20
single_side_bracket	38	17
support_bracket	34	34
puller_right	33	33
falange_larger	33	33
t_bracket	31	31
reinforced_bracket	27	27
falange_piece	21	21
multi_side_bracket	19	19
reductor_wpb_smaller	19	19
reductor_wpb_larger	11	11
carcass_left	8	8
triangle_bracket	5	5
knee_tube	2	2
Volume utilization (%)	Packing time (s)	
79.83	15.807	