**University Mohamed El-Bachir El-Ibrahimi of Bordj Bou Arreridj**

Faculty of Mathematics and Computer Science

Department of Operations Research


# Master Thesis

*presented by:*

**Benabida Sif Eddine**

*Submitted in fulfillment of the requirements for the degree of:*

**Master**

**Branch:** Applied Mathematics

**Speciality:** Methods and Tools for Operations Research

---

# A Hybrid Genetic Algorithm to Solve the Container Loading Problem in Condor Logistics

---

**Defended in June 2025, before the jury composed of:**

Brahmi Boualem — President

Zouache Djaafar — Examiner

Touati Hilal — Supervisor

**2024/2025**

# Dedication

To those who believed in me when I doubted myself.

To my parents, for their unwavering support, sacrifices, and love.

To my professors and mentors, especially **Sir Touati Hilal**, who guided this project with dedication and insight.

To all the silent efforts behind my progress.

*Sif Eddine Benabida*

# Acknowledgment

First and foremost, I would like to express my deep gratitude to **Sir Touati Hilal** for his supervision, guidance, and invaluable advice throughout this project. His insight and constructive feedback played a key role in the success of this work.

I am also thankful to the academic and administrative staff of the **University of Bordj Bou Arreridj, El Bachir El Ibrahimi** and the **Methods and Tools for Operations Research** master's program for the quality of education and support I received.

My appreciation goes to **Condor Logistics** for providing the opportunity and context for this research. Working on the container loading problem within a real operational environment has been a highly enriching experience.

To my family and friends, your encouragement, patience, and belief in me have been a constant source of strength.

Finally, I acknowledge all those who directly or indirectly contributed to the completion of this thesis.

*Sif Eddine Benabida*

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Introduction

In an increasingly competitive and globalized economy, logistics has emerged as a critical success factor for industrial enterprises. Efficient logistics operations reduce operational costs, improve customer satisfaction, and enhance a company's ability to adapt to dynamic markets. In this context, container loading—how goods are packed into transport units—directly impacts delivery efficiency and cost-effectiveness.

Condor Logistics, a division of the Benhamadi Group, plays a key role in managing the transportation and distribution of goods. However, current loading processes at Condor are largely manual and rely on staff experience, leading to inconsistent packing quality, underutilized capacity, and higher transportation costs. This presents a practical optimization challenge suitable for research in operations research.

The core problem tackled in this thesis is the three-dimensional bin packing problem (3D-BPP), commonly referred to in logistics as the Container Loading Problem (CLP). It involves arranging a set of heterogeneous boxes into containers (trucks, in this case) while satisfying various constraints such as volume, weight, orientation, and stacking rules.

The current manual method lacks precision, repeatability, and adaptability to dynamic logistical constraints. As a result, there is a need for an automated, intelligent system that can provide optimal or near-optimal loading plans within reasonable time frames.

The primary objective of this thesis is to develop a robust algorithmic solution to automate and optimize the container loading process at Condor Logistics. This is achieved through:

- Modeling the 3D-BPP with realistic operational constraints.
- Designing and implementing a hybrid genetic algorithm that combines evolutionary strategies with domain-specific heuristics.
- Testing the solution with real-world data from Condor Logistics to assess its performance.
- Integrating the approach into a practical, user-friendly system that can be scaled in future.

On the scientific side, this work contributes to the literature on metaheuristics by adapting the hybrid genetic algorithm of Bortfeldt and Gehring (2001) to a real industrial use case. It also explores the effectiveness of layer-based decomposition strategies, residual space processing, and constraint-aware crossover and mutation operations.

On the practical side, the research offers a working prototype that improves truck space utilization, reduces dependency on manual decisions, and enables scenario simulations for logistics

planning. These outcomes align with Condor's vision of digital transformation in logistics.

The remainder of the thesis is structured as follows:

- **Chapter 2**: Presents the context and challenges of logistics operations at Condor, identifying operational inefficiencies and justifying the need for optimization.
- **Chapter 3**: Reviews academic literature on the 3D bin packing problem, including heuristics, metaheuristics, exact methods, and recent machine learning approaches.
- **Chapter 4**: Describes the mathematical model, genetic algorithm structure, and domain-specific modifications developed to solve the container loading problem.
- **Chapter 5**: Details the implementation process, test scenarios, and experimental results using real data from Condor Logistics.
- **Chapter 6**: Summarizes key findings, discusses limitations, and outlines directions for future work.

# 1

# Analysis of the Context and Problematic of Condor Logistics

## Introduction

One of the most important factors defining a company's success is the efficiency of its logistics operations. Indeed, logistics can be seen as the backbone of the enterprise, as the activity of the latter begins with it –through procurement- ends with it-through delivery- and plays a crucial role in all internal processes.

The origins of the term *logistics* traces back to the ancient Greek *logistikos*, which means *mathematically skilled* or *reasoning related to*. It is derived from the word *logos*, which signifies *reason, calculation, or speech*. This term was used originally back in the 18th century in the military context, where it referred to the art of calculating and planning troop movements, supplies, and army organization. Then it evolved to apply to civilian domains, especially in managing goods and resource flows.

Today, logistics is a strategic instrument that allows enterprises to manage, organize, and optimize physical and informational flows, both within and outside the organization. Regulates the flow of resources, products, and information, as well as allowing traceability, synchronization, and delivery at the right place, right time, and lowest cost.

Benhamadi Group, an Algerian industrial group dedicated to the assembly of electronics and home appliances, construction materials, and agri-food products, is a good example. It was among the top 100 family-run companies in the Arab world in 2020 according to *Forbes Middle East*. One of the key elements of its success has been its creation of its logistics division, Condor Logistics. It was established to provide not only logistics services to affiliated companies, but also to provide them to a broader audience, including individual customers.

This chapter will provide analysis and a critique of the existing operations of Condor Logistics from the point of view of a Master's student of Operational Research, and identify an optimization challenge.

## 1.1 Presentation of Condor Logistics

Condor Logistics is positioned in the Ben Hamadi Group as the logistical backbone that ensures efficiency and smooth operation within its infrastructure. Its activity surpasses flow management and embodies a strategic vision that aims to optimize resources, satisfy internal and external clients, and support the growth of the group by customizing logistical services to meet the needs.

**A diverse and tailored service offering:**

Condor Logistics offers a wide range of services created to meet the needs of the group and its partners. From transporting goods between groups, securing staff transportation, to renting handling equipment and utility vehicles, each activity is designed to ensure the efficiency of the operational system. It also guarantees continuity with municipal contracts in vehicle service and maintenance, to have maximum performance and safety at every step.

**An organizational structure focused on performance**

Condor Logistics implements a comprehensive structured organizational procedure around 6 main departments:

1. **General Management:** for the direction and control of the overall strategy and operational efficiency of the company.
2. **Operations Department:** for the management of the transport activities orchestra putting customer satisfaction as the highest priority.
3. **Maintenance Department:** to guarantee the availability, safety and reliability of the equipment.
4. **Hygiene, Security, and Environment Department:** to satisfy the safety and health of the workplace and ensure environmental protection.
5. **Accounting and Finance Services:** for managing financial transactions and aspects, ensuring accuracy and profitability.
6. **Procurement Department:** for optimizing purchasing / procurement processes, reducing costs, and ensuring supply continuity.

**Figures that demonstrate Condor Logistics' scale**

Knowing a flow of 795 vehicles (of trucks, buses, forklifts, etc.) and 674 collaborators, Condor Logistics satisfies over 21,500 deliveries per year-um and covers nearly 9.6 million kilometers. These statistics show the large scale of the company's operations and its ability to satisfy the logistical needs of the Ben Hamadi Group.

**A key player in a competitive market**

Among significant competitors in the field such as Numilogue, Flèche Bleue, or Bejaia Logistics, Condor Logistics is distinguished by its ability to provide tailor-made and efficient logistics solutions through its integration within the Ben Hamadi Group infrastructure. The strength of its position as a trusted partner has also evolved through collaborating with specialized providers such as suppliers, carriers, etc.

In short, we can say that Condor Logistics is more than just a logistics provider: It is the main link that satisfies the agility, competitiveness, and longevity of the Group.

## 1.2 Strategic and Operational Objectives

Below is a representation of the company's strategic and operational objectives in a structured table format.

| **Strategic Axis** | **Operational Goal** | **Concerned Department** |
|---|---|---|
| Ensure compliance with legal and other applicable requirements | Evaluate regulatory compliance | GM |
| Meet the needs and expectations of our customers and stakeholders | Increase market share | COM |
| | Meet customer needs and expectations | COM |
| | Improve customer loyalty | COM |
| Guarantee health and improve safety of our personnel | Reduce number of HSE policy violations | QHSE |
| | Reduce material damages | OD |
| | Improve emergency response capability and quality | QHSE |
| Prevent and reduce impacts of our activities | Reduce fuel consumption | OD |
| | Decrease oil consumption | MD |
| Control and improve our processes' performance | Achieve objectives | GM |
| | Ensure continuous improvement of staff competence | HR |
| | QHSE involvement and awareness | QHSE |

Table 1.1: Strategic Objectives and Operational Goals

## 1.3 Physical and Information Flows

In this section we will pinpoint the different locations of Condor Logistics activity, the equipment available in the company's infrastructure and its internal and external flows.

### 1.3.1 Logistics Sites

The company's activity transit through these different locations:

1. **Logistics Park:** This is where all available vehicles are collected, including those used for the transportation of goods or people.
2. **Maintenance Workshop:** This is an organized space dedicated to maintaining, inspecting, and repairing equipment, machines, or vehicles. It is equipped with tools, devices, and specialists to ensure the safety, functionality, and longevity of technical assets. This space is very important for preventing breakdowns and optimizing operational performance.
3. **Logistics Platform:** This a centralized warehouse that serves as the nodal point for the flow of goods. Manage storage, sorting, and product distribution, with the aim of optimizing operations between suppliers, production units, and clients.
4. **Production Units:** These are the sites where goods or products are manufactured using raw materials or components. They are equipped with high-performance machines, assembly lines, and specialists with a single goal of transforming resources into final products.
5. **Port:** This is where international exchanges are done, wether it's for collecting raw materials for the company, or exporting final products to foreign countries.
6. **Customer:** It represents all sales points that request final products directly from the company and then either sell these products to the final customer for personal use, or to other selling points that don't connect directly to the company.
7. **Customs Clearance Unit:** This is a team responsible for clearance services, it ensures clearance procedures and customs regulations for goods are respected during their import or export to facilitate international trade.

### 1.3.2 Equipment and Technologies Used

#### Logistics Fleet

Condor Logistics has a diverse **fleet of vehicles** to cater to all transportation needs. The fleet has a broad spectrum that ranges from **vans** for small deliveries to **trucks** with 2.5T to 20T load capacity. **Freight trucks** are used for long international routes with large volumes of goods, while **low-bed trailers** and **flatbeds** are used to carry construction equipment and heavy materials. For urban

deliveries, they use **medium-duty trucks** and also **forklifts** and different **specialized loading vehicles** for warehouse operations. Staff transport is done through **buses** and **light passenger vehicles**.

#### Warehouse Equipment

The logistics platforms are equipped with **material handling systems** to help with operations. These include **clamp forklifts** for pallets, **rotating clamp forklifts** for 180 degree turns and **manual/electric pallet jacks** for short distance transport. For vertical storage **scissor-lift stackers** can place goods high up on warehouse racks and **platform trolleys** for non-palletized goods across the warehouse.

#### Technology Infrastructure

Condor Logistics uses **digital solutions** to help with operations. The **Transport Management System** (TMS) is the backbone for route optimization, shipment tracking and load consolidation, reducing operational costs. Integrated with this is **SAP ERP** for business process management across finance, HR, production and supply chain modules with real time data analytics for decision making. And **Géoflotte** real time fleet tracking to optimize routes, reduce fuel consumption and improve response time to logistical challenges through GPS and telematics.

### 1.3.3 Outbound Flow

Outbound flow refers to all the operations involved in moving finished products from the production site or storage facility to the end customer. This includes inventory management, transportation, delivery and related services. The goal is to meet customer needs efficiently and cost effectively and on time.

In the company there is one outbound flow:

1. **Shipping of finished products:** This is the movement of products from the logistics hub to the customer who ordered.

### 1.3.4 Internal Flow

Internal logistics flow ( also called production flow or intralogistics ) covers all the movement of goods and information within the company's boundaries. This includes stock handling in the warehouse, inventory control and the transfer of goods and components throughout the production network. These flows support the entire process chain of transformation, machining, handling and intermediate storage.

Condor Logistics is involved at three internal levels:

1. **Transfer of finished goods:** This is the movement of products from the production units to the central logistics platform.

2. **Transfer of raw materials:** This is the movement of raw materials from the dry port (located at the logistics hub) to the production sites.

3. **Inventory management:** This includes best practices for different types of inventories: *perpetual*, *cyclical* and *semi-annual*.

### 1.3.5 Inbound Flow

Inbound or upstream flow includes all the activities related to sourcing and procurement of raw materials, parts and supplies for production. This includes managing supplier relationships, planning purchases, receiving goods, storing goods and stock management. The goal is to have all the required resources available when needed for production to run without delay.

The company does this through:

1. **Receiving raw materials:** This is the movement of supplies from the port or local suppliers to the dry port located at the logistics hub.

### 1.3.6 Our Role

During our internship at Condor Logistics, we have been involved in the analysis and optimization of the logistics processes for the shipment and transfer of finished products. We are involved in several stages of the operational flow, mainly in the shipment of finished goods (see Figures 1.1 and 1.2). It begins when a customer requests an order from the commercial department of Condor Electronics. The commercial team then saves the order into the SAP system with all the details, including: customer name, address, phone number, email, delivery address, order number, order date, list of products with quantity, description, price, product references, and transport mode.

Once entered, the external scheduling team schedules the order into the Transportation Management System (TMS) based on driver and truck availability. They add loading voucher numbers, assigned drivers, and their trucks. Meanwhile, the stock planner checks internal resources and defines the capacity needed to meet the delivery date in the TMS. Loading vouchers are then issued and given to the team leader, who distributes them among the warehouse staff responsible for order preparation. This person also determines the manpower and equipment needed to prepare and load each order. The warehouse staff prepare orders in pre-loading zones called “fake trucks”. Then, loading vouchers are given to the staff responsible for physically loading products, who use the necessary equipment.

The driver arrives at the logistics platform with the assigned loading voucher from the scheduling team. Products are loaded into the trucks, and loading is confirmed in SAP. The billing manager prepares the invoice and the loading slip and hands them over to the driver, who proceeds with

delivery. Upon reception, the customer validates both the delivery slip and the invoice, and they are then sent back to the billing department of Condor Logistics for processing.

We are also involved in the transfer of finished products from the Business Unit (B.U.) to the logistics platform. This process includes several coordinated steps, starting with the forecast planning managed by the logistics platform's planner. The forecast plan is done weekly and includes the finished goods to be transferred, quantities, number of rotations, expected dates, and shipment priorities. This plan is communicated to the warehouse staff at the B.U. for execution. Then, the warehouse staff at the B.U. create the transfer orders in SAP and prepare the products physically according to the forecast plan. They also adjust the quantities to be shipped according to the product family and the trailer's capacity and validate these quantities in SAP. Once validated, forklift operators and handlers load the products into the trailers according to the schedule. Finally, the quantities loaded are confirmed in SAP, and delivery slips are issued.

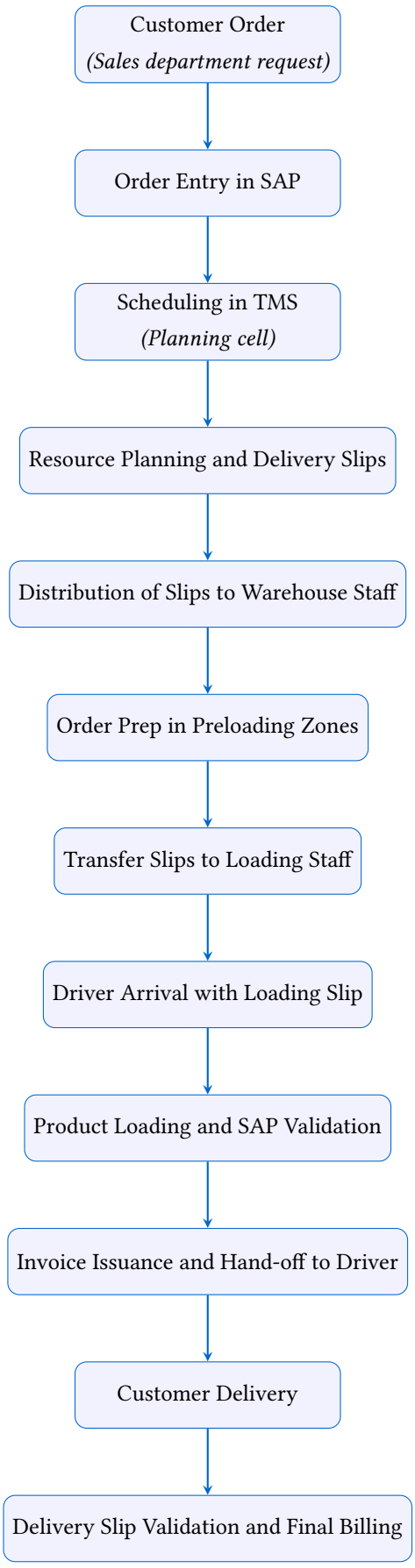



Figure 1.1: **Shipment Process Flow**

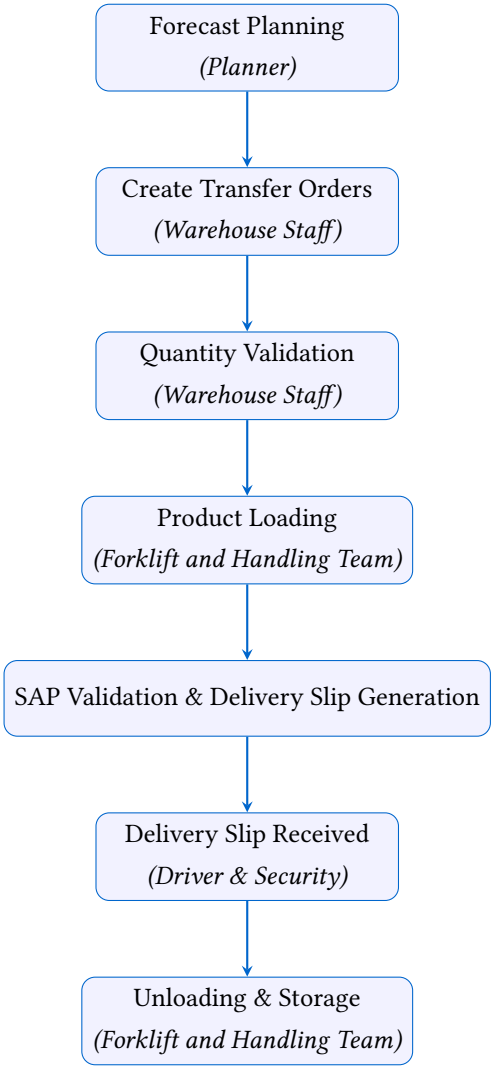



Figure 1.2: **Transfer Process Flow**

## 1.4 Logistical Bottlenecks

1. **Truck Underestimation**
   Bad planning means trucks are under estimated. So delays happen during loading and unloading especially during peak periods. The gap between available and needed cargo capacity disrupts the whole supply chain and increases operational costs due to last minute logistics arrangements. Bad planning means trucks are under estimated. So delays happen during loading and unloading especially during peak periods. The gap between available and needed cargo capacity disrupts the whole supply chain and increases operational costs due to last minute logistics arrangements.

2. **Queueing at Loading Docks**
   The restricted amount of docking space in warehouses pertaining to production keeps creating waiting lines which hinder loading and unloading activities. This issue, in turn, may increase vehicle waiting time, fuel usage, and unmet delivery schedule, thereby lowering efficiency. Reduced efficiency is observed at the distribution center. Idle waiting lines may cause schedule shifts in the production cycle because of postponements associated with raw material unloading. The lack of software that manages dock scheduling leads to simultaneous arrival of trucks, worsening the existing traffic jam.

3. **No Real Time Visibility**
   No WMS implies that there is no monitoring or management of logistic board internal flows on a real-time monitoring basis. The invisibility of facilities severely complicates the accuracy of call center inventory, hampers inter-departmental collaboration, and causes delayed reaction to agile shifts in demand supply or supply chain shocks.

   Real time visibility gaps stifle warehouse supervisor's capability to rectify inefficiencies like improper arrangement of items in relation to demand levels, excessive stock level, packing and picking traffic jams. Additionally, the lack of integration between WMS, ERP and TMS is a transparency hurdle towards a seamless flow of information. From a granular actionable standpoint, firms stand to benefit from employing IoT sensors, RFID tags, cloud based dashboards for stock level influx and efflux as well as warehouse utilization.

4. **Manual Programming of Orders in the TMS**
   Orders are placed and scheduled manually within the Transport Management System (TMS) which is a slow, lacks available resources, and contains several errors. When dealing with a high number of transactions, this inefficiency becomes critical and leads to substential planning inconsistencies and suboptimizations with routes.

   In the case that there is an order cancellation, a last-minute addition, or a re-routing due to blocked roads, manual processes become unresponsive. Due to this, the re-planning process is slow and cumbersome. However, with automation through electronic data interchange (EDI)

and API, an order can be seamlessly extracted from the sales platform, and route optimization occurs in real-time using advanced planning modules with live traffic data and delivery constraints.

5. **Manual Organization of Truck Loading**
   The loading stage is highly dependent on employees' relative experience, which results in unsatisfactory space usage and inefficient loading duration. The operational processes and autonomous methods which could be developed to facilitate the loading activities overlook fundamental restraining factors such as cubed weight limits, delivery order, delicate cargo items, and other critical frangibility factors which mark significant risks for additional value loss and added costs for logistics services.

   Manual loading is dangerous, not just in terms of worker ergonomics but also in relation to the balance of the load and its stability throughout the movement. Sophisticated software for space management creates advanced suggestions using 3D simulation technologies to render optimal box positions that maximize space. These systems are capable of applying rules particular to specific products to design enhanced operational efficiencies while maximizing safety for unstowed containers.

## 1.5 Problem Statement

While comprehensively studying the logistics processes of the company, we noted a number of highly impactful problems—primarily the absence of a mechanism to decide how items needing to be loaded into cargo trucks should be packed in an optimal manner. This gap leads to strategic space and resource mismanagement. Currently, these activities are performed manually, relying on the knowledge and skills of the relevant teams to manage the deficits. While this approach may seem functional for now, it poses significant issues, such as an increased risk of human error, poor decision-making, and considerable time wasted on tedious and complex calculations and processes that often require repeated rearrangement. This time could instead be allocated to more critical tasks that cannot be automated or delegated to machines.

As Operations Research students, we plan to use the skills and knowledge gained during our academic training to tackle this challenge and propose a viable solution. Our aim is to introduce a data-driven approach based on mathematical modeling and algorithmic optimization to improve efficiency and enhance the traceability of the vehicle loading process. Such a solution would reduce human error, enable decision-makers to act more effectively using reliable and measurable outcomes, and allow teams to devote more time to tasks that require strategic or creative thinking.

In this way, our approach aligns with the company's commitment to innovation and continuous improvement, supporting its long-term vision.

### 1.5.1 Problem Formulation

#### Definition

The problem at hands is known as the **Vehicle Loading Problem** (or sometimes called the *Bin Packing Problem*). It has the goal of finding the optimal arrangement of items inside trucks (or bins) in order to maximize the stowage of the latter while satisfying constraints like those related to weight, volume, or stability.

#### Objectives

The main goal of the Vehicle Loading Problem is to use existing space efficiently, adherence to physical constraints, and overall cost reduction. Maximizing the use of available truck space is essential to reducing the number of vehicles required, thereby lowering transportation costs. At the same time, it is crucial to ensure that loading does not exceed the maximum weight capacity, that the volume of the loaded items stays within the truck's limits, and that the load is stable to avoid risks during transport.

Cost minimization involves reducing the number of trucks, fuel consumption and also optimizing associated logistics expenses. Operational efficacity is another important objective, because the simplification of the loading process leads directly to the meeting of the delivery deadlines. Moreover, the problem requires the consideration of operational constraints, such as unloading order, item fragility, and compliance with regulations. To sum up, the primary objective is to find a balance between resource optimization, constraint satisfaction, and cost reduction, while ensuring safe and effective loading.

#### Solution Approach

To deal with this problem in a structured and efficient way, we will follow a logical methodology divided into several key phases. Firstly, we will conduct a review of the literature related to the problem to draw inspiration from existing research and case studies on similar challenges, identify best practices, commonly used methods, and the limitations of existing solutions.

After that comes the collection of relevant data needed to solve the problem, such as the characteristics of the items to be loaded (weight, volume, fragility), truck container specifications (capacity and dimensions), and operational constraints (like stacking rules and allowed orientations). To ensure the efficiency of the modeling process, it is essential to guarantee the quality of this data.

Following data collection, we will develop a mathematical model that formalizes the problem by defining an objective function, such as minimizing the number of trucks or maximizing space utilization. The model must include various constraints—weight, volume, and stability—and be solved using appropriate existing approaches (such as heuristics or genetic algorithms), and tools like Python or CPLEX, adjusted to fit the practical case at hand.

Once the model is formulated, the next step is to test and simulate its performance using historical data. This will involve scenario-based testing by stochastically varying parameters such as order quantity or truck types to assess the model's robustness. The goal of this phase is to enhance the performance of the solution approach based on the test results.

Model validation is a crucial step, involving a comparison of the model's outputs with real-life cases to ensure accuracy and reliability. Domain experts will validate the solution during this phase.

The solution will then be deployed within the company's operations. This includes training relevant teams, integrating the model into daily workflows, and measuring its performance against key performance indicators (KPIs) before and after implementation—such as cost reduction, space optimization, and delivery compliance. We will also identify opportunities for continuous improvement to maximize the solution's impact.

# 2

# Review of the Literature

## 2.1 Origins and Foundations

In combinatorial optimization and industrial operations the problem of container loading - how to pack items into a confined space - has become a major issue in logistics, transportation and supply chain management. This problem is formally known as the Bin Packing Problem (BPP) and has received a lot of attention in the academic community because of its theoretical complexity and practical importance. The three-dimensional variant (3D-BPP) where the goal is to place a set of rectangular boxes into a larger container without overlap and respecting various constraints has direct applications in cargo loading, warehousing and manufacturing. This section describes the early developments in this field, tracing the foundational contributions that shaped the evolution of BPP from its one-dimensional formulation to more realistic three-dimensional industrial models.

The bin packing problem in its classical form goes back to early computer science research on resource utilization. One of the first major contributions to the problem was by [Coffman et al., 1984] who worked on the 1D version of BPP. They gave theoretical bounds and approximation algorithms. Through the analysis of worst case scenarios and polynomial time heuristics Coffman and colleagues established the mathematical foundation for higher dimensional variants. These early papers not only addressed real world packing problems but also started the discussion on NP-hard problems and approximation algorithms.

Following on from this theoretical work the 1990s saw a lot more interest in more complex and realistic versions of the problem. In particular [Bischoff and Ratcliff, 1995] looked at several practical issues that arise in real world container loading. They included weight distribution, box stability and allowed orientations - all important in maritime and air freight. The inclusion of these constraints was a big move from abstract to industrial models. They showed that feasibility of a solution is not just about spatial fit but also about operational safety and integrity of the load especially in dynamic environments like shipping. At the same time algorithmic work was also progressing. The paper by [Martello and Toth, 1990] presented some of the first exact algorithms for the three-dimensional bin packing problem. Using dynamic programming and branch-and-bound

Martello and his colleagues solved simplified instances of the 3D-BPP and found optimal solutions under constrained conditions. These results provided a baseline to evaluate the performance of future heuristics and metaheuristics. They also showed the trade-offs between solution quality and computational cost a theme that is still present in most of the research in this area.

The 1970s to 1990s laid the methodological and conceptual groundwork for 3D bin packing research today. The progression from 1D theoretical models to 3D constraint rich applications shows how this field has evolved and the lasting impact of its pioneers.

## 2.2 Development of Heuristics

As the foundational methods matured the focus in the next decade (1990-2000) was on developing heuristics to tackle larger and more complex instances of the 3D bin packing problem. Heuristics by their nature provide near optimal solutions with much reduced computational requirements compared to exact methods so are very useful for industrial applications where time and flexibility is key.

One of the most interesting was [Gehring and Bortfeldt, 1997] who combined genetic algorithms with the "wall-building" heuristic. This simulated natural evolution with domain specific rules for box placement. The wall-building heuristic mimics human-like stacking by placing boxes against one or more container walls, thus improving space utilization and load stability. By combining stochastic search with structured placement rules Gehring and Bortfeldt's approach was a big step forward in balancing solution quality and computational efficiency.

Meanwhile a full benchmarking of placement heuristics was done by [Pisinger, 2002]. 15 heuristics were tested on a set of benchmark instances from real-world industrial cases. These heuristics included layer-building, volume-sorting and orientation-prioritization strategies. The authors measured the space usage and computation time of each heuristic and also their adaptability to practical constraints like load stability and weight distribution. Their results showed that empirically driven rules can produce good solutions for medium sized problems and that domain knowledge can be embedded in the algorithm to improve performance. Another important contribution from this period was [Moura and Oliveira, 2001] who introduced dynamic stability constraints in the packing process. Unlike static constraints which consider load balance in a stationary state, dynamic stability takes into account movement and external forces that occur during transport – acceleration, tilting or vibrations. Moura's work bridged the gap between what academics study and the messy realities of logistics—especially in maritime and air freight, where unstable loads cause all sorts of operational and safety headaches. By accounting for those dynamic factors, Moura's work took 3D-BPP models beyond just static optimization—and opened the door to much more realistic, much more robust applications.

That decade of research showed just how far the bin packing problem had come. We moved from the foundational concepts and exact methods that got us started to more flexible, more scalable and more aware of their constraints heuristics. That progress wasn't just about making computations

faster-it also made bin packing solutions much more useful in all sorts of industries.

## 2.3 Metaheuristics and Hybridization

In the 2000–2010 decade the field of 3D bin packing moved towards more advanced and computationally powerful solution frameworks, especially metaheuristics and hybrid algorithms. The previous decade laid the foundation for heuristics but it became clear that more flexible and adaptive strategies were needed to tackle the large and constraint rich problem instances that appeared in real world scenarios. Metaheuristics by design are high level abstractions that allow algorithmic components to be tailored to specific problem domains while still being general purpose search algorithms. This is perfect for the 3D-BPP.

A big step forward was made by [Bortfeldt and Gehring, 2006] who introduced genetic algorithms with adaptive mutation operators. Traditional genetic algorithms are good at global exploration but bad at local exploitation and often converge or stagnate in suboptimal regions of the solution space. Bortfeldt's approach solved this by making mutation rates based on population diversity and fitness improvements. This way the algorithm can keep the balance between exploration and exploitation throughout the search process and is more robust and faster converging. The adaptive strategy not only improved the overall solution quality but also allowed to handle more complex packing constraints like heterogeneous item dimensions and orientation rules.

At the same time, work on algorithmic scalability led to the development of parallel search methods. [Crainic et al., 2008] made a big contribution by applying parallel Tabu Search to the 3D-BPP. They used a distributed computing environment to run multiple Tabu Search instances in parallel, each exploring different parts of the solution space and exchanging information periodically. This allowed to diversify and intensify the search and to solve large industrial instances that were previously not computable. They also introduced sophisticated memory structures and adaptive tabu tenure to avoid cycling and to converge faster. Their results showed the importance of parallelism to extend metaheuristics to high dimensional optimization problems with complex constraints.

Another big breakthrough during this period was [Fanslau and Bortfeldt, 2010] who proposed an expert system that combined domain specific physical rules with optimisation procedures. Unlike traditional metaheuristics that only use abstract operators, this hybrid system included physical stability checks, real-time constraint validation and domain logic in the optimisation loop. Through the simulation of real world packing dynamics – load shifts, tilting and container stress limits – Fanslau and Bortfeldt achieved a level of robustness and realism that set a new standard for practical applicability. Their system was particularly useful in industrial environments where physical constraint violations could mean big financial or operational risks. And the expert system architecture allowed for interpretability and user interaction so practitioners could tweak rules and preferences to their specific logistical needs.

Together these made a big step forward in bin packing solutions. The combination of metaheuristic strategies with adaptive, parallel and expert driven components was the maturing of the

field – moving from academic prototypes to real world deployment. These hybrid approaches leveraged the strengths of multiple methods, flexibility, scalability and realism that traditional algorithms couldn't match. So the 2000-2010 period laid the foundation for the next generation of intelligent, domain aware optimisation systems that are still shaping the state of the art in 3D bin packing and related logistics problems.

## 2.4 Exact Methods and Polyhedral Approaches

Building on the metaheuristic and hybrid progress of the last decade, the 2010-2020 period saw a resurgence of exact methods for the 3D bin packing problem (3D-BPP) driven by significant improvements in solvers, computing power and modeling techniques. With commercial and open-source MILP (Mixed Integer Linear Programming) solvers getting more efficient, researchers started to look into solving moderate sized instances of 3D-BPP to optimality – a task that was previously considered impossible due to the combinatorial explosion of the solution space.

One of the main contributions in this direction was [Junqueira and Morabito, 2013] who proposed a MILP model for the 3D-BPP that linearizes the geometric constraints of positioning and non-overlapping of boxes in 3D space. They were able to use state-of-the-art MILP solvers like CPLEX or Gurobi to find exact solutions thanks to the breaking down of the spatial relationships into a set of binary and continuous variables that can be expressed linearly. Although the computational time was still too high for large instances, it was very good for medium sized instances, especially when combined with preprocessing and symmetry-breaking constraints. Having exact solutions was also a nice benchmark to test heuristics and metaheuristics.

Along with this line of research [Martello and Pisinger, 2017] introduced a branch-and-bound algorithm for 3D bin packing. This algorithm featured a new lower bound that reduced the search space and allowed for more aggressive pruning. The bounding used volume-utilization metrics and dynamic programming to check partial packings. The algorithm also used node selection and branching rules based on theory and practice. As a result they were able to cut the computation time down significantly compared to previous exact methods and now the approach is applicable to a wider range of problems.

At the same time the field also saw foundational theoretical work on the 3D-BPP. [Silva et al., 2020] did a complexity analysis of many of the subproblems inside the 3D bin packing framework. By classifying the NP-complete substructures (e.g. item ordering, orientation feasibility, contact stability) they provided a polyhedral view that has since been used to design more targeted and efficient algorithms. This classification allowed for the development of hybrid algorithms that apply exact methods to the tractable parts and approximation methods to the intractable parts. Silva et al. also introduced a modular taxonomy to compare algorithmic performance across different 3D-BPP variants and to standardize the benchmarking.

In total these advances represent a big step forward in exact optimization in 3D packing. The combination of MILP modeling, advanced bounding and complexity theory has not only allowed to

solve more instances but also to better understand the problem space. Heuristics and metaheuristics will always be needed to tackle large industrial problems, but the last decade advances have laid the foundation for hybrid algorithms that combine the precision of exact methods with the flexibility of approximate search. These advances will lead to more intelligent and scalable systems that can deliver optimality and efficiency in real world logistics and container loading.

## 2.5 Machine Learning and Recent Trends

In the latest wave of research (2020-present) machine learning especially deep learning and its variants have become a game changer for the 3D-BPP. As the limitations of combinatorial approaches became apparent with real time industrial demands and high dimensional input data, researchers started to move towards data driven methods that can generalise over different problem instances and learn from experience. This is a big paradigm shift in the field bringing together reinforcement learning, neural networks and hybrid algorithmic frameworks to achieve unprecedented adaptability and performance.

This was achieved by [Hu et al., 2021] who introduced a deep reinforcement learning (DRL) framework dedicated specifically for the 3D-BPP. To avoid traditional reinforcement learning which struggles with the high dimensional action spaces of packing problems, their method used a new form of reward shaping that incorporates spatial awareness. This allows the agent to get more informative feedback about the quality of partial solutions during training and thus converge faster and guide the policy to better packing strategies. The model learned to balance multiple constraints such as spatial fit, stability and orientation on its own and outperformed several heuristics on benchmark datasets. By capturing the sequential and geometric nature of packing decisions, Hu et al's DRL agent showed that intelligent systems can adapt to complex operational settings without manual encoding.

Building on this machine learning momentum [Zhao et al., 2022] did an analysis of 45 AI driven and hybrid algorithms for 3D-BPP and its variants. The evaluation included approaches based on reinforcement learning, supervised learning, evolutionary algorithms and hybrid frameworks that combine machine learning with classical optimization techniques. The results showed that the best approaches tend to combine the global search capabilities of learning based models with the precision and constraint handling strength of traditional operations research tools. For example systems that use machine learning to generate good initial solutions or estimate heuristic parameters and then refine them with MILP solvers or metaheuristics performed well across industrial benchmarks. Zhao et al's work not only showed the benefits of interdisciplinary integration but also provided a methodological framework to evaluate algorithmic trade-offs in practice.

And this is taken to the next level by recent work such as [Zhang et al., 2023] who use the attention mechanisms from Transformer to model and predict the optimal placement sequences. They treat the packing as a sequence generation problem, like language modeling, where each action is placing a specific item in the container. On industrial benchmarks Zhang et al.'s model achieves

high packing efficiency and computational scalability so we can apply deep learning to combinatorial optimization problems.

Together this means a new era of packing algorithms - one that goes beyond static rule based systems to adaptive intelligent decision making. Deep learning not only makes solutions more scalable and flexible but also opens up new possibilities for real time data informed logistics optimisation. As computational power grows and more labeled packing data becomes available the combination of machine learning and operations research will be even more powerful in the next decade for robust, efficient and autonomous packing systems.

# 3

# Solution Approach

## 3.1 Mathematical Model of the Container Loading Problem

The container loading problem (CLP) is a classical optimization challenge in operations research and logistics. It involves determining the optimal way to stow a set of rectangular boxes into one or more containers in order to maximize the utilization of space. Depending on the context, this problem may be classified as a single-container or a multi-container problem. In both variants, a variety of practical constraints must be considered, including volume and weight limitations, box orientation restrictions, and physical feasibility such as support and non-overlapping placement.

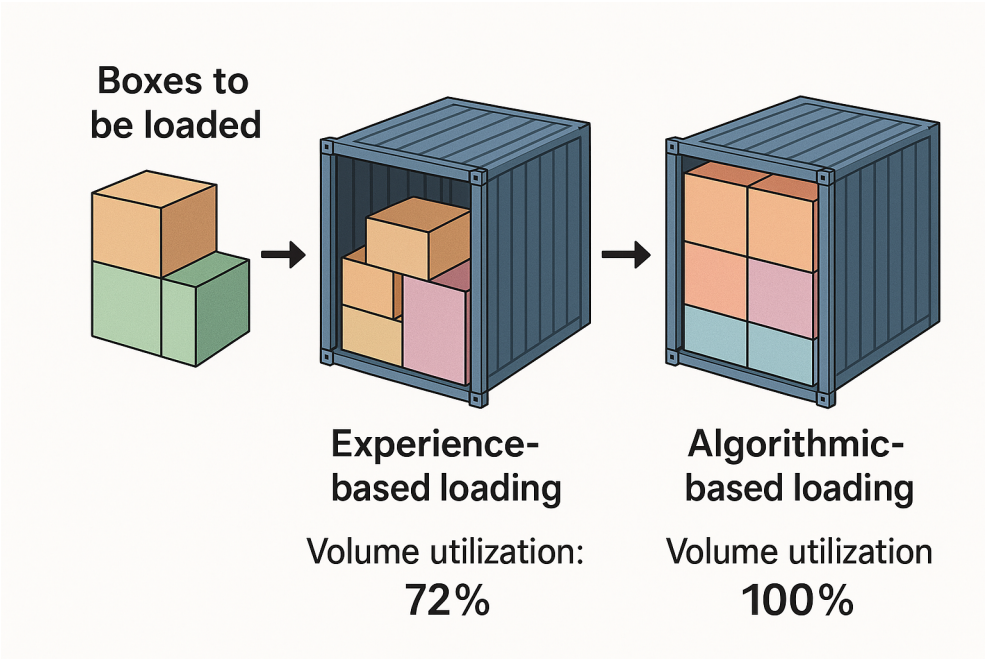



Figure 3.1: Container Loading Problem.

To formulate a mathematical model of the CLP, we first define the decision variables and essential parameters that capture the structure of the problem.

**Let:**

- $B = \{1, 2, \ldots, n\}$ be the set of boxes to be loaded.
- $C = \{1, 2, \ldots, m\}$ be the set of available containers (in the single-container case, $m = 1$).
- $(l_i, w_i, h_i)$ be the dimensions of box $i \in B$.
- $(L_c, W_c, H_c)$ be the dimensions of container $c \in C$.
- $v_i = l_i \cdot w_i \cdot h_i$ be the volume of box $i$.
- $M$ be a sufficiently large constant.

To determine the loading configuration, we introduce the following decision variables:

- $x_i, y_i, z_i$: coordinates of the lower-left-back corner of box $i$ inside its assigned container.
- $r_{ik} \in \{0, 1\}$: binary variable indicating whether rotation $k$ is applied to box $i$ (among the 6 possible orientations).
- $a_{ic} \in \{0, 1\}$: binary variable indicating whether box $i$ is assigned to container $c$.
- $o_{ij} \in \{0, 1\}$: binary variable indicating whether box $i$ is placed strictly before box $j$ in at least one axis to avoid overlap.

**Objective Function:**

The goal is to maximize the total volume of boxes successfully loaded into the containers:

$$\max \sum_{i \in B} \sum_{c \in C} a_{ic} \cdot v_i$$

**Subject to:**

- **Each box is assigned to at most one container:**

$$\sum_{c \in C} a_{ic} \leq 1 \quad \forall i \in B$$

- **Only one orientation is allowed per box:**

$$\sum_{k=1}^{6} r_{ik} = \sum_{c \in C} a_{ic} \quad \forall i \in B$$

- **Boxes must fit within container dimensions:**
  For all $i \in B$ and $c \in C$:

$$\begin{aligned} x_i + l_i^{(r)} &\leq L_c + M(1 - a_{ic}) \\ y_i + w_i^{(r)} &\leq W_c + M(1 - a_{ic}) \\ z_i + h_i^{(r)} &\leq H_c + M(1 - a_{ic}) \end{aligned}$$

  where $(l_i^{(r)}, w_i^{(r)}, h_i^{(r)})$ are the dimensions of box $i$ under rotation $r$.
- **Non-overlapping constraint:**

For all pairs of boxes $(i, j)$ assigned to the same container:

$$o_{ij} + o_{ji} \geq a_{ic} + a_{jc} - 1 \quad \forall i < j, \forall c \in C$$

and $o_{ij}$ implies:

$$x_i + l_i^{(r)} \leq x_j + M(1 - o_{ij}), \quad \text{or analogous for other axes}$$

- **Weight and volume constraints:**
  If $w_i$ is the weight of box $i$ and $W_c^{\max}$ is the capacity of container $c$, then:

$$\sum_{i \in B} a_{ic} \cdot w_i \leq W_c^{\max} \quad \forall c \in C$$

- **Full support constraint:**
  For each box $i$ not on the floor ($z_i > 0$), it must be supported by other boxes beneath it:

$$\text{Area}_{\text{support}}(i) \geq \alpha \cdot \text{BaseArea}(i) \quad \text{with } \alpha \approx 1.0$$

  (This is typically implemented via discretization or via geometric coverage checks.)
- **No floating boxes (gravity constraint):**
  A box $i$ must be placed either on the container floor or on top of another box $j$:

$$\text{If } z_i > 0 \Rightarrow \exists j \text{ such that box } j \text{ supports } i$$

**Discussion:**

In the case of a single-container problem, the model simplifies by fixing $|C| = 1$, in which case the container index $c$ can be omitted. However, the structure and logic of the constraints remain the same. In the multi-container case, the model allows flexibility in box-to-container assignments, subject to capacity constraints. The mathematical formulation aims to ensure a physically feasible, non-overlapping, and stable loading plan while optimizing for maximum space utilization.

Building upon the insights gathered through the contextual analysis and literature review, the following sections present the proposed solution to the Container Loading Problem (CLP) faced by Condor Logistics. The operational challenges identified—such as inefficient manual loading, poor space utilization, and lack of systematic optimization—necessitate a robust computational approach that is both efficient and adaptable to real-world constraints.

The solution is grounded in the framework of combinatorial optimization and leverages the strengths of metaheuristic methods. In particular, a Hybrid Genetic Algorithm (HGA) is adopted due to its proven effectiveness in tackling high-dimensional, non-linear, and constraint-rich problems like the 3D Bin Packing Problem (3D-BPP).

The following sections are divided into three primary parts, corresponding to:

- **General description of Genetic Algorithms:** Introduces the basic concepts of a genetic algorithm, as long as its structure and different operators with their utility.
- **Single-Container Loading Problem:** Focuses on the efficient arrangement of heterogeneous boxes into a single container, ensuring maximal space utilization while respecting stability, orientation, and stacking constraints.
- **Multi-Container Loading Problem:** Extends the single-container logic to situations where a set of items must be distributed across multiple containers, with the aim of minimizing the total number of containers used.

Both components of the solution are implemented in Python and designed to be modular, scalable, and easily integrated into Condor's logistics software infrastructure. Detailed explanations of the solution design, data structures, heuristics, genetic operators, and performance considerations are provided in the following sections, starting with the single-container loading scenario.

## 3.2 Genetic algorithms

Genetic Algorithms (GAs) are stochastic search and optimization techniques inspired by the process of natural selection and genetics. They belong to the broader family of evolutionary algorithms and are particularly suited for solving complex combinatorial and nonlinear problems. A GA operates on a population of candidate solutions, evolving them over time through biologically inspired operations such as selection, crossover, and mutation.

### 3.2.1 Structure of a Genetic Algorithm

A typical Genetic Algorithm follows the steps below:

1. **Initialization:** Generate an initial population of individuals, usually randomly.
2. **Evaluation:** Compute a fitness score for each individual using a problem-specific objective function.
3. **Selection:** Choose individuals from the current population to become parents, based on their fitness.
4. **Crossover:** Create new individuals (offspring) by combining genetic material from two parents.
5. **Mutation:** Apply small random changes to offspring to maintain genetic diversity.
6. **Replacement:** Form a new population by replacing some or all individuals from the previous generation.

7. **Termination:** Repeat steps 2–6 until a stopping criterion is met (e.g., maximum generations or satisfactory fitness).

### 3.2.2 Genetic Operators

The performance and convergence of a GA largely depend on the proper use of its core genetic operators:

- **Selection:** Determines which individuals reproduce. Common techniques include roulette wheel selection, tournament selection, and rank selection. The goal is to favor fitter individuals while preserving population diversity.
- **Crossover (Recombination):** Combines the information of two parent solutions to produce offspring. It allows the algorithm to explore new regions of the solution space. In the context of the 3D Bin Packing Problem, crossover may involve merging sequences of box placements or orientations.
- **Mutation:** Introduces small random changes to an individual to maintain genetic diversity and avoid premature convergence. For example, mutation may alter the order of boxes or change the orientation of a box in a packing solution.
- **Replacement (Survivor Selection):** Determines which individuals survive into the next generation. This can be based on fitness (elitism), diversity, or age.

GAs are particularly attractive for the 3D Bin Packing Problem due to their flexibility in handling constraints, non-linearity, and large combinatorial search spaces. In this project, we implement a **hybrid version** of the standard GA by combining it with domain-specific heuristics and problem decomposition strategies to improve packing efficiency and convergence speed.

## 3.3 Single-Container Loading Problem

The problem of packing goods into transport containers is a common issue in logistics, modelled as the Container Loading Problem (CLP). This is the problem of selecting and placing a subset of rectangular boxes into a single, fixed size container to maximise space usage while adhering to real world constraints such as weight limits, stability and load balance.

In practice CLP instances are 3D knapsack problems, especially when not all items need to be loaded and the objective is to maximise the value or volume of the loaded items. A further distinction is made between homogeneous and heterogeneous sets of boxes. In this work we consider a strongly heterogeneous set, where the number of box types is large and each type only appears a few times, making the packing strategy more complex.

In order to address Container Loading Problems with practical constraints, Bortfeldt and Gehring [2001] created a hybrid genetic algorithm (GA) that incorporates a domain-specific heuristic into the evolutionary process. This approach uses a complex layered stowage plans to represent the solutions, where each layer is a non-overlapping cuboidal region aligned with the container's faces.

The height and width of each layer matches the container's dimensions, and the depth is the size of the box of the layer.

A nice feature of the method is the use of a subordinate heuristic, called the *basic procedure*, to generate feasible stowage layers. This heuristic is used both during the initialization of the population and in the later stages of evolution to complete partial solutions produced by crossover and mutation. Each evolutionary operator is designed to respect problem constraints like box orientation, stacking rules, stability and overall balance so the algorithm always generates feasible and good loading plans.

We chose a hybrid genetic algorithm for the Condor Logistics case because it has been shown to be effective for solving complex combinatorial optimisation problems. On one hand well designed hybrids have been key to the success of genetic approaches in logistics and scheduling tasks [Reeves, 1993, Gehring and Schütz, 1994, Kopfer et al., 1994]. On the other hand genetic algorithms provide a framework in which problem specific constraints can be integrated into the evolutionary search process [Reeves, 1995]. This makes the hybrid genetic approach particularly suitable for the spatial and operational constraints of 3D bin packing in logistics.

This hybrid shows how genetic algorithms can be adapted to a specific domain through tailored representations and operators for solving industrial CLP variants.

### 3.3.1 Solution Concept

The procedural approach can be summarised as follows:

- A hybrid genetic algorithm (GA) is used to generate layer stowage configurations. Each solution is a stack of multiple vertical, cuboid, non-overlapping layers that run along the front and back of the container. These layers are continuous with no gaps and each layer contains one or more boxes that do not extend beyond the boundaries of their layer.
- The height and width of each layer is the same as the container, and the depth of the layer is the depth of the box that defines the layer size. See figures 3.2, 3.3 for an example of this layered structure.
- The creation of stowage configurations relies on an embedded auxiliary procedure, referred to as the *basic heuristic*:
    1. First the basic heuristic is used to generate fully packed stowage plans which are the initial population for the GA.
    2. For subsequent generations problem specific operators like crossover and mutation are used. These operators preserve some of the layers from parent solutions when creating offspring. Offspring may be incomplete so the basic heuristic is used again to fill in the missing layers.
    3. Variations in the crossover and mutation mechanisms give different ways of completing partial solutions. So three versions of the basic heuristic are used throughout the process. Every layer added during the process is created using one of these heuristic variants.

- Both the basic heuristic and genetic operators treat the order of layers in the stowage plan as initially irrelevant. Internally, the solution is represented by a data structure that does not fix layer sequence. The final arrangement of layers within the container is established only at the end.
- The stowing approach uses the following problem representation:

  1. **Container**: $x_c$, $y_c$, $z_c$ for depth, width, and height of the container respectively, and $v_c$ for total volume.
  2. **Boxes**: The complete list of boxes is stored in a vector BList$(b)$ for $b = 1, \ldots, n_b$ where $n_b$ is the total number of boxes. Each box is defined by its 3 dimensions and volume. BList is sorted in descending order by box volume. BAll is the entire box set.
  3. **Rotation variants**: Each box can be oriented in up to 6 ways, indexed from 1 to 6.
  4. **Box placement**: A single placement is described by a structure $pl$ containing:
     - the box identifier $b$,
     - the coordinates $(o_x, o_y, o_z)$ of the reference corner (left-bottom-rear corner),
     - and the rotation variant $rv$.

     Coordinates $o_y$ and $o_z$ are absolute, while $o_x$ is relative to the rear wall of the corresponding layer.
  5. **Layer**: A filled layer is represented by a structure $l$ including:
     - the layer utilization $v_{\text{util}}$,
     - the layer depth $d$,
     - the number of boxes placed $n_{pl}$,
     - and a vector of box placements Pl$(i)$ for $i = 1, \ldots, n_{pl}$.

     The layer depth $l.d$ corresponds to the $x$-dimension of the defining box, and the utilization $v_{\text{util}}$ is calculated as the ratio of the total volume of boxes placed to the volume of the layer ($l.d \cdot y_c \cdot z_c$). The operator $B(l)$ returns the set of boxes placed within the layer.
  6. **Stowage plan**: This is modeled by a structure $s$ that includes:
     - the total packed volume $v$,
     - the number of layers generated $n_l$,
     - the set of layers $L$,
     - the remaining free container depth $x_{c,\text{free}}$,
     - and the set of boxes still unstowed $B_{\text{free}}$.

     Layers within the stowage plan are stored as an unordered set. If the plan is not complete, the remaining free depth and unstowed boxes define the residual problem.

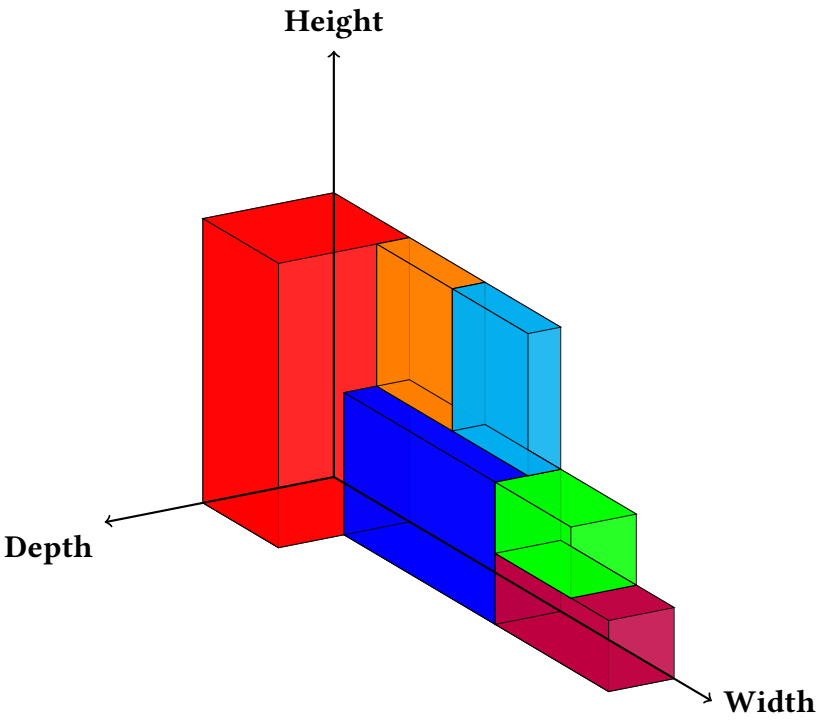



Figure 3.2: 3D visualization of six packed boxes inside a layer.

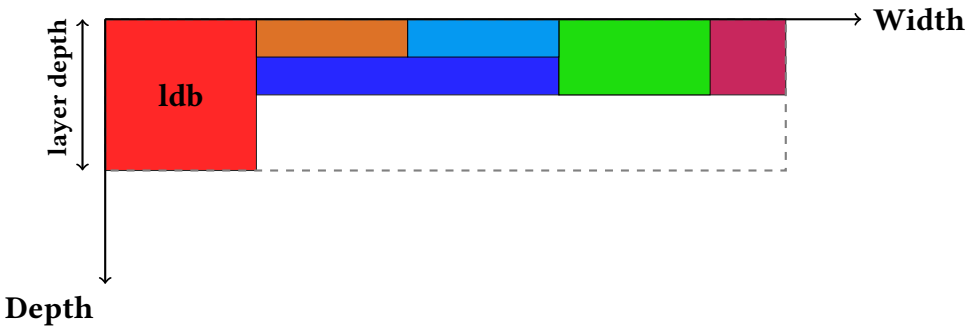



Figure 3.3: Top-down view of the layer

### 3.3.2 Basic heuristic

The stowage plan construction in this method is based on the ideas of Gehring et al. Gehring et al. [1990] but with significant extensions for 3D bin packing. Although the overall framework is inspired by their original heuristic many of the internal rules and operational decisions have been changed to fit the 3D spatial problem.

The packing is done layer by layer, a strategy that simplifies the problem and allows for efficient placement decisions. Each layer is generated in two steps:

1. **Layer Initialization:** First a box is selected – the *layer-determining box* (ldb) – along with one of its feasible rotation variants. The depth of this box is the depth of the entire layer.

2. **Layer Filling:** Once the layer dimensions are fixed the layer is filled using the core procedure of the heuristic, called `fill-layer`. This procedure tries to maximize space usage by placing additional boxes around the ldb and making a tight and stable configuration.

This layer based design not only allows for flexible reordering of box placements but also provides a clear structure for iterative improvement and hybridization with metaheuristics. The details of the `fill-layer` procedure are described in the next section.

#### General algorithm of the basic heuristic

As shown in Algorithm 1, the stowage planning procedure starts with an incomplete configuration $s_{in}$, representing the current state of the container along with the items still to be packed. The core of the method involves incrementally extending this partial plan into a complete one by constructing and evaluating a sequence of packing layers.

The first phase involves identifying all feasible configurations for the next potential layer. Each configuration—referred to as a layer definition—is defined by a box (still available for packing) and one of its rotation variants. These configurations are deemed valid only if the box can be placed in the remaining free space of the container.

For each viable definition of the first new layer, denoted $L_{def1}(i)$ where $i = 1, \ldots, n_{Ldef1}$, a separate complete packing plan is constructed. Starting from $s_{in}$, a new layer is inserted based on the current configuration, and the process continues recursively: at each level, feasible configurations for the next layer are generated, and the most space-efficient layer among them is appended to the plan.

This continues until no more valid configurations are left. At the end, the total volume of packed items is calculated and used to score the solution. All generated plans are stored in $S_{Gen}$ which can be optionally reduced to keep only the top $n_s$ best plans.

#### Heuristic Strategy Variants

Three different variants of the heuristic use this framework for different roles in a genetic algorithm: *start*, *crossover*, and *mutation*.

**Algorithm 1** General algorithm of the basic heuristic

**Input:** Incomplete stowage plan $s_{\text{in}}$, heuristic variant, required number of stowage plans $n_s$
Generate variant-dependent layer-defining list $L_{\text{def1}}$ for the first (additional) layer
Set number of layer definitions $n_{L_{\text{def1}}} \leftarrow |L_{\text{def1}}|$
**if** $n_{L_{\text{def1}}} > 0$ **then**
    Initialise set of generated stowage plans $S_{\text{gen}} \leftarrow \emptyset$
    **for** $i = 1$ **to** $n_{L_{\text{def1}}}$ **do**
        Initialise stowage plan $s \leftarrow s_{\text{in}}$
        Fill layer $l$ for the layer definition $L_{\text{def1}}(i)$
        Update stowage plan and remaining problem:
            $s.n_l \leftarrow s.n_l + 1$
            $s.L \leftarrow s.L \cup \{l\}$
            $s.xc_{\text{free}} \leftarrow s.xc_{\text{free}} - l.d$
            $s.B_{\text{free}} \leftarrow s.B_{\text{free}} \setminus B(l)$
        **repeat**
            Generate variant-dependent layer-defining list $L_{\text{defn}}$ for next layer
            Set number of layer definitions $n_{L_{\text{defn}}} \leftarrow |L_{\text{defn}}|$
            **if** $n_{L_{\text{defn}}} > 0$ **then**
                Initialise set of variants for next layer: $L_{\text{next}} \leftarrow \emptyset$
                **for** $j = 1$ **to** $n_{L_{\text{defn}}}$ **do**
                    Fill layer $l$ for layer definition $L_{\text{defn}}(j)$
                    $L_{\text{next}} \leftarrow L_{\text{next}} \cup \{l\}$
                **end for**
                Select layer $l_{\text{next}}$ from $L_{\text{next}}$ with highest volume utilization
                Update stowage plan and remaining problem:
                    $s.n_l \leftarrow s.n_l + 1$
                    $s.L \leftarrow s.L \cup \{l_{\text{next}}\}$
                    $s.xc_{\text{free}} \leftarrow s.xc_{\text{free}} - l_{\text{next}}.d$
                    $s.B_{\text{free}} \leftarrow s.B_{\text{free}} \setminus B(l_{\text{next}})$
            **end if**
        **until** $n_{L_{\text{defn}}} = 0$
        Determine stowed volume $s.v$
        $S_{\text{gen}} \leftarrow S_{\text{gen}} \cup \{s\}$
    **end for**
    Reduce $S_{\text{gen}}$ to the $n_s$ stowage plans with the highest volume utilization
**else**
    Determine stowed volume $s_{\text{in}}.v$
    $S_{\text{gen}} \leftarrow \{s_{\text{in}}\}$
**end if**
**Output:** $S_{\text{gen}}$

The **start** variant starts from scratch. The input $s_{in}$ is empty in this case, the original problem instance. A number of full packing solutions are generated based on the population size of the genetic algorithm. For the first layer, all valid configurations are considered, for the others only the best one.

The **crossover** variant is applied to a partial solution already containing at least one packed layer. Unlike the start variant, only one complete plan is generated. If multiple feasible extensions are possible, the one with the best space utilization is retained. During layer construction, the list of configurations is trimmed: only the top $q_{ldb1}\%$ are preserved for the first new layer, and the top $q_{ldb2}\%$ for all others.

The **mutation** variant also builds on a partially complete plan but introduces controlled randomness. One layer is added per iteration, selecting a configuration randomly from within the top $q_{ldb3}\%$ of the available sorted list. This introduces diversity into the genetic search without discarding quality completely.

### Layer Configuration Prioritization

All lists are sorted by packing density. The first criterion is the volume of the defining box; larger volumes are preferred. If volumes are equal, then the box with the greater x-dimension is preferred, and in case of a tie, the taller box (z-dimension) is preferred.

### Parameter Roles

The trimming thresholds $q_{ldb1}$ and $q_{ldb2}$ control the trade-off between solution quality and computational effort in the crossover. $q_{ldb3}$ is the selection range for the mutation variant, so we have some randomness but still focus on good packing configurations.

**Filling layers**

**Algorithm 2** Layer Construction Procedure

**Input:** Layer-determining box `ldb`, rotation variant `rv_ldb`, layer depth `d`, set of free boxes `B_free`

**Output:** Layer `l`

Initialize layer $l$: $l.d \leftarrow d$, $l.npl \leftarrow 0$
Initialize residual space stack `SStack`
    Insert layer cuboid as initial residual space
    Place `ldb` with `rv_ldb` into this space
Update set of unstowed boxes: $B\_free \leftarrow B\_free \setminus \{\texttt{ldb}\}$
**while** `SStack` not empty **do**
    Set `scurr` ← top element from `SStack`; remove `scurr`
    **if** `scurr` is still free **then**
        **if** at least one box fits in `scurr` **then**
            Determine placements:
                Placement for `scurr`: box `bcurr`, rotation `rv_curr`
                Optionally, placement for daughter space: box `bsucc`, rotation `rv_succ`
            Place `bcurr`, `rv_curr` in `scurr`
            Update $B\_free \leftarrow B\_free \setminus \{\texttt{bcurr}\}$
            **if** daughter space placement determined **then**
                Update $B\_free \leftarrow B\_free \setminus \{\texttt{bsucc}\}$
            **end if**
        **else**
            Extend residual sister spaces to `scurr`
        **end if**
    **end if**
    **if** `scurr` is now filled **then**
        Update layer $l$:
            $l.npl \leftarrow l.npl + 1$
            Transfer placement from `scurr` to vector $l.Pl$
        Generate three residual daughter spaces for `scurr`
        **if** placement for daughter space determined **then**
            Place `bsucc`, `rv_succ` in daughter space
        **end if**
        Insert daughter spaces into `SStack`
    **end if**
**end while**
Determine layer utilization $l.vutil$
**return** layer $l$

The process of filling a defined layer is conducted through a dedicated procedure (see Alg. 2), which begins by placing a selected box—known as the layer-determining box—within the layer's cuboid volume using a predefined orientation. Additional boxes may subsequently be inserted to complete the layer.

This layer is governed by the concept of residual spaces—unused rectangular regions inside the layer volume. At each step a residual space is selected and a box is placed at its back-left-bottom corner—called the reference corner. Once a box is placed it divides the residual space into three daughter regions (in front, beside and above the box) which are added to a processing stack for future consideration.

To pack efficiently residual spaces are managed using a stack data structure. This last-in, first-out order means newer (and usually smaller) spaces are at the top of the stack so more compact boxes are more likely to fit. Each residual space is defined by its size and the coordinates of its reference corner. When a box is placed the space also stores the box id and its rotation.

During residual space processing four scenarios can occur:

1. If the space is empty and can hold at least one box, two placements are often determined at the same time: one for the current space and one for a future daughter space. Both boxes are removed from the pool but only the current placement is committed to the layer.

2. Sometimes only one box is assigned to the space and all resulting daughter spaces are added to the stack for further evaluation.

3. Sometimes a residual space is already filled. This happens when it was preloaded as a daughter space or is the initial layer-defining placement. Here the recorded placement is added to the layer and the corresponding daughter spaces are inserted as empty.

4. When a residual space can't be matched with any available box an attempt is made to convert part of its area into one or more residual sister spaces—alternative free spaces that originate alongside it. In this case no daughter spaces are generated.

Determining where candidate boxes go, generating new spaces and extending others—that's where the real work happens. And that work needs a clearer explanation of how it's done.

The placement phase is where you decide which boxes go into which spaces. You can put up to two boxes in a residual space, depending on how they fit together and what your criteria are. That involves a series of decisions, made one after the other:

First, you identify the boxes that will actually fit in the space (and there might be just one of those). Next, you choose a suitable space for the second box. Then you decide how each box should be oriented (which way it faces). Finally, you put those boxes in their places.

That decision-making process happens in steps, following a set of rules (R1 through R4). One of the key constraints when you're placing two boxes is avoiding an unstable stack-up. Where only one box is selected, its orientation has to follow rule (R3).

**(R1) Box Pair Selection:** Here we consider all possible box pairs (including "singletons") . Validity is defined as both boxes fitting entirely within the remaining space when placed side by side, front to back, or stacked vertically. From the candidate set, the pair (or single box) with the largest total volume is chosen.

**(R2) Positioning Priority for the Second Box:** Once a pair is chosen, the placement orientation of the second box—either beside, in front of, or on top of the first—is guided by the residual space dimensions. The smallest dimension among the three (x, y, or z) determines priority. For instance, if the x-dimension is smallest, the front placement (along the y-axis) is prioritized. The corresponding daughter space type is selected accordingly. Refer to Fig. 3.4 for a visual illustration, where the green box is the current one and the gray box is the successor.

**(R3) Rotation Variants:** Box orientation is determined by one of two methods, applied in rotation by the genetic algorithm:

- *(R3.1)*: Rotate each box to minimize its x-dimension and, secondarily, its z-dimension. The larger box gets priority, then the smaller one.
- *(R3.2)*: Choose the rotation combination for which the sum of z-dimensions is highest. If there is a tie, then consider the sum of x-dimensions.

For single box placement, either rule can be applied to get the best orientation.

**(R4) Box placement:** The placement of the chosen boxes depends on the variant chosen by R2. If beside or in front variant was chosen, the box with the taller height (z-dimension) is placed in the reference corner of the remaining space, the second box is placed next to it. Otherwise, the box with the larger base area is at the bottom, the second box is placed on top of it.

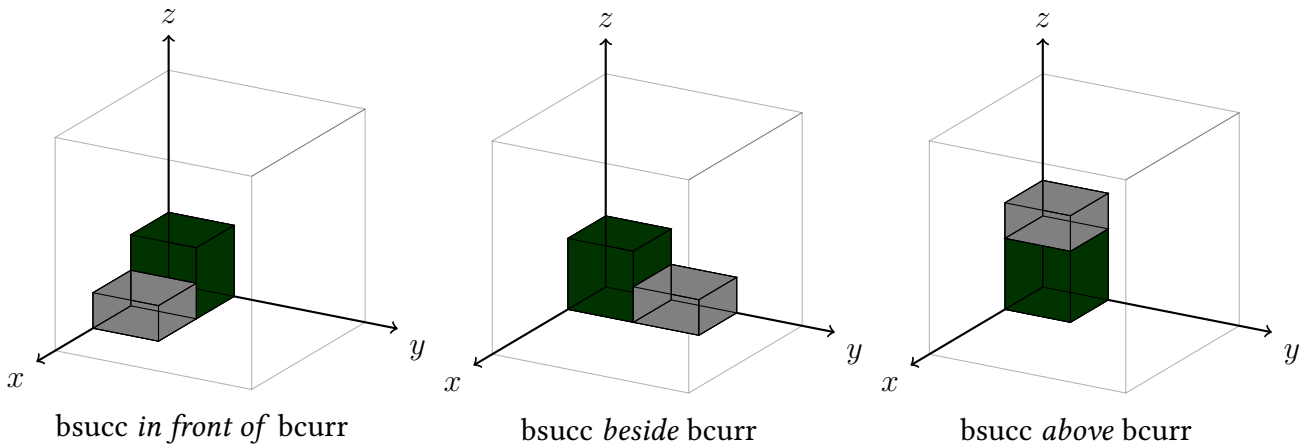



Figure 3.4: Positioning variants for 2 boxes in a residual space

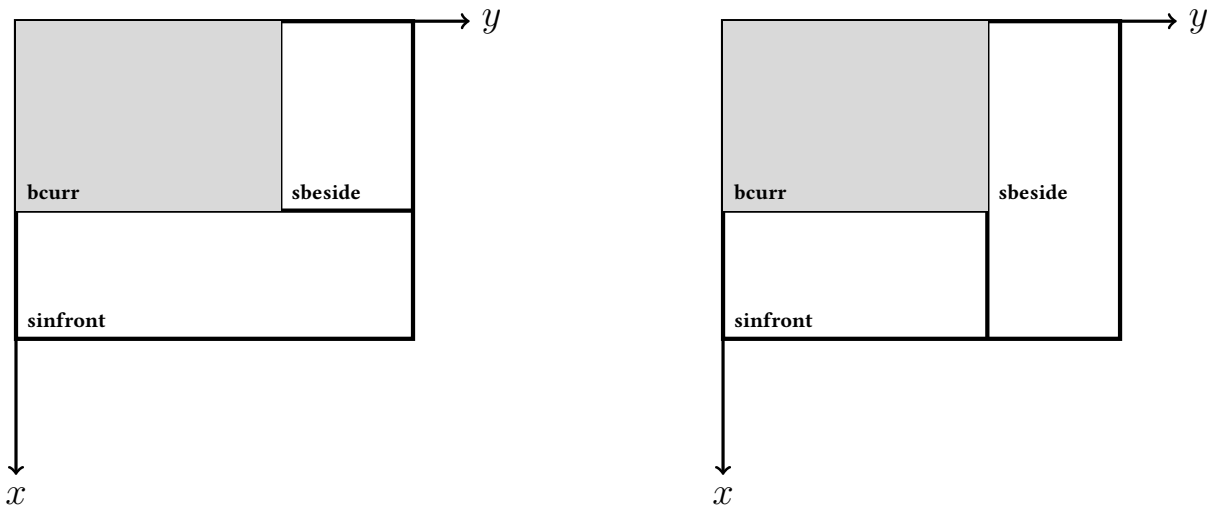
Once a box $b_{\text{curr}}$ has been placed in a residual space $s_{\text{curr}}$, three new subregions—termed daughter residual spaces—are created to represent the remaining usable volume. These are denoted by $s_{\text{infront}}$, $s_{\text{beside}}$, and $s_{\text{above}}$, and are generated using the following rules:

The volume directly above $b_{\text{curr}}$, stretching vertically up to the ceiling of $s_{\text{curr}}$, defines the residual space $s_{\text{above}}$.

When dividing the unoccupied L-shaped area at the base of $s_{\text{curr}}$ (after placing $b_{\text{curr}}$), you split it into two rectangular regions. That division can go one of two ways- in front of large or beside large- depending on which area gets the larger base area (as shown in Fig. 3.5).

At first, the strategy that maximizes the base area of the larger resulting space is chosen. That increases the chances that at least one of those spaces can fit future placements.

However, if the chosen strategy would have the next box, $b_{\text{succ}}$, spilling over into the adjacent space, we adjust course. If you choose to place $b_{\text{succ}}$ in the space beside large, but it would overlap into the space in front, you switch to the in front of large strategy. The same correction is made if $b_{\text{succ}}$ is meant for the space in front and would spill over into the space beside. That adjustment is symmetrical in both directions.




**1. *In front of*** large **2. *beside*** large

Figure 3.5: Subspaces division variants

When a space is no longer able to hold any more boxes, a recovery mechanism is applied to salvage usable areas. Specifically, well defined rectangular parts of the unusable space can be reassigned to its sibling residual spaces that have not been processed yet. This helps to minimize wasted volume in the container.

This space reallocation is only applied between the two residual daughter-spaces $s_{\text{infront}}$ and $s_{\text{beside}}$, both of which come from the same parent residual space $s_{\text{curr}}$ that was previously filled by a box $b_{\text{curr}}$.

Consider the case where the subdivision strategy *in front of large* was initially employed for $s_{\text{curr}}$, and $s_{\text{infront}}$ has subsequently been identified as unfillable. If the corresponding sister space $s_{\text{beside}}$ still remains on the residual space stack, an adaptive strategy is triggered. The system switches to the alternative subdivision variant *beside large* for $s_{\text{curr}}$, thereby reallocating a previously inaccessible cuboidal region—located diagonally opposite to $b_{\text{curr}}$—from $s_{\text{infront}}$ to $s_{\text{beside}}$ (see Fig. 3.6).

A similar adjustment is valid in the reverse direction: if $s_{\text{beside}}$ becomes unusable and $s_{\text{infront}}$ remains active, a mirrored transfer of space can be performed to restore usability.

When you fill a box $s_{\text{curr}}$ with a box $b_{\text{curr}}$, the associated daughter-spaces are usually pushed onto

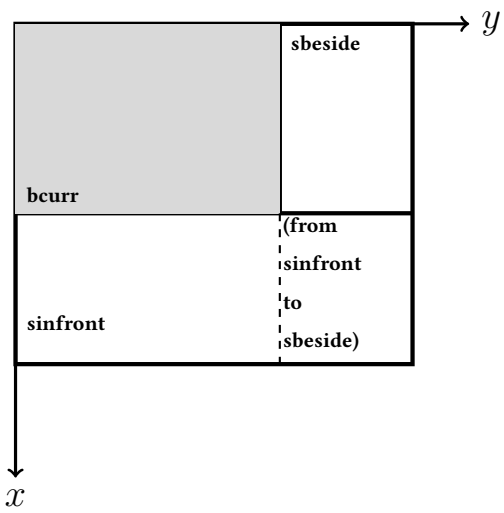



Figure 3.6: Residual spaces extension

the stack for later processing. But under certain geometric and state conditions a new residual space can be merged with an existing one in the stack. This increases the size of the merged residual space without expanding the stack itself.

A new space $s_{\text{new}}$ and an existing stack space $s_{\text{old}}$ can only be merged if neither has a box placement yet.

Furthermore the two spaces must be horizontal and same size to form a single cuboid in the current layer.

If a new residual space can't be merged it's added to the stack in a strict order. The residual space $s_{\text{above}}$—the space above the placed box—is always added first. If the subdivision variant *in front of large* was used during the residual space generation, the remaining daughter-spaces are added with $s_{\text{beside}}$ before $s_{\text{infront}}$. If *beside large* was chosen, the order is reversed.

This ordering ensures that potential donor spaces for future merging or space extension are processed before their possible recipient spaces, so packing is more efficient and less fragmented.

### 3.3.3 Genetic Algorithm

Now that we have the representation scheme for the Genetic Algorithm (GA), we can move on to the algorithmic procedure and operational logic. A flowchart of the process is shown in Algorithm 3.

The population size $n_{\text{pop}}$ remains the same throughout the process. The GA strategy consists of generational replacement where each generation is completely replaced by a new one except for some elite individuals that are preserved through reproduction.

New solutions in the next generation are created using two operators: *crossover* and *mutation*. Mutation is done in two ways: standard mutation and *merger mutation*. And the best solution(s) from the current population, usually $n_{\text{rep}} \approx 1$ are copied directly into the next generation to preserve solution quality.

The generation process goes back and forth between crossover and standard mutation based on their respective probabilities $p_{\text{cross}}$ and $p_{\text{mut}}$ where $p_{\text{cross}} + p_{\text{mut}} = 1$.Once full, $n_{\text{merge}}$ more individuals

are added using merger mutation to add more diversity.

After that the full set of new candidates is pruned down to the top $n_{\text{pop}}$ solutions according to their fitness values. To maintain diversity and avoid redundancy, duplicate solutions are disallowed in all generations, including the initial one. Any duplicate found is excluded and not considered as part of the population.

When the evolutionary loop concludes, the best-performing stowage plan undergoes a final refinement: determining the sequence in which layers should be arranged within the container. If no balance constraint is present, this sequence may be arbitrary. However, when a balance constraint must be respected, the layer ordering must follow the specific method introduced later.

When generating new individuals, the algorithm must balance two important goals: favouring high-performing solutions (based on objective function values) and maintaining genetic diversity to avoid premature convergence. To address both issues, the selection mechanism combines two complementary strategies. The first is **rank-based selection with linear normalization**, which is well known for providing the right selective pressure across a sorted population Whitley [1989], Reeves [1993], Falkenauer [1998]. The second is **uniform random selection**, which keeps the selection probability the same for all individuals regardless of their performance to ensure genetic variation.

The ranking process works as follows. Let $s_i$, for $i = 1, \ldots, n_{\text{pop}}$, represent the individuals in the current population, sorted in descending order based on their objective values (i.e., total stowed volume). Each solution is then assigned a fitness score $f(s_i)$ according to the formula:

$$f(s_i) = f_{\max} - (i-1) \cdot d \quad \text{for } i = 1, \ldots, n_{\text{pop}},$$

where $f_{\max} = \frac{2}{n_{\text{pop}}+1}$ and $d = \frac{2}{n_{\text{pop}}(n_{\text{pop}}+1)}$. These fitness values $f(s_i)$ are then used as probabilities for selection.

For parent selection we use the following strategy. For mutation selection we use rank based probabilities. For crossover selection the first parent is selected based on fitness ranked probabilities and the second parent is selected uniformly at random from the entire population. Also a given parent pair is not allowed to crossover more than once per generation.

This hybrid strategy allows good solutions to propagate through the population while injecting enough randomness to explore new areas of the solution space.

**Algorithm 3** Hybrid Genetic Algorithm for 3D Bin Packing

**Input:** Container data, box data, GA parameters
Generate npop initial stowage plans using the basic heuristic (start variant)
Evaluate the initial population and set it as the current generation
**while** termination criterion is not met **do**
    Copy the best `nrep` individuals to the next generation
    **while** next generation size $<$ npop **do**
        Select operator $op \in \{\text{crossover}, \text{mutation}\}$ according to probabilities `pcross` and `pmut`
        **if** $op$ is crossover **then**
            Select parent plans $p_1$, $p_2$ from the current generation
            Create offspring $o$ by applying crossover to $p_1$ and $p_2$
        **else**
            Select a parent plan $p$ from the current generation
            Create offspring $o$ by applying standard mutation to $p$
        **end if**
        Add offspring $o$ to the next generation
    **end while**
    **for** $i = 1$ to `nmerge` **do**
        Select a plan $p$ from the current generation
        Generate an offspring $o$ by applying merger mutation to $p$
        Add $o$ to the next generation
    **end for**
    Evaluate and reduce the next generation to form the new current generation
**end while**
Determine layer arrangement for the best solution $s_{\text{opt}}$
**Output:** $s_{\text{opt}}$

### Crossover Operator

The crossover mechanism in this genetic algorithm works in two steps to combine parent traits while preserving solution feasibility. The overall logic of the process is shown in Algorithm 4:

**Phase 1: Direct Layer Inheritance** The offspring inherits layers directly from both parents. But this is subject to feasibility checks to avoid conflicts and redundancy. Specifically a layer from a parent solution is eligible to be included in the offspring only if:

1. **Box availability constraint:** Every box in the candidate layer must be in the offspring's set of free boxes ($o.B_{free}$). This ensures no box is used in multiple layers in the offspring.

2. **Spatial constraint:** The geometric dimensions of the layer must not exceed the remaining

container depth in the offspring ($o.xc_{free}$).

If both conditions are met the candidate layer is valid, then we choose the layer with the highest space usage from the valid layers in $L_{next}$ (layers that could be transferred), and add it to the offspring's storage plan. We repeat this until no more valid layers are left.

**Phase 2: Completion via Heuristic Extension**

After the direct inheritance step the offspring is usually not full. To complete the storage plan the basic heuristic—adapted for use in the crossover context—is applied. This heuristic fills the unstowed space in layers from the remaining free boxes to extend the partial solution into a full stowing configuration.

**Redundancy Check**

To keep diversity in the evolving population a validation step is added. If the resulting offspring is an exact copy of one of the parents the crossover operation is considered useless and discarded. In this case a new parent pair is chosen and the process is restarted to ensure meaningful genetic variation is introduced.

**Algorithm 4** Crossover Operator for Stowage Plan Generation

**Input:** Parent plans $p_1$, $p_2$
Initialize descendant $o$:
    $o.n_L \leftarrow 0$
    $o.L \leftarrow \emptyset$
    $o.xc_{\text{free}} \leftarrow xc$
    $o.B_{\text{free}} \leftarrow B_{\text{All}}$
**repeat**
    Identify transferable layers:
        $L_{\text{next}} \leftarrow \{l \mid l \in (p_1.L \cup p_2.L),\ l.d < o.xc_{\text{free}},\ B(l) \subseteq o.B_{\text{free}}\}$
    **if** $L_{\text{next}} \neq \emptyset$ **then**
        Select $l_{\text{next}} \in L_{\text{next}}$ with the highest volume utilization
        Update descendant:
            $o.n_L \leftarrow o.n_L + 1$
            $o.L \leftarrow o.L \cup \{l_{\text{next}}\}$
            $o.xc_{\text{free}} \leftarrow o.xc_{\text{free}} - l_{\text{next}}.d$
            $o.B_{\text{free}} \leftarrow o.B_{\text{free}} \setminus B(l_{\text{next}})$
    **end if**
**until** $L_{\text{next}} = \emptyset$
**if** offspring $o$ is identical to one of the parents **then**
    $o.L \leftarrow p_1.L$ or $p_2.L$
    `CrossoverOk` $\leftarrow$ `False`
**else**
    Apply the basic heuristic (crossover variant) to refine $o$
    `CrossoverOk` $\leftarrow$ `True`
**end if**
**Output:** Descendant $o$, `CrossoverOk`

### Mutation Operator

The mutation process (see Algorithm 5), satisfies genetic diversity by modifying parent solutions through two phases. Two mutation variants are considered: the *standard mutation* and the *merger mutation*, each with different goals and strategies.

#### Standard Mutation

In this approach the new solution is created in two phases:

**Phase 1: Partial Layer Inheritance**

The first phase consists of copying a subset of the best layers (those with the highest space usage) from the parent solution into the offspring. The number of layers to copy is chosen randomly subject

to the following constraints:

- At least one layer must be copied.
- At most $\lfloor p.n_L/2 \rfloor$

These layers are copied verbatim and the inherited layers are inherited. The rest are inherited. The inherited layers are copied verbatim and form the base of the mutant.

**Phase 2: Heuristic-Based Completion**

After the partial copy the incomplete offspring is extended into a full stowage plan using the basic construction heuristic. To introduce randomness and variation different variants of the layer generation process are chosen randomly during this phase. This is the only step where actual mutation—i.e. random change—happens.

Although this ensures diversity, building a stowage plan layer by layer can lead to suboptimal space utilization because of fragmentation. To fix this, another mutation variant is used.

**Merger Mutation**

The goal of the merger mutation is to improve space utilization by reducing the number of layers and increasing their depth. This is how it works: all layers from the parent solution, except two, are copied into the offspring unchanged; one of the omitted layers is chosen randomly from the total set of parent layers; the second omitted layer is also chosen randomly, but only from the 50% of layers with the lowest space utilization.

After the copy, the basic heuristic is invoked to generate a single additional layer, which is fitted into the remaining free container depth. So the offspring generated by merger mutation has one less layer than the parent solution.

This fixes the fragmentation issue of standard mutation and encourages denser packing by generating deeper layers.

**Algorithm 5** Mutation Operator for Stowage Plan Generation

**Input:** Parent plan $p$
Initialize descendant $o$:
    $o.n_L \leftarrow 0$
    $o.L \leftarrow \emptyset$
    $o.xc_{\text{free}} \leftarrow xc$
    $o.B_{\text{free}} \leftarrow B_{\text{All}}$
Randomly choose number of layers to transfer: $n_{lp} \in [1, \lfloor p.n_L/2 \rfloor]$
Initialize transferable layers: $L_{\text{next}} \leftarrow p.L$
**for** $i \leftarrow 1$ to $n_{lp}$ **do**
    Select $l_{\text{next}} \in L_{\text{next}}$ with the highest volume utilization
    Update descendant $o$:
        $o.n_L \leftarrow o.n_L + 1$
        $o.L \leftarrow o.L \cup \{l_{\text{next}}\}$
        $o.xc_{\text{free}} \leftarrow o.xc_{\text{free}} - l_{\text{next}}.d$
        $o.B_{\text{free}} \leftarrow o.B_{\text{free}} \setminus B(l_{\text{next}})$
    Update $L_{\text{next}} \leftarrow L_{\text{next}} \setminus \{l_{\text{next}}\}$
**end for**
Apply basic heuristic (mutation variant) to refine $o$
**Output:** Descendant $o$

### 3.3.4 Constraints Handling

To make the hybrid genetic algorithm practical in real world scenarios we need to add some additional modifications to respect multiple practical constraints. This section explains how the algorithm enforces these constraints during the packing process.

**Orientation Constraints:** Each box can be placed in the container using only a limited number of orientations, from 1 to 6. These are all the possible permutations of the box's 3 dimensions: depth (D), width (W) and height (H). The 6 standard orientations are: `dwh` (original orientation), `wdh`, `dhw`, `hdw`, `whd`, and `hwd`. Before placing a box in a layer the algorithm checks if the selected orientation is in the allowed set for that box. The check is done during the layer construction to make sure invalid rotations are not taken into consideration.

**Stacking Constraints:** Boxes can only be stacked under certain conditions to be safe and sound: some boxes can't be stacked on top of others—for those, the algorithm sets their z-coordinate to 0, so they must sit on the container floor. Some boxes can't have anything stacked on top of them—when one of those is placed, the algorithm won't generate space above it, so no box can be placed on top of it. Every box has a load bearing strength (LBS), which is the maximum weight that can be stacked over it safely (without damaging it). When a box is placed, the algorithm subtracts its weight from the LBS of the box that it was placed over. The placement is not taken into consideration if the

LBS value of that box turns negative. With these rules established, stacking constraints are always respected by the algorithm.

**Stability Constraints:** Load stability is key in 3D packing. A box is stable if its entire base is fully supported—either by the container floor or by one or more boxes below it. The algorithm ensures this by using a placement heuristic that doesn't allow lateral overhangs. So every box is only placed when it can sit completely on a solid surface, so the whole loading process is stable.

## 3.4 Multi-Container Loading Problem

The Multi-Container Loading Problem is another version of the CLP where the goal is to fill multiple identical, or different conatiners with fixed dimensions instead of a single one. Solving this problem efficiently is critical in logistics and distribution where multiple containers need to be filled for a shipment.

To address this challenge, we adopt a greedy strategy that treats the single-container Genetic Algorithm (GA) as a reusable packing engine. The main idea is to iteratively allocate unpacked boxes to new containers, applying the GA independently to each one. This modular approach enables us to take advantage of the strengths of the single-container solution while scaling it to multi-container scenarios with minimal architectural changes. The following subsection details the greedy algorithm used for container-by-container loading.

### 3.4.1 Greedy Genetic Algorithm for Multi-Container Loading

**Pseudocode**

**Algorithm 6** Greedy Multi-Container Loading

$B \leftarrow$ list of all boxes
$C \leftarrow$ empty list of containers
**while** $B \neq \emptyset$ **and** $|C| < max\_containers$ **do**
    $c \leftarrow$ new empty container
    $GA \leftarrow$ initialize Genetic Algorithm with container $c$ and boxes $B$
    $best\_plan \leftarrow$ run $GA$ for $N$ generations
    $packed\_boxes \leftarrow$ boxes placed in $best\_plan$
    $B \leftarrow B \setminus packed\_boxes$
    Add $c$ and $best\_plan$ to $C$
**end while**

**Explanation**

At each iteration a new container is introduced and passed along with the current set of unpacked boxes to a Genetic Algorithm instance. The GA evolves packing solutions to maximise the volume

utilisation of the container. Once the best plan is obtained all successfully packed boxes are removed from the global list. This is repeated until no unpacked boxes remain or the container budget is reached.

This approach preserves the original GA logic from the single-container problem, no algorithm changes are needed and it allows for modularity in both design and implementation.

It's simple, with a greedy loop that's easy to understand and fast as you resolve containers one by one. It's also modular so you can use it directly with the multi-container solver. But it's greedy, where early container assignments can lead to poor space utilisation in later containers due to short-sighted decisions. This means suboptimal global packing because containers are filled sequentially without re-evaluating previous decisions. And it doesn't provide inter-container coordination because it doesn't consider reallocation or swapping of boxes between containers once they are placed which can further limit overall packing efficiency.

# 4

# Implementation and Experimental Validation of the Hybrid Genetic Algorithm for 3D Bin Packing

This chapter presents the complete implementation setup and experimental validation of the proposed Hybrid Genetic Algorithm (HGA) for solving the 3D Bin Packing Problem, including its multi-container extension. The first part details the architecture, object design, and solver mechanisms that were built from scratch in Python. The second part evaluates the performance of this solver through simulations based on real-world logistics data provided by Condor Logistics. By combining implementation insights with experimental results, this chapter offers a comprehensive understanding of the system's practical capabilities and design rationale.

## 4.1 Implementation of the Algorithms

### Programming Environment and Tools

Python 3 is used because it's readable, has broad library support and is easy to prototype. No external optimization frameworks are used, all core components of the solver are implemented from scratch. The most relevant libraries are:

- `copy` – for object duplication in the evolutionary process
- `random`– for parent selection and mutation
- `numpy`– for weighted random selection of parents
- `matplotlib` – only for optional 3D debugging
- `time` – for timing and duration limit

This gives full control over the solver's design and logic which is important for both experimentation and customization.

## Core Object Structure

The algorithm is based on a set of Python classes that model the entities involved in the packing process. These are `box`, which is each item to be packed, including dimensions, weight and orientation constraints; `container`, which is the physical dimensions of the packing space; `rv` and `box_placing`, which is a box's orientation and position in the container; `layer`, which is one packing layer, with all placements and volume metrics; and `stowage_plan`, which is a full packing solution, with all layers and metadata for evaluation. These classes are nested and linked to mirror the hierarchy of a solution: from individual boxes to their positions, layers and the full container configuration.

## Solver Entry Point

The HGA is called through the function `solve_loading_problem` which is the main entry point for the solver. It takes in data about the container, box set and genetic parameters. It converts the data, initializes the GA, runs the evolution and returns the best solution found along with visual data to be interpreted.

## Genetic Algorithm Management

All the evolutionary logic is encapsulated in the `hybrid_GA` class. It creates the population, evaluates the fitness, and the generational loop that applies crossover and mutation. The termination conditions are controlled by a generation cap and a time limit to be adaptable to different runtimes. This class also has internal mappings and state control features like tracking box identities, managing duplicate signatures to avoid duplicate offspring, and recording progress across generations. The goal is to guide the search towards good solutions.

## Multi-Container Solver

In addition to solving the 3D Bin Packing Problem for one container, the system also supports a multi-container version. The function `solve_multi_container` runs multiple independent HGA's, each packing is a subset of the remaining boxes into a new container. It iterates over already packed items and maps them back to their original ids across generations. Each run applies the same evolutionary principles to a smaller and smaller set of unplaced boxes until all boxes are packed or a container limit is reached.

While the internal GA logic remains the same, the multi-container solver adds a higher level logic for the management of box ids across runs, remapping ids and tracking global packing progress. This allows the HGA to handle real world scenarios with multiple loading spaces without changing the core mechanics.

## 4.2 Scenarios Tests and Results

This section presents a series of **simulation-based experiments** to test the **performance and feasibility** of the proposed **hybrid genetic algorithm** in real world logistics scenarios. We use real item and container dimensions from **industrial data** provided by Condor Logistics.

To test the algorithm in different scenarios, two problem types are used. The first is for the **Single Container** problem which tests the algorithm ability to load a single container with a given set of boxes, maximizing **space** and respecting **weight limits** and **box orientation**. The second is the **Multi Container** problem which is more complex as it requires to distribute a larger set of boxes across multiple containers. This scenario tests the algorithm to optimize both **container usage** and **internal packing** in each container, simulating real dispatch operations with **heterogeneous batches**.

Each test case has **quantitative metrics** such as **volume utilization rate**, **number of containers used** (in multi-container scenarios) and **computational time**. The experiments will show the algorithm's **performance and adaptability to different logistics requirements**.

### Configuration

The set up of the GA is defined by its parameters and breaking conditions. For this experiment the population size npop is 50 and nrep = 10 individuals are selected for reproduction in each generation. Crossover is applied with a probability of 0.67 and standard mutation with a probability of 0.33. 10 new solutions are created through merger mutation in each generation.

To control how new layers are generated using the basic heuristic, three parameters—qldb1, qldb2, and qldb3—are defined based on the number of box types in the problem instance. If the number of box types is 20 or less, the values are qldb1 = 100, qldb2 = 30, and qldb3 = 33. If the number of box types is between 21 and 50, the values are qldb1 = 30, qldb2 = 10, and qldb3 = 10. If the number of box types is between 51 and 70, the values are uniformly set to qldb1 = 10, qldb2 = 10, and qldb3 = 10. If the number of box types is 70 or more, the heuristic becomes more selective with qldb1 = 5, qldb2 = 5, and qldb3 = 5.

For the placement rules during solution generation, two rotation rules—(R3.1) and (R3.2)—are used in an alternating manner. (R3.1) is used for the first 50% of the total number of generations and (R3.2) for the remaining 50%. This ensures a balance between exploration and exploitation throughout the evolutionary process.

To prevent the algorithm from running forever, two stopping conditions are used together. Firstly, the algorithm stop when 50 generations are generated from the initial population. Moreover, the computation stops if the total time exceeds 60 seconds. These are checked at the end of each generation. This dual-criteria approach ensures both time efficiency and enough exploration of the solution space.

## Single-Container Configuration and Packing Results

This section presents the results for the one container loading problem with 3 different types of household appliances. The goal is to pack as many as possible in the given container while maximizing the volume and respecting all physical and stacking constraints. The dimensions of the container used in this scenario are 8, 2.5, 2.6 for depth, width, height respectively and the boxes to be packed are heterogeneous where each type has its own dimensions, weight and practical constraints. Table 4.1 shows the input box types specs.

Table 4.1: Box Types and Volume Specifications

| **Box Name** | **Quantity** | **Volume ($m^3$)** | **Total Volume ($m^3$)** |
|---|---|---|---|
| Réfrigérateur CONDOR Double porte – 465 L – Nofrost – INOX | 30.00 | 1.11 | 33.30 |
| Congélateur CONDOR Horizontal CFH-MT26 – 200 Litres – GRIS | 20.00 | 0.46 | 9.20 |
| Machine à laver Hisense top load 20kg / Smart / Inverter | 20.00 | 0.56 | 11.20 |

After running the genetic algorithm-based packing procedure, 44 boxes were packed into the container. All 20 washing machines (shipmentBox3), 20 refrigerators (shipmentBox1) and 4 freezers (shipmentBox2) were packed. But some items couldn't be packed due to space and feasibility constraints, 10 refrigerators and 16 freezers. Table 4.2 shows the volume usage and packing outcome.

Table 4.2: Packing Results Summary

| **Metric** | **Value** |
|---|---|
| Total Container Volume | 52.00 $m^3$ |
| Total Packed Volume | 35.31 $m^3$ |
| Volume Utilization | 67.90% |
| Stowed Boxes | 20 × Washing Machines, 20 × Refrigerators, 4 × Freezers |
| Unstowed Boxes | 10 × Refrigerators, 16 × Freezers |
| Packing Time | 35.437 seconds |

## Multi-Container Packing Results

This section presents the results of packing a heterogeneous set of home appliances into two identical trucks (7-ton Camions). Each truck has a predefined volume of 42.00 $m^3$. The input set consists of large domestic appliances with varying volumes and quantities. The packing was executed sequentially across two containers to optimize volume usage.

### Container Setup

Table 4.3: Container Specifications

| Container Name | Depth (m) | Width (m) | Height (m) |
|---|---|---|---|
| Camion 7T (Container 1) | 7.00 | 2.40 | 2.50 |
| Camion 7T (Container 2) | 7.00 | 2.40 | 2.50 |

### Boxes to Pack

Table 4.4: Boxes to Pack: Key Specifications

| Box Name | Quantity | Volume ($m^3$) | Total Volume ($m^3$) |
|---|---|---|---|
| Réfrigérateur CONDOR Double porte – 465 L – Nofrost – INOX | 30.00 | 1.11 | 33.30 |
| Congélateur Condor vertical – 391 L – Inverter – No Frost | 20.00 | 0.80 | 16.00 |
| Machine à Laver NARDI Frontale – 8Kg | 10.00 | 0.38 | 3.80 |
| Lave Vaisselle ROCCA NARDI – 14 Couverts | 10.00 | 0.38 | 3.80 |
| Cocotte-minute SEB Clipsominut 4.5L | 10.00 | 0.03 | 0.30 |

### Container 1 Results

Table 4.5: Packing Summary for Container 1

| Metric | Value |
|---|---|
| Total Packed Volume | 30.27 / 42.00 $m^3$ |
| Overall Volume Utilization | 72.07% |
| Stowed Boxes | 27 × Réfrigérateur CONDOR, 10 × Cocotte-minute SEB |
| Unstowed Boxes | 20 × Congélateur Condor, 10 × Machine à Laver NARDI, 10 × Lave Vaisselle ROCCA |
| Packing Time | 54.192 seconds |

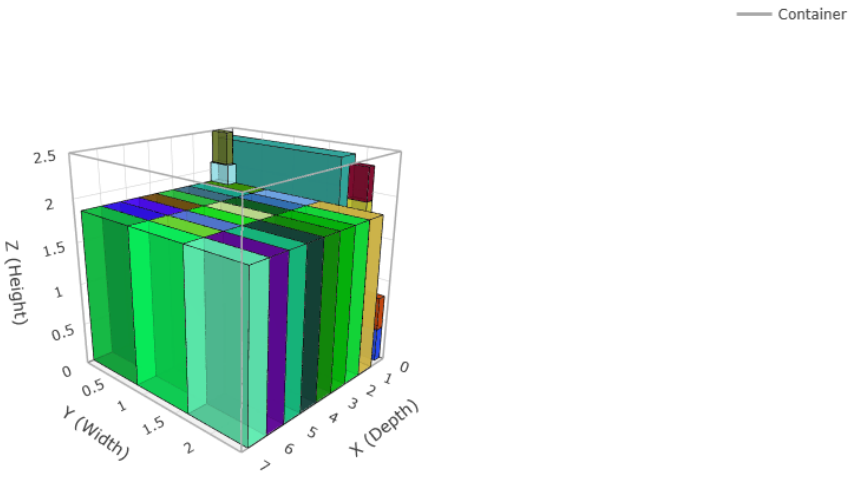



Figure 4.1: Stowage Plan – Container 1

### Container 2 Results

Table 4.6: Packing Summary for Container 2

| Metric | Value |
|---|---|
| Total Packed Volume | 23.53 / 42.00 $m^3$ |
| Overall Volume Utilization | 56.02% |
| Stowed Boxes | 20 × Congélateur Condor, 10 × Machine à Laver NARDI, 10 × Lave Vaisselle ROCCA |
| Unstowed Boxes | None |
| Packing Time | 2.276 seconds |

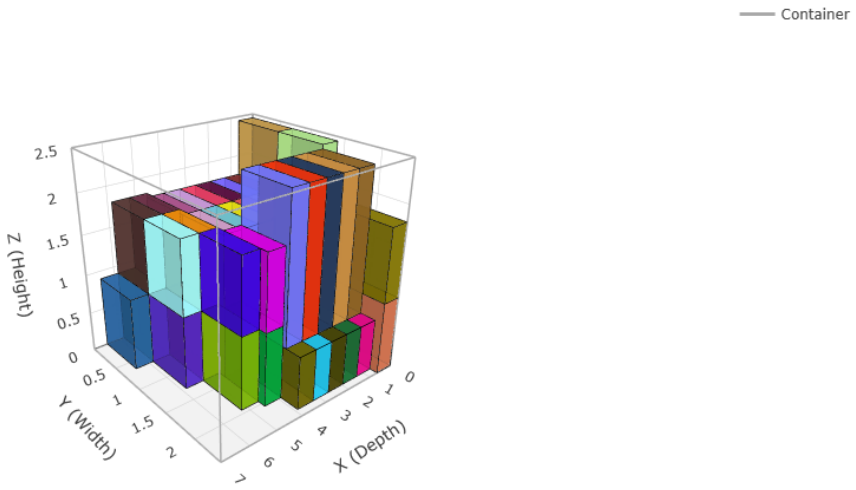



Figure 4.2: Stowage Plan – Container 2

# Conclusion and Future Work

## Conclusion

This paper addressed the operational challenge in the logistics department of Condor: to load containers with different cargo types efficiently and optimized. We introduced a hybrid genetic algorithm (HGA) for the 3D Bin Packing Problem (3D-BPP), based on the Bortfeldt and Gehring approach, adapted to the constraints and operational context of Condor Logistics. The implementation included domain specific heuristics and residual space management to improve space usage in containers. The genetic operators (mutation and crossover) were designed with constraint aware logic to maintain feasibility across generations and to speed up convergence to good solutions. Practical testing showed that the HGA outperformed manual planning in terms of space usage and execution time. The layer based structure of the stowage plans allowed for modular changes, so real time decision making and scenario simulation. The deployment of this algorithm provides a structured, repeatable and scalable framework to reduce manual judgment, risk of human error and overall efficiency. By validating the algorithm with real world data and integrating it into a Python based software prototype, we created a bridge between theory and practice. The tool was not only efficient but also user friendly, so a good basis for future digitalization in the logistics workflow of Condor.

## Future Work

While this research focused on the core challenge of space usage in container loading, there are several directions that can expand the scope and impact of this work:

Future research can go in several directions to enhance the container loading problem. One area is palletization where items are grouped onto pallets before containerization adding complexity through two-tier optimization – item-to-pallet and pallet-to-container which would require custom heuristics. Another direction is dynamic and real-time loading, incorporating real-time data (e.g. traffic, priority changes, cancellations) via APIs and IoT to update stowage plans on-the-fly. The algorithm can also evolve to multi-objective optimization considering trade-offs like loading time, weight balance, stability and delivery order using Pareto-based or NSGA-II algorithms. A human-in-the-loop approach will make the system more flexible so logistics managers can fine tune AI generated plans through an interactive interface. Sustainability can be achieved by fewer trucks

and better fuel efficiency. In cross-docking and just-in-time logistics, packing and scheduling can be combined into a full-stack solution. Machine learning can guide heuristics or operator tuning while reinforcement learning can develop agents that learn optimal strategies through simulations. Finally, extending this to heterogeneous fleets where container sizes vary would require the Vehicle Routing Problem with Loading Constraints (VRPLC).

## Final Thoughts

This thesis lays the groundwork for computational logistics at Condor. It shows how math and computational intelligence can solve real world problems in a practical way. But the full benefit will only be seen when it's embedded in a system of operational processes, digital infrastructure and organisational change. Future developments combining AI, real-time analytics and system integration will make Condor Logistics a data driven and resilient player in a very competitive industry. The journey has just begun – and there's a lot to play for.

# Bibliography


Edward G. Coffman, Michael R. Garey, and David S. Johnson. Approximation algorithms for bin packing: A survey. *Computer Science: Theory and Applications*, 1984.

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

Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, 1990.

Hermann Gehring and Andreas Bortfeldt. A genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, 4:401–418, 1997.

David Pisinger. Heuristics for the container loading problem. *European Journal of Operational Research*, 141:382–392, 2002.

Adriano Moura and José Fernando Oliveira. A hybrid approach to the three-dimensional packing problem. *Annals of Operations Research*, 107:133–152, 2001.

Andreas Bortfeldt and Hermann Gehring. A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 171:1108–1126, 2006.

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

Henning Fanslau and Andreas Bortfeldt. A tree search algorithm for solving the multi-dimensional container loading problem. *OR Spectrum*, 32:711–729, 2010.

Leonardo Junqueira and Reinaldo Morabito. Mixed-integer programming models for the three-dimensional container loading problem. *International Transactions in Operational Research*, 20: 201–222, 2013.

Silvano Martello and David Pisinger. An exact algorithm for the three-dimensional bin packing problem. *INFORMS Journal on Computing*, 29:412–429, 2017.

Elvis Silva, Alysson Costa, and José Fernando Oliveira. The three-dimensional bin packing problem: Mathematical models and heuristics. *European Journal of Operational Research*, 287:37–56, 2020.

Hao Hu, Xiaodong Zhang, Xiaolang Yan, and Longfei Wang. Solving 3d bin packing with reinforcement learning. *IEEE Access*, 9:90348–90356, 2021.

Xinguang Zhao, Ju Long, and Bo Yang. A comprehensive survey on 3d bin packing algorithms. *Artificial Intelligence Review*, 55:4129–4168, 2022.

Yifan Zhang, Libin Qu, and Xiangxiang Zeng. Spatial transformer networks for 3d industrial packing. In *Proceedings of AAAI Conference on Artificial Intelligence*, page 14567–14575, 2023.

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

Colin R Reeves. *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., 1993.

Hermann Gehring and Gerhard Schütz. Zwei genetische algorithmen zur lösung des bandabgleichproblems. *Operations Research Reflexionen aus Theorie und Praxis: Festschrift zum 60. Geburtstag von Hans-Jürgen Zimmermann*, pages 85–128, 1994.

Herbert Kopfer, Giselher Pankratz, and Elmar Erkens. Entwicklung eines hybriden genetischen algorithmus zur tourenplanung. *Operations-Research-Spektrum*, 16:21–31, 1994.

Colin R Reeves. Genetic algorithms and combinatorial optimization. *Applications of modern heuristic methods*, 111:126, 1995.

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

Darrell Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *Proceedings of the third international conference on Genetic algorithms*, 1989.

Emanuel Falkenauer. *Genetic algorithms and grouping problems*. John Wiley & Sons, Inc., 1998.

**Abstract**

This thesis tackles the operational challenge of container loading optimisation at Condor Logistics, where manual packing is inefficient and leads to under-utilisation of space and increased costs. A Hybrid Genetic Algorithm (HGA) is proposed to solve the Three-Dimensional Bin Packing Problem (3D-BPP), adapting the Bortfeldt and Gehring approach to Condor's logistics operations.

The solution combines evolutionary algorithms with domain-specific heuristics:

- Layer-based packing with residual space management
- Constraint-aware crossover and mutation
- Real-world constraints: orientation, stacking, and stability
- Multi-container extension via greedy allocation

Implemented in Python and tested with real industrial data from Condor Logistics, the algorithm showed significant improvements over manual planning. In single-container scenarios, it achieved 67.9% volume utilisation. In multi-container cases, the results reached 72.07% and 56.02% for two 7-ton trucks, respectively. The system reduces human error, ensures repeatable optimisation, and supports scenario simulation for logistics planning.

This research bridges theory and practice, providing a solid foundation for digital transformation in logistics operations. Its modular design allows for future extensions, including dynamic loading, multi-objective optimisation, and integration with IoT systems for real-time logistics decision-making.

**ملخص**

في هذا العمل، قمنا بدراسة وتحليل مشكلة تحميل الحاويات في شركة **كوندور للخدمات اللوجستية**، وهي مشكلة واقعية تتعلق بتوزيع الصناديق داخل الشاحنات بطريقة فعالة تأخذ بعين الاعتبار عدة قيود مثل الحجم، الوزن، الثبات، واتجاه الصناديق. الطرق اليدوية المعتمدة حالياً تُعاني من ضعف في استغلال المساحة وزيادة في التكاليف.

لحل هذه المشكلة، اقترحنا خوارزمية جينية هجينة (Hybrid Genetic Algorithm) مستوحاة من أعمال Bortfeldt and Gehring، حيث قمنا بدمج تقنيات الخوارزمية التطورية مع خوارزميات مجالية خاصة، مع التركيز على تقسيم الطبقات ومعالجة المساحات المتبقية داخل الحاوية.

تم برمجة الخوارزمية باستخدام لغة Python، واختبارها على بيانات حقيقية من الشركة. أظهرت النتائج تحسناً كبيراً في نسبة استغلال الحجم مقارنة بالتحميل اليدوي، حيث وصلت إلى 67.9 % في سيناريو الحاوية الواحدة، و 72.07 % و56.02 % في سيناريوهات شاحنتين من نوع 7 أطنان.

يساهم هذا العمل في دعم التحول الرقمي لعمليات كوندور، ويقدم أساساً متيناً لتطوير نظم ذكية للتخطيط اللوجستي في المستقبل، مع إمكانية إدماج أنظمة الاستشعار وإنترنت الأشياء لتحسين الأداء واتخاذ القرار الأمثل.