Cargo Tetris: A 3D Cargo Container Packing Problem Analysis Using Different Algorithms

Charles Chen[1], Everett Hirano[2], Logan Hosoda[3], Cade Tanaka[4], and Bright Xiang[5]

[1]Department of Mathematics, University of Washington

June 30, 2024

Abstract

SDS, a family-run company, works with China and other countries to ship cargo. Since the business started, the owners have been packing shipping containers based on intuition. Generally, these shipping containers are well-packed due to SDS's managers' years of experience. However, with mathematical modeling, it is possible to make these packing arrangements more space efficient, thereby allowing SDS to pack more cargo per shipping container and raise revenue. Two approaches were explored: a Greedy approach and a layering approach. These models were primed and tested on various conditions, including different cargo types, cargo type packing ratios, and volume constraints. The greedy approach organized items in a high volume to low volume order and then packed items with the most volume first. On the other hand, the layering approach focused on creating discrete stratas/layers of cargo that then could be placed inside the shipping container. The layering approach was able to achieve a high volume occupancy rate while the greedy approach had less success in this regard. However, the greedy algorithm was able to follow the desired cargo distribution more effectively in comparison to the layering approach.

1 Introduction

SDS, an international shipping company based out of China, does international trading with various countries. The company takes requests from foreign consumers, coordinates with partner factories in the production of goods, and mediates the shipping of goods to the customers. The common types of goods that they handle include, but are not limited to, diesel power generators and their corresponding accessory parts for modification and repair, various types of LED lamps and bubbles. During the preliminary packing, all the goods are packed into rectangular containers before being delivered. The most common approach to delivery is via cargo ship, which packs the cargo into shipping containers where the optimal layout is not readily apparent and thus leaves room for different strategies.

1.1 Shipping Container Operations

As it currently stands, the focus of our mathematical model will focus on maximally utilizing the space within a singular shipping container given a list of cargo containers that need to be packed. The reason we are able to make this simplification is due to the fact that SDS are specifying the ratio of cargo types that need to be packed into each shipping container. Due to this operations practice, maximizing shipping container utilization with the specified cargo type ratio for a particular shipping container becomes far more relevant than maximizing the volume utilization across multiple shipping containers.

With this in mind, we formulated a problem statement that encapsulates the purpose of this paper:

Given a list of packages that need to be placed in a shipping container, generate a model that finds a layout that maximizes the volume occupancy within the shipping container such that a desired amount (in our case 90%) of space occupancy is achieved while maintaining a specified volume ratio between various cargo types, safety for fragile items, orientation requirements for specific packages, and total weight of the cargo in a shipping container.

1.2 Problem Objectives

As previously mentioned, the issue with the current approach utilized by SDS is that the packing methods that they employ are highly reliant on the experience and intuition of the managers. Due to the experience of these managers, packing configurations are typically sufficiently optimal and produce satisfactory results. This is less of an issue when the proper personnel are on hand, but becomes a significant detriment when experienced personnel are not present. In order to decouple the company's reliance on experienced personnel for optimal packing, an algorithm capable of producing reasonable packing configurations becomes important.

The objectives of our project are to determine the optimal way to fill a shipping container with 3 different types of cargo. Current challenges faced by the shipping container company include giving precise and replicable instructions to their workers so that they may fill their cargo container in a way that is both efficient and in line with cargo specifications (i.e fragile items, items that need to be oriented a specific way, and weight that is allowed to be above specific items.)

2 Cargo Details/Information

In this section, we will discuss the physical parameters that define and constrain our system.

- The physical shipping container have the following dimensions: 11.8m X 2.34m X 2.69 m
- The shipping container has a total weight limit of 22 metric tons (22,000 kg) and can only be filled to its 90 percent volumetric capacity.[1]
- Cargo must remain upright and can only be rotated in such a way that the top side of the cargo faces upwards.

[1]The volumetric capacity is the total volume of packed cargo we aim to reach. SDS wishes a 90% limit in order to account for cargo that is not perfectly placed, or worker error. This is a soft constraint, so it can be somewhat flexible.

- Fragile items can only be stacked upon themselves, and must otherwise be placed so no other cargo types are stacked above it
- Total shipping percentages should be as close to the desired distribution volume ratios as possible [2]

SDS currently has 3 different kinds of cargo they can utilize to fill this shipping container, with the following dimensions and specifications.

Cargo #	ID	Fragile	Distribution (volume)	Weight/Package (kg/pkg)	Dimensions (mm)
1	YF-3500i	No	40%	43	605x470x500
2	YF-3500i-E	No	50%	48	675x490x510
3	LJ6500A	Yes	10%	7.1	1150x290x230

Table 1. Cargo Names and Specifications

3 Modeling and Packing Strategies

This section will discuss the different approaches for packing a shipping container. Additionally, we will explain what led us to choose our strategy to model.

Initially, we wanted to develop two models: One focused on packing cargo according to the specified cargo ratios (4:5:1), and the other maximizing the volume of cargo packed. Upon further analysis, we found reaching the 22 metric tons weight limit difficult. Filling the container with the highest density cargo never exceeded this limit at 90% space occupancy constraint. Since weight was not an issue, we first focused on developing a naive solutions to this problem and later introducing complexity via exploration of various algorithms.

3.1 Preliminary Approaches

Approach A:
Our first approach involved packing the shipping container in horizontal layers. In this approach, three horizontal layers of cargo 1 were placed into the container, followed by two layers of cargo 2, filling the remaining space with cargo 3. This strategy ensures fragility constraints for cargo 3, so long as it is always the topmost layer. In addition, we are able to somewhat adjust for the various desired cargo ratios by modifying the number of layers of each cargo type we put in according to the target cargo ratio. Total layers can be added and removed to meet the desired 90% capacity constraint.

The primary reason why this approach failed was because of the lack of nuance and fine tuning that this approach is able to make. It is difficult to balance ratios of different cargo types in a way that conforms to the needs of SDS when the packing is done with entire horizontal layers being only a single container type. In order to fix this problem, a packing method would have to have smaller packing layers.

[2]This is a loose parameter, thus some variability is ok

Approach B:
Our second approach looked to solve the issues brought up by approach A by packing the shipping container in vertical layers. In this approach, we look to fill the container in three sections. Each section is comprised of one type of cargo. Due to the cargo not perfectly fitting from edge to edge, we note that the 90% space constraint is generally met without further calculation. However, to allow for flexibility, vertical stacks or rows of cargo can be removed to accommodate different space constraints. Similar to the prior approach, different cargo packing ratios can be met by changing the number of vertical layers of each cargo type. Similar to the previous approach, the fragility constraint of cargo 3 is met since each section is stacked only upon itself. This packing strategy also ensures that cargo can be packed neatly with very little wasted space, and is more realistic for packing containers. Like Approach A, this approach hits space constraints before the weight limit.

3.2 Choosing an Approach Strategy

Our main qualm with strategies A and B were their overly simplistic approach. However, we realized that strategy B had a valuable trait - the solutions would be easy to pack, model, and were in small enough layers to be more customizable to conform to the specified cargo ratio. Packing a shipping container by "sections" of identically oriented cargo of the same type may cut down on needed packing experience and working time.

However, we still wished to look into less-homogeneous solutions to find other space-efficient packing strategies. Thus, we looked towards greedy and branch and bound algorithms [3]. Due to the sheer number of packages we needed to pack into each container, a branch and bound algorithm exploring every combination was inappropriate. This led us to choose a greedy algorithm approach. This approach gives us reasonable run times and explores packing combinations that minimize wasted space. Although a greedy algorithm may result in packing strategies that are high in complexity (many different orientations of cargo), we still saw value in developing a highly efficient algorithm for packing these containers.

With this greedy implementation and strategy B, we intended to develop a "quick and easy" model paired with a "complex and efficient" packing model. Depending on worker experience, packing times, and other factors, either model may be more beneficial to employ.

4 Mathematical Modeling

In this section we discuss researching the greedy algorithm and describe some terminology that will be used throughout the paper.

4.1 Literature Review

The concept of greedy algorithms dates back to the 1960s, with early works by pioneers like Dijkstra, Floyd, and Knuth. In the 1970s, researchers like Cormen, Leiserson, Rivest, and Stein laid the theoretical foundations for analyzing greedy algorithms, identifying problem characteristics

[3]These algorithms are explained in section 4

that make them suitable and establishing performance guarantees. Over the years, greedy algorithms have been applied to a wide range of problems in scheduling, optimization, clustering, and more, with researchers exploring variants like randomized and multi-objective greedy algorithms. Greedy algorithms are widely used in combinatorial optimization problems like knapsack, spanning trees, and set covering, leading to the development of greedy heuristics and approximation algorithms. Notable works like ”Introduction to Algorithms” by Cormen et al., ”Approximation Algorithms” by Vazirani, and books by Schrieb, Viehweger, Ausiello, and others have contributed to the theoretical and practical understanding of greedy algorithms across various domains. On the other hand, the paper titled ”A New Mathematical Model for a 3D Container Packing Problem” by Valentina E. Ocloo et al. utilizes the branch-and-bound algorithm as a critical component of their methodology to solve mixed-integer programming problems, specifically in the context of container packing problems. We figured this approach could be employed to attack the objective as well.

4.2 Greedy Algorithm

A greedy algorithm is a problem-solving approach that makes locally optimal choices at each stage with the hope of finding a global optimum. In other words, it makes the best choice available at the current moment, without considering the potential consequences of that choice on future decisions. Greedy algorithms are generally simple and efficient, but they may not always find the globally optimal solution. In the context of a 3D packing cargo problem, a greedy algorithm is used to pack 3 types of cargos into a container in an efficient manner. The goal is to maximize the utilization of the available space while adhering to the specified cargo ratio, weight constraints and priority requirement. In our greedy approach, we sort the cargos based on the ratio of 4:5:1, and with fragility priority of cargo type 3 which could not be stacked on top with any other type of cargo other than itself. Starting with an empty container, we select the largest sized cargo that is able to fit into the shipping container. We then place the cargo in the container, considering various orientations and positions, choosing the orientation and position that utilizes the space most efficiently. We then move to the next cargo and choose its optimal position and orientation with respect to the former item, by picking the corner points of the new cargo and testing if the corner points are suitable. If the cargo can not be placed, move on to the next item. Continue this process until the cargos packed in the container reach the target space occupancy. This approach assumes that the container ship and all cargos are rectangular prisms, allowing us to have each edge (or projected edge) of the rectangular prism parallel or perpendicular to every single planar surface of the shipping container. This qualifying assumption is important, as it disqualifies irregular cargo and shipping containers from

4.3 Branch and Bound Algorithms

The branch and bound algorithm is employed to partition the feasible solution set into smaller subproblems, solving these subproblems iteratively until the optimal solution is found. The algorithm involves systematically exploring subproblems by branching and creating a decision tree. Each branch represents a partial solution, which must be evaluated and possibly expanded into further branches. The algorithm needs to compute bounds for each subproblem to decide whether it should be explored further or pruned. As the number of boxes increases, the number of branches (subproblems) grows, leading to more computations. According to the result of the paper by Valentina E. Ocloo et al, over 24 boxes being packed would need over 60 minutes to generate a result. In our case, the number of boxes we need to pack is above 100, therefore using the branch and bound algorithm becomes unlikely due to the time concerns.

5 Mathematical Assumptions and Simplifications

This section will cover the changes we have made to our modeling problem to complete our implementations.

5.1 Implicit Constraints

In our project, we identified several implicit constraints that impact the cargo packing process. Firstly, certain types of cargo have orientation requirements, such as only being able to rest with a particular side facing up. Secondly, due to fragility concerns, some cargo can only be placed in the topmost layer of the container, either underneath cargo of the same type or with no other cargo stacked above it. Additionally, the packaging of certain cargo types restricts them to only be stacked upon cargo of the same type. Lastly, we recognized that factors such as worker skill level and the prevalence of irregularly shaped cargo can lead to ”short-packing,” where the maximum capacity of the container is not fully utilized.

5.2 Simplifications

To approach the cargo packing problem more effectively, we made several simplifying assumptions. We assumed that the cargo sizes are fixed and cannot be modified or repackaged. Furthermore, we constrained the number of each cargo type to be a non-negative integer. To reduce the complexity of the problem, we limited the cargo rotation to only horizontal orientations, maintaining a constant height. Initially, we chose to exclude shipping priority from our model, although we acknowledge the potential to incorporate this aspect in future iterations. These simplifications allowed us to concentrate on the primary objective of maximizing space utilization while adhering to the other specified constraints.

5.3 Assumptions

To further simplify our problem objective and narrow the scope of future applications, we’ve made the following simplifying assumptions:

- Cargo size cannot be changed because the contents of each cargo come prepackaged upon its arrival to the loading dock. Additionally, the cargo sizes remain constant for standardization of business practices. Thus, the resulting amounts of each cargo must be represented by whole numbers in our resulting model, often rounded from decimal value.
- Due to the difficulties surrounding the shipping priority, we’ve excluded it from our current model. However, we do intend to give our community partner a range of solutions so that they are able to choose which packing order is most appropriate for their shipping needs.

- The amount of each cargo should be nonnegative, meaning that each cargo type must be packed in the shipping container because using only 1-2 different cargo types does not satisfy our problem of optimizing the space allotted. Additionally, we're aiming for a ratio of 4:5:1 and considering it a soft constraint.
- There is no limit to amount cargo that can be stacked on top of each other.
- The weight distribution does not have to be balanced nor does the desired 4:5:1 ratio correspond to weight.
- The skill level of the workers packing the cargo itself is negligible, thus any packing solution will be considered no matter the visualization difficulty.

6 Modeling

This section covers the strategies of our Approach B (Layering) and Greedy approach.

6.1 Strategy B

To accommodate for volume constraints (in our case, a 90% constraint), a method in python was written to scale down the dimensions of the shipping container. For example, to try and best fit a 70% volume constraint, we will scale down the dimensions of the container until we are left with a smaller container that is 70% of the original volume. We will then try to fill this container as full as possible with the original cargo dimensions.

Approach B's main approach is splitting the shipping container into different sections for each type of cargo (in this case, 3 sections). To create these sections, we made vertical splits of the container. This allows us to provide a simple back-to-front packing instruction, and avoid any fragility violations (in our case, cargo 3 will be in a section where it is only stacked upon itself). These splits are chosen based off of the desired volume ratio of the cargo. In our case, the shipping crate's length was split into three sections who's lengths formed a 4:5:1 ratio to one another. Since the width and height of all sections are identical, splitting the length along these ratios, and filling each section with their respective cargo will all us to loosely follow this constraint.

After finding the section dimensions to the resized container, we begin our orientation tests. In other cases, we realized that some cargo may have extra constraints, such as not being able to be packed on it's side. We wanted our solution to apply to as many types of cargo as possible, thus we only decided to test two different orientations for this strategy. These two strategies are testing length-wise and width-wise fitting, as seen in the image below.

To determine the optimal orientation, we determine the least total waste. This is done by finding the total remainder remaining after placing as many blocks length and width-wise as possible, and adding the remaining lengths on the length and width sides. Of these two orientations, the one with the lower remainder is chosen. The section is then filled with as many of these blocks as possible. The leftover width and length dimensions are stored, and the orientation and dimensions of the cargo are displayed as packing instructions. This process is repeated for all other sections.

After these processes, we will have one area of "leftover space" on the width side of each section if the cargo does not fit perfectly (if there is no leftover space, extra space algorithms will not be run). Checking the two orientations of each cargo, we find the cargo type and orientation

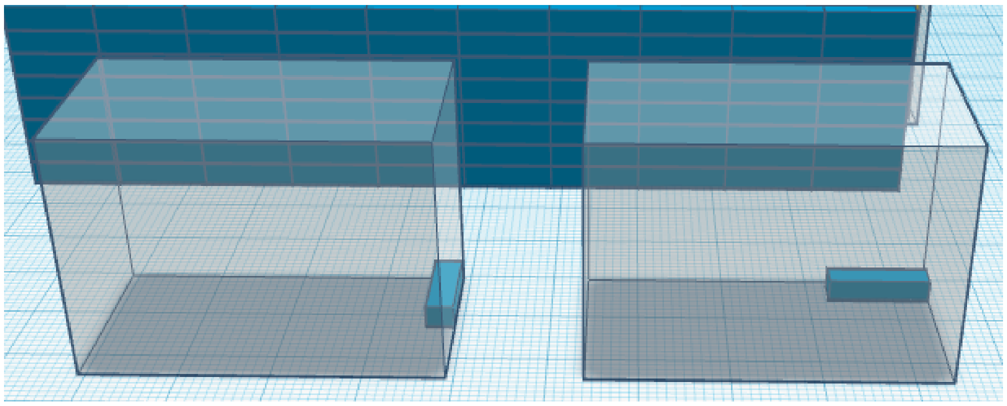
Figure 1. The Two Orientations of Cargo Tested in Strategy B

using the same remainder strategy explained in the prior paragraph. These instructions are also displayed to the user.

Similar to width-wise remaining space, there is also length-wise remaining space at the end of each section if the cargo does not fit perfectly. To minimize complexity, we will assume that the length-wise leftovers of each sections will be totaled at the end of the container. Essentially, sections will be packed back-to-back even if there is leftover space. This way, the leftover length can be addressed as one section in the front most part of the container. We choose cargo type and orientations identically to the algorithm for filling width-wise remaining space.

All placed cargo are totaled in order to provide an overview of the percentage of the container filled, percentage of each cargo's volume, and the total volume packed.

6.2 Greedy Approach

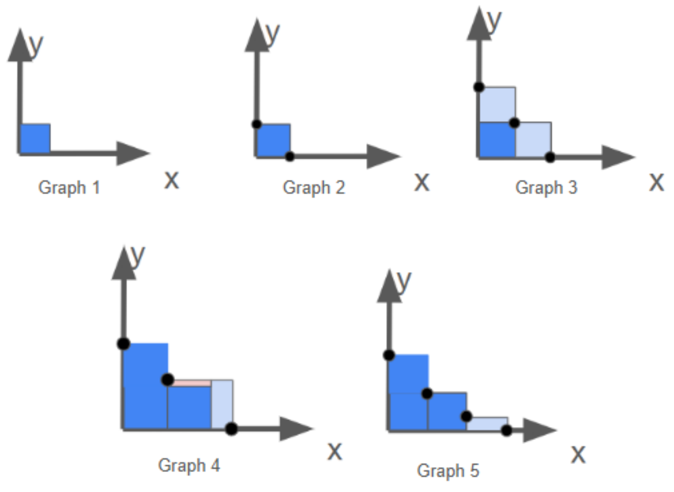

Figure 2. Corner finding and optimal block placing in 2D

The greedy approach is implemented using the approach described in an The Three-Dimensional

Bin Packing Problem by Martello, Pisinger, and Vigo. Fully fleshing out the approach that Martello et al. utilize in their paper would require considerable mathematical reasoning that is beyond the scope of this paper. Instead, we will give a conceptual overview of how a greedy algorithm was employed to solve this problem.

The developed algorithm has two main components: corner point identification and placement goodness evaluation.

Starting with placement goodness evaluation, it is necessarily true that certain cargo placement configuration (which is defined as placement location and orientation) maximize the space utilized. Take, for instance, this two dimensional example in 2, where graph 4 shows a sub optimal way to place the skinny long rectangular cargo, as it wastes a little bit of space that could have otherwise been used by another block due to the rectangular cargo's orientation. A greedy approach would ensure that the block is placed in a way that minimizes the wasted space at every single step.

On the other hand, the corner finding algorithm is essential to the functioning of the greedy approach, as cargo will be placed at these corner points. The corner finding algorithm is performed by first organizing the boxes placed in the two dimensional container in descending order of depth (y-axis) and setting a tracking variable (which we will call x_bar) to 0. Next, the boxes are checked for the x-axis location corresponding to the end of the box. If the x-axis location of the box is greater than x_bar, then this is deemed to be a two dimensional corner point and x_bar is then updated to this new value. This is repeated until all of the boxes in the two dimensional container have been checked.

This algorithm then can be utilized to find corner points in three dimensions by applying the two dimensional corner finding for every unique and discrete box height in the shipping container. For each unique height, the two dimensional corner points are found, which are then treated as potential three dimensional corner points. (Evaluating whether or not a two dimensional corner point is also a three dimensional corner point is discussed in Martello et. al's paper.) Once three dimensional corner points are found, we then place a cargo at one of the identified corner points, rotating the cargo around until we find a configuration that minimizes wasted space. This is repeated until no more cargo can be fit into the shipping container, or until we run out of cargo.

6.3 Software

All code from Approach B was written in Visual Studio Code (Known as "VS Code), which is a free, open source code editor developed by Microsoft. All computer aided design (CAD) visualizations were made using TinkerCAD.

7 Results and Analysis

7.1 Approach B

Running the code for Approach B resulted in a surprisingly efficiently packed container, with relatively little wasted space. Without scaling down the container, we reached a solution that filled 91.2% of the container. Due to the nature of cargo not fitting perfectly, we realized a higher desired volume constraint was needed to reach the desired cargo filled. Altering the desired volume constraint, we realized that 97%, paired with the original 4:5:1 distribution, yielded the closest packed volume at 90.40%.

Desired Volume constraint	Desired Distribution	Modeled Volume Filled	Modeled Distribution
100%	4:5:1	91.17%	3.1 : 5.2 : 1.6
98%	4:5:1	83.58%	3.2 : 4.5 : 2.3
97%	4:5:1	90.40%	2.9 : 4.1 : 2.1
95%	4:5:1	76.77%	3.5 : 4.8 : 1.6
90%	4:5:1	75.60%	3.5 : 5.0 : 4.5

Table 2. Approach B with Fixed Desired Distribution

Taking our closest desired volume constraint (97%), we tested different combinations of the desired distribution to achieve a model as close to the 4:5:1 goal as possible. As seen below, the desired distribution of 5:4.5:0.5 at a 97% desired volume constraint helped us reach a distribution extremely close to the desired volume of 90% and ratio of 4:5:1.

Desired Volume constraint	Desired Distribution	Modeled Volume Filled	Modeled Distribution
97%	4 : 5 : 1	90.40%	2.9 : 4.1 : 2.1
97%	4.5 : 5.5 : 0	89.49%	3.5 : 5.8 : 6.3
97%	5 : 4 : 1	90.10%	3.8 : 4.4 : 1.5
97%	5 : 4.5 : 0.5	90.40%	3.9 : 4.9 : 1.2

Table 3. Approach B with Fixed Desired Volume

Through trial and error, Approach B led us to our first packing solution, coming within just 1.04% of our goal of 90%, whilst also staying close to the desired 4:5:1 ratio.

Our proposed packing instructions[4]:

Section 1
Place 12 x 3 x 5 of Cargo 1 in section 1 with the width side along the container's long side
Place 8 x 1 x 5 of Cargo 2 in section 1 with the width side along the container's width side

Section 2
Place 10 x 3 x 5 of Cargo 2 in section 2 with the width side along the container's long side
Place 4 x 1 x 11 of Cargo 3 in section 2 with the width side along the container's width side

Section 3
Place 2 x 2 x 11 of Cargo 3 in section 3 with the width side along the container's long side
No cargo can fit into these dimensions [5]

[4]This is raw output from the code of Approach B

[5]This means that no dimensions of any cargo (length-wise or width-wise) can fit into the remaining space

End Section:
Place 1 x 2 x 11 of Cargo 3 in section -1 [6] with the long side along the container's width side

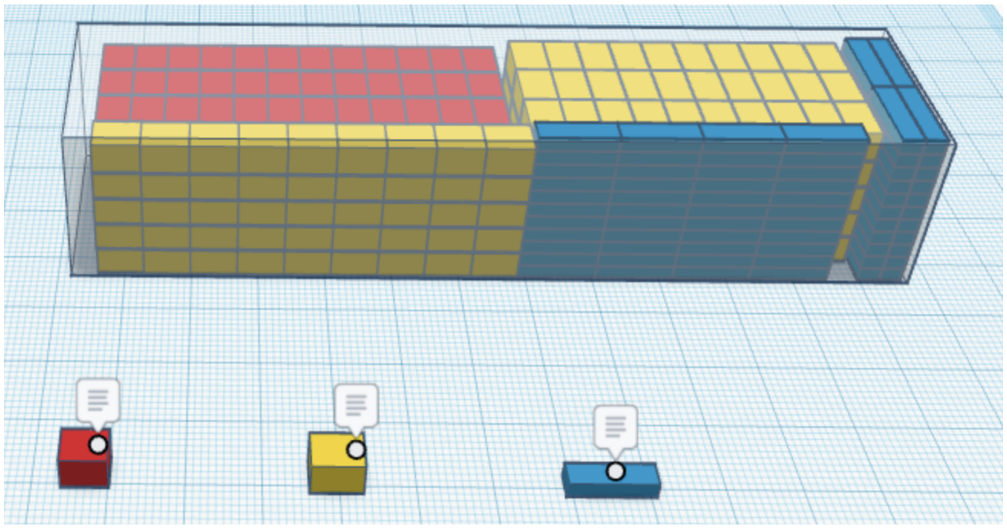

Figure 3. Strategy B's Proposed Solution (Red block: Cargo 1, Yellow Block: Cargo 2, Blue block: Cargo 3)

This packing arrangement results in 88.96% of the shipping container filled, with 3.9:4.9:1.0 cargo ratios. This approach most closely models how SDS wanted their containers packed (90% capacity constraint at 4:5:1 cargo ratios).

7.2 Greedy Approach

We first began testing by running the code for 2 dimensions of the shipping container, filling it with flattened versions of the various cargo types that SDS utilizes in their shipping operations. The results indicate that the greedy approach was able to get a 2 dimensional container filled to more or less 80% under various cargo ratio configurations. Although slightly disappointing, this makes sense as the algorithm in its current implementation always places the box in the corner that is closest to the origin of the shipping container, not necessarily the most optimal corner placement. (This was done due to time constraints, simplifying the problem in a way that was still rational but also reducing the problem complexity in a way that made the preliminary approach still manageable.) Future improvements to this code will place the box in a corner with the orientation that minimizes the wasted space.

[6]"-1" refers to the end section (made of all leftover lengths of other sections)

Cargo Ratio	2D Occupancy
4 : 5 : 1	80.3%
10 : 0 : 0	80.3%
2 : 5 : 3	77.5%
0 : 10 : 0	73.1%

Table 4. 2D Greedy Approach with Variations on Cargo Ratio

With this in mind, we then move onto the three dimensional packing algorithm. As previously mentioned, the three dimensional packing algorithm is simply a higher dimensional implementation of the 2 dimensional algorithm, employing different corner finding and goodness of placement evaluation algorithms, but the underlying principles are the same. Due to the corner placing constraints mentioned above, we did not get the results that we expected for the three dimensional greedy algorithm. We have results posted below:

Cargo Ratio	3D Occupancy
4 : 5 : 1	64.3%
10 : 0 : 0	66.2%
2 : 5 : 3	70.9%
0 : 10 : 0	57.2%

Table 5. 3D Greedy Approach with Variations on Cargo Ratio

Although the results were not as good as we expected, the proof of concept for a greedy algorithm clearly works, clearly indicating that this is a viable strategy in exploring complex packing arrangements.

7.3 Conclusions

In this study, we aimed to tackle the issue of finding the most efficient way to pack three different types of cargo in a shipping container through two distinct solutions. The layering approach focused on providing a straightforward, practical solution that would prioritize ease of packing by dividing the shipping container into distinct sections. Conversely, the greedy approach prioritizes volume efficiency through corner point identification thus optimizing space occupancy. While the current greedy approach has yet to fully realize it's full space occupancy maximizing powers, even if it was able to be truly and fully optimized we still believe that approach B would serve as a better solution due to it's simplicity. In future iterations of this project we would experiment with different cargo types, consider more orientations, and develop a fully automated visualizations of the proposed packing solutions.

Acknowledgments

We would like to express our sincere gratitude to the president of SDS, for proposing this engaging project and for their unwavering support in answering all our questions throughout the process. Their insights and collaboration have been invaluable to the success of this endeavor. We also extend our deepest appreciation to Professor Sara Billey from the University of Washington-Seattle

Department of Mathematics for her passion and dedication to the Discrete Modeling course. Her tireless efforts in providing valuable feedback, ideas, and guidance have been instrumental in shaping our project and enhancing our learning experience. Finally, we would like to acknowledge the hard work and commitment of our project team members: Charles Chen, Everett Hirano, Logan Hosoda, Cade Tanaka, and Bright Xiang. Their collaborative spirit, innovative ideas, and diligent efforts have been essential in bringing this project to fruition. We are grateful for the opportunity to work alongside such talented and dedicated individuals.

8 Appendix

Desired Volume constraint	Desired Distribution	Modeled Volume Filled	Modeled Distribution
100%	4:5:1	91.17%	3.1 : 5.2 : 1.6
99%	4:5:1	86.69%	4.0 : 5.0 : 1.5
98%	4:5:1	83.58%	3.2 : 4.5 : 2.3
97%	4:5:1	90.40%	2.9 : 4.1 : 2.1
96%	4:5:1	77.90%	3.4 : 4.8 : 1.7
95%	4:5:1	76.76%	3.5 : 4.9 : 16.2
94%	4:5:1	76.76%	3.5 : 4.9 : 16.2
93%	4:5:1	76.76%	3.5 : 4.9 : 16.2
92%	4:5:1	67.70%	4.0 : 5.5 : 5.0
91%	4:5:1	67.70%	4.0 : 5.5 : 5.0
90%	4:5:1	67.70%	4.0 : 5.5 : 5.0
89%	4:5:1	67.70%	4.0 : 5.5 : 5.0
88%	4:5:1	67.70%	4.0 : 5.5 : 5.0
87%	4:5:1	67.70%	4.0 : 5.5 : 5.0
86%	4:5:1	67.68%	4.0 : 5.5 : 5.0
85%	4:5:1	60.18%	4.5 : 4.9 : 6.0

Table 6. Full table of Approach B with a fixed desired distribution. From the different desired volume constraints tested, we found 97% the closest to 90%. Note that due to the sizes of containers, different desired volume constraints that are close in percentage produce the same results.

Desired Volume constraint	Desired Distribution	Modeled Volume Filled	Modeled Distribution
97%	4 : 5 : 1	90.40%	2.9 : 4.1 : 2.1
97%	4.5 : 5.5 : 0	89.49%	3.5 : 5.8 : 6.3
97%	5 : 4 : 1	90.10%	3.8 : 4.4 : 1.5
97%	5 : 4.5 : 0.5	90.40%	3.9 : 4.9 : 1.2
97%	5 : 4.6 : 0.4	86.69%	4.0 : 5.0 : 8.0
97%	4.9 : 4.5 : 0.6	86.69%	4.0 : 5.0 : 7.9
97%	4.9 : 4.6 : 0.5	86.69%	4.0 : 5.0 : 7.9

Table 7. Approach B with fixed desired volume. Slowly incrementing and decrementing values to try and get as close to 4:5:1 and 90% as possible, we centered around the 5:4.5:0.5 ratio. Other ratios proved to get closer to our desired ratio, at the sake of the modeled volume filled deviating further from our goal of 90%.

8.1 Greedy Approach Sample Output

```
Corner: (0,0,0)— Box Location: (1, 490, 675, 510)
    Corner: (490,0,0)— Box Location: (1, 1165, 490, 510)
    Corner: (0,0,510)— Box Location: (1, 490, 675, 1020)
    Corner: (0,675,0)— Box Location: (1, 675, 1165, 510)
    Corner: (490,0,510)— Box Location: (1, 980, 675, 1020)
    Corner: (0,675,510)— Box Location: (1, 675, 1165, 1020)
    Corner: (675,675,0)— Box Location: (1, 1165, 1350, 510)
    Corner: (0,0,1020)— Box Location: (1, 675, 490, 1530)
    Corner: (675,675,510)— Box Location: (1, 1165, 1350, 1020)
    Corner: (0,490,1020)— Box Location: (1, 675, 980, 1530)
    Corner: (1165,0,0)— Box Location: (1, 1655, 675, 510)
    Corner: (675,0,1020)— Box Location: (1, 1350, 490, 1530)
    Corner: (675,490,1020)— Box Location: (1, 1165, 1165, 1530)
    Corner: (1165,675,0)— Box Location: (1, 1840, 1165, 510)
    Corner: (0,1350,0)— Box Location: (1, 490, 2025, 510)
    Corner: (1165,490,510)— Box Location: (1, 1840, 980, 1020)
    Corner: (490,1350,0)— Box Location: (1, 980, 2025, 510)
    Corner: (0,1350,510)— Box Location: (1, 675, 1840, 1020)
    Corner: (0,0,1530)— Box Location: (1, 490, 675, 2040)
    Corner: (0,1165,1020)— Box Location: (1, 675, 1655, 1530)
    Corner: (675,1350,510)— Box Location: (1, 1165, 2025, 1020)
    Corner: (1165,980,510)— Box Location: (1, 1840, 1470, 1020)
```

8.2 Relevant Links

Visual Representation of a Approach B solution in TinkerCAD:
`https://www.tinkercad.com/things/42mYetHrNMA-powerful-gaaris-densor/edit?sharecode=PVdpvup_LlL_mhvYfpH5nSRr8wZCA4B1yjvqlus5S6A`

Python Code Github Repository:
`https://github.com/eykh5/Math_381_Packing`

References

[1] Batarlienė, N., & Šakalys, R. (2021). Mathematical model for cargo allocation problem in synchromodal transportation. Symmetry, 13(4), 540. https://doi.org/10.3390/sym13040540

[2] Davidmchapman. (n.d.). GitHub - davidmchapman/3DContainerPacking: A 3D container packing library in C#. GitHub. https://github.com/davidmchapman/3DContainerPacking

[3] Makhova, L., Haykin, M., Glazkova, I., & Domnina, O. (2023). Development of mathematical models for trucks and cargo. Infrastructures, 8(2), 17. https://doi.org/10.3390/infrastructures8020017

[4] Ocloo, V., Fügenschuh, A., & Pamen, O. M. (2020). A new mathematical model for a 3D container packing problem. ResearchGate. https://doi.org/10.26127/btuopen-5088

[5] Zhang, B., Yao, Y., Kan, H. K., & Luo, W. (2024). A GAN-based genetic algorithm for solving the 3D bin packing problem. Scientific Reports, 14(1). https://doi.org/10.1038/s41598-024-56699-7

[6] Martello, S., Pisinger, D., Vigo, D. (2000). The Three-Dimensional Bin Packing Problem. Operations Research, 48(2), 256–267. https://doi.org/10.1287/opre.48.2.256.12386

[7] de Castro Silva, J. L., Soma, N. Y., Maculan, N. (2003). A greedy search for the three-dimensional bin packing problem: the packing static stability case. International Transactions in Operational Research, 10(2), 141–153. https://doi.org/10.1111/1475-3995.00400