PROJECT 1-1

PHASE 3: KNAPSACK

CONTENTS

01
Introduction

Introduction

- In this phase, we were tasked with solving two spatial optimization problems: the **Exact Cover** problem and **Knapsack** problem.

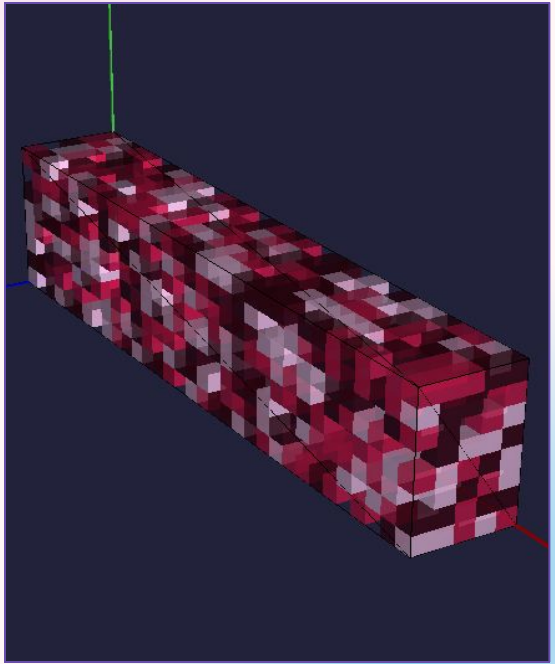
- We will cover why we chose to use **Dancing Links**, how we adapted it to our specific problems, and discuss the solutions we found to each question, if any.

02
Exact Cover

Approach

- We repurposed our Dancing Links implementation from Phase 1, since we theorized that with correct conversion of the 3D pieces and space, Dancing Links is capable of solving the problem.

- Adding an additional dimension to the field significantly increases the size of the matrix and memory requirements.



Implementation

- The only modification that had to be done was representing an extra dimension by extending the matrix along the z-axis.

- Enough columns to represent 3D coordinates had to be added, which are calculated using a formula detailed in our report.

- In order to display the field on the UI, the color ID of the piece or parcel in the matrix is stored.

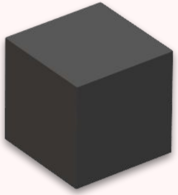
Exact Cover Using Parcels

Parcel A

Dimensions:
1.0m × 1.0m × 2.0m

Parcel B

Dimensions:
1.0m × 1.5m × 2.0m



Parcel C

Dimensions:
1.5m × 1.5m × 1.5m

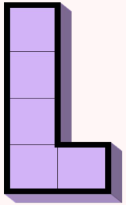
Truck

Dimensions:
16.5m × 2.5m × 4.0m

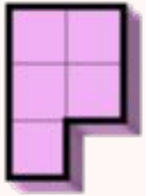
Solution

- The results obtained from running our algorithm indicate that a complete filling of the truck using any possible combination of parcels A, B, and C is impossible.

Exact Cover Using Pentominoes

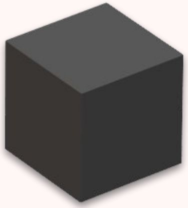


Pentomino L



Pentomino P

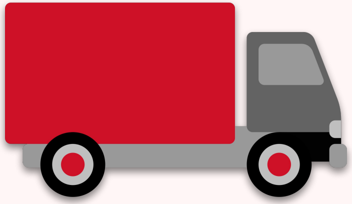
Pentomino T



Cube

Dimensions:
0.5m × 0.5m × 0.5m

Each pentomino is made up of 5 joined cubes of these dimensions



Truck

Dimensions:
16.5m × 2.5m × 4.0m

Solution

- Our algorithm determined that using the 3D pentominoes, the truck can be entirely filled without any gaps.

Placeholder

Will add photo of exact cover solution before final submission

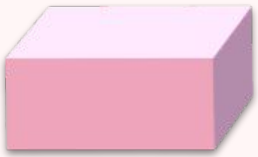
03
Knapsack

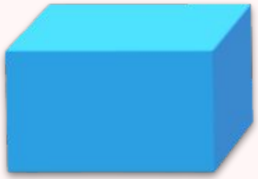
Approach

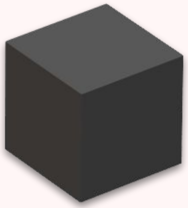
- To compute an approximate solution to the Knapsack problem, we slightly modify the Dancing Links algorithm to sum the assigned weights of every piece used.

- This is because we hypothesized that the difference in the maximum possible weight between using the traditional Dynamic Programming approach, and filling a 3D space and calculating the total weight of pieces used would not be large.

Knapsack Using Parcels

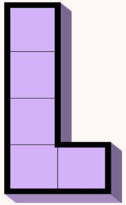


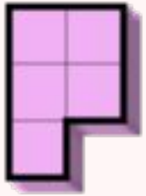




Parcel A	**Parcel B**	**Parcel C**
Value: 3	**Value: 4**	**Value: 5**

Knapsack Using Pentominoes





Pentomino L	Pentomino P	Pentomino T
Value: 3	Value: 4	Value: 5

Solution

- The maximum value we found using parcels was **196** and **1,030** using the 3 types of pentominoes.

- When we compared our parcel solution to an open source project that uses a heuristic Greedy Algorithm approach, we calculated a 17.35% error margin.

04
Who did what?

Placeholder

*Detailed breakdown of coding tasks on canvas