MATH3208 - Optimization Coursework

Azhagesh Azhagesh, Ashwinkrishna
aa9g22@soton.ac.uk, University of Southampton

May 30, 2025

The probelm of finding the optimal orientation and position of ireegular shaped objects in a finite area, is inherently NP complete [3] if you approach this with infinite orientations and positions possible, in a finite abritaty 2d space where the shape of the area can change ie rectangle, triangular 2d plane etc. Even if you want to pack irregular objects in a rectangle 2d space, in the most efficient manner, which reduces complexity as 1. it is not an arbitary shape but a rectange, and, which is fixed, the problem is still combinatorial and NP-hard, and more famously we can redefine this beyond the simple four animal puzzle.

I see this in a more general sense, of wanting to fit four irregular shapes inside a rectangular container, and taking into account the various possible orientations and positions these shapes can take. This is a famous problem, as can be classified as a nesting problem [5]. Now even if we were to remove the rotational contraint, or orientation, this is still a NP-hard problem, so an algorithm that can find the optimal or close to optimal solution with irregular shaped in a rectangular plane in the most efficient manner possible isn't likely to be polynomial time complexity.

Now the approach takes in [4] of using the inclusion-exclusion principal, and using heuristics by measuring the cutouts, the manually trying out all 48 possible permutations and then generating a dataset of the two shapes in the allowed orientations, and then finally using a program to effectively calculate the minimum amount of width needed, such that these shapes fit inside the minimum rectangle possible. This is slightly different from the original paper as the puzzle [2] as that had a hexangonal box where you had to place the pieces, instead of an rectangle.

One of the problems with utilising the inclusion-exclusion and taking this particular approach, is that you have to manually measure the distance, and make a cutout of the irregular shapes, which have scaling errors. Then use these cutouts to try possibe orientations and combinations, manually checking that the shapes do not overlap when you do measure the least width possible. Therefore this process is very time consuming, and isn't feasible for larger amount of shapes. One potential way to mitigate this is, as seen in the first paper [2], model the shapes as a circle which reduces, the measurements needed, but this is also very inefficient and leaves a lot of gap between the shapes, isn't close to being optimal.

So one way to apprach this problem is to in a similar fashion to paper 1 [2], model the shapes roughly, as a rectangular shape. This simplifies the algorithm needed to fit this in the smallest rectangle possible.

So the way I went about my initial solution here, is I used golang as the language, due to it's fast execution speed in comparison to Python, and compile times. I started off with modelling what I was going to do and thought about various edge cases and issues that might occur. So firstly I made a struct type in golang, that defines a rectangle. And this struct has two variables, which are height and width. Then utilising this I make a slice of this custom type, reading the measurements from the pdf, in real life this can be automated with computer vision techniques and openCV, this is what shipping containers do [1], this information of the height and width was then written in manually, but it can be automated with the methods suggested before.

Next there was the issue of possible permuatations of the rectangular shapes, it can either be 0 or 90 degrees, intuitively you can understand due to the symmetrical properties of rectangles it isn't worth trying to do 180 or 360 degrees, also due to our problem of trying to fit the shapes into the smallest space possible, we can conclude that it's best if the faces of the rectangle are parallel so as to minimise the wastage of space. Given we have four shapes we need to consider all possibilities, so 2 raised to 4, is 16 possibilities, which is lower than the 48 considered in this paper [4], (but yes I understand this is worse off long term, as you can approach it dynamically with inclusion exclusion to get better faster results, given you remeasure for every prime.).

So I wrote a combinatorics function that generates all possible combinations, either 0 or 90 degrees in 4 positions.

INSERT IN PICTURE OF LIST 0,0,0,0 etc ... 90,90,90,90

Then for the sake of simplicity, I use a loop to iterate through these possibilities, and if the angle is 0, I add the original width of the shape, if it is rotated, so 90, I then add the original length of the rectangular shape to the width, and at the end I have a list of possibilities, all the total widths and the max height, of the rectangular prism in each case, which is printed at the end. Then taking the minimum of the value, of the width, I print that out to the screen along with the corresponding height of that rectangular prism, along with the area.

References

[1] Lale Akaruna Barıs¸ Evrim Demir¨oza, ˙I. Kuban Altınelb. Rectangle blanket problem: Binary integer linear programming formulation and solution algorithms. *arXiv*, page 42, October 2019.

[2] UUGANBAATAR N INJBAT. The four strongest, mesuem of mongolia. *The mathematical Tourist*, 42:6, 2020.

[3] Bonfim Júnior, Plácido Pinheiro, Rommel Saraiva, and Pedro Pinheiro. Dealing with nonregular shapes packing. *Mathematical Problems in Engineering*, 2014, 07 2014.

[4] Bismark Singh. A mathematical programming approach for mongolia's "the four strongest" puzzle. *INFORMS TRANSACTIONS ON EDUCATION*, 25(3):5, November 2024.

[5] Franklina M.B. Toledo, Maria Antónia Carravilla, Cristina Ribeiro, José F. Oliveira, and A. Miguel Gomes. The dotted-board model: A new mip model for nesting irregular shapes. *International Journal of Production Economics*, 145(2):478–487, 2013.