

General 2D Bin Packing Problem

A difficult but interesting problem

Who are we?

- Hoàng Trần Nhật Minh
- Nguyễn Hoàng Phúc
- Nguyễn Hải Long
- Nguyễn Ngọc Dũng



Introduction

General 2D Bin Packing Problem

Ok, I got "2D", but what is "Bin Packing"?



Items

A lot of “**rectangle**” items,
generally in different **sizes**.
We need to transport them **all**.

Don't disappoint our customers.



\$ 300

\$ 50

\$ 950

\$ 50

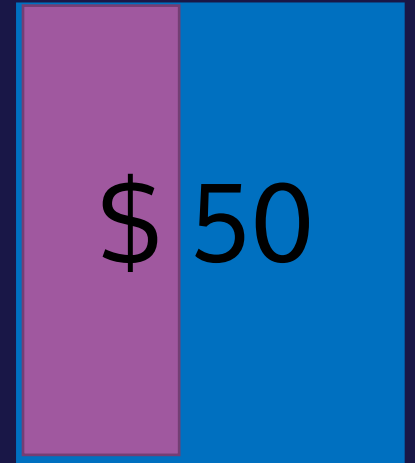
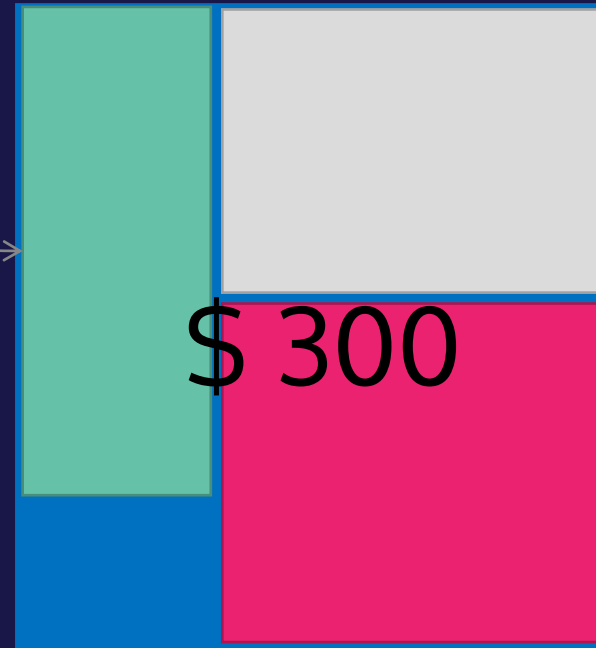
Cars

We have no car, so we rent some.
We can **refund** if we don't use a car.
In general, the cars have different
sizes and **costs**.



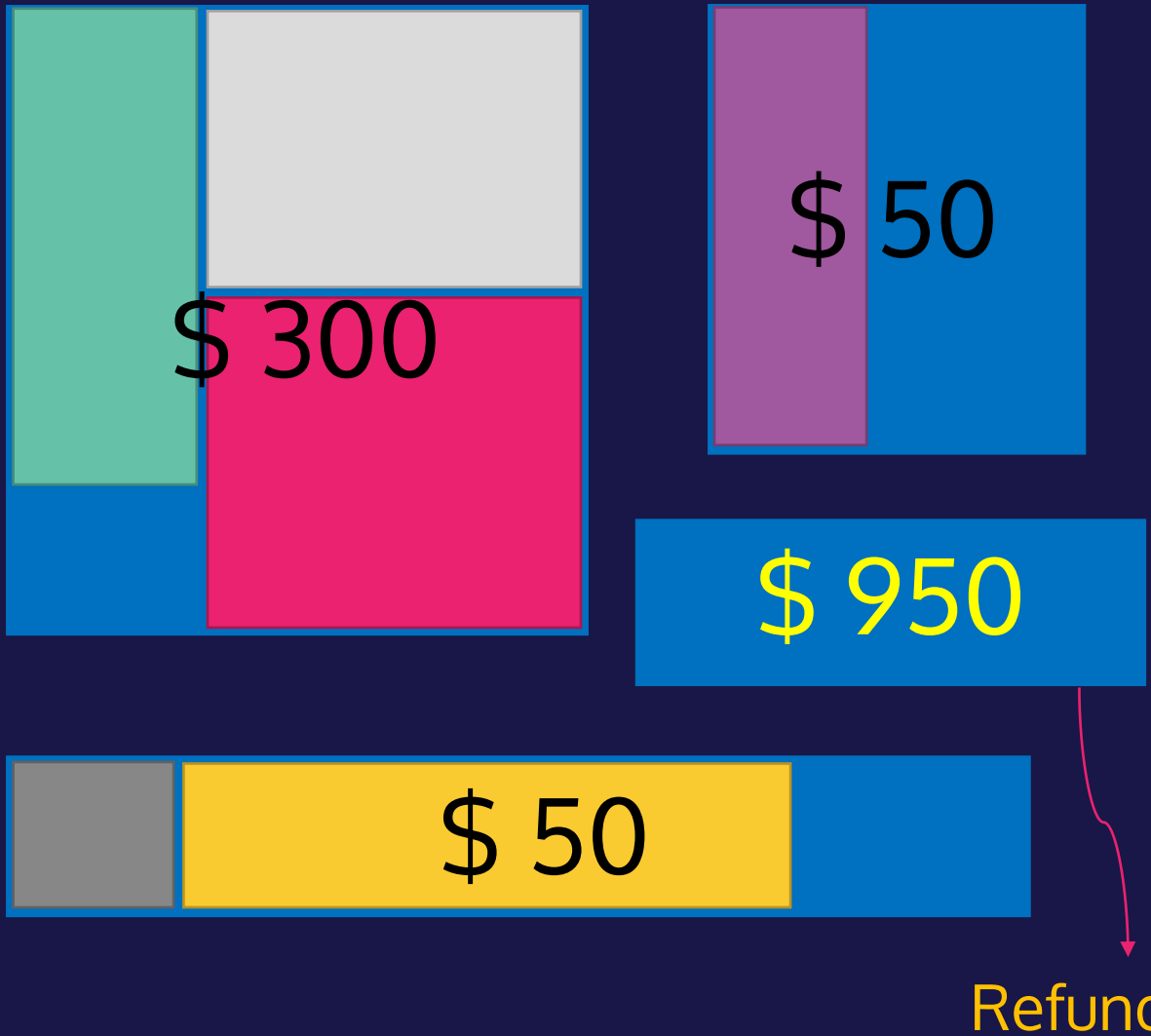
Seriously?

Rotatable
for 90°



Cost = 300 + 50 + 50 = 400 Refund!

Cost and Objective



The **total cost**
= **sum** of rental fees of **used cars**
We want to **minimize** this cost.

$$\text{Cost} = 300 + 50 + 50 = 400$$

Remark



- Each item has a size
- Each car has a size capacity and a cost
- Each item must be in one car
- All items in a car must somehow fit it orthogonally, in which items are rotatable for 90°
- Cost is the sum of fees of all used cars, minimize it

Fun facts! Why “General 2D Bin Packing”?

“Bin”

In articles and papers, instead of “car”, they wrote “bin”.

“2D”

The original version of this problem is **one-dimensional (1D)**, where each item has a “**weight**” instead of “**size**”. It is already an **NP-hard** problem.

“General”

There are **variations** like:

- The cost is the number of cars used (rental fee = 1)
- Items are not rotatable

Which are much **simpler**.

The background is a dark navy blue. It is decorated with numerous diagonal lines and shapes in various colors including yellow, pink, teal, purple, and grey. These elements are scattered across the top and bottom of the image, creating a dynamic, abstract pattern.

Input specification

_sample_data.txt

7 3 $\rightarrow n = 7$: Number of rectangles
2 3 $\rightarrow m = 3$: Number of cars
1 5
2 2 } Side lengths of 7 rectangles
3 2
1 4 $\rightarrow w_6 = 3$: Width of the 6th rectangle
3 2 $\rightarrow h_6 = 2$: Height of the 6th rectangle
4 1
3 5 100 } Side lengths and costs of 3 cars
4 5 150 $\rightarrow W_3 = 5, H_3 = 6$: Width, height of the 3rd car
5 6 200 $\rightarrow c_3 = 200$: Cost of the 3rd car



Data generation

And properties of data

Define the number of rectangles

There will be 75 different input files, each of them has the number of rectangles (**n**) as follow:

- 50 easy and medium files: From 5 to 54 with step 1, i.e. 5, 6, 7, ..., 54
- 10 hard files: From 60 to 330 with step 30, i.e. 60, 90, 120, ..., 330
- 14 very hard and severe files: From 350 to 1000 with step 50, i.e. 350, 400, 450, ..., 1000
- 1 sample file 😊

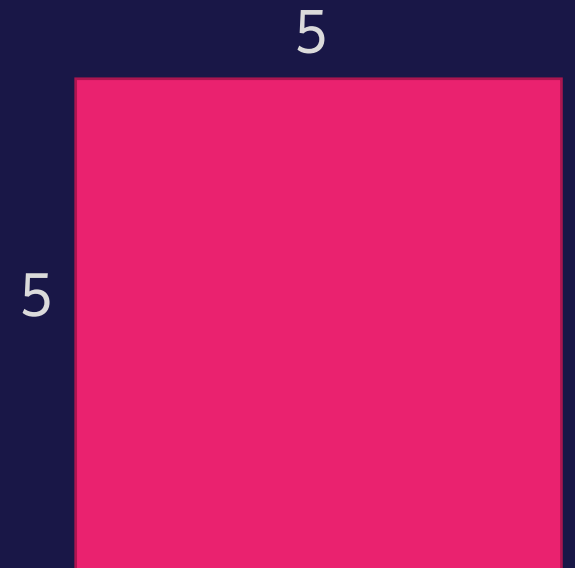
The generating algorithm for each file

- Randomize rectangles
- Randomly put rectangles in cars
- Shrink sizes of cars
- Randomly generate a few more cars and randomly assign costs for all cars

Note that the probability distribution of all random variables in this algorithm is uniform.

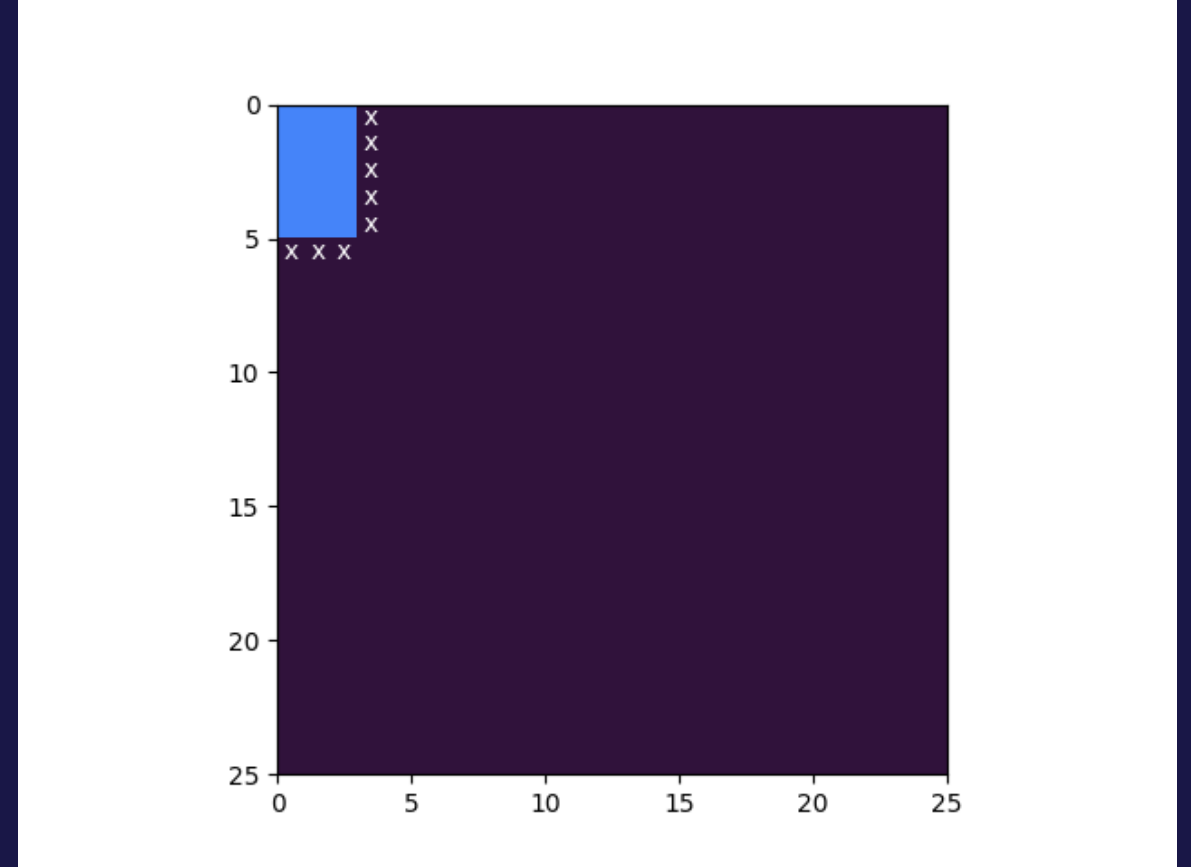
Randomize rectangles

- Randomize a rectangle: each side is a random integer from 1 to 5
 - A rectangle can be 1x1, 5x5 or any size in between
- Repeat that for **n** times to obtain **n** rectangles



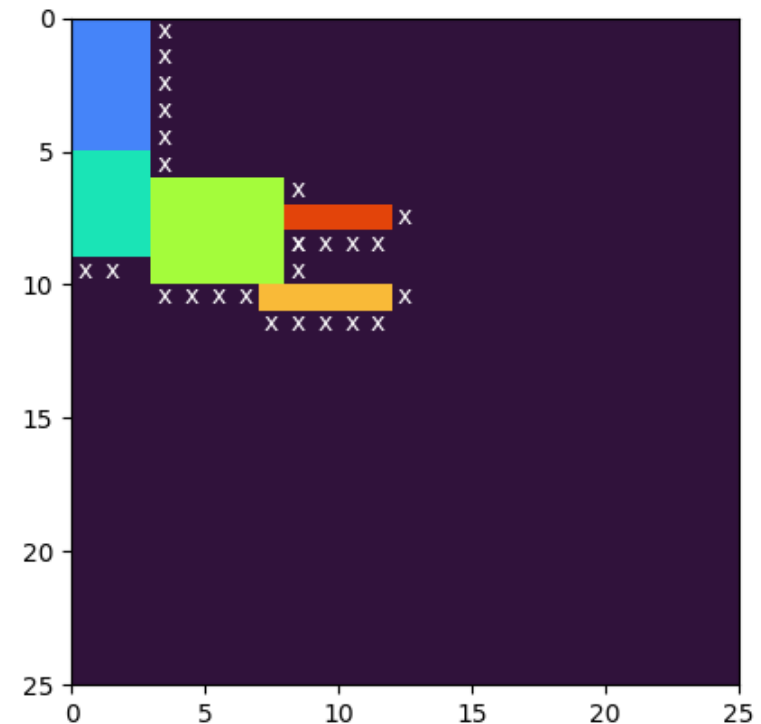
Randomly put rectangles in cars

- Out of n rectangles, randomly pick from 2 to 5 rectangles
- Initialize a 25x25 car (there is a mathematical reason behind it)
- Put the first rectangle in the upper-left corner
- Mark any place that is next to the right or lower bound of the rectangle

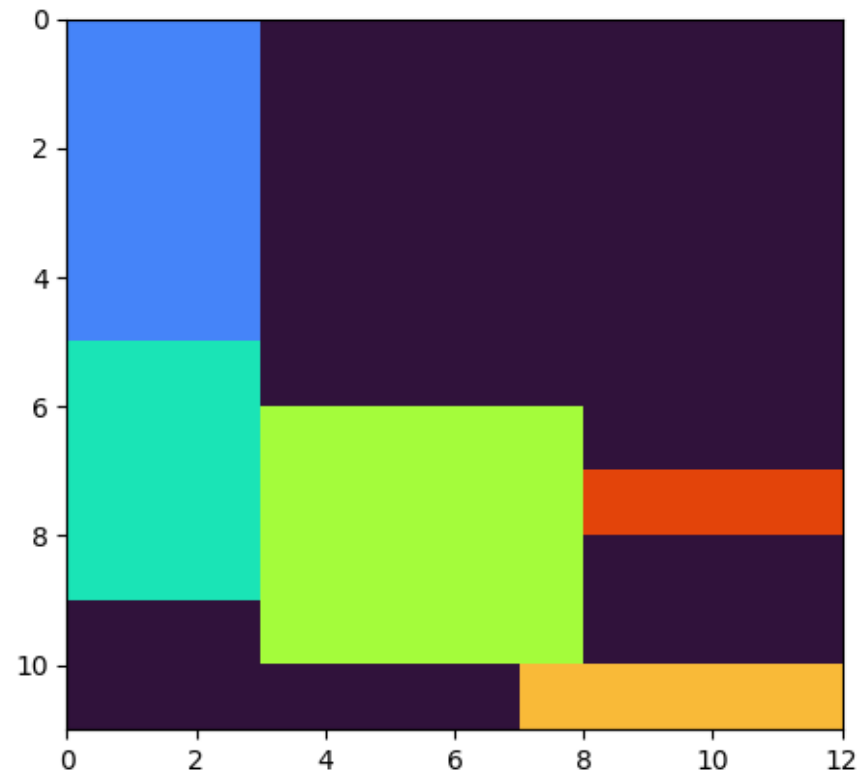
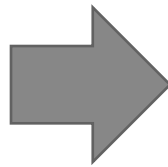
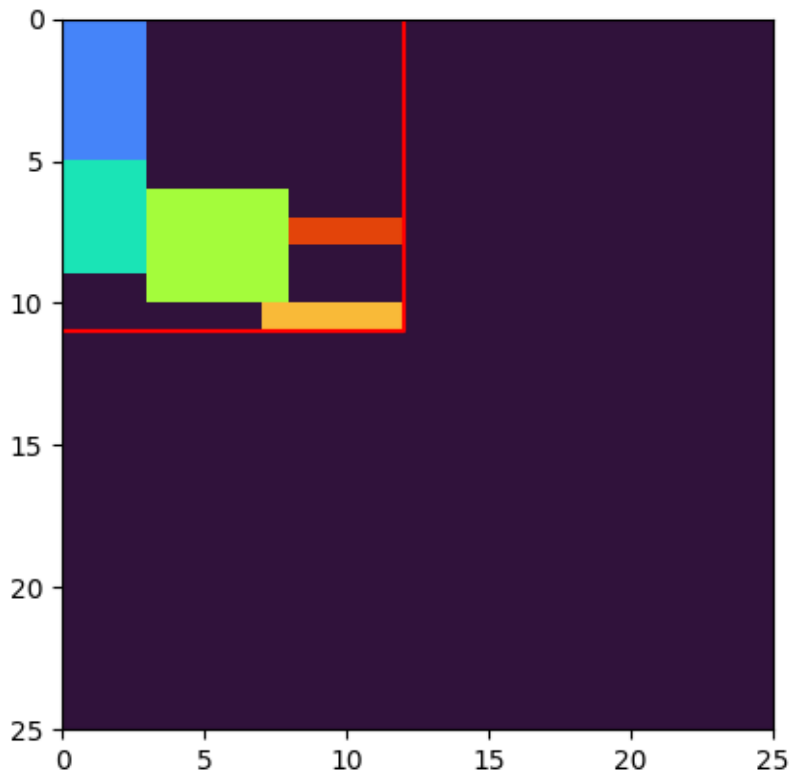


Randomly put rectangles in cars

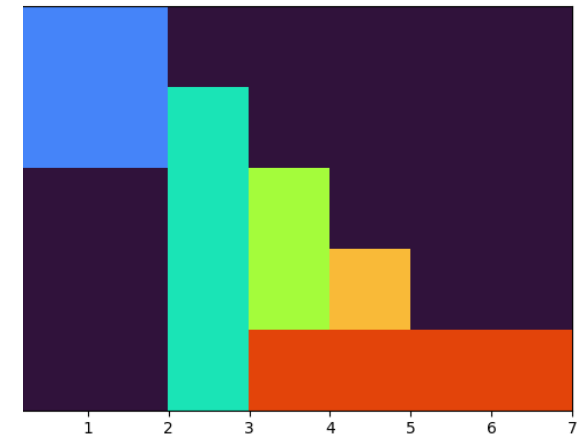
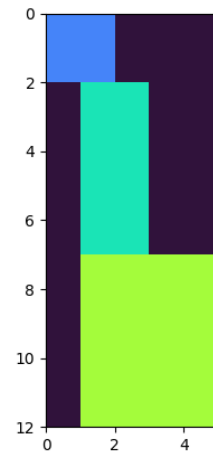
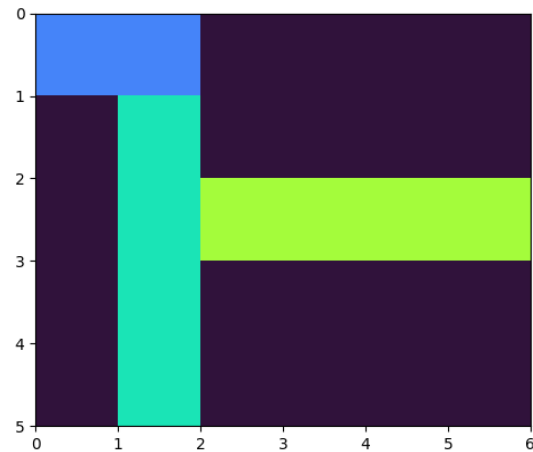
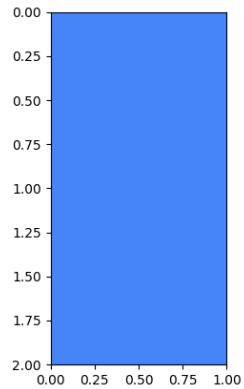
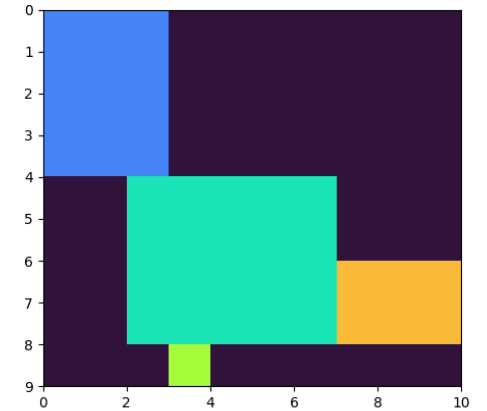
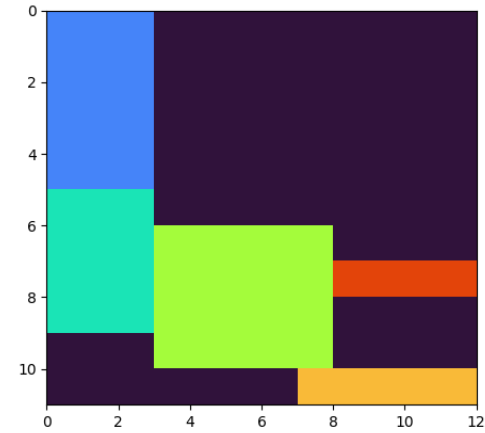
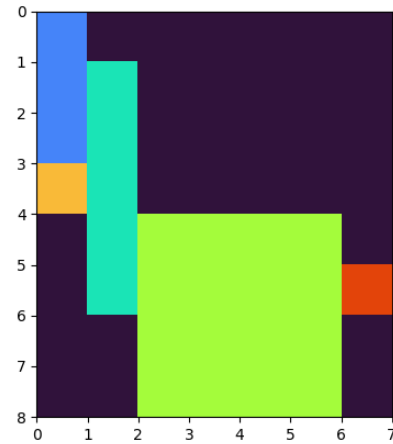
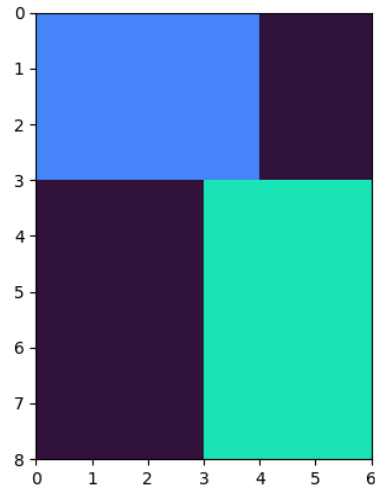
- Out of the rectangles left, choose one and randomly put it in any marked place
- Repeat the step above until there is no rectangle left (from 1 to 4 times of repetition)



Shrink sizes of cars



Some cars of 0030.txt



Randomly generate a few more cars and randomly assign costs for all cars

- Randomly generate a few more cars:

Random each side of each car: From 1 to 25

→ A car can be 1x1, 25x25 or any size in between

"A few" = Ceil of $1/5$ number of the used cars

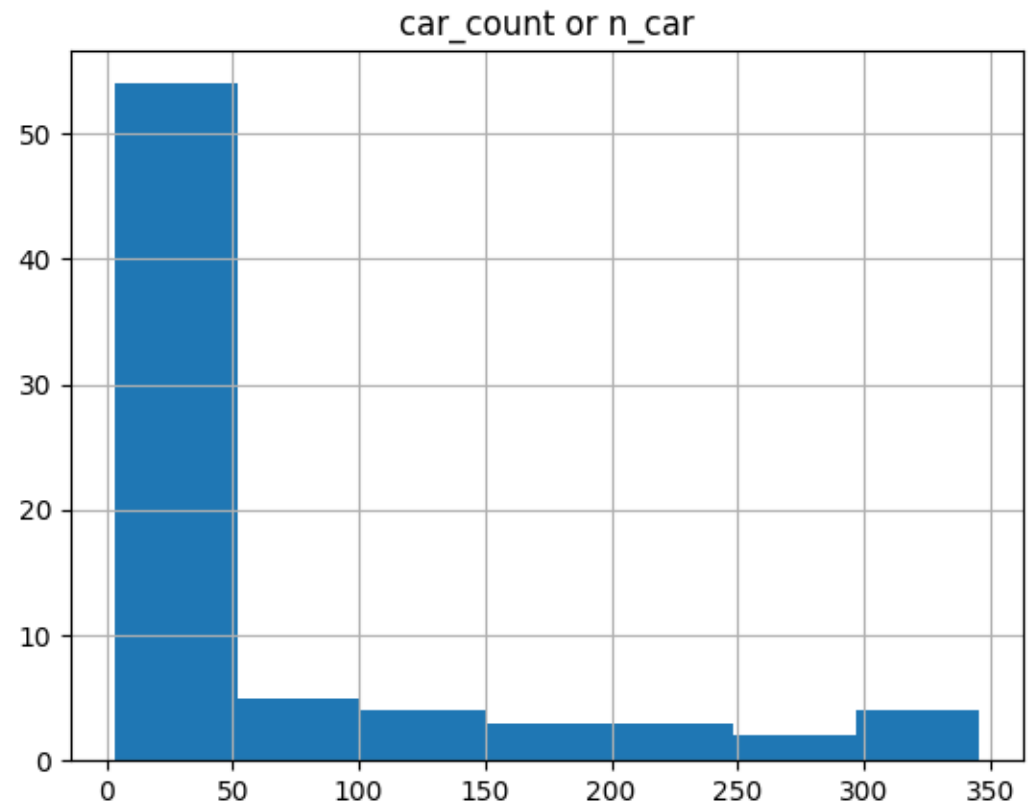
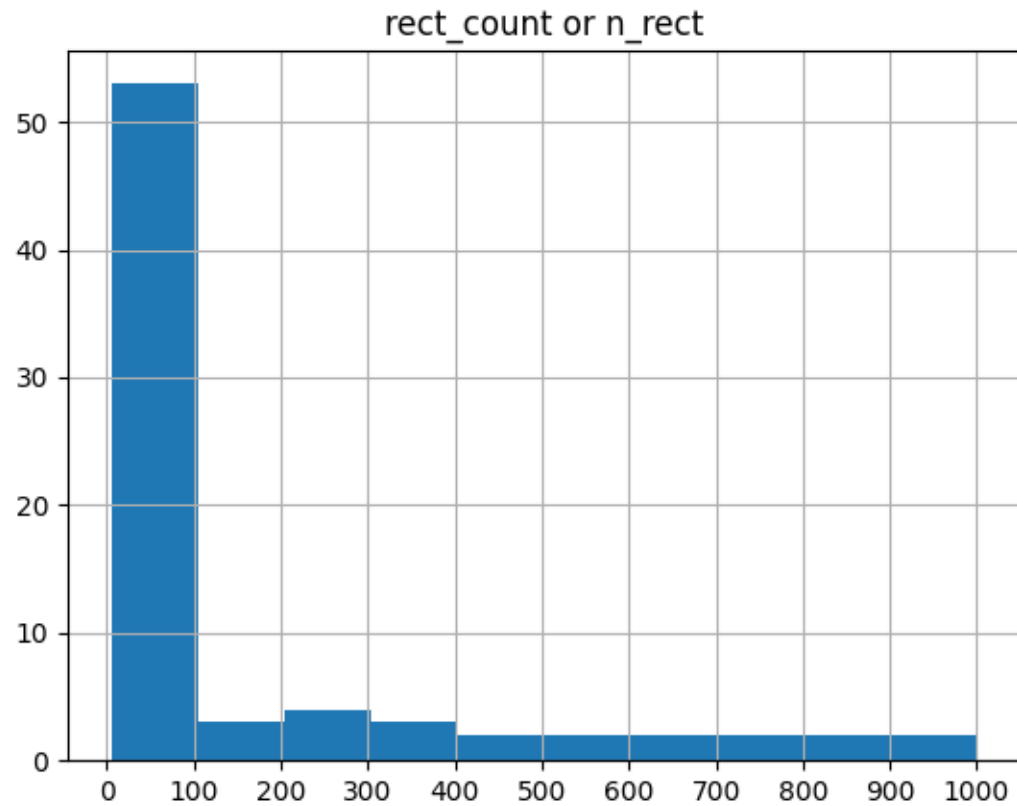
- Randomly assign a cost to every car so far:

From 100 to 1000 with step 50, i.e. 100, 150, 200, ..., 1000

Data properties

- There are a lot more cars than needed
- Theoretically, the maximum number of rectangles that can fit into one car is $25 \times 25 = 625$
 - Backtracking, Brute force, Dynamic programming, or Branch and bound is impossible due to complexity (if implemented, the mean branching factor is 60, the mean depth is 172, so there are about $60^{172} \cong 7 \times 10^{305}$ nodes, for each node, its legitimation is also a very costly backtracking algorithm)
- Multiple "kinds" of rectangle, only with 5 different side-lengths:
Small: 1x1, Big: 5x5, Normal: 3x2, "Annoying": 1x5, ...

Data properties: Size distribution visualized





Problem Formulation

Building CP and MIP model

Denotation

- $R = \{1, \dots, n\}$ is the set of given rectangles

Item i has width w_i and height h_i

- $B = \{1, \dots, m\}$ is the set of available bins

Bin k has width W_k , height H_k and cost c_k

Variables

- $o_i \in \{0, 1\}$ represents the orientation of rectangle i
- l_i, r_i, t_i, b_i are left, right, top and bottom coordinates of item i
- Binary variable u_k is 1 iff bin k is used
- Binary variable p_{ik} is 1 iff item i is placed in bin k
- $y_i = k \in \{1, \dots, m\}$, i.e. item i is placed in bin k

CP model - Constraints

● $o_i = 0 \Rightarrow r_i = l_i + w_i \wedge t_i = b_i + h_i \quad \forall i \in R \quad (1)$

● $o_i = 1 \Rightarrow r_i = l_i + h_i \wedge t_i = b_i + w_i \quad \forall i \in R \quad (2)$

● $y_i = y_j \Rightarrow r_i \leq l_j \vee r_j \leq l_i \vee t_i \leq b_j \vee t_j \leq b_i \quad \forall i, j \in R, i < j \quad (3)$

● $y_i = k \Rightarrow r_i \leq W_i \wedge t_i \leq H_i \quad \forall i, j \in R, k \in B \quad (4)$

● $p_{ik} = 1 \Rightarrow y_i = k \quad \forall i \in R, k \in B \quad (5)$

● $\sum_{i=1}^n p_{ik} \geq 1 \Rightarrow u_k = 1 \quad \forall k \in B \quad (6)$

Constraints

- (1) If an item doesn't rotate, its right = its left + its width and its top = its bottom + its height
- (2) If the item rotates then its right = its left + its height and its top = its bottom + its width
- (3) If two items are placed in the same bin, they can't overlap each other
- (4) If one item is placed in a bin then its right and top coordinates can't exceed the bin
- (5) Item i is placed in bin k
- (6) Bin k is used when at least one item is placed in it

MIP model - Constraints

● $o_i = 0 \Rightarrow r_i = l_i + w_i \wedge t_i = b_i + h_i$ (1); $o_i = 1 \Rightarrow r_i = l_i + h_i \wedge t_i = b_i + w_i$ (2)

To MIP: $r_i = l_i + w_i * (1 - o_i) + h_i * o_i$; $t_i = b_i + h_i * (1 - o_i) + w_i * o_i$

● $y_i = y_j \Rightarrow r_i \leq l_j \vee r_j \leq l_i \vee t_i \leq b_j \vee t_j \leq b_i$ (3)

To MIP: $r_i \leq M * (1 - x_1) + l_j$; $r_j \leq M * (1 - x_2) + l_i$;

$t_i \leq M * (1 - x_3) + b_j$; $t_j \leq M * (1 - x_4) + b_i$;

$x_1 + x_2 + x_3 + x_4 + (1 - x_0) * M \geq 1$;

$x_1 + x_2 + x_3 + x_4 \leq x_0 * M$; $b = 1 - x_0$;

$y_j + x_0 \leq y_i + M * x_{01}$; $y_i + x_0 \leq y_j + M * x_{02}$; $x_{01} + x_{02} \leq x_0$;

MIP model - Constraints

● $y_i = k \Rightarrow r_i \leq W_i \wedge t_i \leq H_i$ (4)

To MIP: $k + x_{00} \leq y_i + M * x_{01}; y_i + x_{00} \leq k + M * x_{02}; x_{01} + x_{02} = x_{00};$

$l_i \leq w_k + x_{00} * M; r_i \leq w_k + x_{00} * M; t_i \leq h_k + x_{00} * M; b_i \leq h_k + x_{00} * M;$

● $p_{ik} = 1 \Rightarrow y_i = k; \text{ To MIP: } y_i = k * p_{ik}$ (5)

● $\sum_{i=1}^n p_{ik} \geq 1 \Rightarrow u_k = 1$ (6)

To MIP: $S = \sum_{i=1}^n p_{ik}$

$S \leq (1 - e) * M; S + e * M \geq 1; u_k = 1 - e;$



Heuristic

The quickness of heuristic + The completeness of backtracking

The heuristic algorithm

Sort

- **Cars** by “economic efficiency” (cost/area), in ascending order.
- **Rectangles** by area, in descending order.

Note that the **for** loops in the next step iterate through those sorted lists.

Best-fit (but sounds like first-fit)

- Initialize all cars, each contains no rectangle.
- **For** each **rectangle**, **for** each **car**, try to contain the **rectangle** in the **car**, if the **rectangle** fits, **contain** it in the **car** and **break** the car loop (to the next rectangle).

“Try to contain”

In a **car**, try to **contain** the **new rectangle** along with the **old rectangle(s)** that are **contained**.
→ Since there is no certain way to fix the places of all the rectangles, a **backtracking algorithm** must be run to check.

But why?

Sort

- The **best cars** will be at the top of the car list.
- The **largest rectangles** will be at the top of the rectangle list.

Best-fit (but sounds like first-fit)

- The **largest rectangles** will fit in some **best cars**.
- When continue to iterate through next rectangles, some **smaller one** could fit in **best cars** too.
→ Try to use the **best cars** as much as possible.

Why sort rectangles?

If we sort **rectangles**, there will be **fewer rectangles** to fit in the **best cars**, since: (some **big** + some *small* rectangles) < a lot of *small* rectangles
→ A certain way to **reduce** the overall running time of the **backtracking algorithm**.



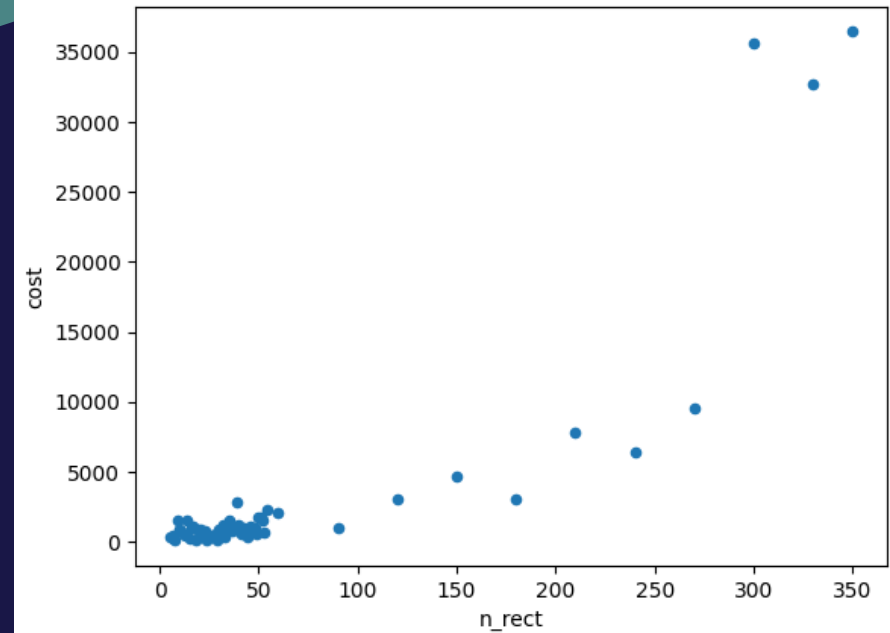
Analysis of algorithms

Practically analyze their performance

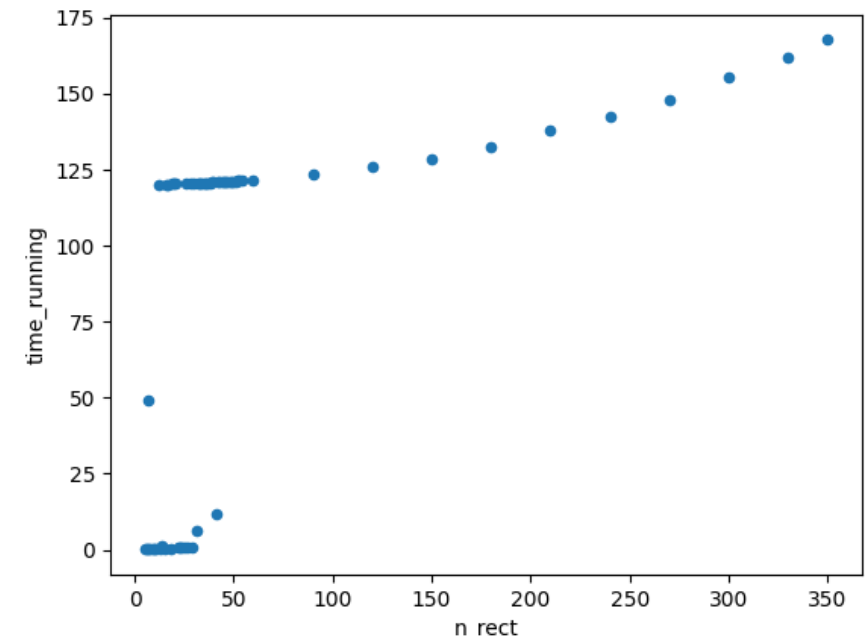
CP model

- Out of 75 tests, the model can solve 62, with a 120-second time limitation
- The objective values (costs) of solved instances are excellent in general
- About half of the solved tests are hard enough for the algorithm to reach the time limit

Cost



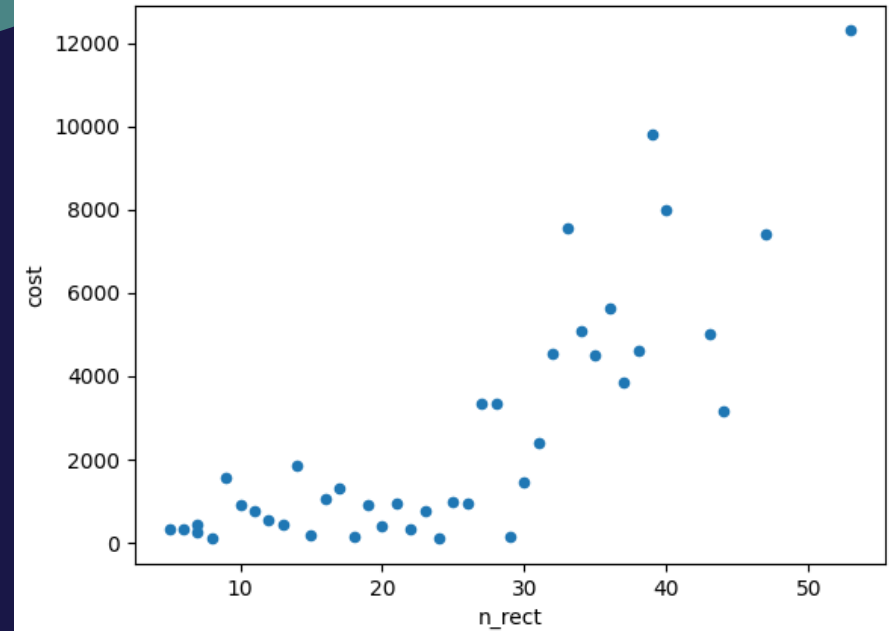
Running
time



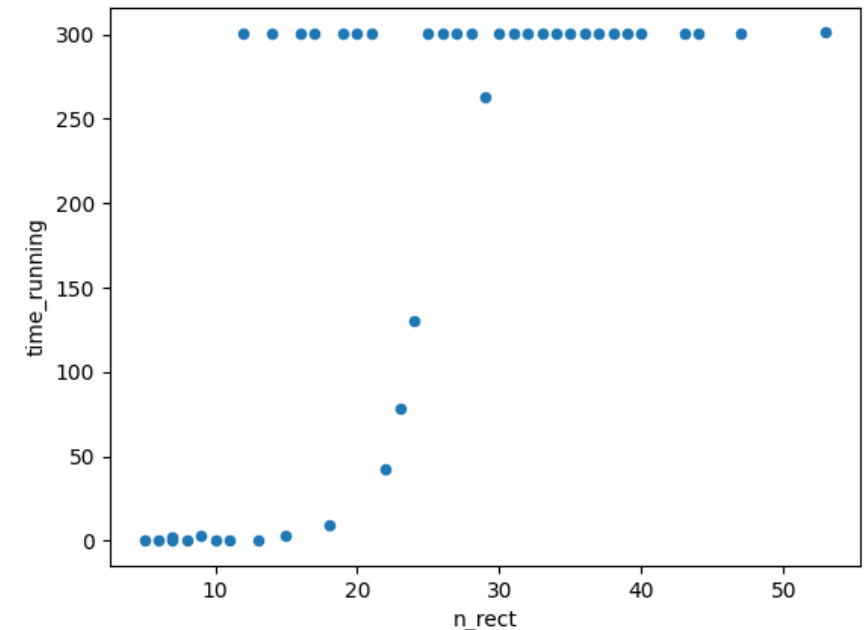
MIP model

- Out of 75 tests, the model can solve 41, with a 300-second time limitation
- The objective values (costs) of solved instances are excellent in general
- About 2/3 of the solved tests are hard enough for the algorithm to reach the time limit

Cost



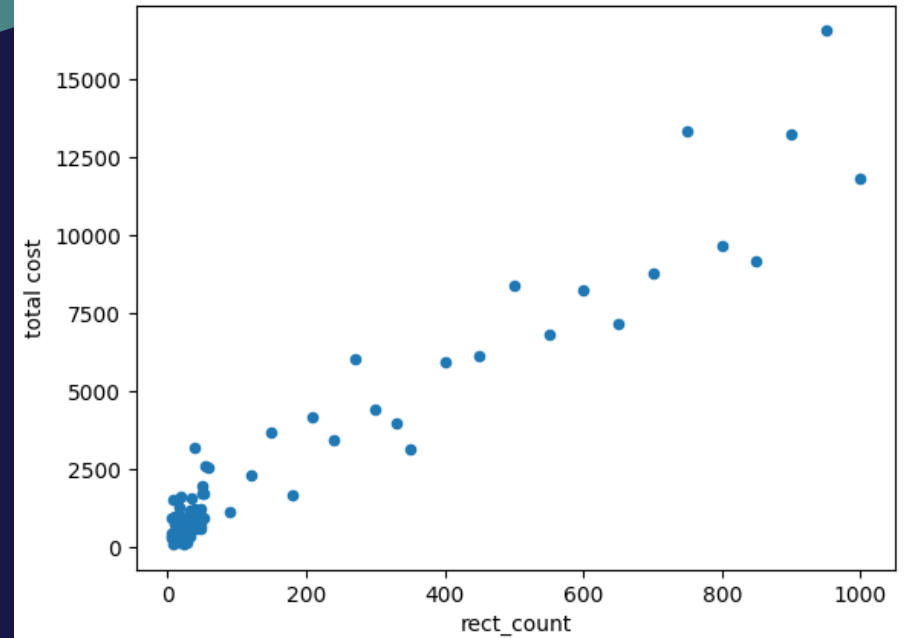
Running
time



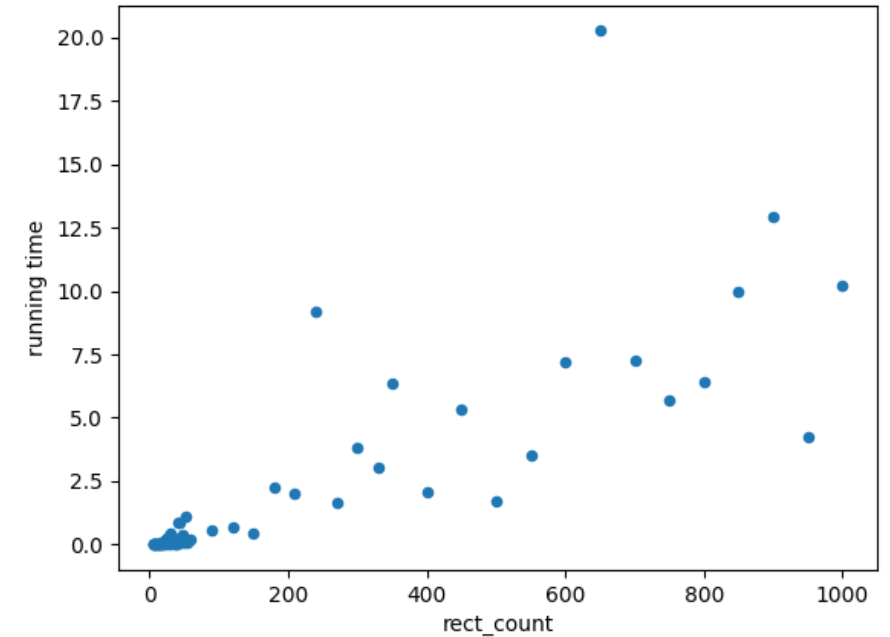
Heuristic

- It can solve **all 75** tests without a time limitation
- The objective values (costs) of instances are **good**
- The running time form an **exponential** curve with the number of rectangles (test size)
- **Fun fact!** The test sizes here are small enough to consider that the running time is **linear** without big differences.

Cost



Running
time





General 2D Bin Packing Problem

You can find interesting **figures**, data **analysis**, and even **things that we tried** to do in
<https://github.com/htnminh/optimization-project>

Thanks for your attention!

