

Report for Project Work
Combinatorial Decision Making and Optimization
Module 1
(SMT Solution)

Yuwei Ke (yuwei.ke@studio.unibo.it)

Yiran Zeng (yiran.zeng@studio.unibo.it)

Contents

1 Introduction.....	1
2 Implementation.....	2
2.1 parameters.....	2
2.1.1 Input.....	2
2.1.2 Output.....	2
3 Normal model Constraints.....	3
3.1 Implied constraint.....	3
3.2 Global constraints.....	4
3.2.1 cumulative.....	4
3.2.2 No Overlapping.....	4
3.2.3 Remove gap.....	5
3.3 Symmetry breaking constraints.....	5
4 Rotation Model.....	6
5 Search.....	6
6 Result.....	7

1 Introduction

VLSI (Very Large Scale Integration) refers to the trend of integrating circuits into silicon chips. A typical example is the smartphone. The modern trend of shrinking transistor sizes, allowing engineers to fit more and more transistors into the same area of silicon, has pushed the integration of more and more functions of cellphone circuitry into a single silicon die (i.e. plate). This enabled the modern cellphone to mature into a powerful tool that shrank from the size of a large brick-sized unit to a device small enough to comfortably carry in a pocket or purse, with a video camera, touchscreen, and other advanced features. In this project, a fixed-width plate and a list of rectangular circuits are given, all the circuits should be put on the plate and the height need to be minimized.

2 Implementation

The code is implemented in python with SMT solver. All the input and output are done in python.

2.1 parameters

2.1.1 Input

The input variables are the following ones:

- w - the width of the silicon plate
- n - the number of circuits
- [width₁, width₂, ..., width_n] - the width of every circuit
- [height₁, height₂, ..., height_n] - the height of every circuit

2.1.2 Output

2.1.2.1 Normal model output

Once the program finished optimization process and find solution, it will give output as following:

```
w h
n
Width1 height1 x1 y1
Width2 height2 x2 y2
.....
Widthn heightn xn yn
```

Among these parameters:

- h is the minimum height the program needs to optimize.
- x_i and y_i correspond to the left bottom point's x-coordinate and y-coordinate of the i circuit.
- w, n, width_i, height_i are same as the input.

And it will generate a picture as Figure 1:

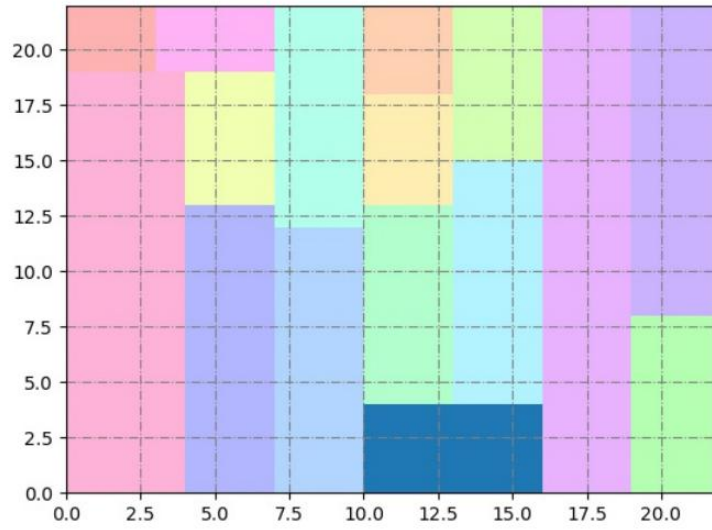


Figure 1:instance 15 with width=22,n=15 and height=22

2.1.2.2 Rotation model output

Output format is as following:

```

w h
n
Width1 height1 x1 y1 Rotation1
Width2 height2 x2 y2 Rotation2
.....
Widthn heightn xn yn Rotation2

```

- Rotation is boolean variable. When it is true, it means the circuit has been rotated. On the contrary, false means it is not rotated.
- Other variables' meaning is same as 2.1.2.1
- It will also generate a picture like Figure 1.

3 Normal model Constraints

3.1 Implied constraint

When taking in to account this problem, we can not ignore the implied constraint of the problem itself. As the plate's width is fixed, all the circuits inside could not exceed the left, right and bottom boundary of the plate. In the solution, the left-bottom corner of plate is the coordinate's origin, and bottom edge is X axis and left edge is Y axis. So there should have formulae:

$$\begin{aligned}
 0 < x_i < w \quad (0 < i \leq n) \\
 x_i + \text{width}_i &\leq w \quad (0 < i \leq n) \\
 y_i + \text{height}_i &\leq h \quad (0 < i < n)
 \end{aligned}$$

3.2 Global constraints

3.2.1 cumulative

In this strip packing problem, it can be treated as resource scheduling problem. Each circuit can be viewed as a task, the horizontal axis of the plate corresponds to the time, the vertical axis corresponds to the capacity, the start time and duration of each task can be viewed as the horizontal axis and width of the circuit, while the amount of resources required corresponds to the height. On the other hand, when the duration be represented by vertical axis and the amount of resource is showed by horizontal one, that is also reasonable. The function `cumulative()` has been realized :

```
def cumulative(start, duration, requirement, capacity):
    c = []
    for u in requirement:
        c.append(
            sum([If(And(start[i] <= u, u < start[i] + duration[i]), requirement[i], 0)
                for i in range(len(start))]) <= capacity
        )
    return c
```

So there have constraints like below:

`cumulative(x, width, height, h)`
`cumulative(y, height, width, w)`

3.2.2 No Overlapping

Circuit cannot overlap with other circuits, so some restrictions are required on the code. That constraint basically states that circuit *i* needs to be either to the left, to the right, below or above circuit *j*.

For each distinct pair of circuit *i* and *j*:

$$x_i + width_i \leq x_j$$

$$x_j + width_j \leq x_i$$

$$x_i + height_i \leq x_j$$

$$x_j + height_j \leq x_i$$

3.2.3 Remove gap

To make sure there is no gap between circuits which means that each circuit is complete fit on another piece of circuit so we also need to add some other constraint.

$$(x_i = 0 \vee x_i = x_j + \text{width}_j)$$

$$\wedge$$

$$(y_i = 0 \vee y_i = y_j + \text{height}_j)$$

3.3 Symmetry breaking constraints

Symmetry may lead to a solution/failure which will have many symmetrically equivalent states. For example, exchange circuits with same width and height; flip the plate horizontally or vertically; rotate the plate 180°. In order to get higher efficiency, we need to reduce the solution and search space by defining some symmetry breaking constraints:

- Sort all circuits by area from largest to smallest, and let the one with the largest area in the lower part of the second largest circuit. In this way, the circumstances including horizontal and vertical flip as well as 180° rotation are all ruled out.

`less_eq([y[maxcircuit],x[maxcircuit]], [y[secondMaxcircuit],x[secondMaxcircuit]])`

- To limit the switching of circuits with same width and height, If the width and height of circuit i and circuit j are the same then if their y's are equal then x_j has to be greater than x_i and if they aren't then y_j has to be greater than y_i. For every pair of circuits:

```
for i in range(n):
    r = []
    for j in range(n):
        if width[i]==width[j]:
            if height[i]==height[j]:
                r.append(i)
    if len(r)>1:
        if minH(r)==i:
            for k in range(len(r)):
                if k>0:
                    symmetry_breaking_same.append(less_eq([y[r[k-1]],x[r[k-1]]],[
y[r[k]],x[r[k]]]))
```

4 Rotation Model

In this model, the only difference with the normal model is that the circuit's width and height be exchanged.

So an array of boolean variable is added which named Rotation. If the rotation[i] is true then it means the i^{th} circuit has been rotated and its width has been swapped with height.

There will be a new implied limitation in this model: if a circuit's height is larger than the plate's width, then it can not be rotated. So in conclusion they can be written as below:

```
width_r = [If(And(rotation[i], width[i] != height[i], height[i] <= w), height[i], width[i])
for i in range(n)]
height_r = [If(And(rotation[i], width[i] != height[i], height[i] <= w), width[i], height[i])
for i in range(n)]
```

5 Search

According to the z3py-tutorial, Optimize() can be used to find the optimal solution for the problem. After adding the constraints described before, the method opt.minimize() can help us to minimize the plate's height.

6 Result

During the python program's operation,if a instance takes more than 5 minutes to find the best solution,the program will automatically stop and run the next instance.

From the chart below,we can find that normal model get a better result than the rotation model.And starting from 11th instance,the rotation model doesn't work well.

