Combinatorial and Decision Making Project Report (SAT/SMT model)

Alberico Arcangelo

July 2020

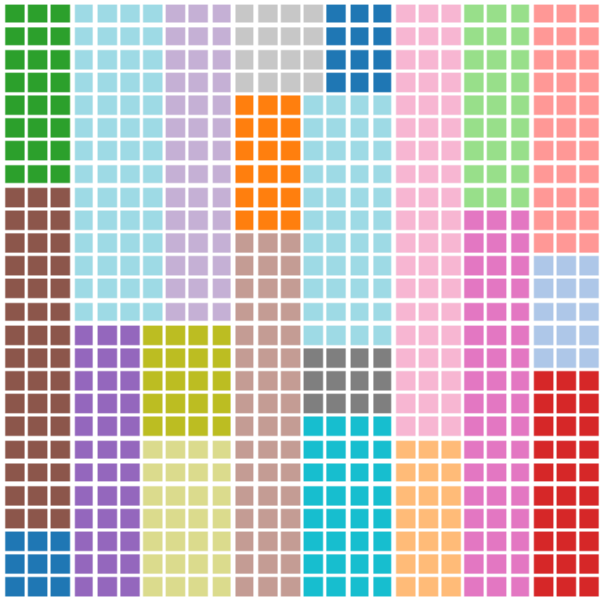

Figura 1: A solution for 26x26 instance

1 Model

The problem of PWP can be rephrased in this way: given a strip of width W and height H, given a set of rectangles $R_1, \ldots, R_n$ with widths w_i and heights h_i, find a way (if exists) to pack these rectangles in the strip without overlapping the rectangles. The variables of the problem are the coordinates of the left-bottom corner of each rectangle in the strip.
After declaring the variables, we can start to write the constraint of our problem. I focused on three classes of constrains: domain, non-overlapping and implied.
Regarding the **domain constraints**, we have to take into account the fact that the rectangles must stay in the strip. So, given the pair of coordinate of the left-bottom corner (x_i, y_i) of the rectangle R_i, we have that:

$$0 \leq x_i \leq W - w_i$$

$$0 \leq y_i \leq H - h_i$$

For the **non-overlapping constraints** we can consider two rectangles overlapping if their projections on the axis have a non empty intersection. So, in order to avoid this we have to impose that for each pair of rectangles R_i and R_j with $1 \leq i \leq j \leq n$:

$$(x_i + w_i \leq x_j) \wedge (x_j + w_j \leq x_i) \wedge (y_i + h_i \leq y_j) \wedge (y_j + h_j \leq y_i)$$

These constraints can be easily translated in the STM-LIB2 syntax. An example of an 8 dimension PWP problem translation can be found in the file `smt8x8.txt` in this folder.
Concerning the **implied constraints**, if we draw an horizontal or a vertical line, the sum of the traversed pieces can be at most W in the horizontal case and H in the vertical one. So, we have to impose that for each $0 \leq i \leq h$:

$$\sum_{\substack{j=0 \\ (y_j \leq i) \wedge (y_j + h_j > i)}}^{n} w_j \leq W$$

for the horizontal case and that for each $0 \leq i \leq w$:

$$\sum_{\substack{j=0 \\ (x_j \leq i) \wedge (x_j + w_j > i)}}^{n} h_j \leq H$$

for the vertical case.
In order to generalize the problem for any instance I used `Z3Py`, which is a Python API for the Z3 theorem prover. This allowed me to extend and automatize the process of creating constraints to any instance provided as input.

2 Results

All the instances have been successfully solved. Most of all took a very short time in the order of seconds while others took some minutes. In the following table there is a summary of each instance with the respective execution time.

Instance	Time (seconds)
8	0.040
9	0.033
10	0.045
11	0.050
12	0.164
13	0.207
14	0.087
15	0.097
16	0.151
17	0.158
18	3.210
19	0.422
20	0.332
21	1.819
22	0.423
23	37.450
24	5.896

Instance	Time (seconds)
25	18.126
26	287.457
27	29.118
28	62.023
29	108.214
30	19.203
31	27.136
32	1109.084
33	2.147
34	31.875
35	32.003
36	52.822
37	3406.093
38	25.577
39	NaN
40	13.852

In red I highlighted the instances which took more than 10 minutes and the NaN means that the instance cannot be solved in less than 1 hour. When I tried to solve them without the implied constraints the things improved dramatically bringing to these results for the most difficult instances:

Instance	Time (seconds)
32	136.190
37	261.932
39	1479.960

3 Generalization

The generalized problem provides the rotation of the rectangles and the handling of pieces of same dimension.
In order to deal with rotation I added a boolean array of variables which indicates if an element is rotated or not and then I added some `if` statement to take into account if each rectangle was rotated or not. I found the SAT/SMT model more natural and slightly easier than the CP model to be modified in order to account the rotation.
Then, regarding rectangles with the same dimension, I forced a relation between the two positions, that is to say if there are two rectangles R_i, R_j with $w_i = w_j$ and $h_i = h_j$ then:

$$x_j \geq x_i + w_i \vee y_j \geq y_i + h_i$$

To check the correctness of this changes, I provided two modified instances in the same folder: one is an 11x11 instance which would be infeasible without rotation and the other is a 12x12 instance with many pieces of equal dimension.