Project Report

Combinatorial Decision Making and Optimization (SMT)

by Hanying Zhang, Nour Bou-Nasr

March, 2021

1 Proposed mainlines

1.1 The variables and the main problem constraints

We first read in *W, H, N, Ws* and *Hs* respectivaly as the width of the whole paper, the height of the whole paper, the number of the small pieces, the widths of the small pieces and the heights of the small pieces. We then created two variables *Xs* and *Ys*, which represent the coordinates of the left-bottom corner of all the small pieces.

$$X_s = [\mathrm{Int}(''X\%i''\ \%\ i)\,\mathrm{for}\,i\,\mathrm{in}\,\mathrm{range}(N)]$$

$$Y_s = [\mathrm{Int}(''Y\%i''\ \%\ i)\,\mathrm{for}\,i\,\mathrm{in}\,\mathrm{range}(N)]$$

1.1.1 The small pieces should not exceed the border of the whole paper

We used *And* together with list comprehension to constraint the value range of *Xs* and *Ys*. We used another list comprehension to make sure that the right borders of all the rectangles don't exceed the right border of the whole paper, also the upper borders of the rectangles don't exceed the upper border of the whole paper.

$$[\mathrm{And}(0 \leqslant X_s[i], X_s[i] < W)\,\mathrm{for}\,i\,\mathrm{in}\,\mathrm{range}(N)]$$

$$[\mathrm{And}(0 \leqslant Y_s[i], Y_s[i] < H)\,\mathrm{for}\,i\,\mathrm{in}\,\mathrm{range}(N)]$$

$$[(X_s[i] + W_s[i] \leqslant W)\,\mathrm{for}\,i\,\mathrm{in}\,\mathrm{range}(N)]$$

$$[(Y_s[i] + H_s[i] \leqslant H)\,\mathrm{for}\,i\,\mathrm{in}\,\mathrm{range}(N)]$$

1.1.2 There should be no overlap between all the small pieces

We used *Or* and list comprehension to make sure that all the small pieces don't overlap. The hint here is that if two small pieces of paper don't overlap horizontally **OR** vertically, then they don't overlap.

$$[\mathrm{Or}(X_s[i] \geqslant X_s[j] + W_s[j], X_s[i] + W_s[i] \leqslant X_s[j], Y_s[i] \geqslant Y_s[j] + H_s[j], Y_s[i] + H_s[i] \leqslant Y_s[j])\mathrm{for}\,i\,\mathrm{in}\,\mathrm{range}(N)\,\mathrm{for}\,j\,\mathrm{in}\,\mathrm{range}(i+1, N)]$$

1.2 Implied constraints

Take the vertical line for instance, the total heights of all the traversed pieces should not be bigger than H, and we should check all the vertical lines. For the SMT problem we use *sum* and list comprehension for these two requirements respectively.

$$[\mathrm{Sum}([\mathrm{If}(\mathrm{And}(X_s[i] \leqslant x, x < X_s[i] + W_s[i]), H_s[i], 0)\mathrm{for}\,i\,\mathrm{in}\,\mathrm{range}(N)]) \leqslant H\ \mathrm{for}\,x\,\mathrm{in}\,\mathrm{range}(W)]$$

$$[\mathrm{Sum}([\mathrm{If}(\mathrm{And}(Y_s[j] \leqslant y, y < Y_s[i] + H_s[j]), W_s[j], 0)\mathrm{for}\,j\,\mathrm{in}\,\mathrm{range}(N)]) \leqslant W\ \mathrm{for}\,y\,\mathrm{in}\,\mathrm{range}(H)]$$

1.3 Global constraints

We don't use any global constraints.

1.4 The best way of searching

We reversed the lines of dimensions of each instance input file as what we did in the CP model (which means that the widest rectangles are palced at the beginning). We also tried to place the constriants about *Xs* in front of those about *Ys*, but that helped little. The model could solve most of the instances in two minutes expect for two instances (37x37 and 39x39). At last we got rid of the two implied constraints and the model could gave all the results (the time consumed was about 1 minute for 37x37 and 6 minutes for 39x39). That makes sense because the two implied constraints are nested *list comprehension*, essentially it's a nested *for* loop even though in *Python* list comprehension runs faster than a naive *for* loop.

1.5 Rotation

We need to set a new bool variable O_i to define if piece i is rotated. When O_i is true, the width and height of piece i is exchanged.

1.6 Multiple pieces of the same dimension

If there are multiple pieces with the same dimension, they should be placed together with each other to form a bigger piece. The problem would be easier as the number of small pieces are smaller and we are just forming new rectangles. But we should also consider that they couldn't exceed the borders of the whole sheet of paper, also if they can't perfectly fit the width or length of the whole paper, we should consider leaving enough space for other small rectangels. We should also consider the direction they grow, for example, two pieces of size 3x4, they could form a bigger piece with size 6x4 or 3x8.

2 Other remarks

2.1 Auxilliary contraints

2.1.1 The small pieces whose heights *Hs[i]* > *0.5 * H* could only be placed horizontally

If the total heights of 2 small pieces are bigger than half the height of the whole sheet, then they could not be placed vertically, which means that the x coordinates are different. Thus we could use global constrint *alldifferent* upon them. More strictly, suppose their width are w_i and w_j respectively, and $w_i < w_j$, then the minimum distance between them should be w_i.

$$[(X_s[j] - X_s[i] \geqslant \min(W_s[i], W_s[j])) \text{ for } i \text{ in range}(N) \text{ for } j \text{ in range}(i+1, N) \text{ if } (H_s[i] > H//2 \text{ and } H_s[j] > H//2)]$$

2.1.2 Encouraging two suitable rectangles to form a whole column

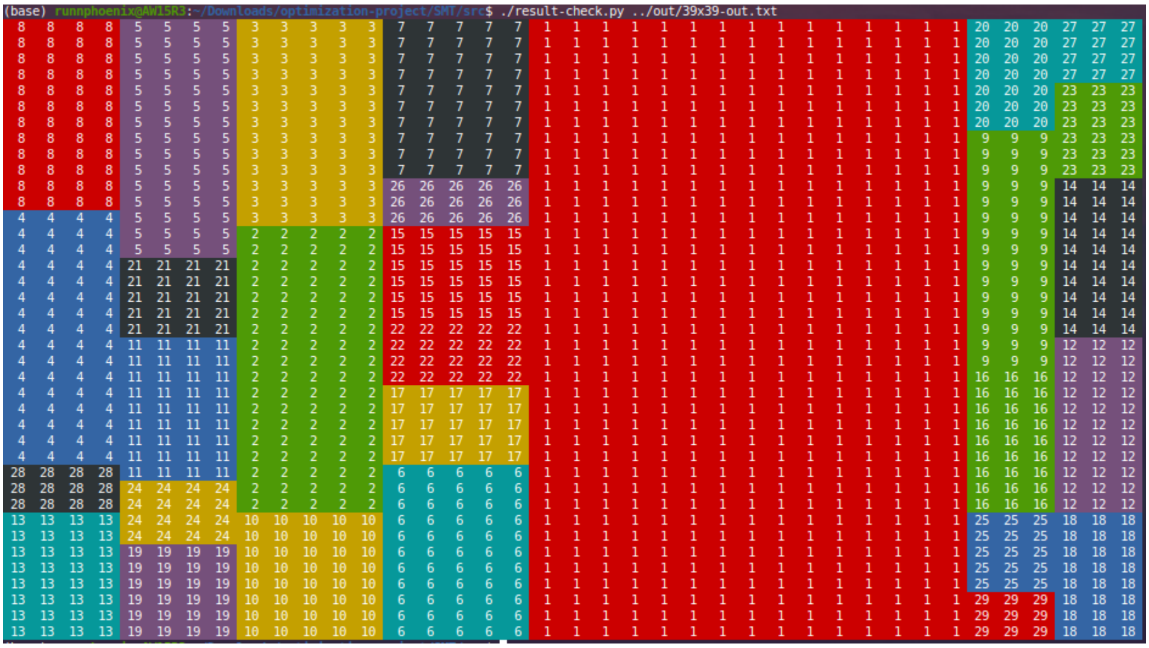
The main idea behind this is that by forming whole columns, to some sense we are actually decresing the size and complexity of the problem. Specifically, we are transforming a problem with size $w_1 \times h$ into a problem $w_2 \times h$, where $w_2 < w_1$. Suppose we have two rectangles with the same width, the height of one rectangle is bigger than 0.5 * H, and the sum of their heights is H, then we try to make them a column.

The problem with this constraint is the new and less complexity problem also has a smaller solution space. It may result in a UNSATISFIABLE situation. In our model we loosed this constraint by also allowing the two rectangles to be placed side by size horizontally. For this project it didn't cause unsatisfication but strictly speaking, this is not a robust constraint.

$[\mathrm{Or}(X_s[j] - X_s[i] == W_s[i],\ X_s[j] == X_s[i])$ for i in range(N) for j in range$(i+1,\ N)$ if$(H_s[i] > H\,//2$ and $H_s[i] + H_s[j] == H$ and $W_s[i] == W_s[j])]$

2.2 Result verification

We wrote a python script file named *result_ check.py* for checkig whether the result we got is right. This file run in terminal and would give a colored representation of the whole sheet. One of the result picture is as following:



The problem with this script file is that some adjacent rectangles are with the same color, as we used only 7 different colors.

3 References

[1] Helmut Simonis and Barry O'Sullivan. Using Global Constraint for Rectangle Packing.