Project Report

Combinatorial Decision Making and Optimization (CP)

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 decision variables Xs and Ys, which represent the coordinates of the left-bottom corner of all the small pieces.

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

We used *forall* in Minizinc to make sure that the right borders of all the small pieces don't exceed the right border of the whole paper, also the upper borders of all the small pieces don't exceed the upper border of the whole paper.

1.1.2 There should be no overlap between all the small pieces

We used *forall* function 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.

We also used *assert* to make sure that the total size of all the small pieces are not bigger than the size of the whole paper.

1.1.3 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.

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 CP problem we use *sum* and *forall* for these two requirements respectively.

1.3 Global constraints

1.3.1 Main constraints

We used *diffn* in Minizinc to make sure all small pieces don't overlap.

1.3.2 Implied constraints

For the global implied constrants, we use *cumulative* constraint in Minizinc. This contraint is originally used for scheduling problems, but it's suitable here. Specifically, we take the left coner coordinate of a small piece as the starting time of a task, the width or height as the duration of the task, the height or width as the resource requirement, and W or H as the capacity.

1.4 The best way of searching

At the beginning we tried several combinations of searching techniques. For variable selection, we tried *dom_ w_ deg* and *first_fail*, for domain selection we tried *indomain_ min* and *indomin_ random*. We also tried *restart* when using *indomin_ random*. The most robust combination we found is *dom_ w_ deg* and *indomain_ random* with *restart*. But there were still about 5 instances which we could not found a solution within 5 minutes.

We then tried another method which was that we placed the bigger pieces before smaller ones. When converting *.txt* input files into *.dzn* input files, we sorted all the dimension lines by their sizes, the bigger the size, the more front the position. We then did the search using *input_ order* and *indomain_ min*, which means that we tried to place the bigger pieces first, and we tried to place them close to the left bottom corner of the whole sheet. In this way we also solved the symmetry problem. The result was better, there's only 1 instance which couldn't be solved within 5 minutes(37x37).

TODO: a plot showing running times

At first we searched Xs and Ys together (Xs + +Ys), then we switched to use *seq_ search*, which means thant we would try to fix *Xs* first, then *Ys* of all the small pieces. The results showed that the second method is better than the first one.

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.

TODO: which one of CP and SMT are easier to implement rotation?

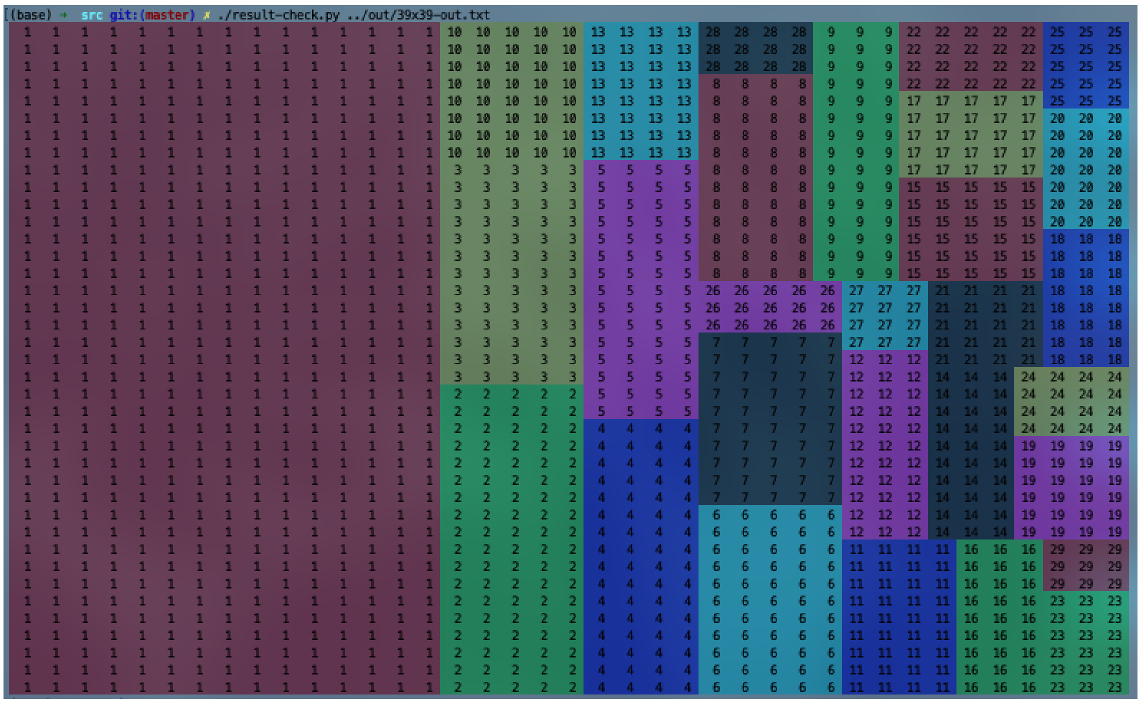
1.6 Multiple pieces of the same dimension

If there are multiple pieces with the same dimension, they should be placed together with each other to form a bigger piece. TODO: the reason…

2 Other remarks

We wrote a python script file named result_check.py for checkig whether the result we got is right.

This file is run in terminal and would give a colored representation of the whole sheet. One of the result picture is as follows:



```
[(base) ➜  src git:(master) ✗ ./result-check.py ../out/39x39-out.txt
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 10 10 10 10 10 13 13 13 13 28 28 28 28  9  9  9 22 22 22 22 22 25 25 25
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 10 10 10 10 10 13 13 13 13 28 28 28 28  9  9  9 22 22 22 22 22 25 25 25
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 10 10 10 10 10 13 13 13 13 28 28 28 28  9  9  9 22 22 22 22 22 25 25 25
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 10 10 10 10 10 13 13 13 13  8  8  8  8  9  9  9 22 22 22 22 22 25 25 25
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 10 10 10 10 10 13 13 13 13  8  8  8  8  9  9  9 17 17 17 17 17 25 25 25
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 10 10 10 10 10 13 13 13 13  8  8  8  8  9  9  9 17 17 17 17 17 20 20 20
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 10 10 10 10 10 13 13 13 13  8  8  8  8  9  9  9 17 17 17 17 17 20 20 20
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 10 10 10 10 10 13 13 13 13  8  8  8  8  9  9  9 17 17 17 17 17 20 20 20
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5  8  8  8  8  9  9  9 17 17 17 17 17 20 20 20
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5  8  8  8  8  9  9  9 15 15 15 15 15 20 20 20
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5  8  8  8  8  9  9  9 15 15 15 15 15 20 20 20
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5  8  8  8  8  9  9  9 15 15 15 15 15 20 20 20
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5  8  8  8  8  9  9  9 15 15 15 15 15 18 18 18
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5  8  8  8  8  9  9  9 15 15 15 15 15 18 18 18
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5  8  8  8  8  9  9  9 15 15 15 15 15 18 18 18
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5 26 26 26 26 26 27 27 27 21 21 21 21 18 18 18
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5 26 26 26 26 26 27 27 27 21 21 21 21 18 18 18
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5 26 26 26 26 26 27 27 27 21 21 21 21 18 18 18
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5  7  7  7  7  7 27 27 27 21 21 21 21 18 18 18
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5  7  7  7  7  7 12 12 12 21 21 21 21 18 18 18
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  3  3  3  3  3  5  5  5  5  7  7  7  7  7 12 12 12 14 14 14 24 24 24 24
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  5  5  5  5  7  7  7  7  7 12 12 12 14 14 14 24 24 24 24
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  5  5  5  5  7  7  7  7  7 12 12 12 14 14 14 24 24 24 24
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  7  7  7  7  7 12 12 12 14 14 14 24 24 24 24
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  7  7  7  7  7 12 12 12 14 14 14 19 19 19 19
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  7  7  7  7  7 12 12 12 14 14 14 19 19 19 19
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  7  7  7  7  7 12 12 12 14 14 14 19 19 19 19
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  7  7  7  7  7 12 12 12 14 14 14 19 19 19 19
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 12 12 12 14 14 14 19 19 19 19
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 12 12 12 14 14 14 19 19 19 19
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 11 11 11 11 16 16 16 29 29 29
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 11 11 11 11 16 16 16 29 29 29
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 11 11 11 11 16 16 16 29 29 29
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 11 11 11 11 16 16 16 23 23 23
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 11 11 11 11 16 16 16 23 23 23
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 11 11 11 11 16 16 16 23 23 23
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 11 11 11 11 16 16 16 23 23 23
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 11 11 11 11 16 16 16 23 23 23
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  4  4  4  4  6  6  6  6  6 11 11 11 11 16 16 16 23 23 23
```

There's still a problem with this script file which is that some adjacent rectangles are with the same color, as we use only 7 different colors(maybe we should use CP again to solve this problem :p). We can also see from this picture that our model is actually working, basiclly speaking, the bigger the small pieces, the closer they are to the left bottom corner of the whole sheet.