Modeling perfect square placement in the Constraint Programming Framework

1.Context

Squaring the square is the problem of tilling an integral square using only other integral squares. A perfect squared square of order n is a square dissected into a finite number n of squares, no two of which are of equal size. The square division is considered to be perfect if the squares are all different sizes. They are considered rare compared to perfect simple squared rectangles of the same order. According to David Gale, "there are about 5,000,000 perfect simple squared rectangles to every such squared square (for order greater than 20)".

2.Problem

The problem aims to optimize the placement of a set of squares with given sizes into a bigger square. The constraints for the problem are:

- the squares placed inside the bigger square must not overlap each other
- all square borders are parallel to the border of the big square

The sum of the square surfaces is equal to the surface of the packing square so that there is no spare capacity.

3.Modeling

The input for our problem will be:

- the size of master square
- the list of sizes of the squares that need to be packed within the bigger one

The problem can be modeled as a constraint satisfaction problem. Our CSP model is defined by:

- a set of variables: $Pi\,(Xi, Y\,i)$ as the placement of the square i
- a set of constraints:

C1: the coordinates for each square should be within the domain

$$Xi \geq 0,\ Y\,i \geq 0\ ,\ \forall\,i$$

C2: the squares must be parallel with the axes

$$Xi \leq masterSquareSize - Size\,i,\ Y\,i \leq masterSquareSize - Size\,i\ ,\ \forall\,i$$

C3: the packed squares must not overlap each other

4. Results

We tried implementing in two ways the constraints in CPLEX:

- Writing the geometric constraints of two rectangles, squares in our case to not collide. We introduced this constraint for every two rectangles

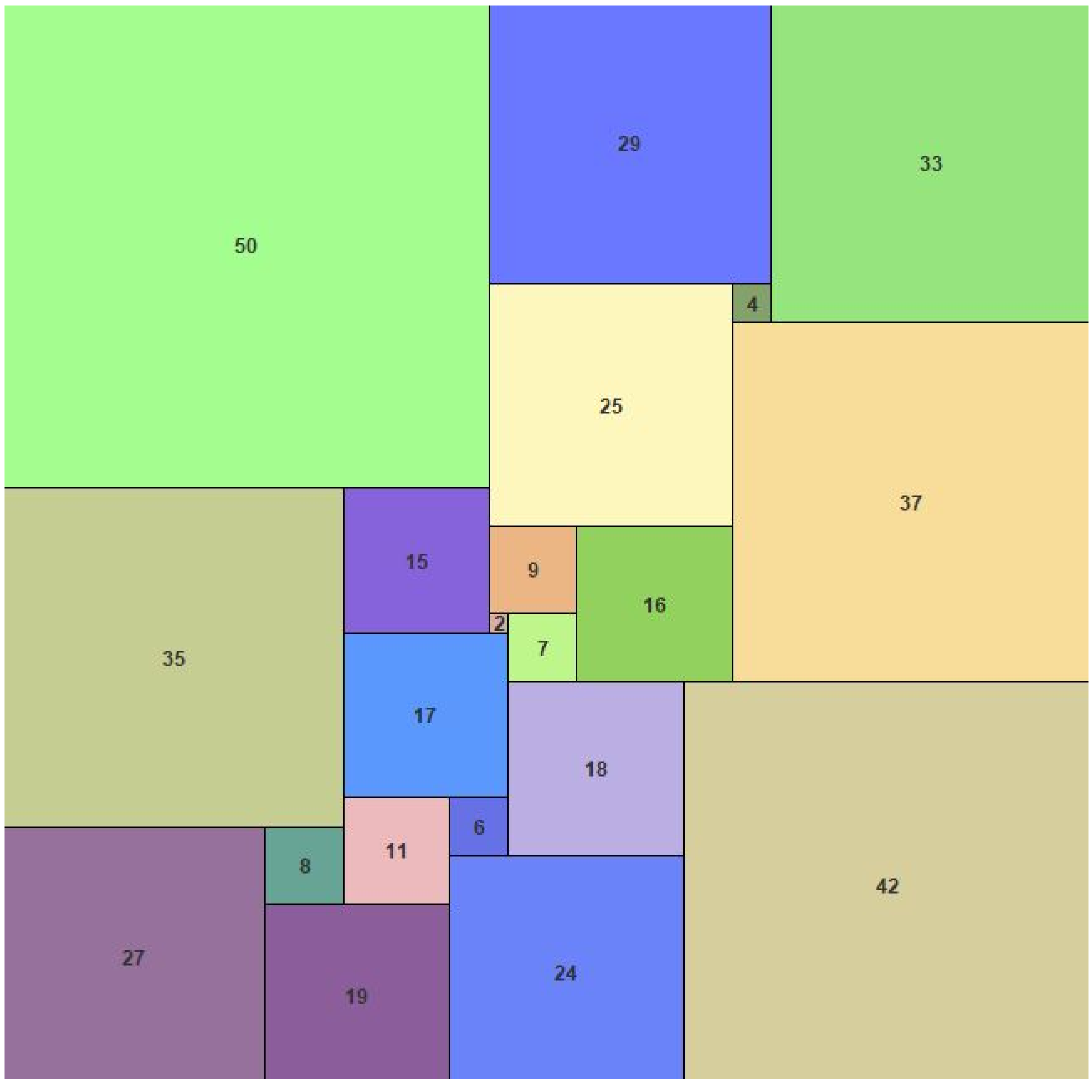
$$Xi \geq Xj + Size\,j \,\|\, X\,i + Size\,i \leq Xj \,\|\, Y\,i + Size\,i \geq Y\,j \,\|\, Y\,i \leq Y\,j + Size\,j\ ,\ \forall i,\ j,\ i > j$$

- Rewriting the same constraints but this time using the max and min operator as proposed here.

$$max(Xi,\ Xj) - min(Xi + Sizei,\ Xj + Sizej) \geq 0 \,\|\, max(Y\,i, Y\,j) - min(Y\,i + Sizei, Y\,j + Sizej) \geq 0,\ \ \forall\, i,j,\ i > j$$

Unfortunately, both sets of constraints introduced too many slack variables for the free version to cope with.

We then had a look at Jacop and Choco, two open-source CSP frameworks that have a Java interface. Here, we found already implemented *diffn* and *cumulative* constraints that are used for solving rectangle packing problems. For all the test instances the CSP model found the solution almost instantly.


50
29
33
4
25
37
35
15
9
16
2
7
17
18
42
6
11
8
27
19
24