Marco Buia
Daniele Ve

Present Wrapping Problem SMT MODEL

Abstract

The Present Wrapping problem can be viewed as finding the position of rectangular pieces order to fit them into the available rectangular paper shape.
Moreover, the presents cannot be rotated nor fall outside of the bounding paper, not even partially.

The matematical model we use with the SMT solver is the same as CP, indeed without glob constraints.
Altought SMT doens't provide a way to guide the search for a solution, Microsoft Z3 solver managed to handle even the most difficult instances in a reasonable time, on the same machine.

Decision variables

The present wrapping problem provides the following input parameters:

- Paper width: W
- Paper heigth: H
- Number of presents: N
- Width and heigth of each k present, respectively $s_{k\,x}$ and $s_{k\,y}$.

We use an $N \times 2$ shaped array of integer decision variables to represent the 2D position of each present.
The bottom-left corner coordinates of a generic k present are represented by $p_{k\,x}$ and $p_{k\,y}$ variables.

Available-paper bounding constraint

The available paper shape constrains the position of present rectangles to be inside the bounding box:

$$\forall k \in N,\ 0 \ \leqslant p_{k\,x} \leqslant W - s_{k\,x}$$
$$\forall k \in N,\ 0 \ \leqslant p_{k\,y} \leqslant H - s_{k\,y}$$

Marco Buia
Daniele Ve

Non-overlapping constraint

The non-overlapping constraint is implemented for each pair of distinct presents.

$$\forall k_1, k_2 \in N,$$
$$p_{k_1\,x} + s_{k_1\,x} \leqslant p_{k_2\,x} \vee \; p_{k_1\,x} \geqslant p_{k_2\,x} + s_{k_2\,x} \vee$$
$$p_{k_1\,y} + s_{k_1\,y} \leqslant p_{k_2\,y} \; \vee \; p_{k_1\,y} \geqslant p_{k_2\,y} + s_{k_2\,y}$$

Partial sums implied contraint

At each moment during search, no column nor row can be assigned to more than its total capacity.
We add the suggested partial sums implied constraint for each row and column.

If the present k occupies the row r, **then** $row_{k\,r}$ value is the heigth of the present, **else** 0.
If the present k occupies the column c, **then** $col_{k\,c}$ value is the width of the present, **else** 0.
Expressed in First Order Logic:

$$(p_{k\,x} \; \leqslant \; r \; \wedge \; p_{k\,x} + s_{k\,x} > r \; \Longrightarrow row_{k\,r} = \; s_{k\,y}) \; \wedge \; (p_{k\,x} \; > \; r \; \vee \; p_{k\,x} + s_{k\,x} \leqslant r \; \Longrightarrow row$$
$$(p_{k\,y} \; \leqslant \; c \; \wedge \; p_{k\,y} + s_{k\,y} > c \; \Longrightarrow col_{k\,c} = \; s_{k\,x}) \; \wedge \; (p_{k\,y} \; > \; c \; \vee \; p_{k\,y} + s_{k\,y} \leqslant c \; \Longrightarrow col_{k\,c}$$

Finally we constrain the sum over each row and column to be equal or lower than the available paper size, width W and heigth H respectively:

$$\forall r \in W, \; \sum_{k}^{N} row_{k\,r} \leqslant H \qquad\qquad \forall c \in H, \; \sum_{k}^{N} col_{k\,c} \leqslant W$$

We compare the model with and without this constraint observing the following:
the solving time increases by up to 200% when the constraint is enabled; this is due to the

high number of added `clauses`.
Our idea is that the implied constraint is already taken into account by the ILP solver, resulting in higher overhead.

Marco Buia
Daniele Ve

Shape rotation variant

In this problems variant we allow rectangular presents to rotate by 90 degrees steps.
To handle this we employ one array of boolean variables, rot, to represent the rotation state of each piece.
We then use $rotshape_{k\,x}, rotshape_{k\,y}$ in place of present shapes $s_{k\,x}, s_{k\,y}$, as an interface to take into account the possible rotation of the presents:

$$(rot_k \implies rotshape_{k\,x} = s_{k\,y}) \ \wedge (\neg rot_k \implies rotshape_{k\,x} = s_{k\,x})$$

$$(rot_k \implies rotshape_{k\,y} = s_{k\,x}) \ \wedge (\neg rot_k \implies rotshape_{k\,y} = s_{k\,y})$$

We prove the correctness of this approach by creating a suitable instance `8x8_rot.txt` th can be solved only if allowing presents to rotate passing the parameter `--allow_rotatior`

Z3 statistics showing the impact of rotations in the 20x20 instance:

- rotations: OFF

  ```
  solved in: 1.107670545578003
  conflicts 3546
  decisions 8507
  propagations 2081392
  binary propagations 1666075
  restarts 28
  ```

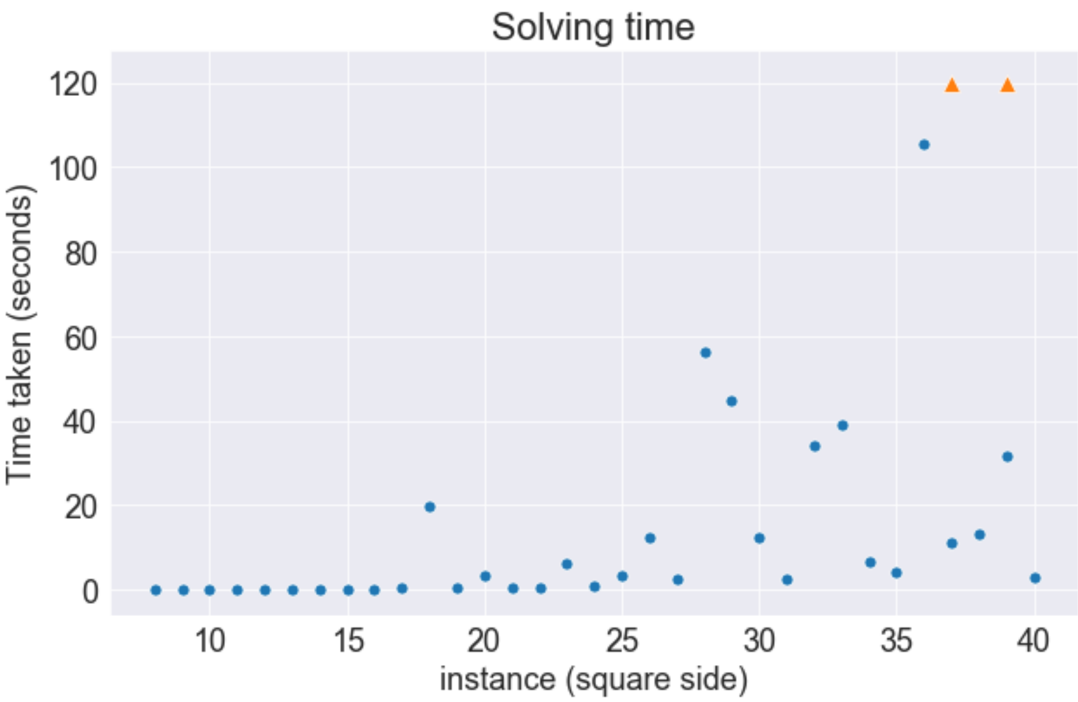
- rotations: ON

```
solved in: 10.154843807220459
conflicts 8503
decisions 24095
propagations 4637732
binary propagations 2269707
restarts 9
```

Results

We managed to solve all the proposed instances.
The figure represent the elapsed time for each instance with rotations and implied constrair disabled.
Tests are performed with a i7 4th gen CPU using 4 threads.

Marco Buia
Daniele Ve



▲ 37x37: 1122 s
▲ 39x39: 3162 s

Z3 statistics about the most difficult instance (39x39) are left:

DPLL solver statistics

```
conflicts 5436
decisions 30083
propagations 5086590
binary propagations 4181210
restarts 42
added eqs 1179373
mk clause 89930
del clause 31448
minimized lits 15835
```

ILP solver statistics

```
arith conflicts 1637
arith row summations 345817
arith num rows 890
arith pivots 10620
arith tableau max rows 890
arith tableau max columns 3227
```