Marco Buiani
Daniele Verì

Present Wrapping Problem SAT MODEL

Abstract

The Present Wrapping problem can be viewed as finding the position of rectangular pieces in 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 SAT encoding poses us the challenge of representing this complex problem with boolean variables only.
This report comments our coding approach using the Z3 Python API.
The program is able to solve the smaller 8x8 to 11x11 instances, while further work is needed to
optimize it for bigger ones.

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 w_k and h_k.

We use an $W \times H \times N$ 3D array of boolean decision variables corresponding to the available position for each present, each occupying one of the N layers.
Bottom-left corner positions are represented by the variable $p_{i\,j\,k}$.

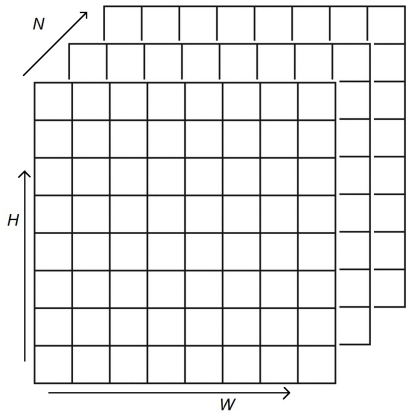

Figure 1: 3D representation of the problem, each layer holds variables for one present. There's one layer for each present.

Shape compactness

For each layer (present) **exactly** one set of variables representing the rectangular shape must
hold.
To correclty implement it, the sets of each layer are found by sliding a window in the plane, much like a 2D convolution.

Set of variables that holds, representing a present k in position $i\ j$:

$$set_{i\,j\,k} = \bigwedge_{x=i}^{i+w_k} \bigwedge_{y=j}^{j+h_k} p_{x\,y\,k}$$

In a layer k, **at most** one set must hold:
(expressed as: can't exist a pair of sets holding at the same layer)

$$atm_k = \neg \bigvee_{i_1=1}^{W-w_k} \bigvee_{j_1=1}^{H-h_k} \bigvee_{i_2=1}^{i_1} \bigvee_{j_2=1}^{j_1} (set_{i_1\,j_1\,k} \wedge set_{i_2\,j_2\,k})$$

In a layer k, **at least** one set must hold:

$$atl_k = \bigvee_{i=1}^{W-w_k} \bigvee_{j=1}^{H-h_k} set_{i\,j\,k}$$

For each layer, **exactly** one (*at least one and at most one*) set must hold:

$$\bigwedge_{k=1}^{N} (atl_k \wedge atm_k)$$

Non-overlapping constraint

For each position, **at most** one layer must hold, i.e. at most one present occupies that position:
(expressed as: can't exist a pair of layers holding at the same position)

$$atm_{i\,j} = \neg \bigvee_{k_1=1}^{N} \bigvee_{k_2=1}^{k_1} (p_{i\,j\,k_1} \wedge p_{i\,j\,k_2})$$

$$\bigwedge_{i=1}^{W} \bigwedge_{j=1}^{H} atm_{i\,j}$$

Total area occupation

In the case the paper area is equal to the total present area, we can force the constraint that for
each position, **at least** one layer must hold:

$$atl_{i\,j} = \bigvee_{k=1}^{N} p_{i\,j\,k}$$

$$\bigwedge_{i=1}^{W} \bigwedge_{j=1}^{H} atl_{i\,j}$$

This *if-then* behavioh is implemented in the python language at model creation time.

Results

We managed to solve from 8x8 instance until the 11x11 while from the 12x12 onward we got `unsat`, probably due to the high number of memory allocations required.

As a future work the model could be simplified, for example one could represent a rectangle by
the variables of its perimeter instead of its area.