Combinatorial Problem Solving - Propositional Satisfiability Project

Sergio Rodríguez Guasch

June 11, 2018

1 Variables

1. $C_{i,j,k,r}$ The kth box has its upper left corner in (i, j) and is (not) rotated, depending on r
2. $X_{i,j,k}$ (i, j) is occupied by the kth box
3. $row_{i,k}$ Is the upper left corner of the kth box in the ith row or before?
4. $col_{i,k}$ Is the upper left corner of the kth box in the ith column or before?

2 Constants

1. $H = \sum_{i=1}^{N} \max(a_i, b_i)$
2. W the given width
3. N number of boxes

3 Constraints

Our first priority is to avoid the solver wasting time on symmetric solutions. For this purpose we will do as we did in the previous deliverables: we will establish an ordering for all the sets of equivalent boxes

$$x_1 \leq x_2 \leq ... \leq x_k$$

$$x_i = x_{i+1} \implies y_{i+1} > y_i$$

In order to implement this constraint we will use the variables $row_{i,k}$ and $col_{i,k}$. First of all, we need to give these variables the appropriate meaning

$$row_{i,k} \implies row_{i-1,k} \tag{1}$$

$$col_{i,k} \implies col_{i-1,k} \tag{2}$$

$$C_{i,j,k,r} \implies row_{i,k} \wedge \neg row_{i+1,k} \tag{3}$$

$$C_{i,j,k,r} \implies col_{j,k} \wedge \neg col_{j+1,k} \tag{4}$$

With these constraints the row and the column of any box can be recovered by simply finding the first pair $row_{i,k}$ and $\neg row_{i+1,k}$ such that these two literals are true, and the same applies to $col_{i,k}$ and $\neg col_{i+1,k}$. Given two identical boxes b_1 and b_2 we can impose that b_1 goes before b_2 as follows:

$$row_{i,b_2} \wedge \neg row_{i+1,b_2} \implies \neg row_{i+1,b_1} \tag{5}$$

$$row_{i,b_2} \wedge \neg row_{i+1,b_2} \wedge row_{i,b_1} \wedge \neg row_{i+1,b_1} \wedge col_{j,b_2} \wedge \neg col_{j+1,b_2} \implies \neg col_{j+1,b_1} \tag{6}$$

Each box must have exactly one upper left corner

$$\sum_{i=0}^{H-1} \sum_{j=0}^{W-1} \sum_{r=0}^{1} C_{i,j,k,r} = 1 \quad \forall k \tag{7}$$

This constraint is implemented as the combination of an at most one constraint with the ladder encoding and an at least one constraint, which is simply a clause containing all the necessary $C_{i,j,k,r}$.
Each position has at most one box. This constraint will avoid overlaps

$$\sum_{k=1}^{n} X_{i,j,k} \leq 1 \quad \forall i,j \tag{8}$$

This constraint is also implemented with the ladder encoding.
Positions that are not appropriate (i.e out of bounds) as the upper left corner for some box should be banned

$$\neg C_{i,j,k,r} \quad \text{if} \quad \neg \text{fits}(i,j,k,r) \tag{9}$$

If some position is used as the upper left corner of some box then all the cells occupied by that box should be marked as used by it

$$C_{i,j,k,r} \implies X_{a,b,k} \quad \forall a,b : \text{inside}(i,j,k,r,a,b) \tag{10}$$

This constraint, together with the constraint from equation 8, will prevent boxes to overlap. In order to give the solver some more additional hints we have imposed two additional constraints: the first box has its upper left corner in the first quadrant, some box must have its upper left corner in $(0,0)$, and if a box is square we will assume that it is not rotated.

4 Minimizing total height

Given that SAT does not have a notion of optimization, as it is a decisional problem, we needed to perform various calls to the solver. In each call we ban

the corresponding $C_{i,j,k,r}$ that would imply that we need more than h rows by adding unit clauses that negates them. We try, from greatest to smaller, all the possible values in an interval determined by two safe bounds

$$[\frac{\sum_{i=1}^{n} a_i b_i}{W}, \sum_{i=1}^{N} \max(a_i, b_i)] \tag{11}$$

Given a solution of a problem where h rows were allowed, it can happen that the actual solution uses less than h rows. In order to speed up a little bit the search, we compute the used height u of our solution and establish the new maximum height as $u - 1$. This especially helps on instances with big boxes, as the upper bound is very large and therefore easy to improve.

5 Results

Our solver is capable to solve all the instances in a reasonable amount of time. By *solving* we mean to find a solution of height h and proving that a solution of height $h - 1$ does not exist (and therefore proving that h is the optimal height). By *reasonable amount of time* we mean 60 seconds or less. All the SAT solutions have been compared with our Integer Linear Programming solutions in order to check that they are correct.

6 Conclusions

This approach seems to be the one that works better among all the three in terms of the ratio performance vs price and availability of the tool. However, this approach is harder in terms of implementation, given the nature of propositional logic. We are also aware that this SAT implementation scales poorly with the number of boxes and the dimensions of the grid. For example, the size of the linear implementation problems did not depend on the size of the grid while the SAT one does. However, it is not clear if it is possible to avoid this complexity in this technology.