August 26, 2022

1 SAT

1.1 A simplified version of the problem

In order to model the problem with SAT, at first we consider a simplified version of the problem. Instead of minimizing the length of the plate with the given width W, we consider a fixed value H for the height and check with SAT if it is possible to insert the given circuits (without being able to rotate them) in the plate within the fixed width W. In this case, we can think of the plate as a rectangle in $\mathbb{R}^2$, with vertices $A(0,0), B(W,0), C(W,H), D(0,H)$.

We can then introduce the atomic propositional formulas x_{ijk} with the intended meaning that x_{ijk} is true if and only if the bottom-left vertex of the k-th circuit is the point (i,j). In general, if the number of circuits is K, this would require to introduce $W \cdot H \cdot K$ formulas, but in order to reduce the total number of atoms, for each circuit k with width $w(k)$ and height $h(k)$, we restrict i and j respectively to the sets $I_k = \{0, \ldots, W - w(k)\}$ and $J_k = \{0, \ldots, H - h(k)\}$. As a consequence the total number of atomic formulas to use becomes

$$\sum_{k=1}^{K} (W - w(k) + 1)\,(H - h(k) + 1) \leq WHK,$$

with the equality holding only when $\forall k \in \{1, \ldots, K\}, w(k) = h(k) = 1$.

1.1.1 Fundamental constraints

We need to model the constraints of the problem by means of several propositional formulas. A solution (if any) of the simplified problem will then be represented by an assignment of truth values to each x_{ijk} which satisfies all of the formulas that model the constraints at the same time.

Our definitions of the atomic formulas x_{ijk} already guarantee that every circuit will be positioned within the bounds of the outer rectangle $ABCD$. In order to guarantee the correctness of the solution we then need to define two fundamental constraints: the first one is that every rectangle must have exactly one bottom-left vertex, and the second one is that any two different circuits must not overlap.

- *Existence and uniqueness of the bottom-left vertex location:* for every circuit $k = 1 \ldots K$, we need to impose the constraint *exactly_one* on the

formulas x_{ijk}, with $i \in I_k$, $j \in J_k$. This amounts to imposing the constraints

$$\bigwedge_{k=1}^{K} \bigvee_{i \in I_k, j \in J_k} x_{ijk} \qquad \text{(at\_least\_one)}$$

and

$$\bigwedge_{k=1}^{K} \bigwedge_{\substack{i,i' \in I_k \\ j,j' \in J_k \\ (i,j) \neq (i',j')}} \neg(x_{ijk} \wedge x_{i'j'k}) \qquad \text{(at\_most\_one)}$$

- *No-overlap constraints:* Let's assume that the k-th circuit has the bottom-left vertex on the point of coordinates (i, j). We can then observe that the k'-th circuit overlaps with the former if and only if its bottom-left corner lies within the rectangle with vertices:
  - $V1(\max(0, i - w(k') + 1), \max(0, j - h(k') + 1))$;
  - $V2(\min(i + w(k) - 1, W - w(k')), \max(0, j - h(k') + 1))$;
  - $V3(\min(i + w(k) - 1, W - w(k')), \min(j + h(k) - 1, H - h(k')))$;
  - $V4(\max(0, i - w(k') + 1), \min(j + h(k) - 1, H - h(k')))$.

  By defining, for each two circuits k, k', the sets

  $$I_{kk'} := \{\max(0, i - w(k') + 1, \ldots, \min(i + w(k) - 1, W - w(k')\}$$

  and

  $$J_{kk'} := \{\max(0, j - h(k') + 1), \ldots, \min(j + h(k) - 1, H - h(k'))\}$$

  we can then define these constraints as:

  $$\bigwedge_{1 \leq k < k' \leq K} \bigwedge_{\substack{i \in I_k \\ j \in J_k \\ i' \in I_{kk'} \\ j' \in J_{kk'}}} (\neg x_{ijk} \vee \neg x_{i'j'k'})$$

A significant drawback of this approach is that the number of constraints tends to become extremely large as the number of rectangles K and the dimensions H, W of $ABCD$ increase. In order to address at least part of this problem, we tried several different encodings for the *at_most_one* constraint, namely the *bitwise*, the *Heule's* and the *sequential* encodings, and the last one experimentally appeared to work better. However, keeping in mind the practical motivation of the problem, we ultimately decided to get rid entirely of the *at_most_one* constraint, and only the *at_least_one* constraint was left there: in this way, in a given solution a circuit would be assigned a list (if any) of possible positions of its bottom-left vertex. This approach in general increases the number of solutions, but in order to obtain a solution of the original problem it is sufficient, for each rectangle k, to choose arbitrarily only one of the possible positions of its bottom-left vertex returned by the the solver.

1.1.2 Symmetry-breaking constraints

Again, since we observed that the time required to generate all the constraints often far exceeded the time needed to actually solve the problem, we opted not to add additional implied constraints that could improve the time required to satisfy the constraints. Moreover, so as not to slow down the constraint generation too much, we decided to work with only two symmetry-breaking constraints: one for the horizontal symmetry and one for the vertical symmetry.

Let X be the set of all variables x_{ijk}. Then the permutations on X

$$\begin{aligned} \pi_h \colon X &\longrightarrow X \\ x_{ijk} &\longmapsto x_{(W-i-w(k))jk} \end{aligned}$$

and

$$\begin{aligned} \pi_v \colon X &\longrightarrow X \\ x_{ijk} &\longmapsto x_{i(H-j-h(k))k} \end{aligned}$$

identify respectively the horizontal and the vertical symmetries of the model.

Unlike with the previous approach with CP, we do not have global constraints at our disposal, and we have to implement the lexicographic constraints "from scratch" . Since we are working with boolean variables, for $x, y \in X$, the condition $x \leq y$ corresponds to the condition $x \to y$, and $x = y$ corresponds to $x \leftrightarrow y$. In this way the lexicographic ordering constraint $\mathrm{lex}([Y_1, \ldots, Y_l], [Z_1, \ldots, Z_l])$ can be implemented for example as:

$$(Y_1 \to Z_1) \wedge \bigwedge_{i=2}^{l} \left(\left(\bigwedge_{j=1}^{i-1} Y_j \leftrightarrow Z_j \right) \to (Y_i \to Z_i) \right).$$

This naive implementation of the constraint, however, produces formulas which grow quadratically in size (atom occurences) with respect to l.

Another possible solution (intuitively similar to the sequential encoding of the *at_most_one* constraints), which keeps the size of the constraint relatively small, is to introduce $l - 1$ new variables S_j that should be true if and only if $Y_i \leftrightarrow Z_i$ for $i = 1, \ldots, j$. In this case the constraint can be defined as:

$$\begin{aligned} \mathrm{lex\_lesseq}([Y_1, \ldots, Y_l], [Z_1, \ldots, Z_l]) := (Y_1 \to Z_1) \wedge (S_1 \leftrightarrow (Y_1 \leftrightarrow Z_1)) \wedge \\ \wedge \bigwedge_{i=2}^{l-1} (S_i \leftrightarrow (S_{i-1} \wedge (Y_i \leftrightarrow Z_i))) \wedge \bigwedge_{i=1}^{l-1} (S_i \to (Y_{i+1} \to Z_{i+1})). \end{aligned}$$

This last encoding makes the size of the constraint grow linearly with respect to l.

The symmetry-breaking constraints are then implemented as:

$$\begin{aligned} &\mathrm{lex\_lesseq}(X, \pi_h(X)); \\ &\mathrm{lex\_lesseq}(X, \pi_v(X)). \end{aligned}$$

Lastly, as a compromise between the need to speed up the generation of the formula to be satisfied and the pruning of the search space, we also tested a "partial" lexicographic ordering constraint: instead of comparing all of the variables in X to $\pi_h(X)$ and $\pi_v(X)$, we compare only the first $f(n)$ variables, where $n = |X|$, and f is a non-decreasing function such that $\forall x \in \{1, \ldots, n\}, f(x) <= x$. In our case we used specifically !!!!!!!!!!INSERIRE LA FUNZIONE (N/2? SQRT(N)?) E I RISULTATI!!!!!!!!!!

1.1.3 Rotations

In order to model the possibility to rotate the circuits when placing them in $ABCD$, we introduce the new set of atoms $Y = \{y_{ijk}\}$ with the intended meaning that y_{ijk} is true if and only if the bottom-left vertex of the rotated k-th circuit is the point (i, j). Analogously to the previous case, in order to reduce the amount of additional atoms, we restrict, for each circuit k, the i and j indices respectively to the sets $I'_k = \{0, \ldots, W - h(k)\}$ and $J'_k = \{0, \ldots, H - w(k)\}$.

The constraints needed to guarantee the correctness of the returned solutions are entirely analogous to what we presented earlier, with the addition of the new variables:

$$\bigwedge_{k=1}^{K} \left(\bigvee_{i \in I_k, j \in J_k} x_{ijk} \vee \bigvee_{i \in I'_k, j \in J'_k} y_{ijk} \right) \qquad \text{(at\_least\_one)}$$

$$\bigwedge_{k=1}^{K} (A_k \wedge B_k \wedge C_k) \qquad \text{(at\_most\_one)},$$

where

$$A_k = \bigwedge_{\substack{i,i' \in I_k \\ j,j' \in J_k \\ (i,j) \neq (i',j')}} \neg(x_{ijk} \wedge x_{i'j'k})$$

$$B_k = \bigwedge_{\substack{i \in I_k \\ j \in J_k \\ i' \in I'_k \\ j' \in J'_k}} \neg(x_{ijk} \wedge y_{i'j'k})$$

$$C_k = \bigwedge_{\substack{i,i' \in I'_k \\ j,j' \in J'_k \\ (i,j) \neq (i',j')}} \neg(y_{ijk} \wedge y_{i'j'k}).$$

When modeling in this way the problem, we get an upper bound on the number of atoms in $Z := X \cup Y$ of $2WHK$, and the number of constraints increases even more with respect to the problem without the possibility to rotate circuits, regardless of the chosen encodings. For this reason, we decided to remove the *at_most_one* constraint in this case too. Again, from a returned solution of the

problem without the *at_most_one* constraint we can easily obtain a solution of the original problem by choosing arbitrarily, for every $k \in \{1, \dots, K\}$, only one of the variables x_{ijk} or y_{ijk} that have been assigned the truth value $\mathbf{t}$.

Regarding the *no_overlap* constraints, we rename the variables in Z as:

$$z_{ijk} := \begin{cases} x_{ijk} & \text{for } 1 \leq k \leq K, \\ y_{ij(k-K)} & \text{for } K+1 \leq k \leq 2K \end{cases}$$

Then we can define the constraints in the same way as before:

$$\bigwedge_{1 \leq k < k' \leq 2K} \bigwedge_{\substack{i \in I_k \\ j \in J_k \\ i' \in I_{kk'} \\ j' \in J_{kk'}}} \left(\neg z_{ijk} \vee \neg z_{i'j'k'} \right),$$

where $\forall k \in \{K+1, \dots, 2K\}$, $w(k) := h(k-K)$ and $h(k) := w(k-K)$.

Concerning symmetry-breaking constraints, we adapt the previous ones (using the "sequential" encoding of the *lex_lesseq* predicate) to the new set of atoms Z:

$$\begin{aligned} &\text{lex\_lesseq}(Z, \pi_h(Z)), \\ &\text{lex\_lesseq}(Z, \pi_v(Z)), \end{aligned}$$

where in this case

$$\begin{aligned} \pi_h \colon Z &\longrightarrow Z \\ z_{ijk} &\longmapsto z_{(W-i-w(k))jk} \end{aligned}$$

and

$$\begin{aligned} \pi_v \colon Z &\longrightarrow Z \\ z_{ijk} &\longmapsto z_{i(H-j-h(k))k}. \end{aligned}$$

As shown before, in the actual Z3Py implementation of the model, we chose not to compare all of the variables in Z, and we limited the comparison to the first !!!!!FUNZIONE!!!!! ones.

Lastly, we considered adding more atoms (a set $R = \{r_k\}$ where r_k is true if and only if the k-th circuit is rotated) and implied constraints, but this has experimentally proven to be ineffective either because they increased significantly the time required to generate the formula to satisfy or because they actually increased the time needed for the Z3 solver to check the satisfiability of the formula.

1.2 Optimization

Since we are limited to check satisfability of propositional formulas, we do not have a direct way to implement optimization, like for example in CP or MIP

models. We identified instead two main possibilities that exploit the bounds on the height of the outer rectangle shown previously (DISTINGUIAMO I CASI ROTATIONS VS NO ROTATIONS?). The first one is performing a linear search starting from the lower bound: we check the satisfiability of the constraints when H coincides with the lower bound; if the constraints are satisfiable, then the returned model identifies an optimal solution of the instance; if they are not satisfiable, then we increment H by 1 and repeat the process until a solution is found. As soon a solution is found, then it is trivially an optimal one (we are assuming here that a solution exists within the lower and the upper bounds, that is, there isn't any rectangle in a given instance whose dimensions are both greater than the fixed width).

RISCRIVERE MEGLIO The second possibility exploits the monotony of the (un)satisfiability of the formula with respect to the height H and consists in performing a binary search between the lower and the upper bounds provided earlier. One thing we noticed in intermediate returned solutions is that often there are several empty lines. We can exploit this fact to partly improve the binary search. Instead of replacing the value of the current upper bound H_t to the height of an optimal solution H^* with the height of the last found solution H_{current}, we can replace H_t with $H_{\text{current}} - E$, where E is the number of empty lines in the current solutions.

If U is the upper bound on the height of $ABCD$, and L is the lower bound, then the first approach requires to solve $U - L + 1$ instances in the worst case, while the second one requires to solve $\simeq \log(U - L)$ instances. For this reason, the second approach is in general preferable. However, the first approach worked particularly well with the specific assigned instances, as the optimal height coincides with the lower bound in all of the instances that we were able to solve within the timout limits of 300 seconds.

1.3 Results

QUI VANNO RISULTATI DEI TEST (+ GRAFICI)