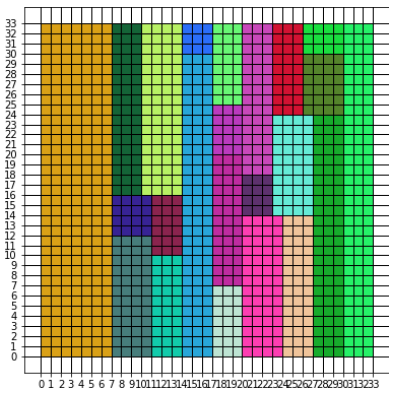
Project Work in Combinatorial Decision Making and Optimization

VLSI problem instances resolved by means of CP, SAT, SMT, and LP encodings



Daniele Baiocco - daniele.baiocco@studio.unibo.it
Memoona Shah - memona.shah@studio.unibo.it
Jacopo Meglioraldi - jacopo.meglioraldi@studio.unibo.it

April 29, 2023

1 Introduction

In this project report, we have solved the VLSI problem if arranging the circuit/chips in a two-dimensional strip. We have solved the problem with four different approaches such as CP, SAT, SMT and MIP. In each model, we have solved the problem as 2D strip packing (rotation of circuits is not allowed) and 2D orthogonal strip packing(rotation allowed by 90° degrees). Results of each model is also demonstrated with and without symmetry-breaking constraints with help of tables and graphs. In this entire document, we will use the following variables for the general approach:

$\forall$ i $\in$ 1..n,

- W, the width of the plate as given in the input file.
- n, the number of circuits that must be placed in the plate
- H, the height of the plate that is to be minimized.
- x_i, an array in which the ith element represents the horizontal position coordinate(bottom-left corner) of the ith circuit
- y_i, an array in which the ith element represents the vertical position coordinate(bottom-left corner) of the ith circuit
- w_i, an array in which the ith element represents the width of the ith circuit
- h_i, an array in which the ith element represents the height of the ith circuit
- Zlr$_{i,j}$, which is true if the ith circuit is placed to the left to jth circuit.
- Zud$_{i,j}$, which is true if the ith circuit is placed below the jth circuit.
- Zlr$_{j,i}$, which is true if the jth circuit is placed to the left of the ith circuit.
- Zud$_{j,i}$, which is true if the jth circuit is placed below the ith circuit.
- r_i which is true if the ith circuit is rotated and false if not.

The work was divided between the three group members in the following way. Each member was a team leader for one part such as Daniele, who was mainly responsible for SAT, Memoona for SMT, and Jacopo for MIP. CP part however was done by both Daniele and Memoona. The constraints encoding was shared on a common platform by each member so it could be implemented by other members. Overall, it was more of a team effort than an individual. The work of all four parts has been completed in around three months time period.

2 CP Model

2.1 Decision variables

The decision variables that have been used are H, x, and y. The semantics of these variables has already been exposed. The Minimum height has been selected by combining two methods. One possibility of assuming the minimum allowed height can be dividing the total area of all the chips summed by plate width. This has been adopted from the paper by Cid-Garcia and Rios-Solis (2021)[1] The variable H has the domain:

$$D(H) \ = \ \{ \ H \ \in \ \mathbb{N} \mid h_{min} \ \leq H \ \leq h_{max} \}$$

where h_{max} and h_{min} are the upper and lower bounds and have been set as the following:

$$h_{min} = \frac{\sum_i h_i w_i}{W} \text{ and } h_{max} = 2[h_{min}]$$

The domains of x_i and y_i are declared as follows:

$$\begin{aligned} D(x_i) \ &= \ \{ \ c \ \in \ \mathbb{N} \mid 0 \ \leq c \ \leq W - w_i \} \\ D(y_i) \ &= \ \{ \ c \ \in \ \mathbb{N} \mid 0 \ \leq c \ \leq H - h_i \} \end{aligned}$$

2.2 Objective function

The objective function is to minimize H, in order to retrieve the lowest height for which all circuits fit within the plate. Its bounds are discussed in the above section.

2.3 Constraints

The core constraint is the

No-overlap constraint The global constraint that has been used to enforce circuits not to overlap is **diffn**.[2] It basically ensures that $\forall\ i, j \in 1..\text{n}$ where $i \neq j$,

$$x_i + w_i \leq x_j \vee y_i + h_i \leq y_j \tag{1}$$

2.4 Implied constraints

The only implied constraint in the model is the The only implied constraint in the model is the cumulative one. It is implied because without it the optimal solution is still found, but it improves the performance if it is added to the constraint model. Without the others constraints, the H found is still the optimal one but the coordinates x and y of the circuits are such that circuits may overlap.

Cumulative constraint It is used twice, in the following ways:

- cumulative(x, w, h, H) which means that
  $\forall u \in D$, with D = [1,2,3,4,...]
  $$\sum_{i \in \{1..n\} | x_i \leq u < x_i + w_i} h_i \leq \text{H} \tag{2}$$
- cumulative(y, h, w, W) which stands for
  $\forall u \in D$, with D = [1,2,3,4,...]
  $$\sum_{i \in \{1..n\} | y_i \leq u < y_i + h_i} w_i \leq W \tag{3}$$

2.5 Symmetry breaking constraints

Symmetries observed in the model were of various types. Given the biggest circuit, in terms of area, it can be placed by the solver in different positions, as described in Figure 1. By constraining its domain in such a way that it is halved (Figure 2), the symmetries shown in Figure 1 are no longer possible. Other symmetries are generated by swappings between circuits which results are still admissible. In order to reduce the number of possible permutations another symmetry-breaking constraint has been introduced.

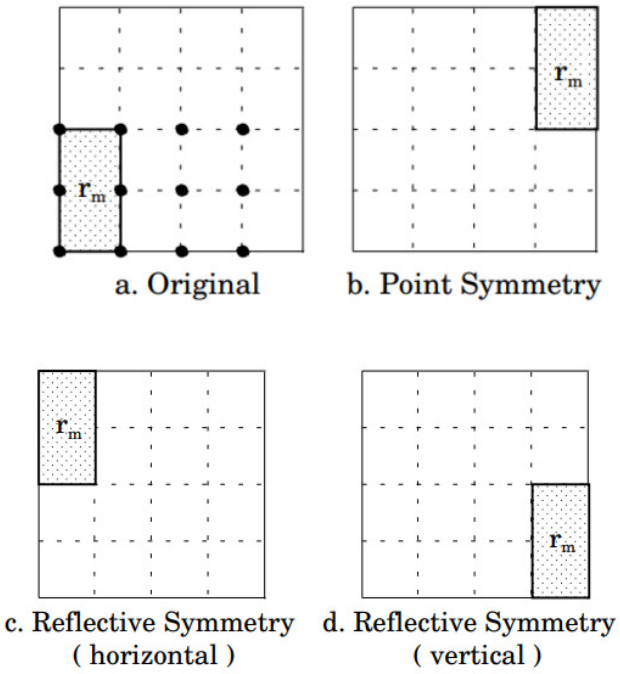


Figure 1: Domain space of circuit c_m. It can be placed in each of the black dots in plate a. This leads to symmetries like the ones in b, c, and d.

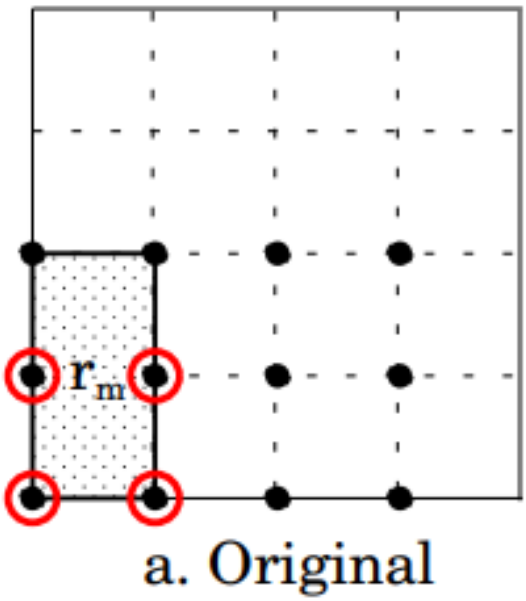


Figure 2: The coordinates domain of circuit c_m is halved in both axis: the only positions allowed are the ones described by the red-circled dots.

Reducing domain of the biggest circuit As has already been discussed, this constraint reduces the number of symmetries concerning the biggest circuit. Given this circuit c_m, we have that
$y_m \leq \lfloor \frac{H - \mathrm{h}_m}{2} \rfloor$
and $x_m \leq \lfloor \frac{\mathrm{W} - \mathrm{w}_m}{2} \rfloor$

Pushing circuits to the right This constraint enforces the number of circuits that lie on the right side of the plate to be higher than the ones on the left side In this way, the number of possible permutations decreases. Considering two sets of circuits, the first one is defined as:

$$\forall i \in [1..n]$$
$$\text{after\_threshold} = \{c_i | x_i + w_i \geq threshold\}$$

the second one is defined as:

$$\forall i \in [1..n]$$
$$\text{before\_threshold} = \{c_i | x_i + w_i < threshold\}$$

Given these sets, the constraint states that

$$|after\_threshold| \geq |before\_threshold| \tag{4}$$

Regarding CP, the threshold was set to $\lfloor \frac{2}{3} * \mathrm{W} \rfloor$

2.6 Rotation

The model was extended with the introduction of new decision variables, bounds and constraints.
The new decision variables are:

- **g**: a grid in which $g_{i,j}$ is the value of the coordinate in the j-th dimension of the left bottom corner of the i-th circuit
- **kind**: a one dimensional array in which $kind_i = i$ if the i-th circuit is not rotated, $kind_i = 2 * i$ otherwise.

Moreover, two new arrays w_rot, h_rot, starting from w and h, have been defined as:

$$\mathrm{w\_rot}_i = \begin{cases} w_i & \text{if } kind_i \leq n \\ h_i & \text{otherwise} \end{cases} \forall \mathrm{i} \in 1..\mathrm{n} \tag{5}$$

$$\text{h\_rot}_i = \begin{cases} h_i & \text{if } kind_i \leq n \\ w_i & \text{otherwise} \end{cases} \forall \text{i} \in 1..\text{n} \tag{6}$$

Given these arrays, the domains of x_i and y_i have been redefined as follows:

$$\begin{aligned} D(x_i) &= \{ c \in \mathbb{N} \mid 0 \leq c \leq W - \text{w\_rot}_i \} \\ D(y_i) &= \{ c \in \mathbb{N} \mid 0 \leq c \leq H - \text{h\_rot}_i \} \end{aligned}$$

The decision variable H has the same domain as the one discussed in a previous section.

Constraints from the previous model Constraints from the previous model have been used with slight different changes. Constraints in the Formulas (1), (2) and (3) were modified by just replacing w and h with respectively w_rot and h_rot. Constraint in Formula (4) has been kept as it is, with value of threshold still set to $\lfloor \frac{2}{3} \text{W} \rfloor$. Constraint concerning the domain reduction of the biggest circuit has been removed.

In the following paragraphs new constraints introduced are listed:

Geost_bb constraint Geost_bb [3] is a global non-overlap constraint for k-dimensional objects. It enforces that no two objects overlap and that all objects fit within a global k-dimensional bounding box. The parameters are

- **k**: the number of dimensions
- **rect_size**: the size of each box in k dimensions
- **rect_offset**: the offset of each box from the base position in k dimensions
- **shape**: the set of circuits defining ith shape
- **x**: the base position of each object. $x[i, j]$ is the position of object i in dimension j
- **kind**: the shape used by each object
- **l**: an array of lower bounds, $l[i]$ is the minimum bounding box for all objects in dimension i
- **u** : an array of upper bounds, $u[i]$ is the maximum bounding box for all objects in dimension i

The k in our case is equal 2 because circuits have two dimensions. The rect_size has been defined as an array of 2n elements such that $\forall k \in \{1..n\}$,

$$\text{rect\_size}_{k,1} = \text{w}_k,\ \text{rect\_size}_{k,2} = \text{h}_k,\ \text{rect\_size}_{k+n,1} = \text{h}_k,\ \text{rect\_size}_{k+n,2} = \text{w}_k \tag{7}$$

The rect_offset is a 2d array of 2n rows. The i-th row tells what is the offset of the i-th circuit. It has been implemented in a way that at each row there is a pair of 0s, because there is no change from the base position in each case. The shape is an array of sets built in the following way:

$$shape_k = \{k\}, \forall k \in \{1..2n\} \tag{8}$$

This means there are no composite shapes: each shape corresponds to a circuit. The two last arguments l and u have been assigned to respectively [0,0] and [W, H].

Kind constraint The array kind has been constrained in the following way: $\forall k \in \{1..n\}$, $kind_k = k \oplus kind_k = k+n$. This means that the i^{th} element of the kind can only represent either the i^{th} circuit or the i^{th} circuit rotated.

Channeling constraint In order to channel the two different ways of representing the output pairs of coordinates g and x, y, a channeling constraint has been introduced for both the axis. For the x, $\forall k \in \{1..n\}$, $\forall\ val \in \{0..upper\_bound(x)\}$, $x_k == val \Leftrightarrow g_{k,1} == val$.

For the y, $\forall k \in \{1..n\}$, $\forall\ val \in \{0..upper\_bound(y)\}$, $y_k == val \Leftrightarrow g_{k,2} == val$

Square circuit constraint It ensures that if a circuit is a square, then it cannot rotate. In a formal way, $\forall i \in \{1..n\}$, if $\text{w}_i == \text{h}_i$ then $\text{kind}_i = \text{i}$.

2.7 Validation

Experimental design The hardware on which the experiments were done has the following specifications:

- CPU: 11th Gen Intel(R) Core(TM) i5-11400F @ 2.60GHz 2.59 GHz
- RAM: 16,0 GB

The experiments were conducted by running a Python script in which we used Minizinc APIs.

The configuration of search strategies for the model without rotation that got the best results was the following: for the choice of which decision variable to pick from the x vector, we used the strategy *dom_w_deg* which choose the variable with the smallest value of domain size divided by weighted degree, which is the number of times it has been in a constraint that caused failure earlier in the search. While, regarding the choice of a value from the variable domain, once the variable has been chosen, we adopted the *indomain_random* strategy, which assigns to the variable a random value from its domain. Both *dom_w_deg* and *indomain_random* has also been adopted for y. For what concerns H, the assignment of a value to this decision variable is dictated also in this time by the indomain_random strategy.

In addition to the search annotations, we used also the restart strategy *restart_luby* with value of the scale equal 450, and a large neighborhood strategy *restart_and_reconstruct*, on x, with the percentage set to 30.

The solver we used is Gecode.

For what concerns the rotation model we used Chuffed instead, because it gave better performances. The annotations implemented are:

- for x, for the choice of a variable first_fail (it chooses the variable with the smallest domain size), for the choice of a value in the domain indomain_min,
- for y the same as x,
- for H, indomain_min,
- for kind, input_order and indomain_min

Also in this case luby was used, with value of the scale equal 150.

Experimental results By executing the script only on the first model (the one without rotation) the performances in terms of computational time are shown in Table 1. After running all the instances in 4 ways:

- with the first model and with symmetry breaking constraints,
- with the first model and with symmetry breaking constraints,
- without the first model and with symmetry breaking constraints,
- without the first model and with symmetry breaking constraints,

what came out is shown in Table 2 and is well displayed in Figures 2.7 and 2.7. Note that the time in sec refers only to the instances that converge within 5 minutes.

instances	n. of timeouts	time in sec (mean)
from 1 to 10	0	0.198
from 11 to 20	0	4.054
from 21 to 30	0	11.562
from 31 to 40	5	155.558

Table 1: computation results on instances

Table 2: Optimal solutions found by each model

ID	Chuffed + SB	Chuffed w/out SB	Gecode + SB	Gecode w/out SB	best solution
1	**8**	**8**	**8**	**8**	**8**
2	**9**	**9**	**9**	**9**	**9**
3	**10**	**10**	**10**	**10**	**10**
4	**11**	**11**	**11**	**11**	**11**
5	**12**	**12**	**12**	**12**	**12**
6	**13**	**13**	**13**	**13**	**13**
7	**14**	**14**	**14**	**14**	**14**
8	**15**	**15**	**15**	**15**	**15**
9	**16**	**16**	**16**	**16**	**16**
10	**17**	**17**	**17**	**17**	**17**
11	**18**	**18**	**18**	**18**	**18**
12	**19**	**19**	**19**	**19**	**19**
13	**20**	**20**	**20**	**20**	**20**
14	**21**	**21**	**21**	**21**	**21**
15	**22**	**22**	**22**	**22**	**22**
16	**23**	**23**	**23**	**23**	**23**
17	**24**	**24**	**24**	**24**	**24**
18	**25**	**25**	**25**	**25**	**25**
19	**26**	**26**	**26**	**26**	**26**
20	**27**	**27**	**27**	**27**	**27**
21	**28**	**28**	**28**	**28**	**28**
22	**29**	**29**	**29**	**29**	**29**
23	**30**	**30**	**30**	**30**	**30**
24	**31**	**31**	**31**	**31**	**31**
25	**32**	**32**	**32**	**32**	**32**
26	**33**	**33**	**33**	**33**	**33**
27	**34**	**34**	**34**	**34**	**34**
28	**35**	**35**	**35**	**35**	**35**
29	**36**	**36**	**36**	**36**	**36**
30	**37**	**37**	**37**	**37**	**37**
31	**38**	**38**	**38**	**38**	**38**
32	40	40	40	40	**39**
33	**40**	**40**	**40**	**40**	**40**
34	**40**	**40**	**40**	**40**	**40**
35	**40**	**40**	**40**	**40**	**40**
36	**40**	**40**	**40**	**40**	**40**
37	61	61	61	61	**60**
38	61	61	61	61	**60**
39	62	61	62	61	**60**
40	113	113	113	120	n/a

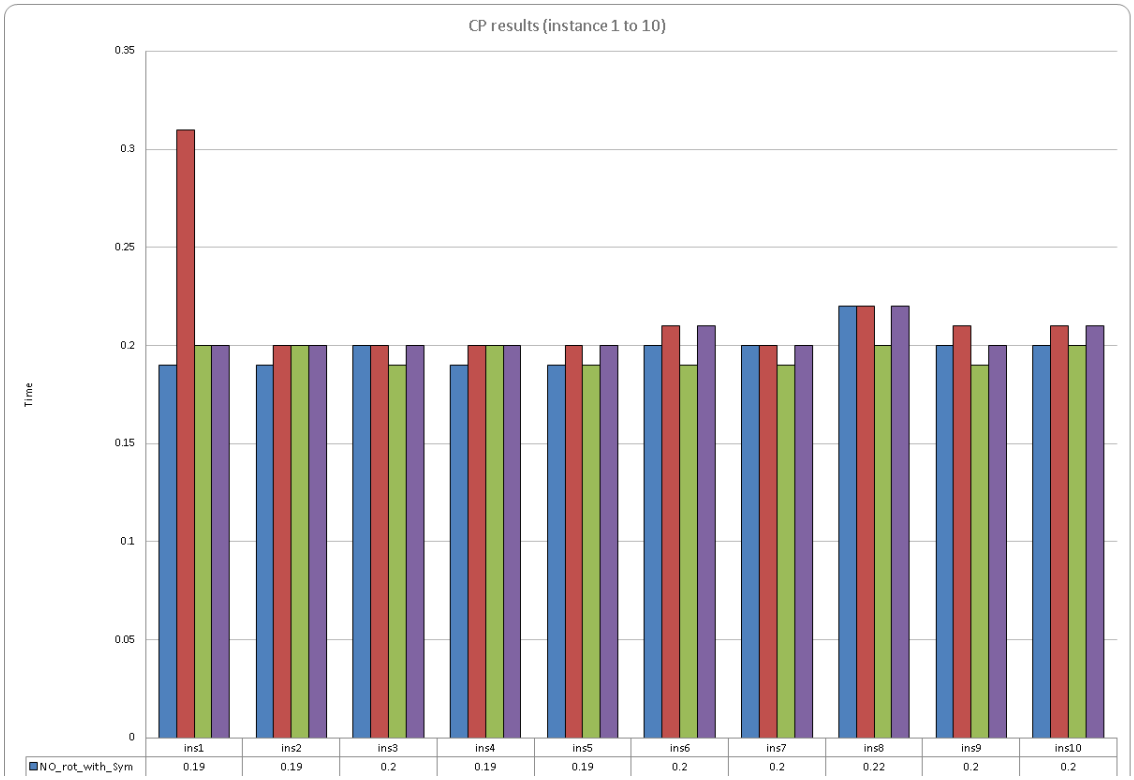
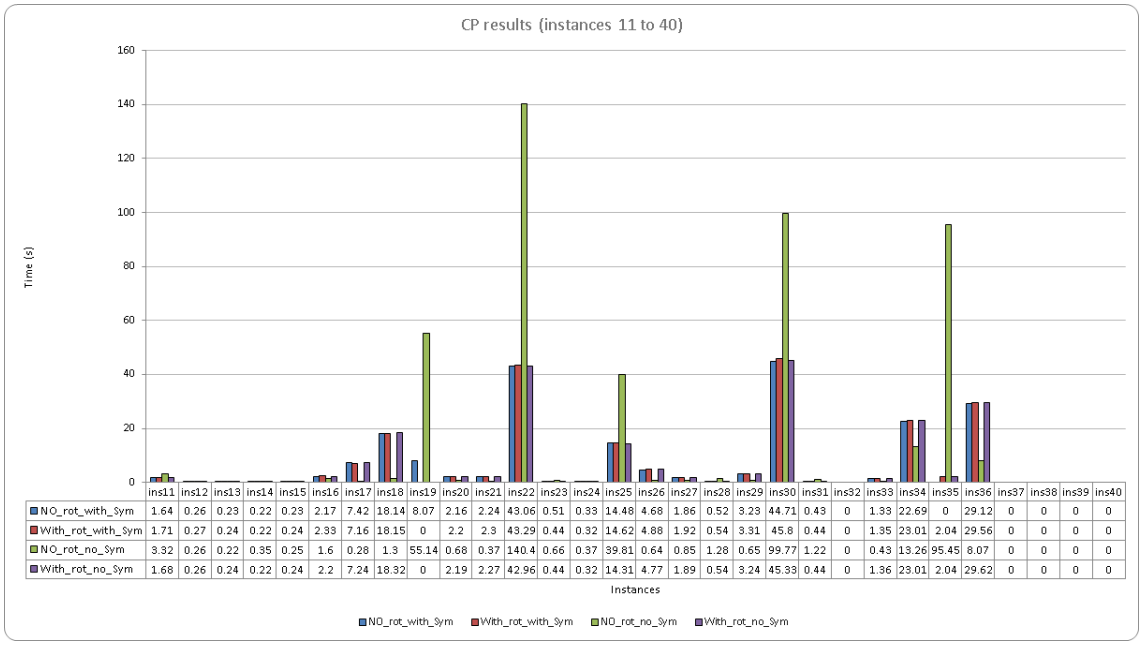


Figure 3: Results of CP



	ins11	ins12	ins13	ins14	ins15	ins16	ins17	ins18	ins19	ins20	ins21	ins22	ins23	ins24	ins25	ins26	ins27	ins28	ins29	ins30	ins31	ins32	ins33	ins34	ins35	ins36	ins37	ins38	ins39	ins40
NO_rot_with_Sym	1.64	0.26	0.23	0.22	0.23	2.17	7.42	18.14	8.07	2.16	2.24	43.06	0.51	0.33	14.48	4.68	1.86	0.52	3.23	44.71	0.43	0	1.33	22.69	0	29.12	0	0	0	0
With_rot_with_Sym	1.71	0.27	0.24	0.22	0.24	2.33	7.16	18.15	0	2.2	2.3	43.29	0.44	0.32	14.62	4.88	1.92	0.54	3.31	45.8	0.44	0	1.35	23.01	2.04	29.56	0	0	0	0
NO_rot_no_Sym	3.32	0.26	0.22	0.35	0.25	1.6	0.28	1.3	55.14	0.68	0.37	140.4	0.66	0.37	39.81	0.64	0.85	1.28	0.65	99.77	1.22	0	0.43	13.26	95.45	8.07	0	0	0	0
With_rot_no_Sym	1.68	0.26	0.24	0.22	0.24	2.2	7.24	18.32	0	2.19	2.27	42.96	0.44	0.32	14.31	4.77	1.89	0.54	3.24	45.33	0.44	0	1.36	23.01	2.04	29.62	0	0	0	0

Figure 4: Results of CP

3 SAT encoding

3.1 Decision variables

The decision variable H has a domain in [min_y..max_y] with min_y, max_y equal to min_y, max_y of cp model.

Atom variables The literals that have been used are $\mathrm{Zlr}_{i,j}$, $\mathrm{Zud}_{i,j}$, $\mathrm{Zlr}_{j,i}$, $\mathrm{Zud}_{j,i}$ whose semantic has been already specified in the Introduction section, along with x and y.

These last literals have the following semantics:

- $\mathrm{x}_{i,e}$ is true if the x coordinate of the base point of c_i is placed in a position that is less than or equal to e
- $\mathrm{y}_{i,f}$ is true if the y coordinate of the base point of c_i is placed in a position that is less than or equal to e

Note that $\forall i \in \{1, ..., n\}$, $\forall e$ s.t. $0 \leq e < \mathrm{W} - w_i$, $\mathrm{x}_{i,e}$ is defined.
Moreover $\forall i \in \{1, ..., n\}$, $\forall f$ s.t. $0 \leq f < \mathrm{H} - h_i$, $\mathrm{y}_{i,f}$ is defined.

3.2 Objective function

The variable to minimize is also in this case H. In order to minimize H at each iteration H is computed as the mean between a lower bound and an upper bound. At the beginning of this recursive process, these bounds are initialized with respectively min_y and max_y. The recursive process is such that if satisfiability is obtained then the upper bound is updated by H, whereas, if the problem is unsatisfiable with that H, then the lower bound is the one that is updated by H. The solution area becomes more and more constrained until the optimal height is obtained.

3.3 Constraints

Propagation constraint This one propagates the truthfulness or the falseness of the $\mathrm{x}_{i,e}$ bounded to a circuit c_i. It states that $\forall i \in \{1, ..., n\}$, $\forall e$ s.t. $0 \leq e < \mathrm{W} - \mathrm{w}_i$

$$\neg x_{i,e} \vee x_{i,e+1} \tag{9}$$

and that $\forall i \in \{1, ..., n\}$, $\forall f$ s.t. $0 \leq f < \mathrm{H} - \mathrm{h}_i$

$$\neg y_{i,f} \vee y_{i,f+1} \tag{10}$$

Relative position constraint It rules the relative position relations that has to exist between each pair of circuits: $\forall i \in \{1, ..., n\}$, $\forall j \in \{i, ..., n\}$, we have that

$$Zlr_{i,j} \vee Zlr_{j,i} \vee Zud_{i,j} \vee Zud_{j,i} \tag{11}$$

No overlap constraint This constraint is the core and ensures that two circuits don't overlap. This constraint was implemented from the paper by Soh, Takehide and Inoue (2010) [4].

$\forall i \in \{1, ..., n\}, \forall j \in \{i, ..., n\}$, $\forall e$ s.t. $0 \leq e < \mathrm{W} - \mathrm{w}_i$

$$\neg Zlr_{i,j} \vee x_{i,e} \vee \neg x_{j,e+w_i} \tag{12}$$

$\forall i \in \{1, ..., n\}, \forall j \in \{i, ..., n\}$, $\forall e$ s.t. $0 \leq e < \mathrm{W} - \mathrm{w}_j$

$$\neg Zlr_{j,i} \vee x_{j,e} \vee \neg x_{i,e+w_j} \tag{13}$$

$\forall i \in \{1, ..., n\}, \forall j \in \{i, ..., n\}$, $\forall f$ s.t. $0 \leq f < \mathrm{H} - \mathrm{h}_i$

$$\neg Zud_{i,j} \vee y_{i,f} \vee \neg y_{j,f+h_i} \tag{14}$$

$\forall i \in \{1, ..., n\}, \forall j \in \{i, ..., n\}$, $\forall f$ s.t. $0 \leq f < \mathrm{H} - \mathrm{h}_j$

$$\neg Zud_{j,i} \vee y_{j,f} \vee \neg y_{i,f+h_j} \tag{15}$$

Reducing the domain of the biggest circuit The base idea is the one described in the cp section, in which the domain of the biggest circuit (in terms of the area) is halved. In this case we have that given this circuit c_m, $x_{m,e}$ is only defined $\forall e$ s.t. $0 \leq e < \lfloor \frac{\mathrm{W} - \mathrm{w}_m}{2} \rfloor$. Moreover $y_{m,f}$ is only defined $\forall f$ s.t. $0 \leq f < \lfloor \frac{\mathrm{H} - \mathrm{h}_m}{2} \rfloor$.

In order to speed up the solver, $\forall i \in \{1, ..., n\}$ such that $\mathrm{w}_i > \lfloor \frac{\mathrm{W} - \mathrm{w}_m}{2} \rfloor$ we states that

$$\neg Zlr_{i,m} \tag{16}$$

The same goes for the heights: $\forall i \in \{1, ..., n\}$ such that $\mathrm{h}_i > \lfloor \frac{\mathrm{H} - \mathrm{h}_m}{2} \rfloor$

$$\neg Zud_{i,m} \tag{17}$$

Same circuits constraint It's a symmetry-breaking constraint For each circuit c_i and c_j, if (w_i , h_i) = (w_j , h_j) we can fix the positional relation of these circuits. We, therefore, say that

$$\neg Zlr_{j,i} \text{ and } \neg Zud_{i,j} \vee Zlr_{i,j} \tag{18}$$

3.4 Rotation

Decision variables A new decision variable **perm** was added to the model to make the rotation of the circuits possible. Its i-th element o tells whether the i-th circuit performs a rotation or not.

Constraints from the previous model Constraints (9), (10) and (11) were kept, whereas the the one that reduced the domain of the biggest circuit has been deleted and the no-overlap and same circuits constraints have been changed. The new constraints are listed as follows:

No-overlap constraint revisited $\forall i \in \{1, ..., n\}, \forall j \in \{i, ..., n\}, \forall e, \forall f$

$$\begin{aligned} perm_i \vee \neg Zlr_{i,j} \vee x_{i,e} \vee \neg x_{j,e+w_i} \\ \neg perm_i \vee \neg Zlr_{i,j} \vee x_{i,e} \vee \neg x_{j,e+h_i} \end{aligned} \tag{19}$$

$$\begin{aligned} perm_j \vee \neg Zlr_{j,i} \vee x_{j,e} \vee \neg x_{i,e+w_j} \\ \neg perm_j \vee \neg Zlr_{j,i} \vee x_{j,e} \vee \neg x_{i,e+h_j} \end{aligned} \tag{20}$$

$$\begin{aligned} perm_i \vee \neg Zud_{i,j} \vee y_{i,f} \vee \neg y_{j,f+h_i} \\ \neg perm_i \vee \neg Zud_{i,j} \vee y_{i,f} \vee \neg y_{j,f+w_i} \end{aligned} \tag{21}$$

$$\begin{aligned} perm_j \vee \neg Zud_{j,i} \vee y_{j,f} \vee \neg y_{i,f+h_j} \\ \neg perm_j \vee \neg Zud_{j,i} \vee y_{j,f} \vee \neg y_{i,f+w_j} \end{aligned} \tag{22}$$

In this way if the solver chose perm_i = False and perm_j = False, these formulas would simplify in exactly the ones from the no-overlap constraint paragraph.

Same circuits constraint revisited It is such that for each circuit c_i and c_j, if (w_i , h_i) = (w_j , h_j) then

$$\begin{aligned} perm_i \vee perm_j \vee \neg Zlr_{j,i} \\ perm_i \vee perm_j \vee Zlr_{i,j} \vee \neg Zud_{j,i} \\ \neg perm_i \vee \neg perm_j \vee \neg Zlr_{j,i} \\ \neg perm_i \vee \neg perm_j \vee Zlr_{i,j} \vee \neg Zud_{j,i} \end{aligned} \tag{23}$$

if instead we have that (h_i, w_i) = (w_j, h_j) then

$$\begin{aligned} perm_i \vee \neg perm_j \vee \neg Zlr_{j,i} \\ perm_i \vee \neg perm_j \vee Zlr_{i,j} \vee \neg Zud_{j,i} \\ \neg perm_i \vee perm_j \vee \neg Zlr_{j,i} \\ \neg perm_i \vee perm_j \vee Zlr_{i,j} \vee \neg Zud_{j,i} \end{aligned} \tag{24}$$

Square circuit constraint This one has already been discussed in the cp section. In SAT it is slightly different: $\forall$ i $\in$ 1..n, if $w_i == h_i$ then

$$\neg \text{perm}_i \tag{25}$$

Positional relation between a pair of circuits Given two circuits c_i, c_j taken at random from the set of all the circuits, they are put in the following relation: $\neg Zlr_{j,i}$, $\neg Zud_{j,i}$

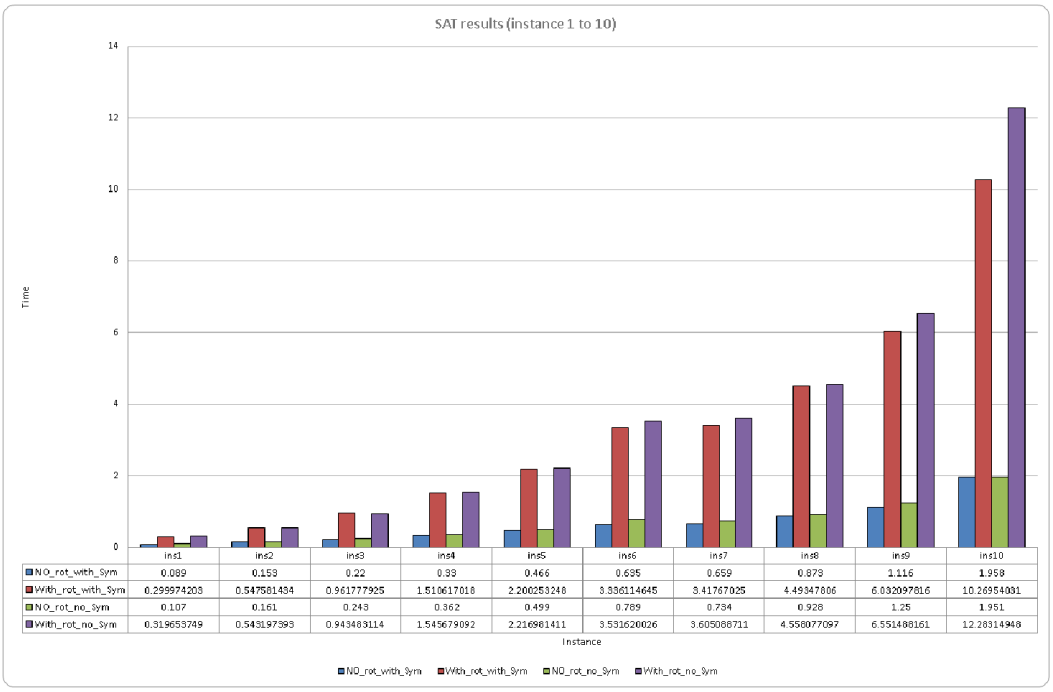
This means that the second circuit is neither to the left of the first one nor under it.

Not allowed to permute constraint Given a randomly picked circuit c_i, it cannot rotate, so $\neg \text{perm}_i$.

3.5 Validation

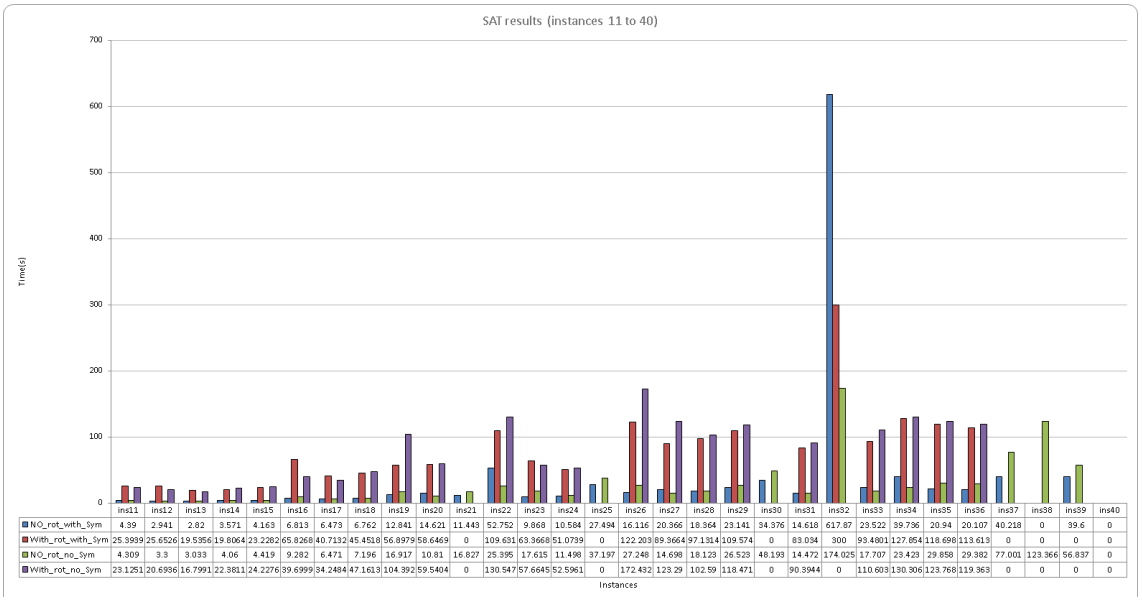
Experimental design The specifications used are the same as the ones listed in the previous section. For what concerns the software, also in this case the experiments were done running a python script.

Experimental results The experiments conducted were 4. The first two regarding the standard model, with and without symmetry breaking constraints, and the other two regarding the rotation model, in the same manner.



	ins1	ins2	ins3	ins4	ins5	ins6	ins7	ins8	ins9	ins10
NO_rot_with_Sym	0.089	0.153	0.22	0.39	0.466	0.695	0.659	0.879	1.116	1.958
With_rot_with_Sym	0.299974203	0.547581434	0.961777925	1.510617018	2.200253248	3.336114645	3.41767025	4.49347806	6.032997816	10.26954031
NO_rot_no_Sym	0.107	0.161	0.243	0.362	0.499	0.789	0.734	0.928	1.25	1.951
With_rot_no_Sym	0.319653749	0.543197393	0.943483114	1.545679092	2.216981411	3.531620026	3.605086711	4.558077097	6.551488161	12.28314948

Figure 5: Results of SAT



	ins11	ins12	ins13	ins14	ins15	ins16	ins17	ins18	ins19	ins20	ins21	ins22	ins23	ins24	ins25	ins26	ins27	ins28	ins29	ins30	ins31	ins32	ins33	ins34	ins35	ins36	ins37	ins38	ins39	ins40
NO_rot_with_Sym	4.39	2.941	2.82	3.571	4.163	6.813	6.473	6.762	12.841	14.621	11.443	52.752	9.868	10.584	27.494	16.116	20.366	18.364	23.141	34.376	14.618	617.87	23.522	39.736	20.94	20.107	40.218	0	39.6	0
With_rot_with_Sym	25.3939	25.6526	19.5356	19.8064	23.2282	65.8268	40.7132	45.4518	56.8979	58.6469	0	109.631	63.3668	51.0739	0	122.203	89.3664	97.1314	109.574	0	83.034	300	93.4801	127.854	118.698	113.613	0	0	0	0
NO_rot_no_Sym	4.309	3.3	3.033	4.06	4.419	9.282	6.471	7.196	16.917	10.81	16.827	25.395	17.615	11.498	37.197	27.248	14.698	18.123	26.523	48.193	14.472	174.025	17.707	23.423	29.858	29.382	77.001	123.366	56.837	0
With_rot_no_Sym	23.1251	20.6936	16.7991	22.3811	24.2276	39.6999	34.2484	47.1613	104.392	59.5404	0	130.547	57.6645	52.5961	0	172.432	123.29	102.59	118.471	0	90.3944	0	110.603	130.306	123.768	119.363	0	0	0	0

Figure 6: Results of SAT

4 SMT Model

In computer science and mathematical logic, satisfiability modulo theories (SMT) is the problem of determining whether a mathematical formula is satisfiable. An SMT instance is a generalization of a Boolean SAT instance in which various sets of variables are replaced by predicates from a variety of underlying theories. SMT formulas provide a much richer modeling language than is possible with Boolean SAT formulas[5]. The difference is that SMT solvers take systems in arbitrary format, while SAT solvers are limited to Boolean equations in CNF1 form

Z3 and Python In the solution of VLSI Circuit arrangement solution, we have used the Z3 library in Python. Z3 is a theorem prover from Microsoft Research with support for bit-vectors, booleans, arrays, floating point numbers, strings, and other data types. Z3 is used for both SAT as we did in our previous solution and SMT-based solution approaches. In the first one, we only use boolean atoms and their CNF form of clauses to solve any problem.

Whereas in SMT, it is possible to solve a problem using a domain-based method.

4.1 Decision Variables

To solve this problem with SMT, first, we need to define the variables we need to model our problem. It is possible to create variables of different kinds such as boolean, constants, real numbers, integers, BITwise vectors, floating point values, and arrays. The variables used in this solution are mainly integers and are stated as have already been discussed in the CP section (2.1).

4.2 Objective Function

The goal of this project was:
(a) to find optimal assignments to all circuit positions and:
(b) to minimize the maximum height of the VLSI board. Therefore, we used *model.optimize()* Z3 function to define our objective function. This instructs the solver to minimize the variable assigned to it.

```
opt = Optimize()
opt.minimize(height)
```

4.3 Constraints

4.3.1 Constraint 1: Non-Overlapping Constraint

In SMT we have implemented the same constraint for non-overlap as discussed in SAT section. Where we create additional variables such as Zlr and Zud which define positional relations between the circuits. After that, we used the same mathematical expressions to encode them as discussed in SAT non-overlap constraint section.

4.3.2 Constraint 2: Domains of x_i and y_i

The domains of x_i and y_i are declared as follows:

$$\begin{aligned} D(x_i) &= \{\ c\ \in\ \mathbb{N} \mid 0\ \leq c\ \leq W - w_i\} \\ D(y_i) &= \{\ c\ \in\ \mathbb{N} \mid 0\ \leq c\ \leq H - h_i\} \end{aligned}$$

And the domain for height is as follows:

$$D(H)\ =\ \{\ H\ \in\ \mathbb{N} \mid h_{min}\ \leq H\ \leq h_{max}\}$$

4.3.3 Symmetry Breaking Constraints

Constraint 3: Cumulative Constraint In mini-zinc, we used a global cumulative constraint to solve this problem. A similar approach has been implemented in this part also. The cumulative constraint is implemented twice once for the x-axis and once for the y-axis. This is, as mentioned earlier, an implied constraint, but it has been added because it improves propagation times.

Constraint 4: Two Biggest circuits: For a pair of largest circuits in terms of area, we restrict their relative position that can create symmetrical solutions. This is done by following a lexicographical Order between the pair such as:

$$x_{large1} > x_{large2} \vee (x_{large1} == x_{large2} \wedge y_{large1} \leq y_{large2})$$

Constraint 5: Two circuits with large enough width If there are two circuits let's say i and j, such that their sum of widths is greater than the allowed plate width, they can not be placed horizontally aligned. For all pairs of circuits (i,j);

$$\begin{cases} \neg Zud_{j,i} \vee \neg Zud_{i,j} & \text{if } w_i + w_j > W \\ Zud_{i,j} \vee Zud_{j,i} \vee Zlri, j \vee Zlrj, i & \text{otherwise} \end{cases} \tag{26}$$

Constraint 6: Symmetry breaking constraint for same size circuits If there are two circuits let's say i and j, such that their widths are equal and so are the heights, then we can define them to be positioned in a specific manner with respect to each other. we define that i^{th} circuit should be left of j or below j. For all pairs of circuits (i,j);

$$\begin{cases} \neg Zlr_{i,j} \vee \neg Zud_{i,j} & \text{if } (w_i == w_j, h_i == h_j) \\ Zud_{i,j} \vee Zud_{j,i} \vee Zlri, j \vee Zlrj, i & \text{otherwise} \end{cases} \tag{27}$$

4.4 Rotation

In this part of the problem, we allow the rotation of circuits which means the optimizer can rotate a circuit by a degree of 90 by swapping their respective height and width. For this, a new array with the name rotation is created to apply the new possibilities. All the constraints remain the same as the previous model without the rotations with this addition.

$$\text{for all i} \in \{\ 1..\text{n}\}:$$
$$w\_r_i = if(rotation_i == True, h_i, w_i)$$
$$h\_r_i = if(rotation_i == True, w_i, h_i)$$

4.5 Validation

Experimental design In this model, we created the decision variables in integer and boolean forms. The x and y positional coordinates are integer vectors, whereas the Zlr and Zud are boolean variables. The input variables such as w and h are integer lists. In this SMT approach, we have made observations based on four different models:

(a) a general model,
(b) a general model with symmetry breaking,
(c) the general model with rotations allowed, and
(d) a General model with rotations allowed as well as symmetry breaking constraints.
The basic constraints defined on these variables in the model(a) are Domain reduction/bound and non-overlap constraints along with two cumulative constraints. In model (b) we have added symmetry-breaking constraints as discussed earlier. This was done to observe the effect of introducing symmetry-breaking constraints in the original model. In model (c) rotation of the circuits was allowed by introducing additional decision variables $rotation_i$ and new lists of w_r_i and h_r_i as the updated heights and widths of respective circuits. The last model (d) was one with rotations allowed as well as symmetry-breaking constraints. The solver used is a Z3 solver with python in google colab. There was a time limit set for 300 seconds, which means any solutions obtained past this time limit are not part of the results shown in the next section.

Experimental results The solutions of all instances found by all four model versions have been shown in table 3. The runtimes can be observed from the graphs.

Table 3: Results of SMT

ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	Total
model (a)	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	n/a	30	31	32	33	34	35	36	n/a	38	n/a	40	40	40	40	n/a	n/a	n/a	n/a	33
model (b)	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	n/a	38	n/a	40	40	40	40	n/a	n/a	n/a	n/a	34
model (c)	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	n/a	24	26	n/a	n/a	29	n/a	n/a	31	n/a	n/a	34	n/a	n/a	n/a	n/a	n/a	n/a	n/a	40	40	n/a	n/a	n/a	n/a	22
model (d)	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	n/a	n/a	n/a	n/a	30	31	n/a	n/a	34	n/a	36	n/a	38	n/a	40	40	40	40	n/a	n/a	n/a	n/a	27
optimal height	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	n/a	38	n/a	40	40	40	40	n/a	n/a	n/a	n/a	34

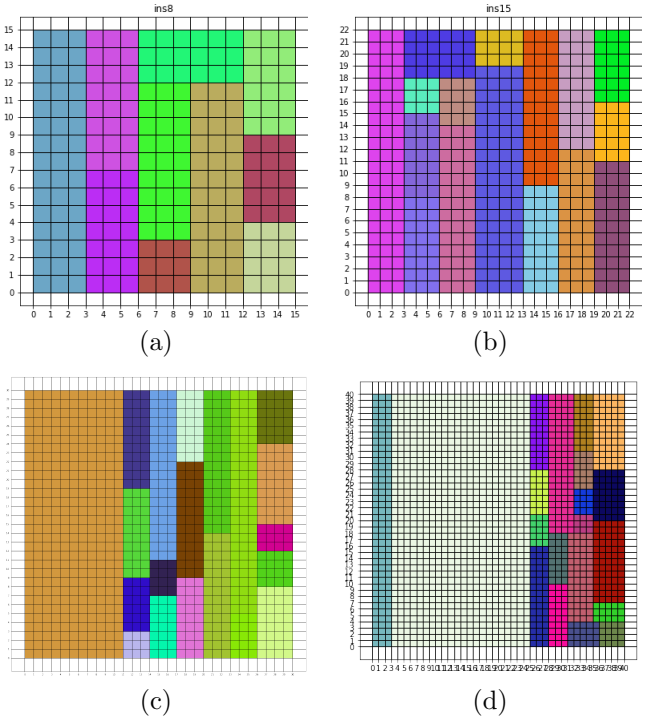
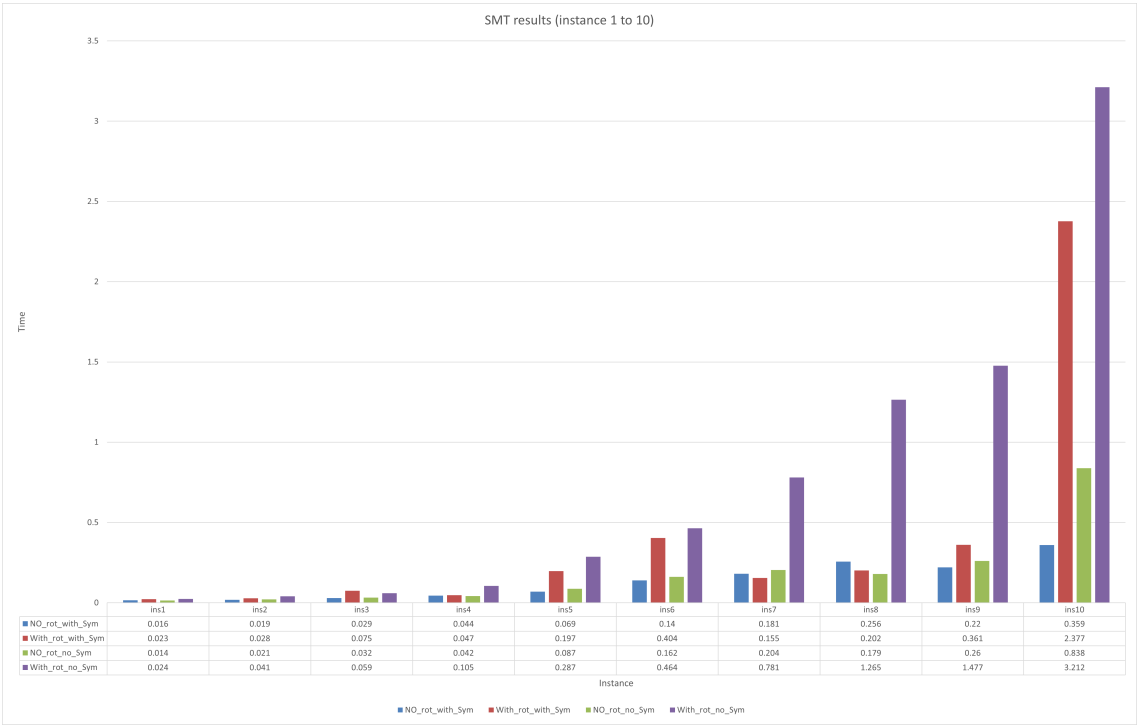


Figure 7: Validation



	ins1	ins2	ins3	ins4	ins5	ins6	ins7	ins8	ins9	ins10
NO_rot_with_Sym	0.016	0.019	0.029	0.044	0.069	0.14	0.181	0.256	0.22	0.359
With_rot_with_Sym	0.023	0.028	0.075	0.047	0.197	0.404	0.155	0.202	0.361	2.377
NO_rot_no_Sym	0.014	0.021	0.032	0.042	0.087	0.162	0.204	0.179	0.26	0.838
With_rot_no_Sym	0.024	0.041	0.059	0.105	0.287	0.464	0.781	1.265	1.477	3.212

Figure 8: Results of SMT

5 MIP Model

Mixed integer programming is a mathematical solution modeling that derives from LP. The objective is to use linear equations to cut hyperspaces of an n-dimensional space in order to limit the search area. This modeling is often used in optimization problems where the optimization function clearly shows a direction to follow in the n-space. MIP uses techniques starting from the same LP problem to find a non-boolean solution to variables and refining them to integer values

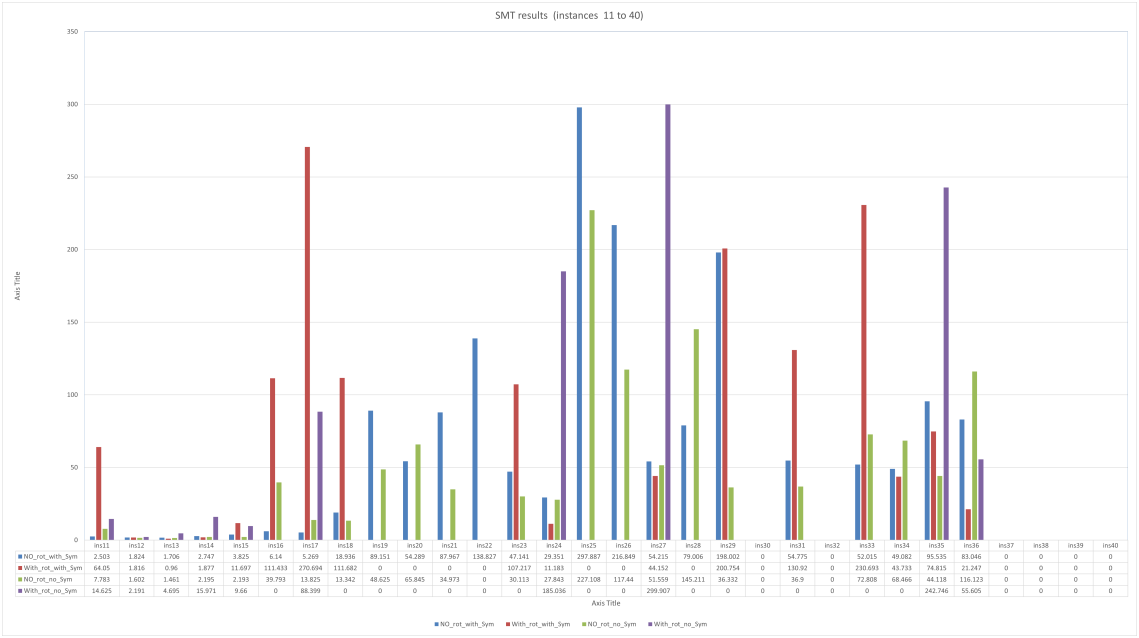


Figure 9: Results of SMT

5.1 Decision Variables

In MIP integer and boolean variables can be used in addition to floating variables of normal LP modeling. Compared with other models presented, the incapacity of a linear function to express some non-linear functions such as OR and IF conditions is bypassed using auxiliary variables as shown in the following sections. The chosen variables are all integer, while auxiliary variables are often Boolean which are treated as 0-1 integer values in the constraints. With a bar above a boolean variable, it's negation is intended ($\bar{r}_i = not(r_i)$):

1. **X,Y coordinate position -** (x_i, y_i): presented in the Introduction 1, common to other models

2. **Rotation -** r_i: presented in Introduction 1. In the model without rotation, this variable is not used assuming a value of 0.

3. **Plate height - H**: As in Introduction 1. With H^* is intended the lower bound of H, also called "ideal H"

4. **Overlap variables -** z_1, z_2, z_3, z_4:These Boolean variables are used to ensure no overlapping between the circuits and express the relative position of two circuits, respectively represent the four sides of the considered circuit: left, right, top, bottom. That differs from other implementations because we need a set of 4 for each pair of circuit to define their relative position due to impossibility of using OR junctions. Each pair of circuits has only one set of this four variables, so $z_{1,ij} = z_{2,ji}$, $z_{2,ij} = z_{1,ji}$ and so on. A 0-value means that the other circuit is positioned left-sided (right, top, or bottom-sided) which respect to the considered one (which is inverse respect to Introduction 1 definition of similar variables due to constraint that are implemented). A combination of the variables can represent all possible situations like top-left relative position or overlap between the two circuits. In the last case, they would be all equal to one.

5.2 Objective Function

The goals to be reached are the same as CP MODEL 2.2. In the case of MIP The objective function has to be linear and is set as:

$$f = H$$

5.3 Constraints

Cumulative Constraints In the MIP part, the cumulative constraints are not efficient to reproduce due to the big number of variables needed to introduce in order to compute the sum for each x position. Bound constraints are implemented instead

Bound constraints To ensure circuits inside the circuit plate two constraints are implemented:

$$x_i + w_i\bar{r_i} + h_i r_i \leq W \tag{28}$$

$$y_i + h_i\bar{r_i} + w_i r_i \leq H \tag{29}$$

The first constraint is only written to stress it, but not really implemented because is already satisfied by the bounds on the x variables

Non-overlap constraints To ensure circuits do not overlap each other the constrains are implemented using the big M method:

$$x_j + w_j\bar{r_j} + h_j r_j \leq x_i + Mz1_{ij} \tag{30}$$

$$x_i + w_i\bar{r_i} + h_i r_i \leq x_j + Mz2_{ij} \tag{31}$$

$$y_i + h_i\bar{r_i} + w_i r_i \leq y_j + Mz3_{ij} \tag{32}$$

$$y_j + h_j\bar{r_j} + w_j r_j \leq y_i + Mz4_{ij} \tag{33}$$

$$z1_{ij} + z2_{ij} + z3_{ij} + z4_{ij} \leq 3 \tag{34}$$

where M is an arbitrary big number that ensures the constraint is always satisfied (or always not satisfied) when present

5.4 Symmetry breaking constraints

Lexicographic constraints of two biggest circuits To reduce the search space a lexicographic constrain over the two biggest circuits $n1$, $n2$ is added:

$$z1_{n1,n2} = 0, \quad z4_{n1,n2} = 0 \tag{35}$$

Lexicographic constraints of similar circuits Circuits with the same dimensions are exchangeable, creating symmetries. To avoid that a lexicographic order is applied to them. For MIP the z_s variables are used to create much simpler constraints:

$$\forall S \quad \forall i,j \in S \qquad z1_{ij} = 0, \quad z4_{ij} = 0 \tag{36}$$

where

$$S_l = \{circuits| \quad w_i\bar{r_i} + h_i r_i = w_j\bar{r_j} + h_j r_j \\ \bigwedge h_i\bar{r_i} + w_i r_i = h_j\bar{r_j} + w_j r_j\}$$

Square circuit constraint As in Symmetry breaking 2.5. MIP implementation is: if $w_i == h_i$

$$r_i = 0$$

Pushing circuits to the right The main theory behind is presented Symmetry breaking constraints 2.5 In MIP it is implemented thanks to an auxiliary binary variable s_side which is true if the circuits belong to the right half:

$$x_i + \frac{w_i\bar{r_i} + h_i r_i}{2} \leq \frac{W}{2} + \frac{W}{2} side_i \tag{37}$$

$$x_i + \frac{w_i\bar{r_i} + h_i r_i}{2} \geq \frac{W}{2} - \frac{W}{2} \bar{side_i} \tag{38}$$

$$n \leq \sum_i 2side_i \leq 2n \tag{39}$$

Pushing big circuits to the left To increase the tightness of the above constraint this constraint can be added which exploits the B biggest circuits. The objective is to have a higher concentration of big circuits in the right half with respect to the left half

$$BIG\_L : \{circuits \in BIG | x_i + \tfrac{w_i\bar{r_i} + h_i r_i}{2} \leq \tfrac{W}{2})\}$$
$$BIG\_R : \{circuits \in BIG | x_i + \tfrac{w_i\bar{r_i} + h_i r_i}{2} \geq \tfrac{W}{2})\}$$

$$\sum_{i \in BIG\_L} 1 \geq \sum_{j \in BIG\_R} 1 \tag{40}$$

Where BIG is the set of B biggest circuits. In order to choose B, the number b of biggest circuits which area summation is less than half of the ideal area is searched and multiplied by 2. The minimum value of b should be 1 for the case in which the biggest circuit has area larger than half of the ideal plate.

$$\sum_{i \in sorted\_circuits}^{b} area_i \leq \frac{WH^*}{2} \tag{41}$$

$$sorted\_circuits : \{circuits | area_i \geq area_{i+1}\}$$
$$BIG : sorted\_circuits[1..2b]$$

The implementation in MIP uses the same variables *side* needed to the previous constraint

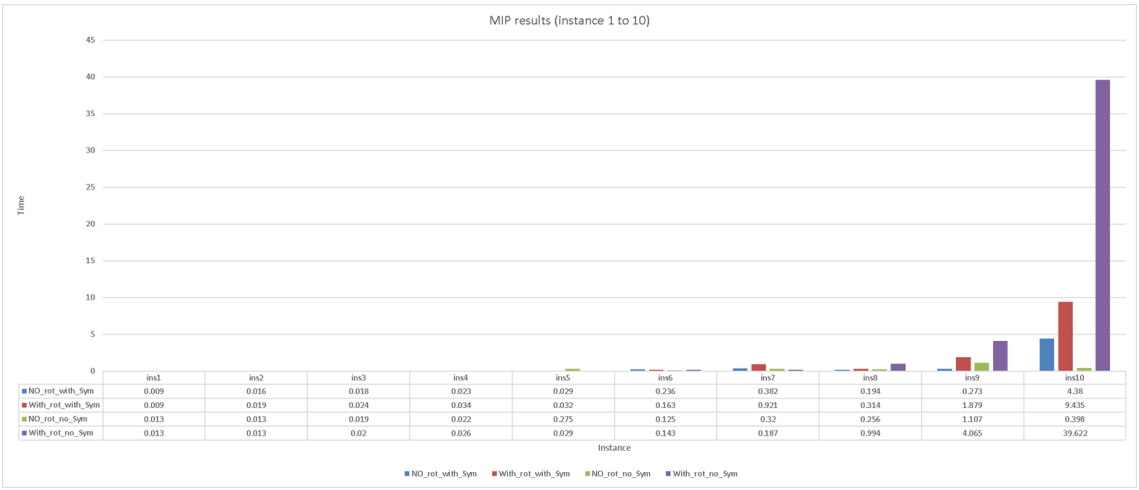
$$0 \leq \sum_{i \in BIG} 2side_i \leq n(BIG) \tag{42}$$

This constraint removes some of the optimal solutions but not all of them and reduce considerably the search space. Considering the worst-case scenario where all circuits have the same area, this constraint and the gradient one reduce the search space to contain only one optimal solution where $n(L) = n(R)$.

5.5 Validation

Experimental design MIP was implemented on python 3.9 using the free package ortools-python [6] that is an API that help interfacing with different solvers. In particular it was used SCIP solver throughout the API. The coding was done using Visual Studio and the windows console. The testing were made on a laptop with the following specification: processor Intel i7 octa-core, 16GB of RAM and VRAM 6GB (not used). Earlier test were done with a much lower performance machine and it wasn't recorder any big increase in the model performances between the two machines. Results were double-checked with a visual representation. Experiments were done by running the model on single more or less difficult instances and then validated against all instances in a consecutive workload. Different version of symmetry breaking constrains were tested, like "pushing circuits to the top" instead of "pushing circuits to the right" but only the ones with best performances are presented.

Experimental results Results reported are for all the combination of model with or without considering the rotation of circuits and with or without adding symmetry breaking constraints. A time limit of 300s was set to the solver in concordance with other model tested. It can be seen that adding symmetry breaking constraints increase the performance of the model in the number of solved instances. Solution of instance 12 using with_rot_no_Sym model has to be taken as solved by chance, probably due to the solver problem approach in exploring the space. As overall observation, model no_rot_with_sym perform better than his counterpart with rotation. The capability of rotate greatly increase the search space.



	ins1	ins2	ins3	ins4	ins5	ins6	ins7	ins8	ins9	ins10
NO_rot_with_Sym	0.009	0.016	0.018	0.023	0.029	0.236	0.382	0.194	0.273	4.38
With_rot_with_Sym	0.009	0.019	0.024	0.034	0.032	0.163	0.921	0.314	1.879	9.435
NO_rot_no_Sym	0.013	0.013	0.019	0.022	0.275	0.125	0.32	0.256	1.107	0.398
With_rot_no_Sym	0.013	0.013	0.02	0.026	0.029	0.143	0.187	0.994	4.065	39.622

Figure 10: Results of MIP

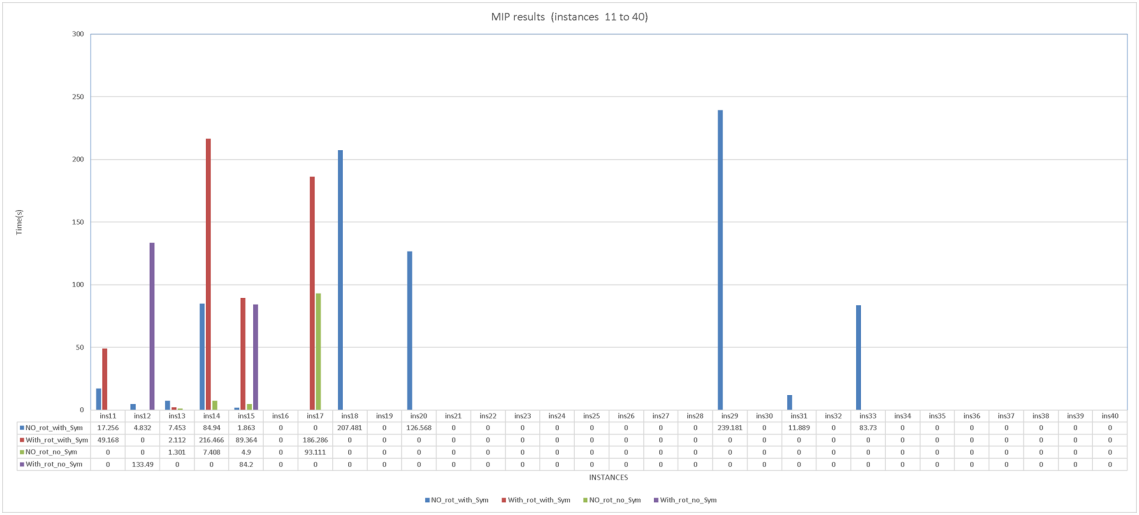


Figure 11: Results of MIP

6 Conclusions

The presented work solved overall 39 instances with the 40^{th} instance resulting unsolved within the time bound. All Solutions found are optimal solutions and none of the instances resulted to be unsatisfiable. Results from all the models are summarized in the graph depicted lin Figure 12 and in the table reported in Tab 4.

1. SAT model achieved best results against the test instances with 39 out of 40 instances completed
2. CP model solved 37 solutions
3. SAT model solved 35 solutions
4. MIP performance are the worst compared to them with only 22 out of 40. That is due to the MIP intrinsic modelling which is very good with wide variables bounds but less optimal with high number of small variables.

Figure 12: Overall solutions by each model compared

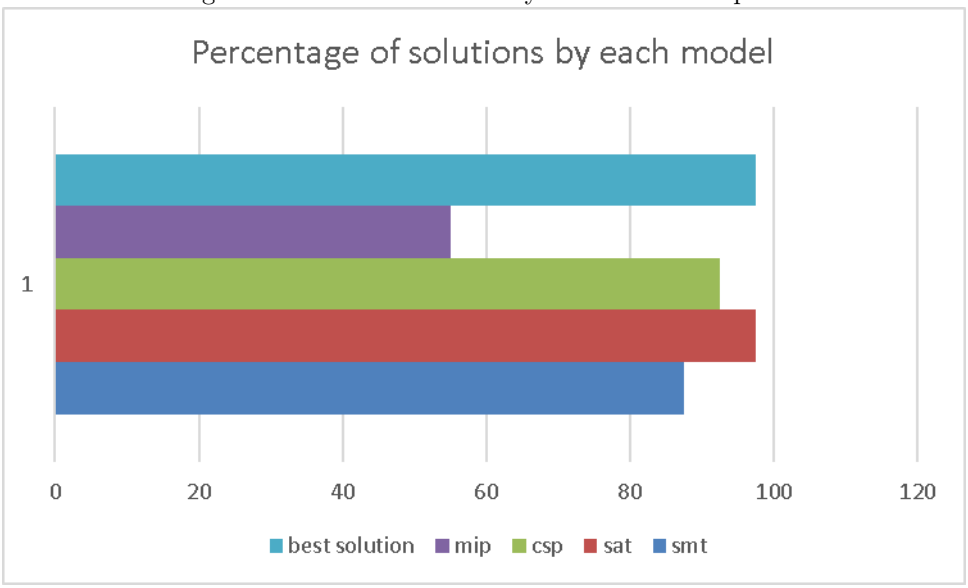


Table 4: Optimal solutions found by each model

ID	smt	sat	csp	mip	best solution
1	**8**	**8**	**8**	**8**	**8**
2	**9**	**9**	**9**	**9**	**9**
3	**10**	**10**	**10**	**10**	**10**
4	**11**	**11**	**11**	**11**	**11**
5	**12**	**12**	**12**	**12**	**12**
6	**13**	**13**	**13**	**13**	**13**
7	**14**	**14**	**14**	**14**	**14**
8	**15**	**15**	**15**	**15**	**15**
9	**16**	**16**	**16**	**16**	**16**
10	**17**	**17**	**17**	**17**	**17**
11	**18**	**18**	**18**	**18**	**18**
12	**19**	**19**	**19**	**19**	**19**
13	**20**	**20**	**20**	**20**	**20**
14	**21**	**21**	**21**	**21**	**21**
15	**22**	**22**	**22**	**22**	**22**
16	**23**	**23**	**23**	**0**	**23**
17	**24**	**24**	**24**	**24**	**24**
18	**25**	**25**	**25**	**25**	**25**
19	**26**	**26**	**26**	n/a	**26**
20	**27**	**27**	**27**	**27**	**27**
21	**28**	**28**	**28**	n/a	**28**
22	**29**	**29**	**29**	n/a	**29**
23	**30**	**30**	**30**	n/a	**30**
24	**31**	**31**	**31**	n/a	**31**
25	**32**	**32**	**32**	n/a	**32**
26	**33**	**33**	**33**	n/a	**33**
27	**34**	**34**	**34**	n/a	**34**
28	**35**	**35**	**35**	n/a	**35**
29	**36**	**36**	**36**	**36**	**36**
30	n/a	**37**	**37**	n/a	**37**
31	**38**	**38**	**38**	**38**	**38**
32	n/a	**39**	n/a	n/a	**39**
33	**40**	**40**	**40**	**40**	**40**
34	**40**	**40**	**40**	n/a	**40**
35	**40**	**40**	**40**	n/a	**40**
36	**40**	**40**	**40**	n/a	**40**
37	**60**	**60**	61	n/a	**60**
38	n/a	**60**	n/a	n/a	**60**
39	n/a	**60**	61	n/a	**60**
40	n/a	n/a	n/a	n/a	n/a
	35	**39**	**37**	**22**	39

References

[1] Nestor M. Cid-Garcia and Yasmin A. Rios-Solis. Exact solutions for the 2d-strip packing problem using the positions-and-covering methodology. *PLoS ONE*, 16, 2021.

[2] Minizinc. 4.2.1. All Different constraint in Global constraints - The MiniZinc Handbook 2.5.3 4.2.1. global constraints¶.

[3] Minizinc. 4.2.1. Geostbb Constraint in Global constraints - The MiniZinc Handbook 2.5.3 4.2.1. global constraints¶.

[4] Takehide Soh, Katsumi Inoue, Naoyuki Tamura, Mutsunori Banbara, and Hidetomo Nabeshima. A sat-based method for solving the two-dimensional strip packing problem. *Fundam. Inform.*, 102:467–487, 01 2010.

[5] Dennis Yurichev. *SAT/SMT by Example*. 2022.

[6] Google. Integer Optimization— OR-Tools — Google developers¶.