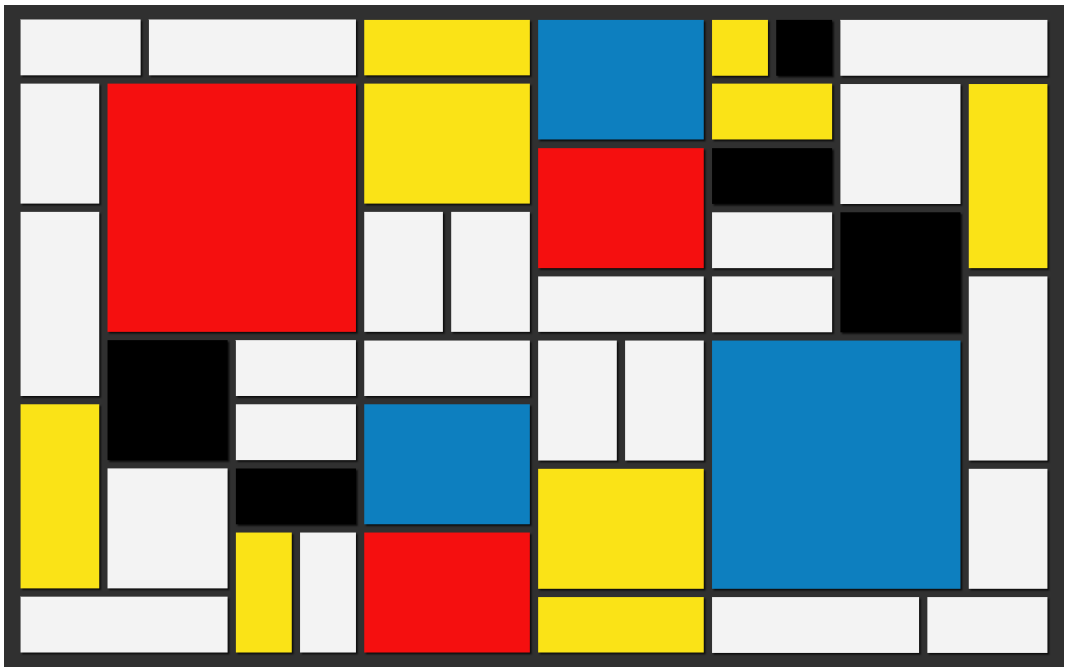
Very Large Scale Integration

SAT



Flavio Pinzarrone flavio.pinzarrone@studio.unibo.it
Enrico Pallotta enrico.pallotta@studio.unibo.it
Giuseppe Tanzi giuseppe.tanzi@studio.unibo.it

Alma Mater Studiorum - University of Bologna

Contents

1 Introduction

VLSI (Very Large Scale Integration) refers to the trend of integrating circuits into silicon chips. A typical example is the smartphone. The modern trend of shrinking transistor sizes, allowing engineers to fit more and more transistors into the same area of silicon, has pushed the integration of more and more functions of cellphone circuitry into a single silicon die (i.e.plate).
So, given a fixed-width plate and a list of rectangular circuits, decide how to place them on the plate so that the length of the final device is minimized(improving its portability).
Consider two variants of the problem.
In the **first**, each circuit must be placed in a **fixed orientation** with respect to the others. This means that, an $n \times m$ circuit cannot be positioned as an $m \times n$ circuit in the silicon plate.
In the **second case**, the **rotation is allowed**, which means that an $n \times m$ circuit can be positioned either as it is or as $m \times n$.
This problem is also known in literature as the ***"strip packing problem"***, a 2-dimensional geometric minimization problem. Given a set of axis-aligned rectangles and a strip of bounded width and infinite height, determine an overlapping-free packing of the rectangles into the strip **minimizing its height**.

The **SAT** approach proposed in this project is an implementation of the paper "A SAT-based Method for Solving the Two-dimensional Strip Packing Problem (2SPP)" by Takehide Soh et al [1].
In this paper the approach to tackle the 2SPP consists in considering an alternative version of the problem in which both the maximum width and height are given (two-dimensional orthogonal packing problem 2OPP).
Since with SAT solvers we can only determine the satisfiability of a given problem, to get to optimal height of a 2SPP we solve a sequence of 2OPPs with increasing maximum height.

Algorithm 1 Solve 2SPP

```
h ← lower_bound
while h ≤ upper_bound ∧ ¬solver.is_sat() do
    instatiate_model(h) % 2OPP with fixed height
    solve_model()
    h ← h + 1
end while
```

2 Problem formulation

Here we define some notation that will be used in the description of the model, listing below the parameters (which can be read from the instance files) and the variables we used.
We will strictly follow the mathematical notation of the paper.

2.1 Problem parameters

- $n \in \mathbf{N}$ number of circuits to place
- $W \in \mathbf{N}$ maximum plate width
- $w_i \in \mathbf{N}$ width of the i^{th} circuit
- $h_i \in \mathbf{N}$ height of the i^{th} circuit

2.2 Decision variables

- x_i bottom left x coordinate of the i^{th} circuit
- y_i bottom left y coordinate of the i^{th} circuit

2.3 Objective function

The objective function of this problem is to minimize H. As we said, the optimization of the height is done by solving a sequence 2OPPs, so we defined the lower bound and the upper bound of H as follows:

- The lower bound for H is obtained when there aren't no empty spaces among the circuits and all the plate is fitted

$$lb = \frac{\sum_i h_i * w_i}{W}$$

- The upper bound for H is obtained where all the circuits are on a single column:

$$ub = \sum_i h_i$$

Hence:

$$lb \leq H \leq ub \tag{1}$$

2.4 Main problem constraints

In this section we provide a description of the basic constraints which need to be enforced in order to find a solution.
Basically there are two main constraints:

2.4.1 Domain constraint

Each circuit must be placed in a position (x_i, y_i) such that it does not exceed the bounds.

$$D(x_i) = \{v \in \mathbf{N} \mid 0 \leq v \leq W - w_i\} \; \forall i \in \{1, ..., n\} \tag{2}$$

$$D(y_i) = \{v \in \mathbf{N} \mid 0 \leq v \leq H - h_i\} \; \forall i \in \{1, ..., n\} \tag{3}$$

2.4.2 No overlapping

Circuits should not overlap with each other.

$$\bigwedge_{i,j \in [1,n] \;\mid\; i<j} (x_i + w_i \leq x_j) \;\vee (x_j + w_j \leq x_i) \;\vee (y_i + h_i \leq y_j) \;\vee (y_j + h_j \leq y_i) \tag{4}$$

3 SAT encoding

The translation of the 2OPP into a SAT problem is done using ***order encoding***.

3.1 Atoms encoding

$\forall i, j \mid i \neq j$ and $e \in \{0, ..., W - w_i\}$, $f \in \{0, ..., H - h_i\}$ we define the following atoms:

- $lr_{i,j}$: true if the i^{th} circuit is placed on the left of the j^{th} one.
- $ud_{i,j}$: true if the i^{th} circuit is placed under the j^{th} one.
- $px_{i,e}$: true if $x_i \leq e$
- $py_{i,f}$: true if $y_i \leq f$

Before starting with the encoding of constraints, lets do a quick analysis of the size of this encoding. Moreover, a consideration on the number of clauses will be presented afterwards. Considering the variables used to indicate the relative position between a pair of circuits (lr and ud) we have $n^2 + n^2$ variables.
At this value we have to sum also the variables used to encode the exact position of a circuit, for px we have

$$\sum_{i=1}^{n}(W - w_i + 1) = N * W * \sum_{i=1}^{n}(-w_i + 1)$$

which can be simplified to $n * W$, and also for py we can assume to have $n * H$ variables. The total size of this encoding is

$$n^2 + n^2 + n * W + n * H = 2n^2 + (W + H)n$$

That can be considered as an $O(n^2)$.

3.2 Constraints encoding

Using the atoms defined above we can thus implement both domain and no overlapping constraint in the following way:

1. **Domain constraint** (2) (3): For each i^{th} circuit we add the clauses:

$$\bigwedge_{e=0}^{W-w_i-1} (\neg px_{i,e} \vee px_{i,e+1})$$

$$\bigwedge_{f=0}^{H-h_i-1} (\neg py_{i,f} \vee py_{i,f+1})$$

   For example, given a circut with $(w_i, h_i) = (3, 2)$ and $(W, H) = (5, 5)$ we have:

$$(\neg px_{i,0} \vee px_{i,1}) \wedge (\neg px_{i,1} \vee px_{i,2})$$

$$(\neg py_{i,0} \vee py_{i,1}) \wedge (\neg py_{i,1} \vee py_{i,2}) \wedge (\neg py_{i,2} \vee py_{i,3})$$

   The paper does not explicit it, but the following constraint is needed in order to force the solver to place all the circuits:

$$\bigwedge_{e=W-w_i}^{W-1} px_{i,e}$$

$$\bigwedge_{f=H-h_i}^{H-1} py_{i,f}$$

2. **No overlapping** (4) For each pair of circuits such that $i \neq j$ we add the following constraints:

- $lr_{i,j} \vee lr_{j,i} \vee ud_{i,j} \vee ud_{j,i}$
- $\bigwedge_{e=0}^{W-w_i-1} \neg lr_{i,j} \vee px_{i,e} \vee \neg px_{j,e+w_i}$
- $\bigwedge_{e=0}^{W-w_i-1} \neg lr_{j,i} \vee px_{j,e} \vee \neg px_{i,e+w_j}$
- $\bigwedge_{f=0}^{H-h_i-1} \neg ud_{i,j} \vee py_{i,f} \vee \neg py_{j,f+h_i}$
- $\bigwedge_{f=0}^{H-h_i-1} \neg ud_{j,i} \vee py_{j,f} \vee \neg py_{i,f+h_j}$

Considering the constraints above, the number of clauses of a SAT-encoded 2OPP are $O(n^2)$, as reported in [1].

3.3 Symmetry breaking

The paper also proposes four techniques to reduce the search space. Here, we decided to implement three of them, according to the best result shown in the paper, section 5.

1. **Domain reduction** : reduces the domain of the rectangle with maximum area (with index m) defined by w_m and h_m to:

$$D(x_m) = \{a \in \mathbf{N} \mid 0 \leq a \leq \lfloor \frac{W - w_m}{2} \rfloor\} \tag{5}$$

$$D(y_m) = \{a \in \mathbf{N} \mid 0 \leq a \leq \lfloor \frac{H - h_m}{2} \rfloor\} \tag{6}$$

   By applying this reduction, if $w_i > \lfloor \frac{W-w_m}{2} \rfloor$ we can set $lr_{i,m} = false$ adding the clause $\neg lr_{i,m}$ ($\neg ud_{i,m}$ in the other direction).

2. **Large rectangles** : If we are given large rectangles which satisfy $w_i + w_j > W$, we can add the clause $\neg lr_{i,j} \wedge \neg lr_{j,i}$.

3. **Same rectangles** : If we are given rectangles with the same dimension $(w_i, h_i) = (w_j, h_j)$, we can fix the positional relation of rectangles. Thereby, we can add the clause $\neg lr_{j,i} \wedge (lr_{i,j} \vee \neg ud_{j,i})$.

3.4 Rotation encoding

The paper doesn't present a solution to the variant of the problem in which each circuit can be rotated of 90°.
By the way, we implemented a solution also for this variant following almost the same reasoning done for the *cp* model.
We added a variable for each circuit r_i : true if the i^{th} circuits it's rotated.
All the constraints are basically changed in this way:

- if the considered circuit can be rotated (i.e. $h_i \leq max\_width$) then the constraint is changed in a Xor like way, considering the normal constraint when r_i is false, and using h_i instead of w_i when r_i is true.
- if $h_i > max\_width$, the considered circuit cannot be rotated, add the normal constraints plus this one:

$$\neg r_i$$

  The latter constraint is added even in case of square circuit or if $w_i > H$.

Here there are the added constraints to handle the case in which a circuit can be rotated.

1. **Domain constraint** For each i^{th} circuit :
   - $\left(\bigwedge_{e=0}^{W-w_i-1} (\neg px_{i,e} \vee px_{i,e+1}) \wedge \neg r_i\right) \vee \left(\bigwedge_{e=0}^{W-h_i-1} (\neg px_{i,e} \vee px_{i,e+1}) \wedge r_i\right)$
   - $\left(\bigwedge_{f=0}^{H-h_i-1} (\neg py_{i,f} \vee px_{i,f+1}) \wedge \neg r_i\right) \vee \left(\bigwedge_{f=0}^{H-w_i-1} (\neg px_{i,f} \vee px_{i,f+1}) \wedge r_i\right)$
   - $\left(\bigwedge_{e=W-w_i}^{W} (px_{i,e}) \wedge \neg r_i\right) \vee \left(\bigwedge_{e=W-h_i}^{W} (px_{i,e}) \wedge r_i\right)$
   - $\left(\bigwedge_{f=H-h_i}^{H} (py_{i,f}) \wedge \neg r_i\right) \vee \left(\bigwedge_{f=H-w_i}^{H} (py_{i,f}) \wedge r_i\right)$
2. **No overlapping** For each pair of circuits such that $i \neq j$ we add the following constraints:
   - $\left(\bigwedge_{e=0}^{W-w_i-1} (\neg lr_{i,j} \vee px_{i,e} \vee \neg px_{j,e+w_i}) \wedge \neg r_i\right) \vee \left(\bigwedge_{e=0}^{W-h_i-1} (\neg lr_{i,j} \vee px_{i,e} \vee \neg px_{j,e+h_i}) \wedge r_i\right)$
   - $\left(\bigwedge_{e=0}^{W-w_j-1} (\neg lr_{j,j} \vee px_{j,e} \vee \neg px_{i,e+w_j}) \wedge \neg r_i\right) \vee \left(\bigwedge_{e=0}^{W-h_j-1} (\neg lr_{j,j} \vee px_{j,e} \vee \neg px_{i,e+h_j}) \wedge r_i\right)$
   - $\left(\bigwedge_{f=0}^{H-h_i-1} (\neg ud_{i,j} \vee py_{i,f} \vee \neg py_{j,f+h_i}) \wedge \neg r_i\right) \vee \left(\bigwedge_{f=0}^{H-w_i-1} (\neg ud_{i,j} \vee py_{i,f} \vee \neg py_{j,f+w_i}) \wedge r_i\right)$
   - $\left(\bigwedge_{f=0}^{H-h_j-1} (\neg ud_{j,i} \vee py_{i,f} \vee \neg py_{i,f+h_j}) \wedge \neg r_i\right) \vee \left(\bigwedge_{f=0}^{H-w_j-1} (\neg ud_{j,i} \vee py_{i,f} \vee \neg py_{i,f+w_j}) \wedge r_i\right)$
3. **Symmetry breaking**: In case of rotations, only the first part of the domain reduction constraint has been implemented (using again a XOR like notation), this is due to the fact that whenever we want to implement a constraint that relies on the comparison between shapes of two (or more) rectangles, the complexity of the constraint becomes huge.

3.5 Position extraction

Once the SAT solver has found a feasible solution for the model with height H, we need to extract the actual positions (x_i, y_i) for each circuit, converting back from order encoding the variables $px_{i,e}$ and $py_{i,f}$.
This conversion works by simply considering as value for the coordinate $x(/y)$ the **first** value of $e(/f)$ such that $px_{i,e}(/py_{i,f})$ is True.

4 Results and performances

Here we show the results obtained by the model, without using symmetry breaking clauses:

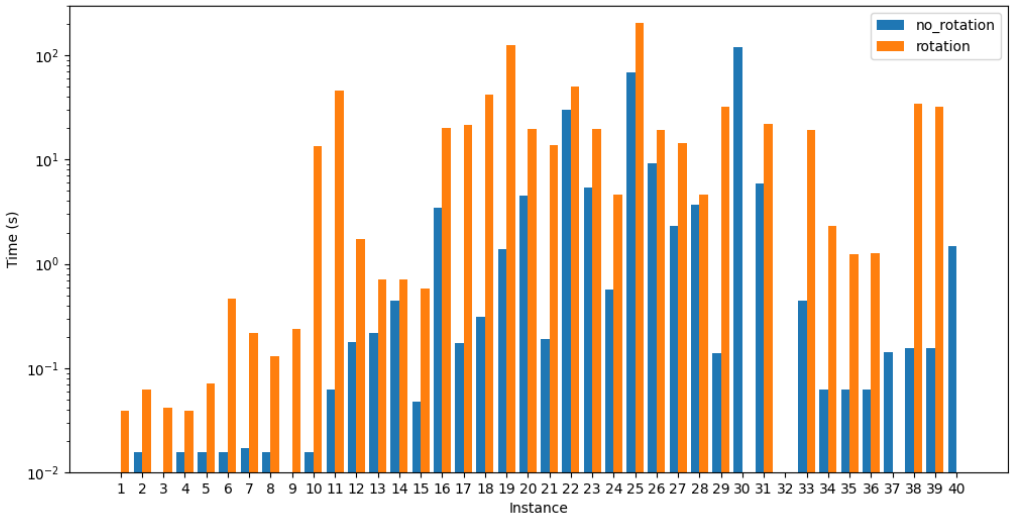


Figure 1: Performance without using symmetry breaking

n°	No-rot	Rot	n°	No-rot	Rot
1	0.00	0.04	21	0.19	13.83
2	0.02	0.06	22	30.18	50.50
3	0.00	0.04	23	5.35	19.57
4	0.02	0.04	24	0.56	4.59
5	0.02	0.07	25	69.02	206.03
6	0.02	0.46	26	9.28	19.29
7	0.02	0.22	27	2.32	14.36
8	0.02	0.13	28	3.72	4.59
9	0.00	0.24	29	0.14	32.12
10	0.02	13.50	30	119.05	—
11	0.06	45.88	31	5.96	21.97
12	0.18	1.72	32	—	—
13	0.22	0.70	33	0.44	19.45
14	0.44	0.71	34	0.06	2.31
15	0.05	0.58	35	0.06	1.24
16	3.47	20.22	36	0.06	1.27
17	0.17	21.38	37	0.14	—
18	0.31	42.08	38	0.16	34.18
19	1.38	126.48	39	0.16	32.13
20	4.53	19.65	40	1.49	—

Table 1: Results of model without symmetry breaking constraints (time in seconds)

The results are very good, this model is able to solve all instances except for the 32^{nd} one , considering the simple version of the problem with no rotation allowed. In case of rotation, of course, the time required to solve the instances increases and we were not able to solve the instances number : 30 - 32 - 37 - 40.

Here, instead, we can see the result obtained by the model that uses also symmetry breaking clauses described above.

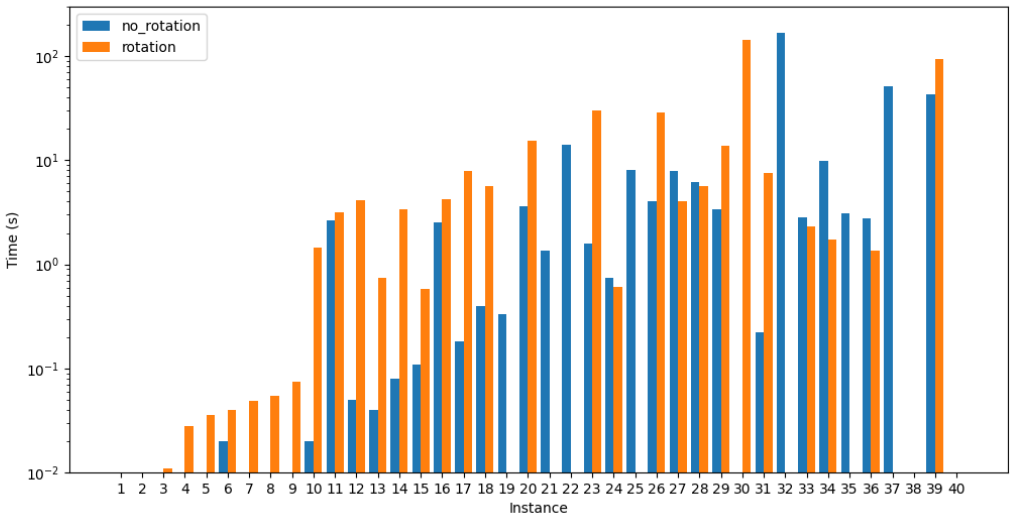


Figure 2: Performance using symmetry breaking

n°	No-rot	Rot	n°	No-rot	Rot
1	0.01	0.01	21	1.36	—
2	0.01	0.01	22	14.22	—
3	0.01	0.01	23	1.58	30.42
4	0.01	0.03	24	0.74	0.6
5	0.01	0.03	25	8.07	—
6	0.02	0.04	26	4.04	28.67
7	0.01	0.05	27	7.89	4.00
8	0.01	0.05	28	6.24	5.68
9	0.01	0.07	29	3.41	13.89
10	0.02	1.45	30	—	142.1
11	2.62	3.14	31	0.22	7.61
12	0.05	4.13	32	167.55	—
13	0.04	0.74	33	2.8	2.31
14	0.08	3.40	34	9.82	1.74
15	0.11	0.58	35	3.06	—
16	2.54	4.25	36	2.74	1.35
17	0.18	7.98	37	51.62	—
18	0.40	5.66	38	—	—
19	0.33	—	39	42.72	94.07
20	3.61	15.31	40	—	—

Table 2: Results of model with symmetry breaking constraints (time in seconds)

Introducing symmetry breaking constraints we were able to solve 37 out of 40 instances, the model fails to find a solution in the time limit for the instances 30, 38 and 40. Considering the case with rotation, we can see again a deterioration in performance as the model is slower and able to find only 31/40 solutions.
In the end, we selected as **best model** the one that **doesn't use symmetry breaking constraints**

since, as we can see from the table below, the simple model resulted to be more performing in both cases with and without rotation.

	Simple model	Sym Breaking model
No Rotation	**13.98**	30.95
Rotation	**49.29**	76.98

Table 3: Average time of all solutions of both models (time in seconds)

4.1 Hardware

All the tests were performed on a laptop equipped with an Intel i7-1185G7 CPU.

References

[1] 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.