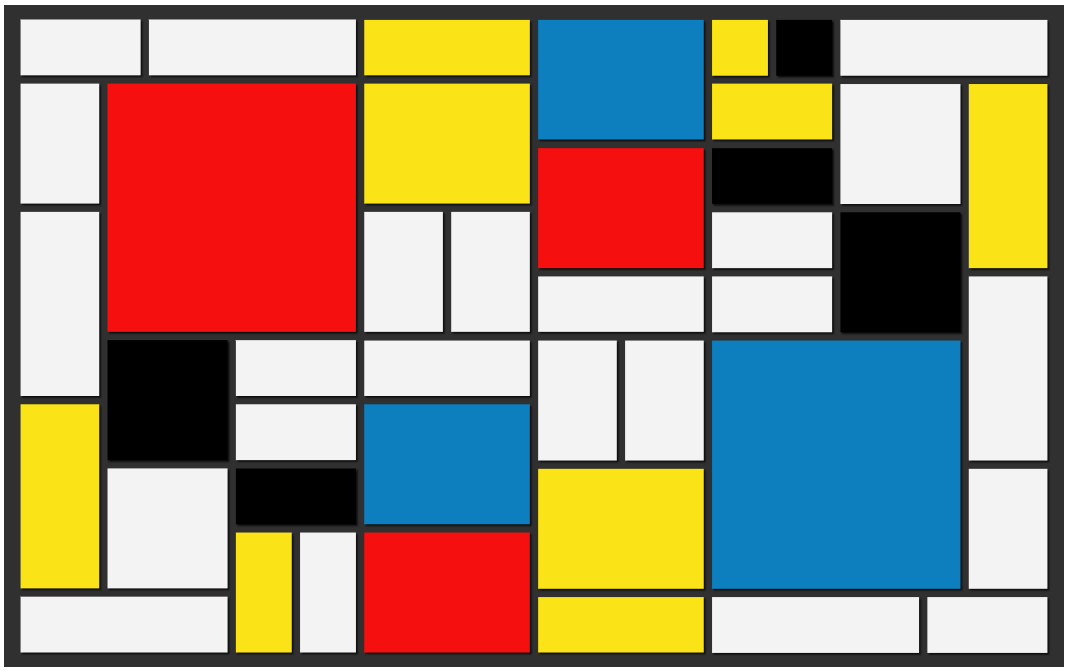
Very Large Scale Integration

Satisfiability Modulo Theory



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 $m \times n$.

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.

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

- H plate height, to be minimized
- 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. To reduce the search space, we can define 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 Constraints

Each circuit should not exceed the plate bounds, neither in width nor in height.

$$\bigwedge_{i=1}^{n} 0 \leq x_i \leq W - w_i \tag{2}$$

$$\bigwedge_{i=1}^{n} 0 \leq y_i \leq H - h_i \tag{3}$$

2.4.2 No overlapping constraint

Circuits should not overlap with each other.

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

2.5 Implied Constraints

Since we noticed that this problem can be seen as a task scheduling problem, the following cumulative constraint can be enforced.

2.5.1 Cumulative constraint over the columns

The parallelism with the scheduling problem is the following:

- the time axis is the plate width,
- tasks are circuits,
- task durations are circuits widths,
- task resource requirements are circuits heights,
- resource capacity is the plate height;

we enforced a cumulative constraint in our model as follows:

$$\sum_{i=1 \ | \ x_i \leq u < x_i + w_i}^{N} h_i \leq H \qquad u \in [1, W] \tag{5}$$

2.6 Symmetry breaking constraints

Scaling up with the problem dimension we noticed that the presence of symmetries in the solutions significantly slows down the search process. The first and most obvious symmetries to be identified are the ones obtained by flipping vertically or horizontally the plate. Nevertheless, performing some tests it came out that dealing with them doesn't result in an increase of performances. For this reason only the following symmetries have been considered when implementing the model.

- **One pair of rectangles**
  This symmetry breaking constraint is an implementation of a constraint in the paper "A SAT-based Method for Solving the Two-dimensional Strip Packing Problem (2SPP)" by Takehide Soh et al [1].
  We can fix the positional relation between only one pair of rectangles. We chose the pair of the biggest circuits, as they are the most difficult to place.

$$lex \leq ([y_1, x_1], [y_2, x_2]) \tag{6}$$

- **Rectangles of same dimension**
  This symmetry breaking constraint considers the case in which you have two rectangles of same dimension:
$$lex \leq ([x_1, y_1], [x_2, y_2]),\ w_1 = w_2 \wedge h_1 = h_2 \tag{7}$$

2.7 Rotation model

In order to handle the case in which there is the possibility for each circuit to be rotated, we had to manage the problem as efficiently as possible. For this reason we slightly changed the model. Furthermore, in this model, sorting by area of instances has led us to the best performance.

Decision variables

In addition to the decision variables described in section 2.2, we defined other variables:

- $rotation_i$ `True` if the i^{th} circuit is rotated, `False` otherwise
- $width_i$ effective width of the i^{th} circuit on the plate
- $height_i$ effective height of the i^{th} circuit on the plate

2.7.1 Main model constraints

In addition to the *no overlapping* constraint described in the model without rotation, we defined other constraints.
These two constraints force the rotation of a circuit if the i^{th} boolean variable of the `rotation` vector is `True`.

$$\bigwedge_{i=1}^{n} rotation_i \rightarrow width_i = h_i \ \wedge \ height_i = w_i \tag{8}$$

$$\bigwedge_{i=1}^{n} \neg rotation_i \rightarrow width_i = w_i \ \wedge \ height_i = h_i \tag{9}$$

2.7.2 Implied Constraints

After performing some tests, the cumulative over the columns didn't lead to any advantage, while the cumulative over the rows led us to better results, so we decided to implement only the cumulative over the rows.

Cumulative constraint over the rows

The parallelism with the scheduling problem is the following:

- the time axis is the plate height,
- tasks are circuits,
- tasks duration are circuits heights,
- tasks resource requirements are circuits widths,
- resource capacity is the plate width;

we enforced a cumulative constraint in our model as follows:

$$\sum_{i=1 \ | \ y_i \leq u < y_i + h_i}^{N} w_i \leq W \qquad u \in [1, H] \tag{10}$$

2.7.3 Symmetry Breaking Constraints

In addition to the symmetry breaking constraints described in the model without rotation, we defined another constraint:

- If a circuit is a square, its rotation is useless, so we forced it not to be rotated.

$$\bigwedge_{i=1}^{n} width_i = height_i \rightarrow \neg rotation_i \tag{11}$$

The figures below represent an example of the same instance solved both with circuit rotation and without.

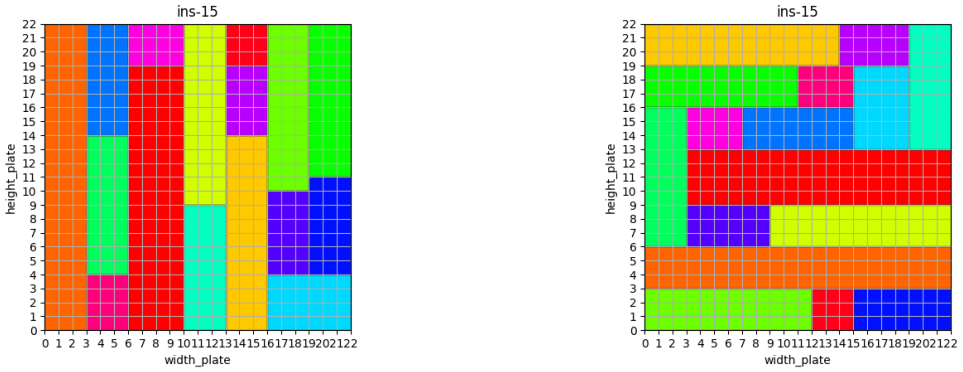


Figure 1: Instance 15 solution without (left) and with rotation (right)

3 SMT implementation

Taking inspiration from [1], we considered an alternative version of the problem in which both the maximum width and height are given (two-dimensional orthogonal packing problem 2OPP).
Since we noticed worst performance with the use of an *Optimizer*, 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
```

Hence, if we find a solution we will be sure that it will be the optimal one as we start from the lower bound of h.

3.1 Search strategy and solvers

To implement our SMT model we decided to use both Z3 Python API and SMT-LIB v2, a language which allows us to try different SMT solvers on the same model. Our solution is completely independent of the specific solver, indeed, after generating the model into the corresponding version of SMT-LIB, we tried Z3 and CVC5 to compare the performance of both solvers.
CVC5 [2] is an efficient open-source automatic theorem prover for Satisfiability Modulo Theories (SMT) problems. It can be used to prove the satisfiability (or, dually, the validity) of first-order formulas with respect to (combinations of) a variety of useful background theories. CVC5 is the successor of CVC4 and is intended to be an open and extensible SMT engine.
While Z3 [3] is an efficient Satisfiability Modulo Theories (SMT) solver from Microsoft Research. Z3 is a solver for symbolic logic, a foundation for many software engineering tools.

4 Results and performances

Using the models and the solvers we presented above we obtained the results shown in the graphs below.

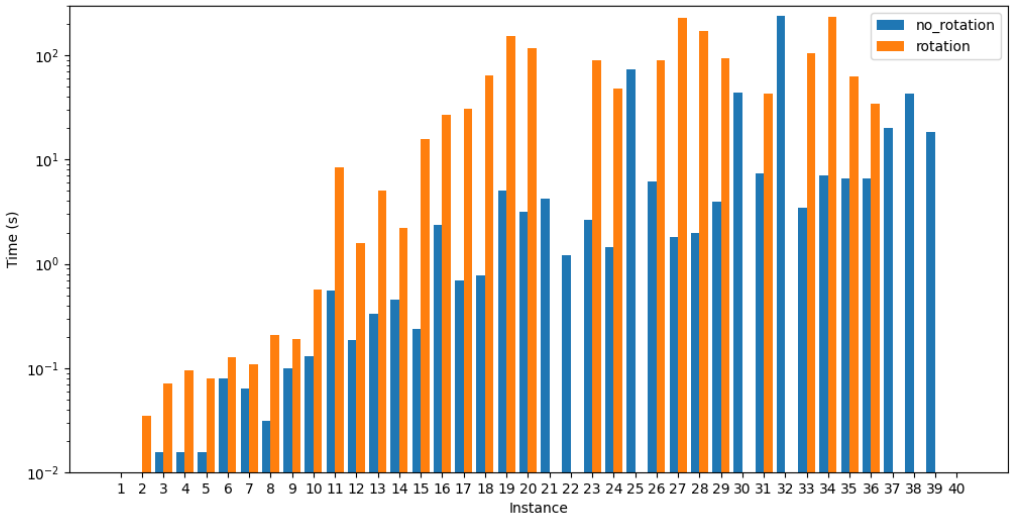


Figure 2: Performance of Z3

CVC5 wasn't able to solve many instances within the timeout; so we chose Z3 as our solver for the final model, since it obtained the best results, solving 39 instances with the model without rotation. While handling possible circuit rotations resulted in longer computation times and, in some cases, no optimal solution was found within the timeout. In particular, in the case where rotation wasn't allowed, it was able to solve all the instances except for the 40^{th}. On the other hand, enabling the possibility of rotating the circuits, it was able to solve all the instances except for the 21^{th}, the 22^{th}, the 25^{th}, the 30^{th}, the 32^{th}, the 37^{th}, the 38^{th}, the 39^{th} and the 40^{th}.

n°	No-rot	Rot	n°	No-rot	Rot
1	0.00	0.00	21	4.18	—
2	0.00	0.03	22	1.21	—
3	0.02	0.07	23	2.66	89.38
4	0.02	0.09	24	1.46	47.79
5	0.02	0.08	25	73.51	—
6	0.08	0.13	26	6.12	89.62
7	0.06	0.11	27	1.80	229.89
8	0.03	0.21	28	1.97	173.52
9	0.10	0.19	29	3.92	94.04
10	0.13	0.57	30	44.00	—
11	0.55	8.49	31	7.46	43.46
12	0.19	1.57	32	242.47	—
13	0.33	5.07	33	3.46	104.98
14	0.46	2.23	34	7.09	233.89
15	0.24	15.91	35	6.59	62.86
16	2.39	26.79	36	6.58	34.16
17	0.69	31.04	37	20.31	—
18	0.77	63.76	38	42.79	
19	5.10	154.70	39	18.64	—
20	3.19	116.82	40	—	—

Table 1: Results with Z3 (time in seconds)

4.1 Hardware

All the tests were performed on a laptop PC equipped with a Intel i5-1135G7 CPU, splitting the computational effort on 4 threads, as we noticed this resulted in faster computation times.

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.

[2] Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 28th International Conference, TACAS 2022, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Munich, Germany, April 2-7, 2022, Proceedings, Part I*, volume 13243 of *Lecture Notes in Computer Science*, pages 415–442. Springer, 2022.

[3] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.