VLSI Report

Using CP & SMT models to solve VLSI problems of increasing difficulty: SMT section.

Alessandro Maggio[1] and Serban Cristian Tudosie[2]

[1]alessandro.maggio5@studio.unibo.it
[2]serban.tudosie@studio.unibo.it

August 2021

SATISFIABILITY MODULO THEORIES

This type of problem can be also translated efficiently in SMT. In particular we used the Z3 Python library in order to do that. We want to compare, at the end of the analysis, which method will be faster to solve the given instances. In this case is not possible to apply particular heuristics during the searching phase.

Problem formulation

The problem formulation is the same of CP. We retieve the lexical convention for reasons of clarity of presentation.

Parameters:

- n number of chips
- w maximum plate width
- $chips\_w_i$ width of the i_th chip
- $chips\_h_i$ height of the i_th chip

Variables:

- h plate's height that must be minimized
- x_i bottom left x coordinate of the i_th chip
- y_i bottom left y coordinate of the i_th chip

The coordinates, after the assignment, represent a complete solution of a problem instance. The height variable is crucial to evaluate the solution quality.

Constraints:

$$0 \leq y_i \leq h - chips\_h_i \qquad i \in [1, n] \tag{1}$$

$$0 \leq x_i \leq w - chips\_w_i \qquad i \in [1, n] \tag{2}$$

$$\forall i, j.$$
$$y_i + chips\_h_i \leq y_j \bigvee y_j + chips\_h_j \leq y_i \tag{3}$$
$$\bigvee x_i + chips\_w_i \leq x_j \bigvee x_j + chips\_w_j \leq x_i$$

$$\forall j \sum_{i \mid y_i \leq j \leq y_i + chips\_h_i} chips\_w_i \leq w \qquad j \in [0, h) \tag{4}$$

$$min\_h \leq h \leq max\_h \tag{5}$$

Problem formulation for SMT

The implementation of the VLSI problem is different in the case of SMT. The possibility to use Python allowed us to keep the whole solution elegantly in a class.
The parameters and the variables are the same as the model in the Constraint Programming formulation.
We have the same constraints given by the equations (1)(2)(3) for the domain and the not overlapping respectively. We do not need to add the extra constraints for the domain since we can initialize the variables respecting (1)(2). So adding two more constraints as in (Code **??**) would be redundant. The difference appears in the translation of the cumulative constraint following the equation (4). This time we performed a sum over rows and a sum over columns which resulted in a faster computation by the solver (6)(7).

$$\forall u \sum_{i \mid y_i \leq u \leq y_i + chips\_h_i} chips\_w_i \leq w \qquad u \in [0, h), i \in [1, n] \tag{6}$$

$$\forall u \sum_{i \mid x_i \leq u \leq x_i + chips\_w_i} chips\_h_i \leq h \qquad u \in [0, w), i \in [1, n] \tag{7}$$

Rotation case

We follow the approach the same approach also in the case of SMT since it allows a clean re-modulation of the problem without adding redundant code. So in order to do so we added the two arrays and the following constraint to take in account rotations.

$$\begin{gathered}\forall_{i\in[1..n]} \\ (rotations_i \wedge chips\_w\_true_i = chips\_h_i \wedge chips\_h\_true_i = chips\_w_i) \\ \otimes \\ (\neg(rotations_i) \wedge chips\_w\_true_i = chips\_w_i \wedge chips\_h\_true_i = chips\_h_i)\end{gathered} \tag{8}$$

Very small code changes have been needed in order to implement this feature with Z3 thanks to the high modularity of an Object Oriented paradigm.

Optimization & Performances

In order to minimize the variable h we tried to solve the problem starting from min_h and gradually increasing h until its upper bound max_h, if a solution has not been found with the current height.
Although the expected performances were much worse than CP, mainly due to the poor search strategy customization, SMT revealed quite good overall performances on many instances.

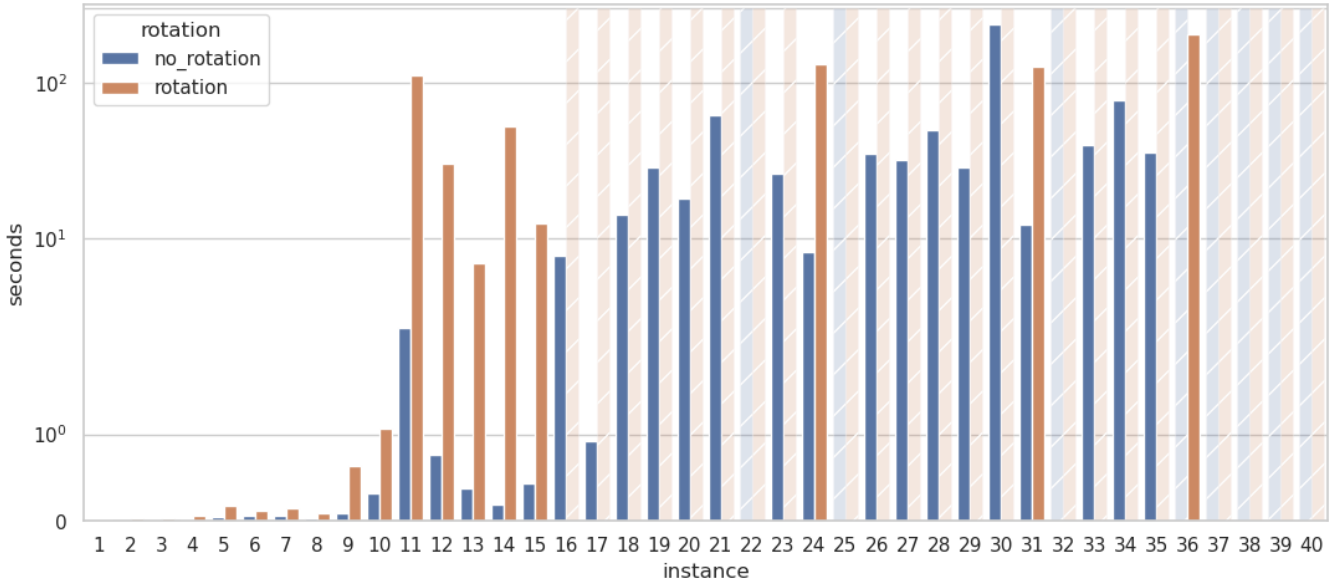

Figure 1: SMT Solving Times

The collected data can be analysed more accurately in the following table (at 3 significant digits):

	SMT	
Instance	No Rotation	Rotation
1	.009	.019
2	.009	.027
3	.014	.027
4	.021	.061
5	.041	.170
6	.062	.125
7	.059	.146
8	.032	.096
9	.087	.634
10	.326	1.06
11	2.69	110
12	.769	30.1
13	.379	6.99
14	.196	51.9
15	.437	12.6
16	7.70	-
17	.921	-
18	14.1	-
19	28.7	-
20	17.9	-
21	61.7	-
22	241	-
23	25.9	-
24	8.21	130
25	-	-
26	34.8	-
27	32.0	-
28	49.4	-
29	28.4	-
30	236	-
31	12.3	-
32	-	-
33	39.9	-
34	76.5	-
35	35.8	-
36	-	48.1
37	-	-
38	-	-
39	-	-
40	-	-

Table 1: SMT solving Time Results

Conclusions

Satisfiability Modulo Theories realized by means of Z3Py allowed to build very accurate domains for all the variables, removing much search space and reducing the computational time. The double constraint of sum over rows and over columns (implied constraints) contribute significantly to push on the performances and make a meaningful comparison between CP and SMT. We strongly believe that the main drawbacks of this approach come from a poor search strategy.
Between the instances 18 and 35 there is not a relevant time difference, it means that the pruning is effective but the situation changes for the rotation-allowed case: even in a case like instance 17 (less than 1 second) the other case remained unsolved. Since the standard solution would be a feasible assignment, we think the search is not efficient and a heuristic may help to attempt, at least for a limited number of steps, to find the standard case before to try rotating chips.
Anyway it is hard to assert that there exists a general method which globally outperforms every other techniques, it clearly emerges a strong domain dependency and a interesting margin of improving. See the main Report for a complete comparison (Github folder).