**Giovanni Minelli**

# SAT and CP-based approach to VLSI problem

## Project report for Combinatorial Decision Making and Optimization Module 1

giovanni.minelli2@studio.unibo.it

# 1 SAT-based approach

## 1.1 Implementation

This part of the project make use of of the Z3 theorem prover through the Python APIs: **Z3Py**. The previous script to read the instance and write the output solution has been reused. The solving part instead is now integrated in the same file but for conformity the name of input output variables were preserved.
Since my limited experience with SAT solvers in general I choose to solve the problem following an interpretation of the problem given by scientific literature, aiming for the optimality. name of input output variables were preserved.
The paper i relied on is [1], which not only explain the general approach used but also integrates many techniques in a way to reduce the search space.
From the experimental results section in the paper is also possible to see how that method was able to solve many problem instances including two open problems: their minimum heights are found and proved to be optimum.

## 1.2 Step-by-step resolution

The problem faced in the paper is a 2SPP (aka 2DSP - Two dimensional strip packing problem) which can be seen as an equivalent instance of the VLSI circuit problem of the assignment. The approach used apply a translation from it into a multiple resolution of a decision problem (2OPP) where the question to answer is "Does exist a displacement to pack the input rectangles in a strip of fixed height?" instead of querying what is the optimal height. Such idea was reproduced in the implementation but still in a step by step procedure to better evaluate the model built as suggested in the assignment.

### 1.2.1 Variables, Main constraints and objective function

For the formalization of the problem has been used the variables: $lr$ and $ud$.
The used approach, instead of capturing the position of the circuits as in the CP-based section, make use of variables able to describe the space and inter-relations.
$lr_{i,j}$ is true if $r_i$ are placed at the left to the $r_j$.
$ud_{i,j}$ is true if $r_i$ are placed at the downward to the $r_j$.

Using a coordinate system starting from the bottom left corner, for each circuit $i$ identified by the tuple $(x_i, y_i)$, the domain of positions in the plate is restricted to:

$$D(x_i) = \{a \in \mathbb{N} \mid 0 \leq a \leq W - w_i\}$$
$$D(y_i) = \{a \in \mathbb{N} \mid 0 \leq a \leq H - h_i\}$$

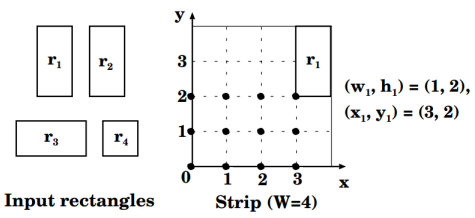



Figure 1: Domain representation of an example.

To constrain the solution to a feasible displacement we have to avoid the overlapping, and so for each rectangles $r_i$, $r_j$ (i > j), the following constraints are used:

$$lr_{i,j} \vee lr_{j,i} \vee ud_{i,j} \vee ud_{j,i}$$
$$\neg lr_{i,j} \vee px_{i,e} \vee px_{j,e+w_i}$$
$$\neg lr_{j,i} \vee px_{j,e} \vee px_{i,e+w_j}$$
$$\neg ud_{i,j} \vee py_{i,f} \vee py_{j,f+h_i}$$
$$\neg ud_{j,i} \vee py_{j,f} \vee py_{i,f+h_j}$$

where the $px$ and $py$ represent a wise encoding for the displacement of a circuit (order encoding). In fact instead of capturing conflict points the constraints are encoded by representing conflict regions which lead to a large saving in the number of clauses used.
$px_{i,e}$ is true if $r_i$ are placed at less than or equal to e.
$py_{i,f}$ is true if $r_i$ are placed at less than or equal to f.

Then it's necessary a constraint for the ordering of such variables:

$$px_{i,e} \vee px_{i,e+1}$$
$$py_{i,f} \vee py_{j,f+1}$$

with e and f which take values in the domain space of each $i$ circuit.

The suggested method of the paper to find the optimal value to answer the 2DSP problem is by iteratively call a 2OPP and make use of an additional variable $ph$.
$ph_o$ is true if all rectangles are packed at the downward to the height $o$.
The following constraints allow to valorize it properly:

- $ph_o \vee py_{i,o-h_i}$ in case of existence of a valid displacement with fixed height $o$ the $py$ variable must be true for all circuits at the extent of the domain
- $ph_o \vee ph_{o+1}$ for order encoding

Since the 2OPP instances can span in the range *min_height* to *max_height* the optimal result can be found just iterating over the $ph$ values, returning the minimum index $o$ at which the variable is True.

### 1.2.2 Variables domain

From the structure of the problem itself we have that the feasibility of each circuit can be reduced operating on each domain. More precisely, each circuit can be displaced only at positions inside the plate keeping in consideration also it's sizes. Therefore, with a cycle over each circuit, the maximal extrema of the domain value are added as True clause for both px and py in case of a possible feasible displacement in the bounds of the plate. Instead, if the size is grater than a dimension of the plate, the same domain value is added to the solver with a False value causing the failure of the single decision problem.

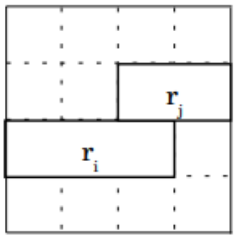
To mention something more in this regard, since the iterations for creation of the clauses could include literals also out of the domain allowed (e.g. $lr_{i,j} \vee px_{i,e} \vee px_{j,e+w_i}$ ), at first each domain included much more values than the necessary all of which had to be properly negated during the domain definition phase resulting in useless time spending.
To prevent that, the iterations for the clauses have been limited and also the resulting domain formulation for px and py variables become much more light. In practical terms the

solver statistics showed less memory usage and much less variables used.
Also, an important clause needed to reach a solution is the one which constraint the position of a rectangle j in case of $lr_{i,j}$ or $ud_{i,j}$ valorized to True: respectively $px_{j,wi-1}$ or $py_{j,hi-1}$ should be False

### 1.2.3 Symmetries

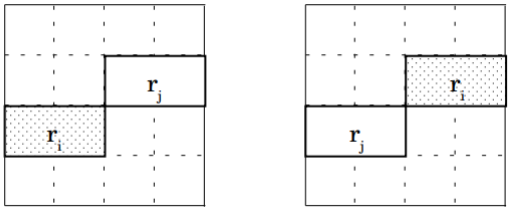
Different techniques to reduce the search space are evaluated considering symmetries and relations of rectangles.

- Large rectangles **(LR)** - Reducing the possibilities for placing large rectangles, both in horizontal and vertical direction. For each rectangle $r_i$ and $r_j$, if $w_i + w_j > W$ we can not pack these rectangles in the horizontal direction, then we can eliminate useless overlapping constraints.



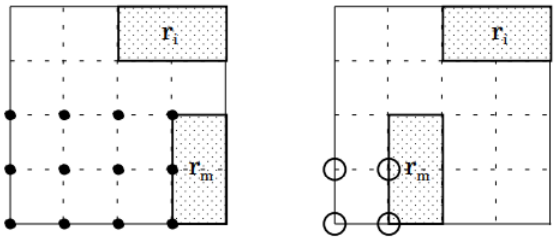

$$\cancel{lr_{i,j} \vee lr_{j,i}} \vee ud_{i,j} \vee ud_{j,i}$$
$$\cancel{\neg lr_{i,j} \vee px_{i,e} \vee px_{j,e+w_i}}$$
$$\cancel{\neg lr_{j,i} \vee px_{j,e} \vee px_{i,e+w_j}}$$
$$\neg ud_{i,j} \vee py_{i,f} \vee py_{j,f+h_i}$$
$$\neg ud_{j,i} \vee py_{j,f} \vee py_{i,f+h_j}$$

- Same rectangles **(SR)** - Breaking symmetries for same-sized rectangles. For each rectangle $r_i$ and $r_j$, if $(w_i, h_i) = (w_j, h_j)$ we can fix the positional relation




$$lr_{i,j} \cancel{\vee lr_{j,i}} \vee ud_{i,j} \vee ud_{j,i}$$
$$\neg lr_{i,j} \vee px_{i,e} \vee px_{j,e+w_i}$$
$$\cancel{\neg lr_{j,i} \vee px_{j,e} \vee px_{i,e+w_j}}$$
$$\neg ud_{i,j} \vee py_{i,f} \vee py_{j,f+h_i}$$
$$\neg ud_{j,i} \vee py_{j,f} \vee py_{i,f+h_j}$$
$$\neg ud_{i,j} \vee lr_{j,i}$$

- Largest rectangle **(LS)** - Reducing the domain for the largest rectangle. We can force the position of the largest rectangle between a pair by removing some constraints. Allow the largest circuit to be displaced only at his maximal domain the other one will be affected as well with a domain reduction.




$$\begin{gathered}
\cancel{lr_{i,j}} \vee lr_{j,i} \vee ud_{i,j} \vee ud_{j,i} \\
\cancel{\neg lr_{i,j} \vee px_{i,e} \vee px_{j,e+w_i}} \\
\neg lr_{j,i} \vee px_{j,e} \vee px_{i,e+w_j} \\
\neg ud_{i,j} \vee py_{i,f} \vee py_{j,f+h_i} \\
\neg ud_{j,i} \vee py_{j,f} \vee py_{i,f+h_j}
\end{gathered}$$

### 1.2.4 Search

The search process have been the most hard detail to implement. The referenced paper assume an iterative approach of search for the optimal height, guided by SAT or UNSAT result of the many 2OPP (by use of a bisection method in the min/max height bounds).
My aim, following the requirements of the project assignment, was to make a single call to the solver with the unified clauses of each 2OPP. But, since the solver try to minimize the effort in the solution of the SAT problem the result obtained with the first encoding resulted always in the maximal height as solution and instead tightening the constraints resulted in an UNSAT result given by the unsatisfiability of the initials 2OPPs under optimal value. To overcome that, after many attempts, the *ph* variables have been modified to be always True except for the optimal height. Then iterating over the solution heights, that variable is put in relation with a property which have to be True in case of a proper height value able to satisfy the instance. Such case can indeed be verified checking the truthness of the extrema of the domain of *py* variable for each circuit (which size is tied to the size of the same circuit). Such clauses constraining the problem seemed correct with small instances but unfortunately at bigger ones the output solution wasn't optimal. Curiously in many cases the result was lower then the *max_height* value, and just reducing the space range of optimal height in which the solver has to search, the optimal value solution is always returned. In my opinion the bad encoding could have influenced the results given by the implementation.

### 1.2.5 Hypothetical model with rotation allowed

The extension of the model to include also rotations isn't faced in the paper but just cited as a future work extension.
A possible resolution method could be the one of consider the different combinations of circuits rotated, and execute separately the 2OPP on each one. Then, by using the cited bisection method forward the execution of only the branches which terminates with a SAT result for the current height value.

Implementing such relaxation in the model can instead be done encoding the circuit rotation with an hot encoded variable (a boolean for each rotation possible of each circuit) which value state about the rotation used for the solution and then during the clause creation could be considered interchangeable the value of height and width (such clause must relate such choice to the truthness of one rotation variable).

## 1.3 Results

The experimental results were collected considering the backjumps of the solver, the optimality of the solution and time of solving used by the SAT solver. Anyway just the instances from 1 to 16 were evaluated for each model combination since the execution time of the python script itself for the solver construction (formulating and adding the clauses) become very long without returning good results: the 16th instance exceeded 20 minutes of computation. Up to the eleventh the whole script execution last less then 300 seconds. The data collected were primary aimed to confront the different techniques (here showed just the best symmetry combination) and also it's possible to see how a different ordering can affect the failures/backjumps of the solver process but not the time of solving or the results obtained, differently from the CP-based approach.

| | ORDERED | | | SYM ORDERED (best LRH+SR) | | |
|---|---|---|---|---|---|---|
| N° | TIME (s) | BACKJUMPS | OPT | TIME (s) | BACKJUMPS | OPT |
| **1** | 0,02 | 5 | Y | 0,02 | 3 | Y |
| **2** | 0,05 | 1 | Y | 0,05 | 1 | Y |
| **3** | 0,10 | 10 | Y | 0,10 | 8 | Y |
| **4** | 0,16 | 93 | Y | 0,14 | 37 | N |
| **5** | 0,58 | 227 | N | 0,66 | 229 | N |
| **6** | 0,89 | 327 | N | 0,94 | 283 | N |
| **7** | 1,39 | 193 | N | 1,40 | 256 | N |
| **8** | 2,03 | 29 | Y | 2,11 | 162 | Y |
| **9** | 2,37 | 151 | Y | 2,43 | 137 | Y |
| **10** | 4,51 | 869 | N | 4,65 | 103 | N |
| **11** | 12,56 | 327 | N | 10,13 | 434 | N |
| **12** | 14,29 | 148 | N | 13,70 | 148 | N |
| **13** | 18,61 | 319 | N | 18,07 | 319 | N |
| **14** | 22 | 304 | N | 21,10 | 304 | N |
| **15** | 29,83 | 263 | N | 29,72 | 1611 | N |
| **16** | 56,59 | 2230 | N | 56,52 | 2230 | N |

Table 1: The complete tests results can be found in the SAT sources folder

# References

[1] A SAT-based Method for Solving the Two-dimensional Strip Packing problem
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.163.7772&rep=rep1&type=pdf