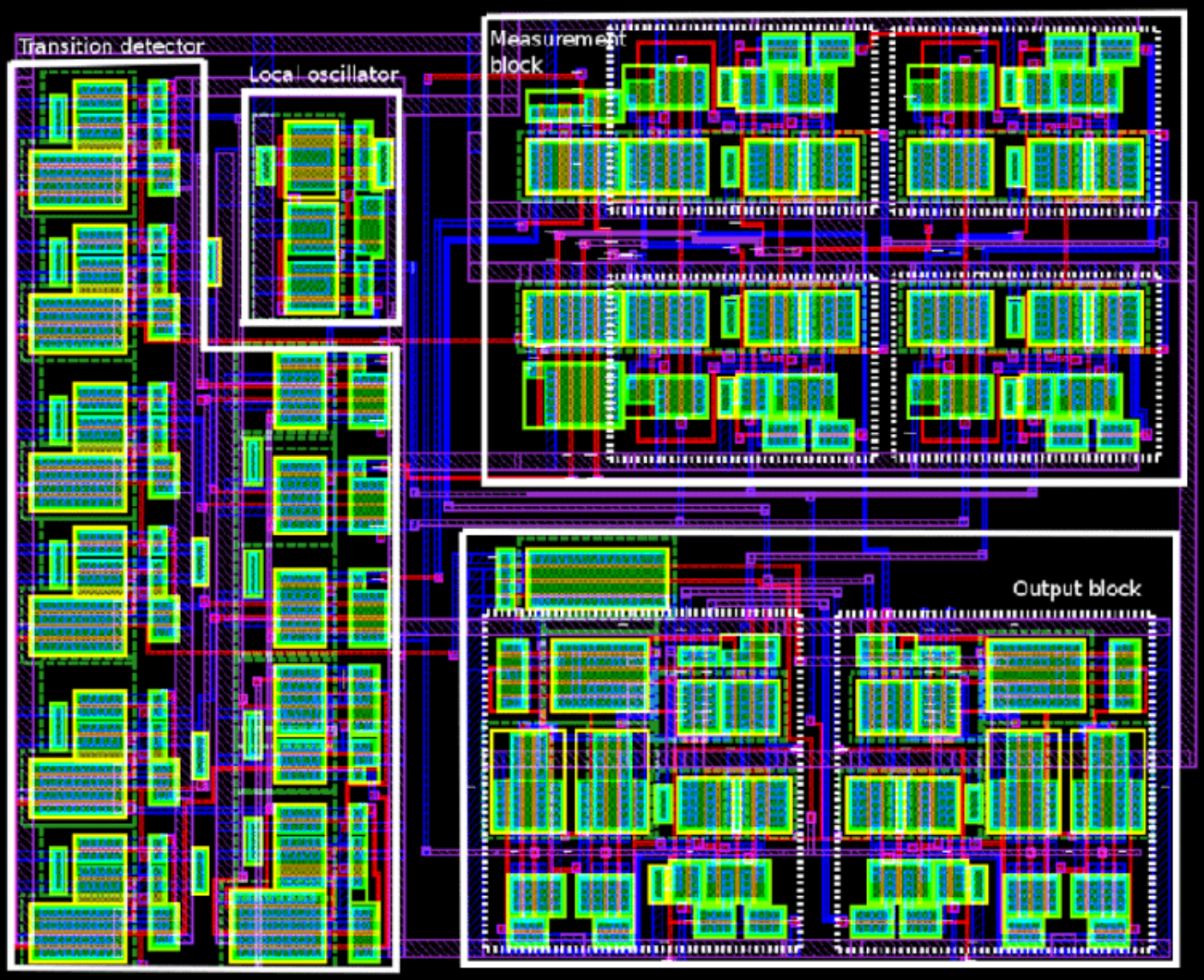
VLSI

A collection of approaches to solve the 2D strip packing problem

Artificial Intelligence: Combinatorial Decision Making and Optimization

Alessandro D'Amico,
Sfarzo El Husseini,
Ana Slovic,
Andrea Virgillito



Alma Mater Studiorum

VLSI

A collection of approaches to solve the 2D strip packing problem

by

Alessandro D'Amico,
Sfarzo El Husseini,
Ana Slovic,
Andrea Virgillito

Student Name	Student Number
D'Amico	0001025050
El Husseini	xxxxxxxxxx
Slovic	xxxxxxxxxx
Virgillito	xxxxxxxxxx

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

Contents

1

Assignment Description

An introduction...

2

CP

3

SAT

3.1. Domain and search strategy

This approach was largely inspired by some concepts presented in the paper **A SAT-based Method for Solving the Two-dimensional Strip Packing Problem** [2].
In SAT it is not possible to specify a function to minimize as instead we've done in CP. In this case it is necessary to define a **lower bound** and an *upper bound* to explore the domain. The width of the strip W is predefined (and therefore fixed) while H can be chosen as follows:

$$H_{LB} = \lceil \frac{\sum_{i \in \{0,..,N-1\}} A_i}{W} \rceil = \lceil \frac{\sum_{i \in \{0,..,N-1\}} width_i \cdot height_i}{W} \rceil$$

$$H_{UB_{naive}} = \sum_{i \in \{0,..,N-1\}} height_i$$

$$H_{UB_{naive-rotation}} = \sum_{i \in \{0,..,N-1\}} min(width_i, height_i)$$

$$H_{UB_{rotation}} = min(H_{UB_{naive-rotation}}, H_{UB})$$

Where N is the number of rectangles in the current instance, A_i the area of the i-th rectangle, H_{LB} and H_{UB} are respectively the lower bound and the upper bound for the strip height. H_{LB} represents the best case possible, in which all the rectangles compact to a big rectangle having no empty spaces. $H_{UB_{naive}}$ represent the worst possible case, in which all the rectangles are placed one over the other. $H_{UB_{naive-rotation}}$ is equivalent to $H_{UB_{naive}}$ but takes in consideration a possible rotation of each rectangle (on the shortest side). H_{UB} is computed solving a relaxed version of the problem: the rectangles are placed in the lowest position available starting from the bottom left corner: this method is known as Bottom-Up Left-Justified, described by Baker et al. [1] . H_{UB} is computed fastly and provides a bound which is way better than $H_{UB_{naive}}$ (this advantage is more evident with instances having many rectangles). $H_{UB_{rotation}}$ is the upper bound for the height in case the rotations are taken into consideration. Known H_{LB} and H_{UB}, the *bisection method* will be used, so the height of the first attempt will be

$$H_{attempt_0} = \lfloor \frac{H_{LB} + H_{UB}}{2} \rfloor$$

and if an assignment for the variables is found, then the new attempted height will be

$$H_{attempt_1} = \lfloor \frac{H_{LB} + H_{attempt_0}}{2} \rfloor$$

, otherwise (in case of failure) it will be

$$H_{attempt_1} = \lfloor \frac{H_{attempt_0} + H_{UB}}{2} \rfloor$$

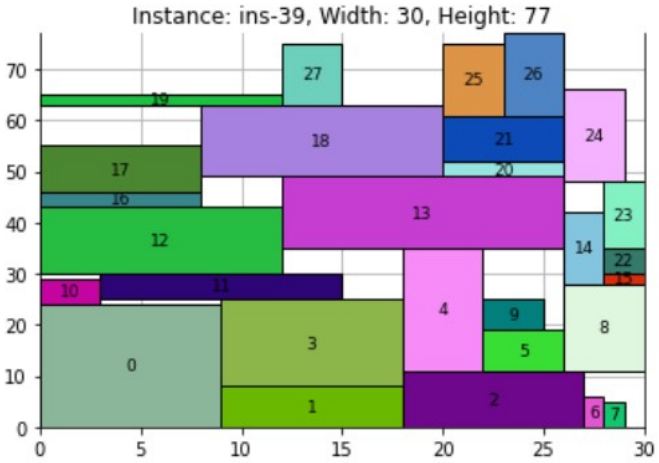


Figure 3.1: Image describing the way in which H_{UB} is computed

In case of rotations it is sufficient to use $H_{UB_{rotation}}$ instead of H_{UB}.
Instead of using the bisection method we could have started from the actual lower bound H_{LB} and then increment it up to reaching the upper bound, but this is a good method just if we use the strong hypothesis that the given instances can be placed with a height near to H_{LB}.

3.2. Encoding

We've chosen the order encoding to represent natural numbers: in this way we have a direct and easy way to convert numbers from the original to the SAT representation (and viceversa), also simplifying the comparison between numbers.
An example of order encoding having 4 as the biggest representable natural number:

$$0 \rightarrow 1111 \rightarrow enc_0 = TTTT$$
$$1 \rightarrow 0111 \rightarrow enc_1 = FTTT$$
$$2 \rightarrow 0011 \rightarrow enc_2 = FFTT$$
$$3 \rightarrow 0001 \rightarrow enc_3 = FFFT$$
$$4 \rightarrow 0000 \rightarrow enc_4 = FFFF$$

Using this encoding it is easy to see that the encoding of a natural number $n \in [0, n_{max}]$ is

$$[\underbrace{F..F}_{\text{n}}\ \underbrace{T..T}_{n_{max}\text{-n}}]$$

Using the 0-based indexing for the elements of the encoding enc_a of a natural number a, we have that

$$enc_{a,i} \rightarrow enc_{a,i+1}, \qquad i \in \{0, n_{max} - 2\}$$

rewritten as a pure CNF clause:

$$\neg enc_{a,i} \vee enc_{a,i+1}, \qquad i \in \{0, n_{max} - 2\}$$

It is also worth noticing that

$$enc_{a,j} \Leftrightarrow a \leq j, \qquad j \in \{0, n_{max} - 1\}$$

For each rectangle r_i, the length of the encoding of the x coordinate of the i-th rectangle is

$$|enc_{x_i}| = W$$

And the length of the encoding of the y coordinate of the i-th rectangle is

$$|enc_{y_i}| = H$$

And to the describe the relative position of the rectangles in the strip we add two kinds of variables:

- the left-right $lr_{i,j}$ which is true if r_i is at the left of r_j (and therefore r_j is at the right of r_i)
- the up-down $ud_{i,j}$ which is true if r_i is above r_j (and therefore r_j is below r_i)

3.3. Constraints

We should take into account that **all** the whole rectangle and not just the left-bottom corner must fit into the width W and height H of the strip, therefore the following constraints must be satisfied (for each rectangle r_i):

$$enc_{x_i,W-width_i} \wedge ... \wedge enc_{x_i,W-1}$$

$$enc_{y_i,H-height_i} \wedge ... \wedge enc_{y_i,H-1}$$

Then we add the order encoding constraint described in the previous section (for each rectangle r_i):

$$\neg enc_{x_i,j} \vee enc_{x_i,j+1} \qquad j \in \{0,..,W-2\}$$

$$\neg enc_{y_i,k} \vee enc_{y_i,k+1} \qquad k \in \{0,..,H-2\}$$

Finally, we have the **no-overlap** constraints:
Every rectangle has a relative position w.r.t. another, so (for each rectangle r_i,r_j with $i < j$ and $i, j \in \{0, N-1\}$):

$$lr_{i,j} \vee lr_{j,i} \vee ud_{i,j} \vee ud_{j,i} \tag{3.1}$$

We should now chain the meaning of the relative position variables with all the other variables and to do that we can think about that, as an example, if a rectangle is at the left of another, the left rectangle left-bottom corner has a distance from the bottom-left corner of the right rectangle that is at least equal to the width of the left rectangle (we can then extend this reasoning to the up-down positioning)

$$lr_{i,j} \Rightarrow x_i + width_i \leq x_j \tag{3.2}$$

$$lr_{j,i} \Rightarrow x_j + width_j \leq x_i \tag{3.3}$$

$$ud_{i,j} \Rightarrow y_i + height_i \leq y_j \tag{3.4}$$

$$ud_{j,i} \Rightarrow y_j + height_j \leq y_i \tag{3.5}$$

3.3.1. Encoding of inequalities in SAT

if we have an inequality of the form

$$x + c \leq y$$

where $c \in \{0..n_{max}\}$ and so that $|enc_x| = |enc_y| = n_{max}$, then

$$x \leq z, \qquad z \in \{n_{max} - c, .., n_{max}\}$$

$$y \geq t, \qquad t \in \{0, .., c\}$$

which can be rewritten as

$$x \leq z, \qquad z \in \{n_{max} - c, .., n_{max}\}$$

$$\neg(y \leq k), \qquad k \in \{0, .., c-1\}$$

$$(y \leq c + s) \rightarrow (x \leq s), \qquad s \in \{0, .., n_{max} - c - 1\}$$

The formulas above are rewritable as literals, each one corresponding to a simple inequality:

$$enc_{x,z} \qquad z \in \{n_{max} - c, .., n_{max} - 1\}$$

$$\neg enc_{y,t} \qquad t \in \{0, .., c-1\}$$

$$enc_{y,c+s} \rightarrow enc_{x,s} \qquad s \in \{0, .., n_{max} - c - 1\}$$

which can be rewritten as

$$enc_{x,z} \qquad z \in \{n_{max} - c, .., n_{max} - 1\}$$

$$\neg enc_{y,t} \qquad t \in \{0, .., c-1\}$$

$$\neg enc_{y,c+s} \vee enc_{x,s} \qquad s \in \{0, .., n_{max} - c - 1\}$$

The inequalities are therefore transformed in the following formulas:

- Formula 3.2 becomes

$$\begin{aligned} &lr_{i,j} \rightarrow enc_{x_i,z} && z \in \{W - width_i, .., W - 1\} \\ &lr_{i,j} \rightarrow \neg enc_{x_j,t} && t \in \{0, .., width_i - 1\} \\ &lr_{i,j} \rightarrow (\neg enc_{x_j,width_i+s} \vee enc_{x_i,s}) && s \in \{0, .., W - width_i - 1\} \end{aligned} \tag{3.6}$$

- Formula 3.3 becomes

$$\begin{aligned} &lr_{j,i} \rightarrow enc_{x_j,z} && z \in \{W - width_j, .., W - 1\} \\ &lr_{j,i} \rightarrow \neg enc_{x_i,t} && t \in \{0, .., width_j - 1\} \\ &lr_{j,i} \rightarrow (\neg enc_{x_i,width_j+s} \vee enc_{x_j,s}) && s \in \{0, .., W - width_j - 1\} \end{aligned} \tag{3.7}$$

- Formula 3.4 becomes

$$\begin{aligned} &ud_{i,j} \rightarrow enc_{y_i,z} && z \in \{H - height_i, .., H - 1\} \\ &ud_{i,j} \rightarrow \neg enc_{y_j,t} && t \in \{0, .., height_i - 1\} \\ &ud_{i,j} \rightarrow (\neg enc_{y_j,height_i+s} \vee enc_{y_i,s}) && s \in \{0, .., H - height_i - 1\} \end{aligned} \tag{3.8}$$

- Formula 3.5 becomes

$$\begin{aligned} &ud_{j,i} \rightarrow enc_{y_j,z} && z \in \{H - height_j, .., H - 1\} \\ &ud_{j,i} \rightarrow \neg enc_{y_i,t} && t \in \{0, .., height_j - 1\} \\ &ud_{j,i} \rightarrow (\neg enc_{y_i,height_j+s} \vee enc_{y_j,s}) && s \in \{0, .., H - height_j - 1\} \end{aligned} \tag{3.9}$$

The first formula of each term can be avoided because the right-hand side of that implication is already made true by the first described constraint, therefore we finally obtain

- Formula 3.6 becomes

$$\begin{aligned} &\neg lr_{i,j} \vee \neg enc_{x_j,t} \qquad t \in \{0,..,width_i - 1\} \\ &\neg lr_{i,j} \vee \neg enc_{x_j,width_i+s} \vee enc_{x_i,s} \qquad s \in \{0,..,W - width_i - 1\} \end{aligned} \tag{3.10}$$

- Formula 3.7 becomes

$$\begin{aligned} &\neg lr_{j,i} \vee \neg enc_{x_i,t} \qquad t \in \{0,..,width_j - 1\} \\ &\neg lr_{j,i} \vee \neg enc_{x_i,width_j+s} \vee enc_{x_j,s} \qquad s \in \{0,..,W - width_j - 1\} \end{aligned} \tag{3.11}$$

- Formula 3.8 becomes

$$\begin{aligned} &\neg ud_{i,j} \vee \neg enc_{y_j,t} \qquad t \in \{0,..,height_i - 1\} \\ &\neg ud_{i,j} \vee \neg enc_{y_j,height_i+s} \vee enc_{y_i,s} \qquad s \in \{0,..,H - height_i - 1\} \end{aligned} \tag{3.12}$$

- Formula 3.9 becomes

$$\begin{aligned} &\neg ud_{j,i} \vee \neg enc_{y_i,t} \qquad t \in \{0,..,height_j - 1\} \\ &\neg ud_{j,i} \vee \neg enc_{y_i,height_j+s} \vee enc_{y_j,s} \qquad s \in \{0,..,H - height_j - 1\} \end{aligned} \tag{3.13}$$

3.4. Symmetry Breaking Constraints

In order to make the computation faster it is necessary to prune the search space reducing the number of clauses and literals involved (contrarily to CP, where adding constraints makes the detection of unfeasible solutions faster).
We've applied 3 kinds of Symmetry Breaking Constraints:

3.4.1. Large Rectangles (LR) Symmetry Breaking

If the sum of two rectangles widths is bigger than the width of the strip (i.e. $width_i + width_j$>W) we can assume that they can't be placed one on the left or right side of the other and we can delete some of the constraints:
Formula 3.1 is rewritten as

$$\cancel{lr_{i,j}} \vee \cancel{lr_{j,i}} \vee ud_{i,j} \vee ud_{j,i}$$

- Formula 3.10 is deleted

$$\neg lr_{i,j} \vee \neg enc_{x_j,t} \qquad t \in \{0,..,width_i - 1\}$$
$$\cancel{\neg lr_{i,j} \vee \neg enc_{x_j,width_i+s} \vee enc_{x_i,s}} \qquad s \in \{0,..,W - width_i - 1\}$$

- Formula 3.11 is deleted

$$\neg lr_{j,i} \vee \neg enc_{x_i,t} \qquad t \in \{0,..,width_j - 1\}$$
$$\cancel{\neg lr_{j,i} \vee \neg enc_{x_i,width_j+s} \vee enc_{x_j,s}} \qquad s \in \{0,..,W - width_j - 1\}$$

- Formula 3.12 is kept

$$\neg ud_{i,j} \vee \neg enc_{y_j,t} \qquad t \in \{0, height_i - 1\}$$
$$\neg ud_{i,j} \vee \neg enc_{y_j,height_i+s} \vee enc_{y_i,s} \qquad s \in \{0, H - height_i - 1\}$$

- Formula 3.13 is kept

$$\neg ud_{j,i} \vee \neg enc_{y_i,t} \qquad t \in \{0,..,height_j - 1\}$$
$$\neg ud_{j,i} \vee \neg enc_{y_i,height_j+s} \vee enc_{y_j,s} \qquad s \in \{0,..,H - height_j - 1\}$$

We can also apply the same reasoning to the height, because if $height_i + height_j$>H we can assume that those rectangles can't be placed one above or under the other and we can delete these constraints:
Formula 3.1 is rewritten as

$$lr_{i,j} \vee lr_{j,i} \vee \cancel{ud_{i,j}} \vee \cancel{ud_{j,i}}$$

- Formula 3.10 is kept

$$\neg lr_{i,j} \vee \neg enc_{x_j,t} \qquad t \in \{0,..,width_i - 1\}$$
$$\neg lr_{i,j} \vee \neg enc_{x_j,width_i+s} \vee enc_{x_i,s} \qquad s \in \{0,..,W - width_i - 1\}$$

- Formula 3.11 is kept

$$\neg lr_{j,i} \vee \neg enc_{x_i,t} \qquad t \in \{0,..,width_j - 1\}$$
$$\neg lr_{j,i} \vee \neg enc_{x_i,width_j+s} \vee enc_{x_j,s} \qquad s \in \{0,..,W - width_j - 1\}$$

- Formula 3.12 is deleted

$$\neg ud_{i,j} \vee \neg enc_{y_j,t} \qquad t \in \{0, height_i - 1\}$$
$$\cancel{\neg ud_{i,j} \vee \neg enc_{y_j,height_i+s} \vee enc_{y_i,s}} \qquad s \in \{0, H - height_i - 1\}$$

- Formula 3.13 is deleted

$$\neg ud_{j,i} \vee \neg enc_{y_i,t} \qquad t \in \{0,..,height_j - 1\}$$
$$\cancel{\neg ud_{j,i} \vee \neg enc_{y_i,height_j+s} \vee enc_{y_j,s}} \qquad s \in \{0,..,H - height_j - 1\}$$

3.4.2. Same Sized Rectangles (SR) Symmetry Breaking

If two rectangles are such that $width_i = width_j$ and $height_i = height_j$ we can choose to have rectangle i left-under j:
Formula 3.1 is rewritten as

$$lr_{i,j} \vee \cancel{lr_{j,i}} \vee ud_{i,j} \vee ud_{j,i}$$

- Formula 3.10 is kept

$$\neg lr_{i,j} \vee \neg enc_{x_j,t} \qquad t \in \{0,..,width_i - 1\}$$
$$\neg lr_{i,j} \vee \neg enc_{x_j,width_i+s} \vee enc_{x_i,s} \qquad s \in \{0,..,W - width_i - 1\}$$

- Formula 3.11 is deleted

$$\cancel{\neg lr_{j,i} \vee \neg enc_{x_i,t} \qquad t \in \{0,..,width_j - 1\}}$$
$$\cancel{\neg lr_{j,i} \vee \neg enc_{x_i,width_j+s} \vee enc_{x_j,s} \qquad s \in \{0,..,W - width_j - 1\}}$$

- Formula 3.12 is kept

$$\neg ud_{i,j} \vee \neg enc_{y_j,t} \qquad t \in \{0,..,height_i - 1\}$$
$$\neg ud_{i,j} \vee \neg enc_{y_j,height_i+s} \vee enc_{y_i,s} \qquad s \in \{0,..,H - height_i - 1\}$$

- Formula 3.13 is kept

$$\neg ud_{j,i} \vee \neg enc_{y_i,t} \qquad t \in \{0,..,height_j - 1\}$$
$$\neg ud_{j,i} \vee \neg enc_{y_i,height_j+s} \vee enc_{y_j,s} \qquad s \in \{0,..,H - height_j - 1\}$$

Adding just a new formula:

$$ud_{i,j} \rightarrow lr_{j,i}$$

rewritable as

$$\neg ud_{i,j} \vee lr_{j,i}$$

3.4.3. Largest Sized Rectangle (LS) Symmetry Breaking

We can also assume to have the biggest rectangle (in terms of area) $largest$ placed in the left bottom half of the rectangle. This makes it possible to restrict the domain:

$$enc_{x_{largest},i}, \qquad i \in \{\lfloor \frac{W - width_{largest}}{2} \rfloor, W - 1\}$$

$$enc_{y_{largest},j}, \qquad j \in \{\lfloor \frac{H - height_{largest}}{2} \rfloor, H - 1\}$$

And delete some overlapping constraints:
Formula 3.1 is rewritten as

$$\cancel{lr_{i,largest}} \vee lr_{largest,i} \vee \cancel{ud_{i,largest}} \vee ud_{largest,i}$$

- Formula 3.10 is deleted

$$\cancel{\neg lr_{i,largest} \vee \neg enc_{x_{largest},t} \qquad t \in \{0,..,width_i - 1\}}$$
$$\cancel{\neg lr_{i,largest} \vee \neg enc_{x_{largest},width_i+s} \vee enc_{x_i,s} \qquad s \in \{0,..,W - width_i - 1\}}$$

- Formula 3.11 is kept

$$\neg lr_{largest,i} \vee \neg enc_{x_i,t} \qquad t \in \{0,..,width_largest - 1\}$$
$$\neg lr_{largest,i} \vee \neg enc_{x_i,width_{largest}+s} \vee enc_{x_{largest},s} \qquad s \in \{0,..,W - width_{largest} - 1\}$$

- Formula 3.12 is deleted

$$\neg ud_{i,largest} \vee \neg enc_{y_{largest},t} \qquad t \in \{0,..,height_i - 1\}$$
$$\neg ud_{i,largest} \vee \neg enc_{y_{largest},height_i+s} \vee enc_{y_i,s} \qquad s \in \{0,..,H - height_i - 1\}$$

- Formula 3.13 is kept

$$\neg ud_{largest,i} \vee \neg enc_{y_i,t} \qquad t \in \{0,..,height_{largest} - 1\}$$
$$\neg ud_{largest,i} \vee \neg enc_{y_i,height_{largest}+s} \vee enc_{y_{largest},s} \qquad s \in \{0,..,H - height_{largest} - 1\}$$

3.5. A variant: rotations

We've introduced a new variable r for each instance, it is True if the rectangle is rotated, False if it is not. For each of the rectangles, if the height (without rotations) is bigger than the dimension of the whole container, the rotation of that same rectangle is avoided
Therefore, we've rewritten the formulas (0-4) in the following way, considering that if a rectangle is rotated, its height and width are inverted:

- Formula 3.10 is rewritten as

$$\neg R_i \rightarrow (\neg lr_{i,j} \vee \neg enc_{x_j,t}) \qquad t \in \{0,..,width_i - 1\}$$
$$\neg R_i \rightarrow (\neg lr_{i,j} \vee \neg enc_{x_j,width_i+s} \vee enc_{x_i,s}) \qquad s \in \{0,..,W - width_i - 1\}$$
$$R_i \rightarrow (\neg lr_{i,j} \vee \neg enc_{x_j,t}) \qquad t \in \{0,..,height_i - 1\}$$
$$R_i \rightarrow (\neg lr_{i,j} \vee \neg enc_{x_j,height_i+s} \vee enc_{x_i,s}) \qquad s \in \{0,..,W - height_i - 1\}$$

- Formula 3.11 is rewritten as

$$\neg R_j \rightarrow (\neg lr_{j,i} \vee \neg enc_{x_i,t}) \qquad t \in \{0,..,width_j - 1\}$$
$$\neg R_j \rightarrow (\neg lr_{j,i} \vee \neg enc_{x_i,width_j+s} \vee enc_{x_j,s}) \qquad s \in \{0,..,W - width_j - 1\}$$
$$R_j \rightarrow (\neg lr_{j,i} \vee \neg enc_{x_i,t}) \qquad t \in \{0,..,height_j - 1\}$$
$$R_j \rightarrow (\neg lr_{j,i} \vee \neg enc_{x_i,height_j+s} \vee enc_{x_j,s}) \qquad s \in \{0,..,W - height_j - 1\}$$

- Formula 3.12 is rewritten as

$$\neg R_i \rightarrow (\neg ud_{i,j} \vee \neg enc_{y_j,t}) \qquad t \in \{0,..,height_i - 1\}$$
$$\neg R_i \rightarrow (\neg ud_{i,j} \vee \neg enc_{y_j,height_i+s} \vee enc_{y_i,s}) \qquad s \in \{0,..,H - height_i - 1\}$$
$$R_i \rightarrow (\neg ud_{i,j} \vee \neg enc_{y_j,t}) \qquad t \in \{0,..,width_i - 1\}$$
$$R_i \rightarrow (\neg ud_{i,j} \vee \neg enc_{y_j,width_i+s} \vee enc_{y_i,s}) \qquad s \in \{0,..,H - width_i - 1\}$$

- Formula 3.13 is rewritten as

$$\neg R_j \rightarrow (\neg ud_{j,i} \vee \neg enc_{y_i,t}) \qquad t \in \{0,..,height_j - 1\}$$
$$\neg R_j \rightarrow (\neg ud_{j,i} \vee \neg enc_{y_i,height_j+s} \vee enc_{y_j,s}) \qquad s \in \{0,..,H - height_j - 1\}$$
$$R_j \rightarrow (\neg ud_{j,i} \vee \neg enc_{y_i,t}) \qquad t \in \{0,..,width_j - 1\}$$
$$R_j \rightarrow (\neg ud_{j,i} \vee \neg enc_{y_i,width_j+s} \vee enc_{y_j,s}) \qquad s \in \{0,..,H - width_j - 1\}$$

3.5.1. Squares Symmetry Breaking

In the version with rotations, We've added a novel constraint that makes it possible to avoid rotations when the rectangle is actually just a square (in this way we avoid the symmetry obtained simply rotating it). In order to apply this symmetry breaking strategy it is necessary to check if $width_i = height_i$: in that case the additional variable r_i is removed from the constraints.

3.6. Results

All instances have been solved (with at least one of the 4 described SAT constraints encodings) but instance 40. Even if sometimes the Symmetry Breaking techniques improve the solver speed, it may not always be the case, this is due to:

- **Thermal throttling**: a common feature for non-industrial PCs
- **CPU job priority scheduling**: the PC is not executing just a single process, but has to manage the timing of many system and app processes, therefore some of the tasks involving the benchmark may have been postponed. For the last reason in the image are shown both the time taken by the solver alone and the time taken by the solver and the encoding part togheter.

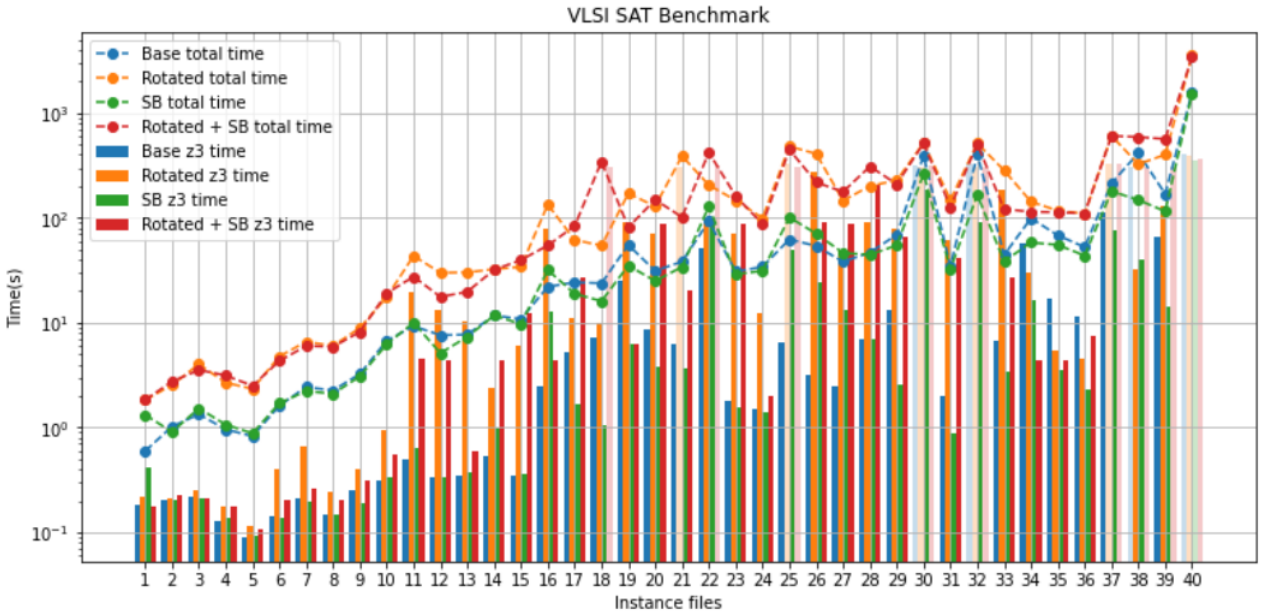


Figure 3.2: Benchmark on all the described SAT constraints encodings

4

SMT

5

LP

6

Conclusion

A conclusion...

References

[1] Brenda S. Baker, E. G. Coffman Jr., and Ronald L. Rivest. "Orthogonal Packings in Two Dimensions". In: *SIAM Journal on Computing* 9.4 (1980), pp. 846–855. DOI: 10.1137/0209064. eprint: https://doi.org/10.1137/0209064. URL: https://doi.org/10.1137/0209064.

[2] Takehide Soh et al. "A SAT-based Method for Solving the Two-dimensional Strip Packing Problem". In: *Fundam. Inform.* 102 (Jan. 2010), pp. 467–487. DOI: 10.3233/FI-2010-314.

A

Source Code Example

Adding source code to your report/thesis is supported with the package `listings`*. An example can be found below. Files can be added using* `\lstinputlisting[language=<language>]{<filename>}`*.*

```
"""
ISA Calculator: import the function, specify the height and it will return a
list in the following format: [Temperature,Density,Pressure,Speed of Sound].
Note that there is no check to see if the maximum altitude is reached.
"""

import math
g0 = 9.80665
R = 287.0
layer1 = [0, 288.15, 101325.0]
alt = [0,11000,20000,32000,47000,51000,71000,86000]
a = [-.0065,0,.0010,.0028,0,-.0028,-.0020]

def atmosphere(h):
    for i in range(0,len(alt)-1):
        if h >= alt[i]:
            layer0 = layer1[:]
            layer1[0] = min(h,alt[i+1])
            if a[i] != 0:
                layer1[1] = layer0[1] + a[i]*(layer1[0]-layer0[0])
                layer1[2] = layer0[2] * (layer1[1]/layer0[1])**(-g0/(a[i]*R))
            else:
                layer1[2] = layer0[2]*math.exp((-g0/(R*layer1[1]))*(layer1[0]-layer0[0])
                    )
    return [layer1[1],layer1[2]/(R*layer1[1]),layer1[2],math.sqrt(1.4*R*layer1[1])]
```

B

Task Division Example

If a task division is required, a simple template can be found below for convenience. Feel free to use, adapt or completely remove.

Table B.1: Distribution of the workload

	Task	Student Name(s)
	Summary	
Chapter 1	Introduction	
Chapter 2		
Chapter 3		
Chapter *		
Chapter *	Conclusion	
	Editors	
	CAD and Figures	
	Document Design and Layout	