VLSI Problem

CP and SAT formulations

Angelo Quarta

01 April 2025

Table of Contents

1. VLSI
2. CP Formulation
   - Decision Variables
   - Optimization Variable
   - Main Constraints
   - Implied Constraints
   - Symmetry Breaking
   - Search Strategy
3. SAT Formulation
   - Decision Variables
   - Order Encoding
   - SAT Optimization

VLSI

Problem Definition

VLSI

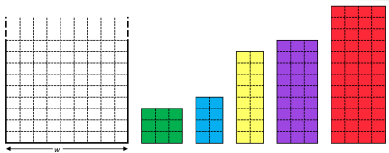
VLSI (Very Large Scale Integration) refers to the trend of integrating circuits into silicon chips.

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.

Each circuit must be placed in a fixed orientation with respect to the others, therefore it cannot be rotated.

Instance Format

An instance of **VLSI** is a text file consisting of lines of integer values. The first line gives w, which is the width of the silicon plate. The following line gives n, which is the number of necessary circuits to place inside the plate. Then the following n lines represent the horizontal and vertical dimensions of the circuits.



CP Formulation

Parameters

If n is the number of circuits, we define the circuits as a matrix c in the form:

$$c_{i,j} \qquad i \in \{1, \ldots, n\}, j \in \{0, 1\}$$

Each rectangle is simply parameterized by its width and height, where w represents the maximum width for a given instance.
Such parameters are set by reading the instance file.

Decision Variables

Moreover, the coordinates of the circuits are defined by the sequences X and Y:

$$X_i \qquad i \in \{1, \ldots, n\}$$
$$Y_i \qquad i \in \{1, \ldots, n\}$$

Optimization Variable

The goal of the optimization process is the minimization of the height H:

$$\max\{y_i + c_{i,1}\} = H \qquad \forall i \in \{1, \ldots, n\}$$

So, in order to achieve it faster, we introduce a lower and an upper bound.

Upper Bound

$$h_{\text{max}} = \sum_i c_{i,1}$$

Lower Bound

$$a = \sum_i c_{i,0} \cdot c_{i,1}$$

$$h_{\text{min}} = \max\left(\left\lceil \frac{a}{w} \right\rceil, \max c_1^T\right)$$

where c_1^T is used to denote a column of the c matrix.

So, $H \in \{h_{\text{min}}, \ldots, h_{\text{max}}\}$

Main Constraints

Preventing circuits from overflowing involves constraints based on their width and height:

$$x_i + c_{i,0} \leq w \qquad \forall i \in \{1, \ldots, n\}$$

Non-overlap constraints can be seen as collision detection between Axis-Aligned Bounding Boxes by enforcing four positional constraints for each pair of circuits:

$$\neg(x_i < x_j + c_{j,0} \wedge x_i + c_{i,0} > x_j \wedge y_i < y_j + c_{j,1} \wedge y_i < y_j + c_{,j+1} \wedge y_i + c_{i,1} > y_j)$$

However, Minizinc offers a global constraint to manage such situation called **diffn**, which, given four vectors defining circuits positions and dimensions enforces a non-overlap behavior between them:

$$\text{diffn}(x, y, c_0^T, c_1^T)$$

Implied Constraints

Implied Constraints

cumulative(x, c_0^T, c_1^T, H) cumulative(y, c_1^T, c_0^T, w)

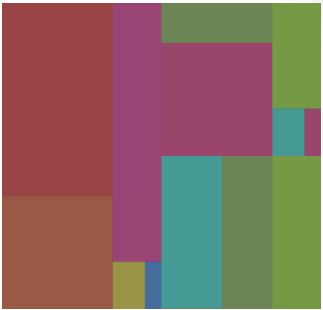
Reflection Symmetry

(a) Original

(b) Vertical

Reflection Symmetry

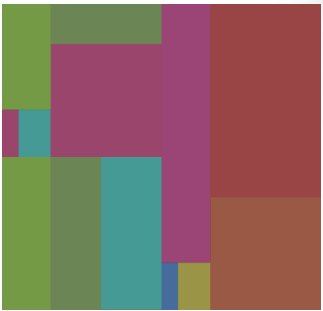
(a) Original



(b) Horizontal

Reflection Symmetry

(a) Original



(b) Horizontal + Vertical

Reflection Symmetry

(a) O

(b) V

(c) H

(d) H + V

$$X_k \leq \frac{w - c_{k,0}}{2} \qquad Y_k \leq \frac{H - c_{k,1}}{2}$$

where k represents the index of the largest circuits.

Same-sized Circuits Symmetry

Permutations of equal chips would provide identical solutions from a graphical point of view:

$$c_{i,0} = c_{j,0} \ \wedge \ c_{i,1} = c_{j,1} \implies$$

$$[x_i, y_i] \leq_{\text{lex}} [x_j, y_j] \quad \forall i, j \in \{1, \ldots, n\},\ i < j$$

Search strategy

In packing problems it is usually better to place cumbersome items first. We define the sequence a of circuit areas:

$$a_i = c_{i,0} \cdot c_{i,1} \qquad \forall i \in \{1, \ldots, n\}$$

Therefore, the searching strategy consists of sorting the circuits according to their area in decreasing order and then feed them to the solver in the input order.

SAT Formulation

Direct Encoding

The most naive encoding of the VLSI problem is the direct encoding, where each module's position is represented by explicit Boolean variables $x_{i,j,k}$. In this encoding, the index k denotes which of the n circuits we are considering, while the indexes i and j map the coordinates where the k-th circuit is located.

Decision Variables

Considering that SAT formulation does not rely on optimization variables, the height $h \in \{h_{\text{min}}, \ldots, h_{\text{max}}\}$ is considered to be fixed for each instance of the problem to be solved.

We define $bx_{i,j}$ and $by_{i,j}$ as decision variables, where the $i \in \{1, \ldots, n\}$ index represents the i-th circuit. On the other hand, j has a different meaning according to the decision variable:

$$
\begin{aligned}
bx_{i,j} \quad & \forall i \in \{1, \ldots, n\}, j \in \{1, \ldots, w\} \\
by_{i,j} \quad & \forall i \in \{1, \ldots, n\}, j \in \{1, \ldots, h\}
\end{aligned}
$$

Meaning of the Variables

Let w_i and h_i be the width and the height of the i-th circuit, let x_i and y_i be the integer coordinates on a plane of a single circuit, we use them to understand the meaning behind the boolean variables employed to describe the problem. Order encoding relies on l and u representing the relative positions among the circuits, where $l_{i,j}$ denotes that the circuit i is on the left with respect to the circuit j.

$$bx_{i,j} \iff (x_i > j) \qquad \forall i \in \{1, \ldots, n\}, j \in \{1, \ldots, w\}$$

$$by_{i,j} \iff (y_i > j) \qquad \forall i \in \{1, \ldots, n\}, j \in \{1, \ldots, h\}$$

$$l_{i,j} \iff (x_i < e) \vee (x_j < e + w_i) \qquad \forall e \in \{0, \ldots, w - w_i\},\ \forall i, j \in \{1, \ldots, n\}$$

$$u_{i,j} \iff (y_j \leq f + h_i) \vee (y_i \leq f) \qquad \forall f \in \{0, \ldots, h - h_i\},\ \forall i, j \in \{1, \ldots, n\}$$

Order Encoding

Boundaries Constraints

$$\bigwedge_{j=0}^{w-1} \neg\, bx_{i,j} \;\vee\; bx_{i,j+1} \qquad \forall i \in \{1,\ldots,n\}$$

$$\bigwedge_{j=0}^{h-1} \neg\, by_{i,j} \;\vee\; by_{i,j+1} \qquad \forall i \in \{1,\ldots,n\}$$

$$\bigwedge_{j=w-w_i}^{w} bx_{i,j} \qquad \forall i \in \{1,\ldots,n\}$$

$$\bigwedge_{j=h-h_i}^{h} by_{i,j} \qquad \forall i \in \{1,\ldots,n\}$$

Order Encoding

Non-overlapping Constraints

$$\bigwedge_{i \neq j} \neg l_{i,j} \vee bx_{j,w_i-1} \qquad \forall i, j \in \{1, \ldots, n\}$$

$$\bigwedge_{e=0}^{w-w_i} \neg\, l_{i,j} \vee bx_{i,e} \vee \neg\, bx_{j,e+w_i} \qquad \forall i, j \in \{1, \ldots, n\}$$

$$\bigwedge_{i \neq j} \neg\, u_{i,j} \vee by_{j,h_i-1} \qquad \forall i, j \in \{1, \ldots, n\}$$

$$\bigwedge_{f=0, i \neq j}^{h-h_i} \neg\, u_{i,j} \vee by_{i,f} \vee \neg\, by_{j,f+h_i} \qquad \forall i, j \in \{1, \ldots, n\}$$

SAT Optimization

Given that the optimization variable's value is bounded, the optimization within the search interval can be carried out using two main strategies:

Linear Search
Starting from the minimum bound, we scan the entire horizon until the first feasible solution.

Binary Search
Leveraging the fact that the interval consists of a sorted sequence of numbers, then the searching procedure will select the mean value of the interval and look for the optimal solution in the left sub-interval or the right one according to the feasibility of the problem.

Linear Search

The linear search algorithm relies on the fact that we are sure that the lower bound we chose gives us an infeasibility as a result.

Thus, the adding a fix delta to the previous step's lower bound, we check the feasibility of the model, if it is **not** feasible, the algorithm proceeds with another iteration, otherwise, the first point on the line in which does not report an infeasibility is our optimal solution.

Binary Search

In the first iteration of the algorithm the searching space is restricted to the first half of the considered interval.

Going on with the subsequent iterations, two cases arise: if the mean point of the interval is feasible, the upper bound shifts to the previous mean point.

Binary Search

Once the searching space is restricted, the second case might show up: the upper bound shifts to the previous mean point if the point is not feasible.

Binary Search

Once the interval is restricted, the iteration starts again by selecting the mean point.

Finally, the algorithm reaches convergence when the lower and the upper bound of the interval collapse in a single point.