VLSI: VERY LARGE SCALE INTEGRATION

CDMO REPORT - MIXED INTEGER LINEAR PROGRAMMING

Lorenzo Pratesi
lorenzo.pratesi2@studio.unibo.it

Artificial Intelligence

University of Bologna

June 2022

Contents

1 Introduction

VLSI (Very Large Scale Integration) refers to the problem 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 plate is minimized. We will address two variants of the problem; in the first each circuit must be placed in a fixed orientation with respect to the others, while in the second case each circuit could be also rotated by 90° degrees.

The solutions provided in this work are implemented using PuLP (4), a linear programming toolkit written in Python. PuLP provides an interface to define LP/MIP models in Python, but it also provides and interface to communicate with the most used solvers.

Three different solvers have been used in this work:

- **COIN-OR/CBC**: an open source mixed integer programming solver based on the branch and cut algorithm (2);
- **CPLEX**: a proprietary mathematical optimizer based on distributed parallel algorithms, developed by IBM (1);
- **GUROBI**: a proprietary LP, QP and MIP optimizer based on several state-of-the-art algorithms (3)

In Section 2 we will discuss how the problem is formally defined by means of its inputs and how the outputs are structured. in Section 3 we will see how the models are defined together with their constraints. In Section 4 we will see the all the experimental setups and their results over the provided instances.

2 Problem Description

The input of the problem is structured as follows:

- w: width of the plate;
- n: number of circuits;
- m: a $n \times 2$ matrix where m_{i1} and m_{i2} are the width and the height of the i-th circuit respectively.

The aim of the problem is to fill the plate with all the circuits by minimizing the total height without exceeding its width w.

The instances of the problem are structured in the following way:

w
n
m_{11} m_{12}
m_{21} m_{22}
...
m_{n1} m_{n2}

The outputs of the problem will be structured in the following way:

w h
n
x_1 y_1 m_{11} m_{12}
x_2 y_2 m_{21} m_{22}
...
x_n y_n m_{n1} m_{n2}

where x_i and y_i are the coordinates of the bottom-left corner of the i-th circuit.

Note that, if rotation of circuits is allowed, the dimension of a circuit could be inverted.

3 Modelling

We are now going to introduce the models and the constraints used to address the VLSI problem with a MILP approach. We are going to represent each circuit placed in the plate by its bottom-left coordinate. Three types of models have been defined:

- **Complete**: the model that uses all the constraints described below, without allowing rotation of circuits;
- **Standard**: like the Complete model, but without cumulative constraints
- **Rotation Standard**: like the Standard model, but allowing rotation of circuits;
- **Rotation Complete**: like the Complete model, but allowing rotation of circuits;

Following the LP paradigm, all the constraints have been implemented in the canonical form $\sum_{i=1}^{n} a_i x_i \leq \beta_i$ where a_i and β_i are constants while x_i are variables. Even if PuLP allows to define equations in other forms, I decided to use the canonical form as an exercise.

In order to create tighter bounds, some useful constants have been defined:

- $area_{min} = \sum_{i=1}^{n} m_{i1} m_{i2}$
- $min_h = max(max(m_{02}, ..., m_{n2}), \lfloor area_{min}/w \rfloor)$
- $max_h = \sum_{i=1}^{n} m_{i2}$

If rotation is allowed:

- $area_{min} = \sum_{i=1}^{n} m_{i1} m_{i2}$
- $min_h = max(max(min(m_{01}, m_{02}), ..., min(m_{n1}, m_{n2})), \lfloor area_{min}/w \rfloor)$
- $max_h = \sum_{i=1}^{n} max(m_{i1}, m_{i2})$

where $area_{min}$ represents the minimum area of the plate covered by the circuits, min_h represents the minimum height possible and max_h represents the maximum height possible.

3.1 Variables

For each circuit i, we are going to define two sets of integer variables. For the width axis, $x_i \;\; i \in \{1, ..., n\}$ variables have been defined, while for the height axis, $y_i \;\; i \in \{1, ..., n\}$ variables have been defined. Each variable has a lower and upper bound defined as:

$$\forall_{i \in \{1,...,n\}} \; 0 \leq x_i \leq w - m_{i1}$$

$$\forall_{i \in \{1,...,n\}} \; 0 \leq y_i \leq max_h - m_{i1}$$

If rotation is allowed:

$$\forall_{i \in \{1,...,n\}} \; 0 \leq x_i \leq w - min(m_{i1}, m_{i2})$$

$$\forall_{i \in \{1,...,n\}} \; 0 \leq y_i \leq max_h - min(m_{i1}, m_{i2})$$

The maximum height covered by all the positioned circuits has been defined as an integer variable h. Our aim is to minimize that variable and its bounds are:

$$min_h \leq h \leq max_h$$

If rotation is allowed, for each circuit i a Boolean variable r_i is defined, representing the circuit orientation.

3.2 Constraints

3.2.1 Bounds constraints

We know that each circuit of the plate must be placed in non-negative coordinates and its position plus its size can not exceed the maximum plate size along an axis. When rotation is not allowed, the upper bound of the variables of the width axis already ensures that behavior. Otherwise also a bound for that axis must be defined. The following constraints implement the mentioned behavior:

- $y\_bounds \equiv \forall_{i \in \{1,\dots,n\}}\ y_i - h \leq -m_{i2}$

If rotation is allowed:

- $x\_bounds \equiv \forall_{i \in \{1,\dots,n\}}\ x_i - m_{i1}r_i + m_{i2}r_i \leq -m_{i1} + w$
- $y\_bounds \equiv \forall_{i \in \{1,\dots,n\}}\ x_i - h - m_{i2}r_i + m_{i1}r_i \leq -m_{i2}$

3.2.2 No overlap constraint

To ensure that the circuits will not overlap, for each pair of circuits i, j we must ensure that $x_i + m_{i1} \leq x_j \vee y_i + m_{i2} \leq y_j$. So if one of those inequalities holds, it means that j is placed after i along one axis, and this is sufficient for stating that the two do not overlap. To implement the logical "or" operator, it has been employed the so called "Big-M" trick. We define a variable $b_i \in \{0, 1\}$ for each formula j and we impose a new constraint $\sum_{j=1}^{m} b_i \geq 1$; then for each formula j in the canonical form, we impose $\sum_{i=1}^{n} a_{ij}x_i - \beta_j \leq M_j(1 - b_j)$ where M_j is a big enough constant and it is usually defined as $M_j = -\beta_j + \sum_i max(a_{ij}l_i, a_{ij}u_i)$ where l_i and u_i are respectively the lower and upper bounds of the variable x_i. For the "no overlap" constraint, for each pair of circuits four new variables $b_i \in \{0, 1\}$ and four M_i new big-M constants, specified as above, have been defined. The following constraints ensure the aforementioned behavior:

$$
\begin{aligned}
-b_{ij1} - b_{ij2} - b_{ij3} - b_{ij4} &\leq -1, \\
x_i - x_j + M_1 b_{ij1} &\leq M_1 - m_{i1} \\
y_i - y_j + M_2 b_{ij2} &\leq M_2 - m_{i2} \\
x_j - x_i + M_3 b_{ij3} &\leq M_3 - m_{j1} \\
y_j - y_i + M_4 b_{ij4} &\leq M_4 - m_{j2}
\end{aligned}
$$

If rotation is allowed:

$$
\begin{aligned}
-b_{ij1} - b_{ij2} - b_{ij3} - b_{ij4} &\leq -1, \\
x_i - x_j + M_1 b_{ij1} - m_{i1}r_i + m_{i2}r_i &\leq M_1 - m_{i1} \\
y_i - y_j + M_2 b_{ij2} - m_{i2}r_i + m_{i1}r_i &\leq M_2 - m_{i2} \\
x_j - x_i + M_3 b_{ij3} - m_{j1}r_j + m_{j2}r_j &\leq M_3 - m_{j1} \\
y_j - y_i + M_4 b_{ij4} - m_{j2}r_j + m_{j1}r_j &\leq M_4 - m_{j2}
\end{aligned}
$$

where $1 \leq i \leq n$ and $i + 1 \leq j \leq n$.

3.2.3 Cumulative constraint

The cumulative constraint is defined as $cumulative(s, d, r, b)$ where:

- s is an array of integer variables, representing the starting time of each job
- d is an array of integer constants, representing the time duration of each job
- r is an array of integer constants, representing the resources needed for each job;
- b is an integer representing the maximum amount of resources that could be allocated at each time t.

For our problem, this constraint could be used as an implied constraint. In particular, if we think:

- the two axis of the plate (x and y) as temporal lines,
- each circuit as a job and its coordinates as a starting time in the relative axis,
- the circuit's size along a particular axis as duration and the other one as a resource
- the axis bound as b.

we can define one cumulative constraint per axis and obtain the the expected results.

The cumulative constraint is defined as:

$$\forall_{t \in \{early,\dots,late\}} \sum_{i=1}^{n} v_i \, r_i \leq b$$
$$v_i = \begin{cases} 1 & if \ s_i \leq t \ \wedge \ t+1 \leq s_i + d_i \\ 0 & otherwise \end{cases} \tag{1}$$

where $early$ is the minimum among all the lower bounds of s_i and $late$ is the maximum among all the sums between the upper bounds of s_i and d_i.

Before introducing the MIP formulation, we must first define some MIP encodings of v_i

First term encoding Given s_i and t, we want to define a new variable k_i so that

$$k_i = \begin{cases} 1 & if \ s_i \leq t \\ 0 & otherwise \end{cases} \tag{2}$$

To do so, we can rely again on the "Big-M" trick, but this time we just impose:

$$s_i \leq t + M_1(1 - k_i)$$
$$-s_i \leq -t - 1 + M_2 k_i$$

where M_1 and M_2 are big enough constants. In this way, if $k1 = 1$ then $s_i \leq t$ must hold, otherwise $s_i \geq t + 1$.

Second term encoding Given s_i and t, we want to define a new variable k_i so that

$$k_i = \begin{cases} 1 & if \ t+1 \leq s_i + d_i \\ 0 & otherwise \end{cases} \tag{3}$$

We rely again on the "Big-M" trick, but this time we impose:

$$s_i \leq t - d_i + M_2 k_i$$
$$-s_i \leq -t - 1 + d_i + M_1(1 - k_i)$$

where M_1 and M_2 are big enough constants. In this way, if $k1 = 1$ then $s_i + d_i \geq t + 1$ must hold, otherwise $s_i + d_i \leq t$.

Binary logical AND constraint Given two variables $q \in \{0, 1\}$ and $p \in \{0, 1\}$ we would like to define a new variable k so that:

$$k = \begin{cases} 1 & if \ p = 1 \ \wedge \ q = 1 \\ 0 & otherwise \end{cases} \tag{4}$$

By imposing the following constraints we ensure the above behavior:

$$1 - k \leq 2 - p - q$$
$$k \leq p$$
$$k \leq q$$
$$0 \leq k \leq 1$$

Putting all together We are ready to join together the above encodings to implement the cumulative constraint.

For each $t \in \{early, ..., late\}$ and $i \in \{1, ..., n\}$ with $n = |s|$, three new variables $k_{ti1}, k_{ti2}, k_{ti3} \in \{0, 1\}$ and four big-M constants $M_{ti1}, ..., M_{ti4}$ have been defined. The cumulative constraint have been implemented by adding the following formulas:

$$
\begin{gathered}
cumulative(s, d, r, b) \equiv \\
s_i \leq t + M_{ti1}(1 - k_{ti1}), \quad -s_i \leq -t - 1 + M_{ti2}k_{ti1} \\
s_i \leq t - d_i + M_{ti3}k_{ti2}, \quad -s_i \leq -t - 1 + d_i - M_{ti4}(1 - k_{ti2}) \\
1 - k_{ti3} \leq 2 - k_{ti1} - k_{ti2} \\
k_{ti3} \leq k_{ti1} \\
k_{ti3} \leq k_{ti2} \\
\sum_t r_i k_{ti3} \leq b
\end{gathered}
$$

To conclude, we defined two cumulative constraints, one for each axis of the plate:

$$cumumative_x \equiv cumulative(\{x_1, ..., x_n\}, \{m_{11}, ..., m_{n1}\}, \{m_{12}, ..., m_{n2}\}, h)$$

$$cumumative_y \equiv cumulative(\{y_1, ..., y_n\}, \{m_{12}, ..., m_{n2}\}, \{m_{11}, ..., m_{n1}\}, w)$$

4 Experimental setup and Results

In this section there will be showed the experimental setups and the results obtained on the instances provided. The experimental setup is featured by the use of three different solvers: COIN-OR/CBC (2), GUROBI (3) and CPLEX (1). For each of those solvers, every model have been tested over all the provided intances.

Table 1 shows some solving statistics obtained over all the configurations described above. The best configuration features the Complete model with the GUROBI solver. In general the use of the cumulative constraint resulted beneficial for almost all the configurations.

Figure 1 shows the solving times over all the instances computed by each solver using the Complete model. We can immediatly notice that COIN-BC/CBC had a lot of issues in solving all the instances; while CPLEX and GUROBI performed very similarly.

Table 1: Timing statistics using different solvers and models

Model	**Solver**	**Solved instances**	**Max solving time**	**Avg solving time**
Standard	COIN-BC/CBC	9	62.74925971031189	8.048
Standard	CPLEX	26	274.9458239078522	29.7912
Standard	GUROBI	32	259.7245583534241	28.2866
Complete	COIN-BC/CBC	26	296.1142113208771	163.3959
Complete	CPLEX	31	201.5385286808014	40.5729
Complete	GUROBI	**33**	241.59940147399902	41.8321
Rotation Standard	COIN-BC/CBC	9	287.76986145973206	52.0557
Rotation Standard	CPLEX	21	166.85631656646729	16.0019
Rotation Standard	GUROBI	28	189.13197875022888	30.889
Rotation Complete	COIN-BC/CBC	16	294.78941226005554	154.257
Rotation Complete	CPLEX	16	284.6473376750946	43.1296
Rotation Complete	GUROBI	18	294.73363161087036	64.4808

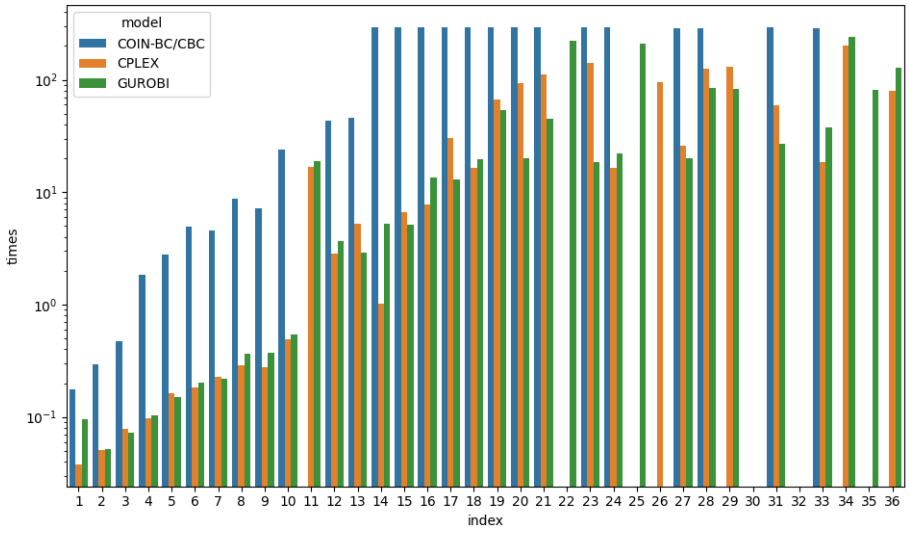

Figure 1: Solving times of the Complete model using different solvers over all the instances

References

[1] IBM ILOG Cplex. V22.1.0: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2022.

[2] John Forrest, Ted Ralphs, Haroldo Gambini Santos, Stefan Vigerske, John Forrest, Lou Hafer, Bjarni Kristjansson, Jpfasano, EdwinStraver, Miles Lubin, Rlougee, Jpgoncal1, Jan-Willem, H-I-Gassmann, Samuel Brito, Cristina , Matthew Saltzman, Tosttost, Bruno Pitrus, Fumiaki MATSUSHIMA, and To-St. *coin-or/Cbc: Release releases/2.10.8*. Zenodo, May 2022.

[3] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022.

[4] Stuart Mitchell, Stuart Mitchell Consulting, and Iain Dunning. Pulp: A linear programming toolkit for python, 2011.