5 MIP Model

We attempted to utilize a framework that is independent of the solvers, as suggested in the description of the VLSI problem. During our investigation, we encountered Pyomo.

An open-source, Python-based modeling language for optimization with a wide range of optimization features. Simply modifying the solver's name would be sufficient to solve the problem with another solver.

We tried three different solvers (GLPK, CBC, IPOPT) and compared the result.

We aimed to stabilize the optimization algorithm based on the variables' probable lower and upper bounds.

5.1 Decision variables

We have boolean and integer variables.

The Boolean variables are used to encode logical operators.

Model.rots represents rotation to the number of circuits.

Model.ors encodes some logical or relation in our constraints which will be described in the next section.

For the integer variables, we have model.height (with min height and max height as lower and upper bounds respectively as calculated before)

We have model.x and model.y variables which hold the x and y coordinates of the left bottom corner of each circuit respectively.

Model.x has the domain [0, width- min(x_dims_init)]

Model.y has the domain [0, max_height - min(y_dims_init)]

We stored every circuit dimension in x_dims_init, y_dims_init provided by instance txt file.

We have model.x_dims, model.y_dims which both have the domain [0, dims_upper_bound]

dims_upper_bound have been calculated before: max(max(x_dims_init), max(y_dims_init))

5.2 Objective function

Our purpose is to minimize model.height

5.3 Constraints

As usual, we have two main constraints for not breaking the boundaries of the box which are:

$$\forall i: \text{model.x}[i] + \text{model.x\_dims}[i] \leq \text{width}$$

$$\forall i: model.y[i] + model.y\_dims[i] \leq model.height$$

For encoding the constraint for the circuits not to overlap we need to satisfy:

$$\begin{aligned}\forall(i, j): &\ model.x[i] + model.x\_dims[i] <= model.x[j]\\ &\vee model.y[i] + model.y\_dims[i] <= model.y[j]\\ &\vee model.x[j] + model.x\_dims[j] <= model.x[i]\\ &\vee model.y[j] + model.y\_dims[j] <= model.y[i]\end{aligned}$$

So, we encode them in a linear way:

$$\begin{aligned}\forall(i, j): &\ (model.x[j] + model.x\_dims[j] <= model.x[i] + M\_x * (1 - model.ors[i, j, 0])\\ &\wedge model.x[i] + model.x\_dims[i] <= model.x[j] + M\_x * (1 - model.ors[i, j, 1])\\ &\wedge model.y[j] + model.y\_dims[j] <= model.y[i] + M\_y * (1 - model.ors[i, j, 2])\\ &\wedge model.y[i] + model.y\_dims[i] <= model.y[j] + M\_y * (1 - model.ors[i, j, 3])\end{aligned}$$

while considering that at least one of them should be true:

$$\forall(i, j): \sum_{k=0}^{4} ors[i, j, k] \geq 1$$

For the symmetry breaking we again considered placing the bottom left coordinates of the biggest circuit in the bottom left quarter of the board. For this we used these linear constraints:

$$x[biggest] \leq width/2$$
$$y[biggest] \leq model.height/2$$

5.3 Rotation

In order to model the rotation of circuits, we added a boolean variable called model.rot for each circuit to encode if that specific circuit was rotated or not. So these new constraints were added to the previous model:

$$\forall i: model.x\_dims[i] == (1 - model.rots[i]) * model.x\_dims[i] + model.rots[i] * model.y\_dims[i]$$

$$\forall i: model.y\_dims[i] == (1 - model.rots[i]) * model.y\_dims[i] + model.rots[i] * model.x\_dims[i]$$

5.4 Results

IPOPT solver had a unique performance and solved all 40 samples in 18 seconds overall.

GLPK and CBC solved 9 and 11 instances in total.

Except instances number 9th and 13th, GLPK was faster than CBC.

Through trial and error, we spent a lot of time and effort determining the best approach to specify the variables and constraints in this framework and addressing the issues.

	GLPK	CBC	IPOPT	GLPK_run_time	CBC_run_time	IPOPT_run_time
INS-01	8	8	8	0.116	0.339	0.703
INS-02	9	9	9	0.053	0.42	0.063
INS-03	10	10	10	0.071	1.168	0.057
INS-04	11	11	11	0.467	1.837	0.059
INS-05	12	12	12	0.406	4.047	0.072
INS-06	13	13	13	15.901	19.892	0.067
INS-07	14	14	14	0.117	43.109	0.071
INS-08	-	15	15	-	5.82	0.072
INS-09	16	16	16	277.814	14.648	0.069
INS-10	-	17	17	-	202.982	0.082
INS-11	-	-	18	-	-	0.097
INS-12	-	-	19	-	-	0.095
INS-13	20	20	20	154.856	82.143	0.101
INS-14	-	-	21	-	-	0.11
INS-15	-	-	22	-	-	0.115
INS-16	-	-	23	-	-	0.126
INS-17	-	-	24	-	-	0.724
INS-18	-	-	25	-	-	0.133
INS-19	-	-	26	-	-	0.144
INS-20	-	-	27	-	-	0.148
INS-21	-	-	28	-	-	0.143
INS-22	-	-	29	-	-	0.159
INS-23	-	-	30	-	-	0.143
INS-24	-	-	31	-	-	0.13
INS-25	-	-	32	-	-	0.203
INS-26	-	-	33	-	-	0.158
INS-27	-	-	34	-	-	0.16
INS-28	-	-	35	-	-	0.156
INS-29	-	-	36	-	-	0.157
INS-30	-	-	37	-	-	0.283
INS-31	-	-	38	-	-	0.143
INS-32	-	-	39	-	-	0.241
INS-33	-	-	40	-	-	0.146
INS-34	-	-	40	-	-	0.184
INS-35	-	-	40	-	-	0.199
INS-36	-	-	40	-	-	0.185
INS-37	-	-	60	-	-	0.23
INS-38	-	-	60	-	-	0.251
INS-39	-	-	60	-	-	0.225
INS-40	-	-	90	-	-	1.552