Combinatorial Decision Making and Optimization

Project Report

VLSI Design

Gee Jun Hui Leonidas Yunani
Anna Fabris

Academic Year: 2020 / 2021

Contents

1 CP (Normal)

The normal CP model is one where rotation of the circuits is not allowed. The model has been designed to use as many global constraints as possible. Symmetry breaking has also been applied where applicable to improve the final performance.

1.1 Considerations on the Height

By assuming that no gaps are allowed between the circuits, the min height is obtained by taking the sum of the circuits' area and dividing it by the given max width.

If gaps are allowed between the circuits, the board's height can be minimised by the solver to find the minimum height of the board.

1.2 Constraints

- 2 `cumulative` constraints are used to find the circuits' x and y coordinates respectively.

```
constraint cumulative(
    start_x, CIRCUIT_WIDTHS,
    CIRCUIT_HEIGHTS, MIN_HEIGHT
);

constraint cumulative(
    start_y, CIRCUIT_HEIGHTS,
    CIRCUIT_WIDTHS, MAX_WIDTH
);
```

- 2 `forall` constraints are used to ensure that the circuits' x and y coordinates along with their corresponding widths and heights do not exceed the board's max width and height.

```
constraint forall(c in CIRCUITS)(
    start_x[c] + CIRCUIT_WIDTHS[c] <= MAX_WIDTH
);

constraint forall(c in CIRCUITS)(
    start_y[c] + CIRCUIT_HEIGHTS[c] <= MIN_HEIGHT
);
```

- A `diffn` constraint is used to remove overlaps between the circuits. This prevents the solver from searching for solutions whereby the circuits are placed on top of each other.

```
constraint diffn(
    start_x, start_y,
    CIRCUIT_WIDTHS, CIRCUIT_HEIGHTS
);
```

- A `lex_lesseq` constraint is used to remove the vertical symmetry of the board. This reduces the total number of solutions by half for each instance.

```
constraint lex_lesseq(
    start_y, reverse(start_y)
);
```

- A search strategy using `most_constrained` and `indomain_min` is applied to the circuits' x and y coordinates.

1.3 Final Considerations

The final model is able to solve 32/40 instances with a time constraint of 300 seconds.

A lexicographical constraint was also tested to remove horizontal symmetry from the board. However, although the number of total solutions was reduced, the number of total instances solved did not increase.

In fact, the number of total instances solved worsened and hence, the constraint was not included in the final model.

2 CP (Rotation)

The rotational CP model is one where rotation of the circuits is allowed. A global constraint that allows for the rotation of circuits has been used to build a generalisable model.

2.1 Considerations on the Height

By assuming that no gaps are allowed between the circuits, the min height is obtained by taking the sum of the circuits' area and dividing it by the given max width.

If gaps are allowed between the circuits, a model using the `cumulative` constraint can be used to first minimise the board's height. After finding the minimum height, it can then be applied to a second model using the `geost_smallest_bb` constraint.

2.2 Circuits' Rotations

Circuits with widths and heights that are equivalent do not have any rotations assigned to them. This reduces the number of total solutions as square-shaped circuits occupy the same relative area regardless of orientation.

2.3 Constraints

- A `geost_smallest_bb` constraint is used to pack the circuits with rotations allowed into the board without gaps.

```
constraint geost_smallest_bb(
    k,
    rect_size,
    rect_offset,
    shape,
    x,
    kind,
    l,
    u
);
```

- A `forall` constraint is used to ensure that only shapes with certain rotations are accepted as valid.

```
constraint forall(obj in OBJECTS)(
    kind[obj] in valid_shapes[obj]
);
```

- A search strategy using `most_constrained` and `indomain_min` is applied to the rotation of the circuits.

2.4 Final Considerations

The final model is able to solve 17/40 instances with a time constraint of 300 seconds.

A search strategy using `most_constrained` and `indomain_min` was also tested on the circuits' coordinates.

However, this only led to a worsening of the model's performance and hence, was not included in the final model.

3 SAT

3.1 Considerations on the Height

The problem was solved assuming that no gaps are allowed between the circuits, the plate height is thus obtained by taking the maximum between the sum of the circuits' areas divided by the plate width and the height of the highest circuit.

To solve cases for which this does not apply the code could be modified by adding a for loop that tries to solve the problem with increasing heights, starting from the calculated height.

3.2 Variable Encoding

Each circuit corresponds to a boolean variable indicating if the given circuit is or not located in that specific position. Each position in the plate is indicated by two lists of boolean variables to indicate the horizontal and vertical coordinates.

This creates a grid with true values only where the given circuit is placed ($gird[i][j][c]$ will be true if some parts of circuit c occupies position (i, j), false otherwise).

3.3 Constraints

For the following constraints *width* and *height* will be used to indicate the width and the height of the plate while *c_width* and *c_height* will be used to indicate the width and the height of a given circuit. To indicate the number of circuits C will be used, *grid* is the variable that is used to indicate the model.

- **At most one** circuit can occupy each place of the grid.

$$\bigwedge_{i=1}^{width} \bigwedge_{j=1}^{height} \bigwedge_{0 < c1 < c2 \leq C} \neg(grid_{i,j,c1} \wedge grid_{i,j,c2})$$

- Every circuit must be placed in its entire form in one of its possible positions. The possible positions of each circuit are indicated with $positions_n$, and **at least one** of them must be placed in the plate.

$$positions_{i,j} = \bigwedge_{c1=1}^{c} \bigwedge_{ii=i}^{i+c1\_width} \bigwedge_{jj=j}^{j+c1\_height} grid_{c1,ii,jj},$$

$$\bigwedge_{c1=1}^{c} \bigwedge_{i=1}^{width_{temp}} \bigwedge_{j=1}^{height_{temp}} \bigvee positions_{i,j},$$

$$width_{temp} = width - c1\_width + 1,$$

$$height_{temp} = height - c1\_height + 1$$

In practice there's no need to check that each circuit is in **exactly one** of its possible positions, because it would always lead to a worse solution.

3.4 Rotation

A simple idea to allow the model the possibility of rotation is doubling each circuit swapping its width and height coordinates and adding a constraints to avoid placing the same circuit twice.

A possibly better solution could consist in a list of boolean variables that is the same length as the number of circuits and keeps track of whether the circuit is rotated or not. Given this list, it's easy to find the actual width and height of each circuit and using those in the constraints.

3.5 Final Considerations

The current model was able to solve 20/40 instances with the time constraint of 300 seconds.

To improve this results another enconding of the variables was also tested: the circuits were represented with their left-bottom corners in the plate grid (as it was then used in SMT), but it had worse performance than this current model explained above.

Test concerning the implied constraints also seemed to generally worsen the solving time (although in some specif case they do help), so they were removed from the code.

4 SMT

4.1 Considerations on the Height

The problem was solved assuming that no gaps are allowed between the circuits, the plate height is thus obtained by taking the maximum between the sum of the circuits' areas divided by the plate width and the height of the highest circuit.

To solve cases for which this does not apply the code could be modified by adding a for loop that tries to solve the problem with increasing heights, starting from the calculated height.

4.2 Variable Encoding

Each circuit can be represented as its lower-left corner coordinates c (c_x, cy) and a number identifying the circuit (n).

4.3 Constraints

For the following constraints dc (dc_x, dc_y) will be used to define each circuit own given dimension and dp (dp_x, dp_y) for the plane given dimension. The list of circuits will be defined as C.

- **Inside**

  Each circuit's bottom-left corner needs to be in the grid so that all the circuit is contained in the plate. This means that every circuit bottom-left corner needs to have both of its coordinates equals or bigger than 0 and smaller than the plate (considering also that the whole circuit needs to be inside the grid):

  $$\forall c \in C : c_x \geq 0$$
  $$\forall c \in C : c_x + dc_x \leq dp_x$$
  $$\forall c \in C : c_y \geq 0$$
  $$\forall c \in C : c_y + dc_y \leq dp_y$$

- **Overlap**

  Each circuit cannot overlap with another circuit. This means that the distance between the two corners needs to be at least as big as the dimension of the first circuit.

  Given than circuit 1 x coordinate is bigger than circuit 2 x coordinate $(c1_x \geq c2_x)$ this constraints must be respected:

  $$\forall c1, c2 \in C \wedge c1 \neq c2 \wedge c1_x \geq c2_x : c1_x - c2_x \geq dc2_x$$

  For the opposite case $(c1_x < c2_x)$ the same reasoning can be applied. The same two case can be seen for the y coordinate of the circuits. It's enough that **at least one** this cases is true for the circuits to avoid overlapping.

- **Implied**

  The implied constraints were already given: the sum of the horizontal/vertical sides of the traversed circuits, can be at most the one of the plate. This means that for every row i the sum of all the lengths of the circuits that meet the following conditions has to be at most $plate_{width}$:

  $$\forall row \in rows : (\sum_{c \in C} dimension_{row,c}) \leq plate_{width},$$

  $$dimension_{ic} = \begin{cases} d_x, & \text{if c in row i} \\ 0, & \text{otherwise} \end{cases}$$

  The same needs to be done for the columns considering y coordinates and the circuit height.

4.4 Rotation

To allow the model the possibility of rotation we can add a list of boolean variables that is the same length as the number of circuits and keeps track of whether the circuit is rotated or not. Given this list, it's easy to find the actual width and height of each circuit and using those in the constraints.

4.5 Final Considerations

The implied constraints seem to generally worsen the solving time (although in some specif case they do help), so they were removed from the code.

The model was able to solve 37/40 instances with the time constraint of 300 seconds. Many of which can be solved in less than 60 seconds.