Report

Niccolai Lorenzo, Pozzoli Davide

Project report for
Combinatorial Decision Making and Optimization
Module 1
SAT

Artificial Intelligence
University of Bologna
Italy

Contents

Introduction

This is the Report for the first module of the Combinatorial Decision Making and Optimization exam for July 2021 for the SAT model.

1 Setting

We chose to model the problem in SAT, so we are using the z3 library in python. Similarly to the CP part of the project also in this case we use `matplotlib` python package to plot a graphical representation of the solution. We also reuse all the method created for the input output in CP since the instances are the same and the solutions we need to provide have the same structures.

2 7 points

As advised in the paper describing the project, we are going to tackle the seven points that brought us to the final implementation of the SAT model.

2.1 Point 1: Variables and Main constraints

For the variables we took inspiration from the sudoku example that we have seen during the course, in which there were `p[i][j][n]` boolean variables, with `i,j,n = 1..9`, this meaning that `p[1][1][1]` being true encodes that a '1' is placed in the cell `[1][1]`. In VLSI case we have something more complex than numbers: rectangles of various sizes. So we used `p[i][j][2*n´+1]` boolean variables:

- `i = 0..width`: width being in the input data
- `j = 0..height`: height is a variable nonetheless, but we will explain how we can create a fixed number of variables
- `n = 0..(2*n+1)`: `n` being the total number of circuits to place, we double the total variables because variables from 0 to `n` encode the starting point (i.e. bottom left corner) of the circuit and from `n` to `2*n` encode the occupied surface of the respective circuit. The extra variable encodes the blank space.

To sum up,thinking of the board as a matrix and given a cell in position `[0][0]` in an instance with 4 circuits to place:

- `p[0][0][1]` being true encodes that the left bottom corner of circuit number 1 is placed in such position
- `p[0][0][4]` being true encodes that some circuit number 1 area different from the left bottom corner of circuit number 1 is placed in such position

We set a fixed height in each instance of z3, to solve we start from the minimum bound of the board height (`min_h` is the maximum height among the input circuits) and we try iteratively to increase the height until we find a solution, using this method and stopping at the first solution found we assure optimality.

Constraints in propositional logic that we used, the first one states that for each "cell" of the board only one value can be encoded, meaning that with fixed `x` and `y` only one of the `2*n+1` variables can be true. This is done using the `exactly_one` function that adds the following constraints to the solver:

- $p[i][j][0] \vee p[i][j][1] \vee \dots p[i][j][(2 * n + 1)]$ with `i` and `j` varying all over their ranges
- $\neg(p[i][j][x] \wedge p[i][j][y])$ with `i` and `j` varying all over their ranges, `x` and `y` assuming indexes for every possible combination in range `0..2*n+1`

The second constraint states a circuit has only one position, this is doing using again the `exactly_one` function:

- $p[0][0][h] \vee p[0][1][h] \vee \dots p[width][height][h]$ with `h` varying over `n`, encoding a circuit bottom left corner
- $\neg(p[i][j][h] \wedge p[i1][j1][h])$ with `i` and `i1` assuming indexes for every possible combination in range `0..width`, `j` and `j1` assuming indexes for every possible combination in range `0..height` and `h` varying over `n`, encoding a circuit bottom left corner

The third constraint states that a circuit can not exceed the bounds (width and height):

- $p[0][0][h] \vee p[0][1][h] \vee \dots p[width - sizexh][height - sizeyh][h]$ with `h` varying over `n`, encoding a circuit bottom left corner, and `sizexh` and `sizeyh` encoding respectively the two dimensions of circuit `h`. Subtracting the size of the circuit from the ranges of the first two indexes it is guaranteed that the circuit will not exceed bounds.

The fourth and last constraint deals with overlapping of the circuits, this is where the other unused variables come in handy:

- $p[i][j][h] \Rightarrow p[k][u][h + n]$ with fixed h we have:
  - `i` that varies over `0..width-sizexofh`
  - `j` that varies over `0..height-sizeyofh`
  - `k` that varies over `i..i+sizexofh`
  - `u` that varies over `j..j+sizeyofh`

  this with the exception of when both `k` and `u` are equals to `i` and `j`, since that is the bottom left corner.

this constraint forces the rest of a circuit area (given the starting point) to correspond to the variable that we reserved for such case.

2.2 Point 2: Implied constraint

Even if in CP the introduction of such implied constraint saved some execution time, i with our SAT encoding of VLSI problem is not possible to state such constraint.

2.3 Point 6: Hypothetical model with rotation allowed

With rotation allowed each circuit can be considered as 2 different instances of such circuit. An idea would be to try to fit only one of these 2 possible circuits in the board, this can be done in propositional logic, even if this leads to a overhead in time since the boolean variables would be duplicated.

3 Results

The results with this model are not very good, having the 5 minutes limit we got stuck before the 10^{th} instance. Possibly we are tackling the height issue in a wrong way, since a lot of time is used to try solving the board with height which will not do.