THE AMERICAN UNIVERSITY IN CAIRO
الجامعة الأمريكية بالقاهرة

CSCE330401/ Digital Design II

A Simple Simulated-Annealing Cell Placement Tool

Fall 2022

Supervised by:
Dr. Mohamed Shalan

TA: Eng. Mohamed Abd Elmonem

Yara Ali 900183117
Yasmina Ramdan 900171547

Table of content

1. Introduction……3

2. Algorithm ……3

3. Implementation……..4

4. Cooling Rate Results……5

1. Introduction:

   We developed a simulated annealing cell placement tool that minimizes the total wire length used in connecting the cells in a grid using the half-perimeter of the smallest bounding box containing all pins for a net and depends on the cooling effect.

   Our work on Git Hub:

   https://github.com/YaraYahia17/DDII_Project

2. Algorithm:

   We create an initial random placement. Then mark the current temperature as the initial one and while is it still greater than the final one, we swap cells and calculate the change in the wire length. If it is a negative change, we accept the new wire length as the final one and if not, we reject with probability $(1 - e^{-\Delta L/T})$ and schedule temp.

3. Implementation:

The algorithm is divided into three steps. First, it parses a text file that contains the number of cells, number of connections, grid size (to create the 2d array) , and a line for each net that describes the number of cells in the net and which cells are there. Secondly, we created a 2d array initially empty that represents the initial grid then we implemented a random function that loop over a list of the cells and put them in random locations in the 2d array. Then, we started calculating the wire length using the half-perimeter of the smallest bounding box containing all pins for each net (HPWL). We did that by selecting the net's cells and getting the difference of the minimum and maximum x and between the minimum and maximum y then adding them to get the initial total wire length. Third, we started doing the simulated annealing algorithm and scheduling. As long as the initial temperature calculated by (500*initial cost) is greater than the final temperature, as the temperature decreases by (0.95*current temperature) each time the number of swaps per temperature ends. We first picked any two cells and swapped these cells. Then, calculate the total wire length again, if it is less than the previous wire length, then

we pick it as the best solution. If no, we reject with probability calculated in the code by the difference in the wire length and current temperature. After getting the best total wire length by iterating over the nets, we place the new cell locations on the grid. Then present the final cell placement on the 2d array.

We tried reducing the algorithm complexity by mapping the cells to the nets they appeared at and then when swapping any two cells, it check the map first and instead of going for the whole new grid to calculate the wire length. It go over the cells of the nets associated to the swapped cells to calculate the wire length again.

4. Results:

   We ran the three different test files. In each run it successfully changed the cells placement and reduced the wire length.