EECE583 - Assignment 2: Simulated Annealing Placement

Jose Pinilla

I. Introduction

This program performs the placement of benchmark circuits on a target matrix of cell sites using Simulated Annealing. The structures chosen for the Data are a Graph for the input circuit and a list for the target Cell sites. In the graph a Node represents each Cell and directed edges between the Nodes represent the connections. Each node has the *"nets"* and *"cost"* attributes. The "nets" attribute is a list of the Cells from which there are connections to the Node, this helps to keep track of the Nets for which a new calculation of the cost is necessary when a swap is made. The *"cost"* attribute is the Half-Perimeter of the bounding box of the Net originating from that Node. The *"cost"* attribute is initialized on a random placement and added together to get the *Total Cost* which is then incrementally modified on every change by subtracting the *"cost"* related to the nodes involved in the swap and then adding the new values.

The target cell sites matrix is represented by a list of *Block* elements, every Block keeps track of the cell assigned to it, if any, the state of occupancy, and its physical X and Y locations for graphical connections and cost calculation. In this way the data representations assimilate the input circuit for placement, and the target device topology. To find a Block's list index from the X and Y location it is simply the result of: Y * Columns + Y

Program Usage Syntax:
python placerA2.py -i "file" [-q] [-t int] [-s int]
-q Quiet Mode
-s: Inital Random Placement Seed
-t: Initial Temperature

II. Basic Algorithm

The simulated annealing algorithm found in the *_startplacement()* function is the main element of this program. After performing a random placement with any given seed as the program argument (-s "Seed") or the default "30" value, the program calculates the *Total Cost* of the obtained placement. The outer loop of the following computation obeys the minimal Temperature condition and executes *k* iterations of random swaps by selecting a random Cell as a victim and a random Site (Free or occupied) as a target.

The acceptance function (1) is then greater than 1 for every cost better than the previous cost and will randomly accept bad moves depending on the *rand* value generated on every iteration. A previous comparison of the cost delta is done to avoid computing the exponent in the following step and therefore improving performance.

$$rand > e^{\frac{-\Delta}{T}} \tag{1}$$

A. Data and Tuning

The best obtained results are shown below.

TABLE I
Time for execution

Benchmark	time(s)
alu2	110
apex1	235
apex4	685
C880	64
cm138a	31
cm150a	33
cm151a	32
cm162a	31
cps	447
e64	125
paira	164
pairb	137
Total	2094

TABLE II
Cost values

Benchmark	Cost
alu2	1176
apex1	8405
apex4	15537
C880	1360
cm138a	71
cm150a	173
cm151a	45
cm162a	210
cps	9489
e64	2876
paira	6600
pairb	6656
Average	4383.17

The parameters chosen during the tuning process are as follow.

Initial T	10
T limit	0.1
T update	0.99*T
K	N^(4/3)+(5*It)
Extension	Fixed Range Window of size M/2

It: Temperature update iteration
M: Size of the target matrix

The main conclusions about the values presented lie on T, K and the size of the Window Range. It is important to

have a number of iterations proportional to the number of cells to be placed and it is also beneficial to increase the number of iterations as the temperature decreases to allow more exploration as the probabilities of taking bad moves decreases. K is increased by 5 on every T update, the value 5 is completely arbitrary. The temperature range (10-0.1) was chosen with a high initial temperature but the update function rapidly decreases this value and contrary to a subtraction update the temperature decreases by a smaller value every iteration. The window range is explained on Section IV.

III. Graphics

Three graphic elements are present in the program, a static Graph showing the input circuit, a Cost plot, and a graphic representation of the resulting placement including connections. Since the dynamic (Cost and Placement) elements are a workload for the algorithm, these are disabled by default and give the user the option to show or stop updating them during the computation with the ”Plot” and ”Draw” buttons. These elements are always displayed at the end of the execution even if the corresponding buttons are not pressed. A more ”performance-centered” version can be executed with the quiet mode switch (-q). Results are always appended to the *results.txt* file in the working directory and an external script is used to compute the average of all benchmarks.

IV. Additional work

A fixed size implementation of a window range was used to optimize the process. The chosen window size is half the size of the cell matrix and instead of randomly generating a Cell Site X and Y the random generator gives a number between negative window size and window size and adds it to the axis value of a randomly chosen populated cell. In this way there are no swaps between two free nodes, and swaps are constrained to nearby cell sites.