THE AMERICAN UNIVERSITY IN CAIRO

School of Sciences and Engineering

CSCE 3304-01 Digital Design II

By:

Mazen Hassan - 900191797

Miriam Sameh Youssef - 900191957

Fall 2022

Table of contents

Algorithm Overview

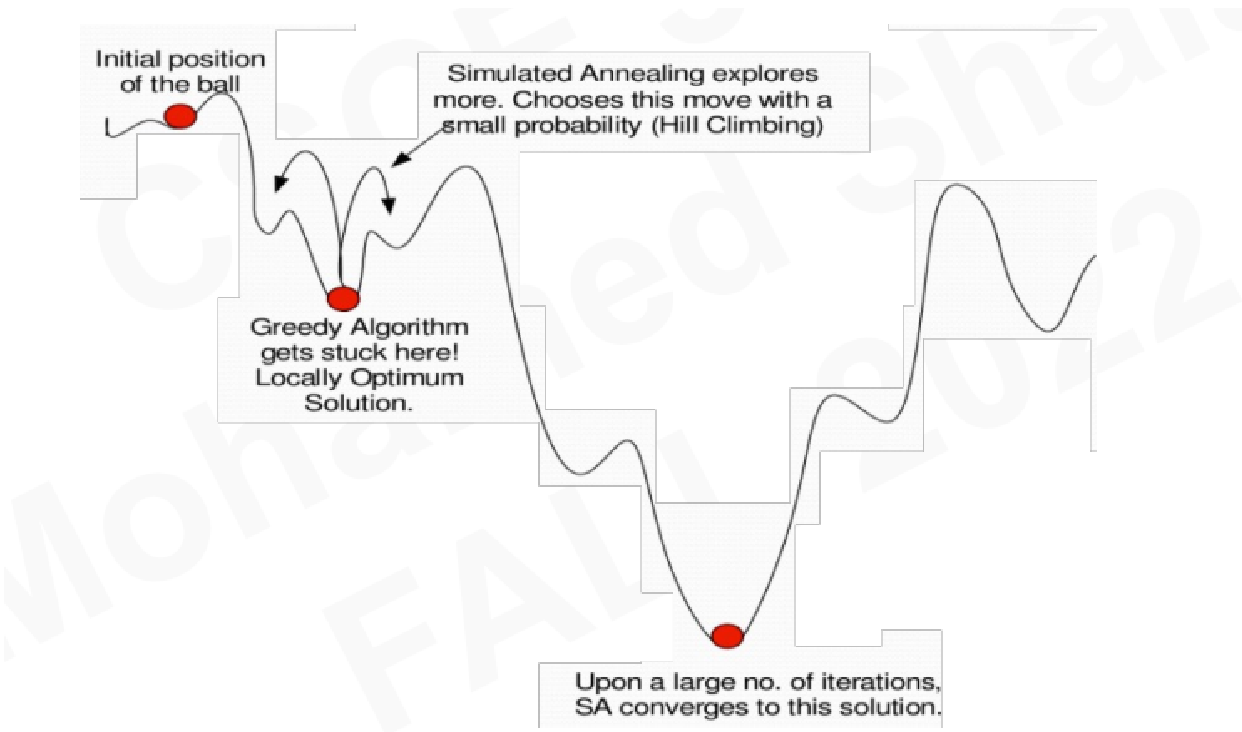
The simulated annealing procedure is an optimization technique that simulates the gradual cooling of metals, which is characterized by a gradual decrease in the atomic motions that lower the density of lattice defects until a lowest-energy state is obtained. Similar to this, the simulated annealing algorithm provides a new possible solution (or neighbor of the present state) to the issue under consideration by changing the current state in accordance with a preset criterion at each virtual annealing temperature. The Metropolis criteria is then used to determine if the new state should be accepted, and this process is repeated until convergence. The metropolis is calculated in our project by raising e to the negative of the difference between the proposed wire length and the current wire length over the current temperature.

The simulated annealing is an optimization algorithm that indicates that randomization is included into the search process. Because of this, the technique works well for nonlinear objective functions, unlike other local search algorithms.It adjusts a single solution and scans the relatively local region of the search space until the local optima is found, much like the stochastic hill climbing local search algorithm. It may accept less desirable alternatives as the current working solution, unlike the hill climbing algorithm.

Simulated Annealing vs. Greedy algorithm

The fundamental difference between greedy search and simulated annealing is that, whereas greedy search always selects the best proposal, simulated annealing has a chance of rejecting it and selecting a worse one . By leaving the local optimum, this aids the algorithm's search for a global optimum. Temperature and other factors are only a supplemental benefit (or drawback) of SA.

Simulated annealing algorithm has the option to first discover the region for the global optima, escape the local optima, then hill climb to the optima itself. The likelihood of accepting poorer solutions starts high at the beginning of the search and lowers with the progress of the search.



Lecture 17, Prof. M. Shalan

Implementation

The first part of the program is the parsing of the text files in order to extract the needed relevant information for the rest of the program to work. Firstly the user enters the name of the file they want to parse. Secondly, the regex library is used to extract all the digits from the first line of the program, and that allows us to get the number of nets, cells, and the dimensions of our board. The program then initialises a board where are all elements are '---' with the dimensions that were extracts from the txt file using the initialisearray() function. The program then iterates over every line, and extracts the components that need to be place onto the board, while also removing unwanted white spaces and line terminators. The program then randomly generates two indices for our rows and columns, and proceeds to place the components from the generated list onto the board, while checking that the location of the placement is a valid location, where a '---' symbol exists, denoting that this point on the board is an empty location that has not had a component placed on it

The program than proceeds to calculate the initial distance on the random initial board that was generated. It does this by calling the calcdistance function. The calcdistance function takes the list that indicates how the components are connected, as well as the board itself, and caluclates the half perimeter distance of a rectangle that has the elements that form the net in question. This gives us the total wire length between the elements in that one specific connection. A loop is created that loops over all the connections in our list, and the half perimeter distance is calculated for each of them. The sum of all those values gives us the initial total wire length of our board

The next step in the program is the annealing algorithim, denoted by the thermal_an() function. The function takes the board, the connection list, the initial wire length, the number of nets, and number of cells. The function then calculates t_initial and t_final, as well as the number of moves. The program then proceeds to go into a while loop, on the condition that current temperature >final temperature, and does the following:

1. Swaps two random components in our board
2. Calculates the wire length after the swap
3. Calculates the delta_l difference between initial wire length and wire length after the swap
4. If delta_l is negative, it is accepted, or it is rejected with probability 1-exp(delta_L/temperatures)
5. We then check the number of moves against the number of iterations. If they are equal iterations are reset, and the temperature=temperature*0.95
6. This is done until the condition of the while loop is met, after which the program prints out the final placement and the final wire length of the board

Final Results

The outputs of all the text files are included in the results folder in the github repo as well as the bonus implementation. It should be noted that in d2.txt, number of moves was 10*260=2600, meaning that for the temperature to change it would require 2600 iterations each time. The complexity of the program results in a extended time requirement for D2 to produce a result with its original parameter, thus its result is with 350 as the number of moves, as well as the next step temperature being 0.75 instead of 0.95 for testing purposes, indicating that the algorithm does decrease the total wire length correctly, however the actual result that would be produced when setting moves as well as the next step temperature to its accurate number would be a lower number, as the program would run through more iterations

Bonus Implementation

We added the graphical representation of the cells on the board in this below graph. By looping through the rows and columns in the board and checking if the place is not empty "---" then we plot using the scatter function.