**Assignment 1 (CSI 5137 [B] – AI Enabled Software Verification and Testing**

Ravisha Sharma (Student Id: 300162406)

Syeda Zainab (Student Id: 300303434)

**Objective**

The objective of this report is to summarize the implementation of a solver for Travelling Salesman problem using Hill Climbing, which is a metaheuristic search algorithm. The report covers an overview of traveling salesman problem, hill climbing algorithm, explanation of our approach and its comparison with Random search algorithm.

**Traveling Salesman Problem (TSP)**

If we have a list of cities and the distance between each pair of cities, the Traveling salesman Problem is to find the shortest distance such that the salesman visits each city exactly once and then returns back to the original city. It is an NP hard problem. TSP can be symmetric or asymmetric. In this report, we only focus on symmetric travelling salesman problem which means that the distance between two cities is same in opposite directions.

**Hill Climbing Algorithm and why this approach**

Hill Climbing is a greedy local search algorithm. There are various types of hill climbing such as:

* Simple Hill Climbing
* Stochastic Hill Climbing
* Steepest Ascent Hill Climbing

In Hill Climbing algorithm, the state of the neighbour node is evaluated at each step. If the neighbour node optimizes the current cost or is better than the current state, it moves to the neighbour node. Otherwise, it stays in the same state. The process is continued till a solution is found or there is no new operator left to apply. This algorithm does not remember previous states so there is no backtracking involved.

We opted for this algorithm as it is very popularly used for optimization problems in the field of AI and has been widely used for solving travelling salesman problem. Also, this algorithm is simpler and has the ability to generate good results.

**Our Approach**

Our implementation of TSP using Hill Climbing involves various functions. An overview of each of the functions is provided below: -

1. get\_euclidean\_distance (p, q): This function accepts two co - ordinates and returns the Euclidean distance between these points. This function is used to construct a distance matrix which consists of distance between every pair of cities/points.
2. get\_random\_soln (distance\_matrix): This function accepts the distance matrix that includes distance between every pair of cities and returns a random solution for travelling salesman problem such that each city is travelled exactly once.
3. path\_length (distance\_matrix, soln): This function accepts the distance matrix and the solution to travelling salesman problem and returns the length of the specific solution.
4. find\_Neighbors(solution): This function accepts an existing solution and generates neighbouring solutions to that solution.
5. find\_best\_neighbor (distance\_matrix , neighbors): This function accepts the distance matrix and a list of neighbors from the getNeighbours function and finds out the best neighbour and best route length.
6. hill\_climbing(distance\_matrix): This function accepts the distance matrix and applies the hill climbing approach to it. Since we already have all the functions ready for this approach, we start by generating a random solution to this problem. We then generate neighbouring solutions by using the getNeighbors () function and identify the best neighbour by using the find\_best\_neighbor () function. As long as the best neighbor is better than the current solution, same steps are repeated, and the current solution is updated with best neighbor. Finally, current solution and its route length is returned.

**Representation, Fitness and Operator**

1. Representation:

* We start with a list of cities where we have co – ordinates of each of the cities.
* We form a distance matrix that that specifies the distance between every pair of cities.
* We find out a random solution to travelling salesman problem such that each city is visited exactly once.
* We find all the neighbouring solutions to the existing random solution.
* We find the best neighbor from amongst the existing neighbors.
* Finally, we implement hill climbing algorithm where we perform all the above listed steps in the given order. Till the time, the best neighbor is better than the current solution, same steps are repeated, and the current solution is updated with best neighbor.

1. Operator:

* At each step in the algorithm, an inspection of the neighbors is performed, and a selection is done.
* If the total route length on choosing a neighbour A is shorter than the total route length on selecting other neighbouring solutions, neighbour A is selected.
* The above steps are repeated, and the current solution is updated with the best neighbour and the current route length is updated with the best neighbour’s total route length till the time best neighbour is better than the current solution.

1. Fitness Function: As shorter route is considered better for TSP. The fitness function is defined as the inverse of the total distance/ route length. The higher is the value, the better the algorithm is.

**Results**

We tried running our travelling salesman problem on four datasets namely pr76, a280, berlin52 and burna14. We plotted the route lengths and directed graphs of hill climbing approach for each of these datasets. Please refer to **Figure 1** for the directed graph and output on pr76 dataset and **Figure 2** for berlin 52 dataset. Rest of our outputs and directed graphs are available in the ‘Output\_images.docx’ file. We have also plotted the fitness versus iteration graph and the same is available in ‘Output\_images.docx’ file.

A picture containing chart

Description automatically generated



**Figure 1: Directed graph using hill climbing for pr76 dataset**

Text

Description automatically generated with low confidence

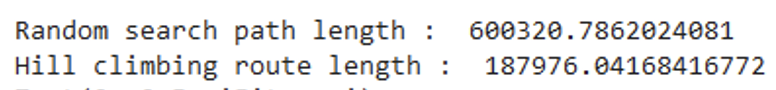
****

**Figure 2: Directed graph and route length using hill climbing on berlin 52 dataset**

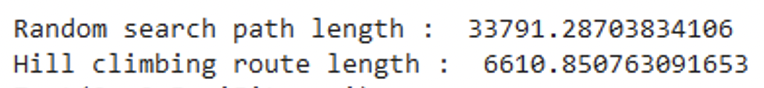
***Note:*** *Due to large number of nodes in pr76 dataset, we did not plot co -ordinate values in the directed graph.*

**Comparison with Random Search Algorithm**

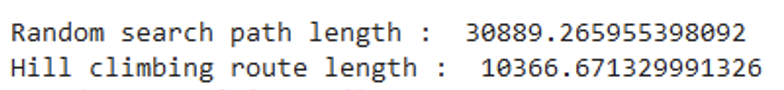
We also compared the total route length for hill climbing versus random search on the aforementioned datasets. For each of the datasets, hill climbing generated a more optimum solution (a shorter total route length) for travelling salesman problem. Please refer to **Figures 3 to 6** for the results on all the four datasets.



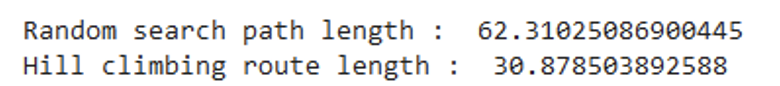
**Figure 3: Total route length on pr76 dataset**

****

**Figure 4: Total route length on a280 dataset**

****

**Figure 5: Total route length on berlin52 dataset**

****

**Figure 6: Total route length on burna14 dataset**

**References**

1. <https://en.wikipedia.org/wiki/Travelling_salesman_problem>
2. <https://www.javatpoint.com/hill-climbing-algorithm-in-ai>
3. <https://towardsdatascience.com/how-to-implement-the-hill-climbing-algorithm-in-python-1c65c29469de>
4. <https://www.annytab.com/hill-climbing-search-algorithm-in-python/>