**DESCRIPTION:**

For the given Travelling salesman problem, we implement two search strategies namely simple and sophisticated to find the optimal solution.

**EXPLORED STRATEGIES:**

**Genetic Algorithm:**

The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population evolves, toward an optimal solution.

After having implemented genetic algorithm with 5000 generations, and a mutation probability of 0.02 with a population size of 25 it was observed that the populations are not converging to give the fittest solution The tournament selection is done at random from the initial population, using the tournament size . Hence, other strategies were considered to implement.

**A\* algorithm**

For A\* search, the evaluation function is defined as f(n)=g(n)+h(n). Though g(n) can be computed easily by using the cost functions given because g(n) is simply the cost from start city to the current city, h(n) is the estimated cost of the path from current city to goal city which is not yet determined. Hence obtaining a good heuristic requires a lot of domain knowledge. Thus, other strategies were considered.

**IMPLEMENTED STRATEGIES**

The program functions as follows:

1. The script prompts the user to enter the input data to the console
2. Based on the strategy selected, simple or sophisticated algorithms are implemented.

**Simple Strategy:**

To implement a simple search strategy, the concept used here is the nearest neighbor search.

The nearest neighbor search algorithm also known as greedy search works as follows:

For given list of cities,

1.select a city as current city

2. Find the shortest edge connecting the current city.

3. We set the new city as current city

4. We add the previous current city to the list “visited”

5. We repeat this process until all cities are visited once and only once.

Implementation:

1. We create a list of cities, L from 0 to N-1.
2. Since we need to find all the possible routes starting from any city and then compute the optimal, we iterate the following procedure from 0 to N-1.
3. We take a city from L and then remove the city from L.
4. Add the city to visited.
5. We then compute the list of all cities along adjacent to the current city and the corresponding distances.
6. We choose the city with minimum distance as our next current city. This minimum distance is added to optimal distance
7. We remove the current city from list L and iterate over the process.
8. We add the distance of the last city to the first city after the steps 3-7 and compute optimal distance
9. By repeating steps 3-8, we get N tours with N optimal distances

10.The path with minimum distance is taken as the optimal path

**RESULTS:**

|  |  |  |  |
| --- | --- | --- | --- |
| Number of cities | C1 | C2 | C3 |
| 10 | 426  426  426 | 58  58  58 | 1050  1050  1050 |
| 30 | 466  466  466 | 156  156  156 | 33350  33350  33350 |
| 60 | 526  526  526 | 247  247  2642 | 277300  277300  277300 |
| 120 | 646  646  646 | 424  424  12752 | Not required as per project specification |

**Sophisticated algorithm:**

When the user prompts to use a heuristic search strategy, we implement our simulated annealing algorithm.

Algorithm:

Simulated Annealing uses a random sampling . The algorithm works as ollows:

1. We first generate a random solution
2. We calculate the cost of the solution using our cost functions,
3. We then generate another neighbouring solution at random.
4. We calculate the cost of the new solution.
5. We compare the new and old costs.

6.A If new cost>old cost, we may probably move to new solution.

6.B If new cost<old cost , we move to the new solution.

7. We repeat step steps 3-6 until the maximum effort bound condition is satisfied or solution is obtained.

With the help of this pseudocode, the algorithm was implemented.

**Implementation**:

1.We take the initial Temperature to be 10000

2. The cooling factor taken for our program is 0.99

3. The lower end of temperature is taken as 0.1.

4. We generate an initial route at random from the list of N cities and calculate its length.

5.We take this initial route , distance as best.

6. We iterate the loop until the meb is reached.

7. We then swap cities randomly from the route and assign it to new route

8.The new cost is computed for the new route.

9. if the new cost is less than the old cost, we take the new route and new cost as best.Else, we use the probability function.

10. The probability function a is given by exponent of ((new cost-old cost)/T), where T iterates on the loop updating Tmin to tmin\*the cooling factor every time.

11.This process is repeated until the conditions are met.

**RESULTS**

|  |  |  |  |
| --- | --- | --- | --- |
| Number of cities | C1 | C2 | C3 |
| 10 | 227  426  227 | 64  64  64 | 970  970  970 |
| 30 | 307  307  307 | 268  268  268 | 32510  32510  32510 |
| 60 | 428  565  582 | 582  582  582 | 273820  273820  273820 |
| 120 | 668  6 | 1194  1194  1194 | Not required as per project specification |

**REFERENCES**

<http://katrinaeg.com/simulated-annealing.html>

<https://www.slideshare.net/kaalnath/comparison-of-tsp-algorithms>

<http://www.obitko.com/tutorials/genetic-algorithms/index.php>

https://www.mathworks.com/matlabcentral/fileexchange/25542-nearest-neighbor-algorithm-for-the-travelling-salesman-problem