|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Number of cities | Cost to traverse all cities in Brute force | Time needed for Brute force (seconds) | Cost of Genetic algorithm | Time needed to execute in genetic algorithm (seconds) | Number of generations used to calculate | Cost to traverse Minimal spinning tree | Time needed in MST  (Seconds) |
| 10 | 23 | 0.288319 | 36 | 0.000086 | 100 |  |  |
| 11 | 23 | 3.469 | 43 | 0.000222 | 1000 |  |  |
| 12 | 27 | 42.287 | 59 | 0.000890 | 1000 |  |  |
| 13 |  | More than 5 minutes | 64 | 0.001552 | 10000 |  |  |
| 14 |  |  | 70 | 0.001613 | 10000 |  |  |
| 15 |  |  | 75 | 0.016556 | 100000 |  |  |

**Introduction:** In this project our goal was to implement and find a generic way to solve the traveling salesman problem using three different algorithms. One is using brute force technique another one is using genetic algorithm and the other one is using minimal spinning tree. Using the first two of them I tried to get some results but couldn’t able to solve the minimal spinning tree. In the Brute force technique the challenge was to implement the permutation of some number. I tried some technique that convert the characters into number then I’m calculating the shortest path from starting to all other nodes and then coming back to the starting node.

Here is the experiment table for the algorithms for a round trip from one node to all other nodes in the given graph.

**Brute Force:**

In the brute force I’m doing a permutation to all of the cities and then reading the edges from the file. After that I’m building a adjacency matrix from the edges. Then I’m calculating the cost of the edges from the adjacency matrix and finding the minimum cost for a round trip from starting vertex and again I’m returning there. The result is given above. I’m getting result of 12 cities before the execution time reached more than 5 minutes.

**Genetic Algorithm:**

In the genetic algorithm I’m dividing the array of the vertices and then I’m adding the values of the both array one by one to the main array of the vertices. I tried to do it like the merge-sort. Merge sort divide the array and then sort the array when there’s two elements but I’m not sorting the array I’m just adding the values from both of the arrays one by one to the main array. For example if the main array contains 4 vertices. I denoted the vertices by number. In the file characters denote the graph but I converted them into number for the calculation. If there are 4 vertices in then there will be 1 array consists of 1, 2 another will consists of 3 & 4. Then I’m merging the array by adding one element from first array and another from second array, so the new array will be now 1 3 2 4. I’m doing this as per generations is given in the command line. I couldn’t do the algorithm with numTours & percent parameter for the shortage of time.

**Minimal spinning Tree:**

I couldn’t able to finish this program due to the shortage of time. I was trying to implement the prim’s MST algorithm.

**Conclusion:** It can be seen in the brute force technique that all possible routes are visited somehow either it is optimal or not that’s why it’s taking so much time when the number of vertices is higher. I didn’t ever get the result for more then 12 cities in the graph. In the Genetic algorithm the time is taking less but it’s not optimal in the perspective view of the cost to visiting all the cities. I tried to implement a technique that much similar with merge-sort. I showed some result using that one. For the MST I tried but couldn’t able to solve it properly.