CS6380: TSP using Genetic Algorithm

V.C.Ashwin
EE16B041
Muqeeth Mohammed
EE16B026

September 15, 2019

1 Introduction

We have implemented a genetic algorithm to find the best tour of the TSP. The following indicate the various stages of the genetic algorithm:

- Generate initial pool
- Selection
- Crossover breeding
- Mutation
- Next-Generation

2 Constructive Methods

We have used 3 different types of heuristic methods to come up with the initial pool to perform better on both the euclidean and non-euclidean distances.

2.1 Double Tree algorithm

This method is an Minimum spanning tree based algorithm and gives best results when implemented on euclidean distances where the Δ-inequality is satisfied. It is a 2-approximation of the optimal solution *i.e* it gives a path whose cost is atmost twice the optimal cost. The MST of the graph is constructed and a tour is generated from it, starting from a node arbitrarily. We have used the Prim's algorithm for MST-implementation and DFS for tree traversal.

2.2 Modified Greedy Approach

This heuristic method starts from a randomly chosen node and adds it to the tour. Next we select the best neighbours greedily in each iteration. For every node, we iterate through all the edges between the currently added nodes in the tour and find the best position for the current node. This works better than the normal greedy approach since we consider the min-cost tour after each addition of a node.

2.3 Random Heuristics

To introduce a certain randomness in the initial pool, we have generated tours randomly by shuffling the cities. This is done to introduce tours which may ultimately lead us to the optimal tour.

3 Perturbation Methods

3.1 2-Edge Exchange (2-Opt)

We take a given tour and make it 2-optimal by exchanging all possible combinations of edges taken 2 at a time. The minimum cost tour amongst these ${}^{N}C_2$ exchanges is chosen and replaced with the current tour in the population

4 Genetic Algorithm

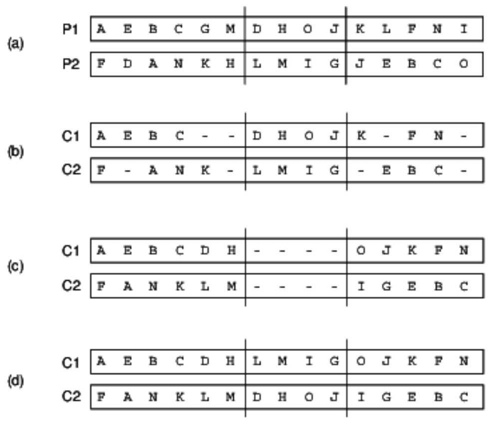
4.1 Selection

We have defined $fitness$ - as the inverse of the tour cost. The selection of parents for crossover is done based on a $roulette - wheel$ based selection. It is similar to rotating a wheel and picking the value which the pointer indicates. The current population is sorted in descending order of fitness and their cumulative probabilities are generated. Now, a pick is selected at random and the tour just above the pick is selected. This cumulative sum based method replicates the roulette-wheel mechanism in picking the tours with higher fitness frequently.

4.2 Crossover

We have used the $NWOX$- Non Wrapping Order Crossover technique as it showed faster convergence than the Partially Mapped crossover (PMX). Let $P1, P2$ be the parents and $C1, C2$ be their children. Now we select 2 indices a, b ($a \leq b$) to retain the gene from the parents. $C1$ is searched for the locations of the values $P2[a : b]$ and replaced with holes. Similarly $C2$ is searched for the locations of values $P1[a : b]$ and replced with holes. Now, we slide these holes in $C1, C2$ into the gene region $[a : b]$ as shown in the below figure. Replace the gene of $C1$ with that of $P2$ and $C2$ with that of $P1$.

Figure 1: NWOX Example



4.3 Mutation

After crossover, we perform the 2-edge exchange over the crossed children to introduce mutation. This ensures that the new children always tend to have the minimum cost possible within their tours.

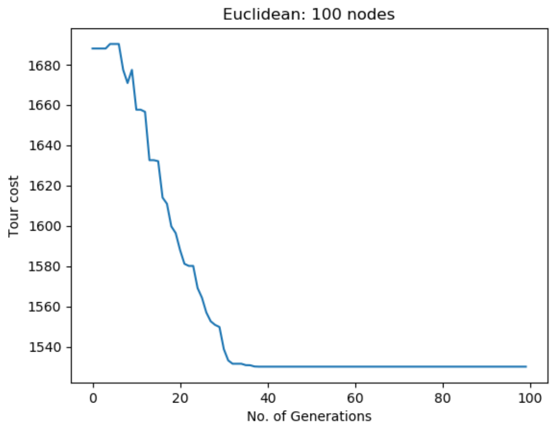
4.4 Elitism and Next-Generation

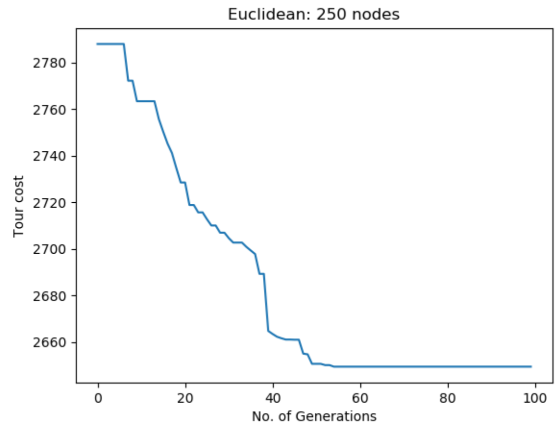
The next generation of population is created by taking the crossover children and also retaining the parents from the previous generation. This is because propagating the best tours of the parents gives a higher chance to end up with the optimal tour.

5 Results

The following results were obtained on different test cases for 100 generations. The initial population was taken in the ratio 1 : 3 : 1 from the MST algorithm, Greedy approach and randomised tours respectively. The next-generation was populated using half of the best parents and the remaining half of the crossed children.

5.1 Euclidean Test Cases





5.2 Non-Euclidean Test Cases