SHARIF UNIVERSITY OF TECHNOLOGY
PHYSICS DEPARTMENT

Bonus Problem Set

Student name: *Ali Fayyaz - 98100967*

Course: *Computational Physcis - (Spring 2023)*
Due date: *June 24, 2023*

Exercise 12.1

Open Traveling Salesman Problem

Use genetic algorithm with elitist selection.

(a) Plot the shortest path vs. the number of iterations of the genetic algorithm.

(b) Plot the time it takes to a reach a final solution as a function of n (the number of cities). A final solution is one that hasn't changed during the simulation.

Answer. The following functions were defined:

- **initialize_points():**

  This simple function generates n random points on a 2D grid; returns an array of shape $(n, 2)$.

- **cost():**

  Calculates the total cost of a route. The cost of a route is simply defined as the Euclidean distance between two consecutive points in the route. According to stackoverflow, *numpy.linalg.norm()* was used.

- **mutate():**

  Takes route, as an array of length n, and swaps two random points, based on the given mutation probability. The mutation probability has a default value of 0.08.

- **rank_selection():**

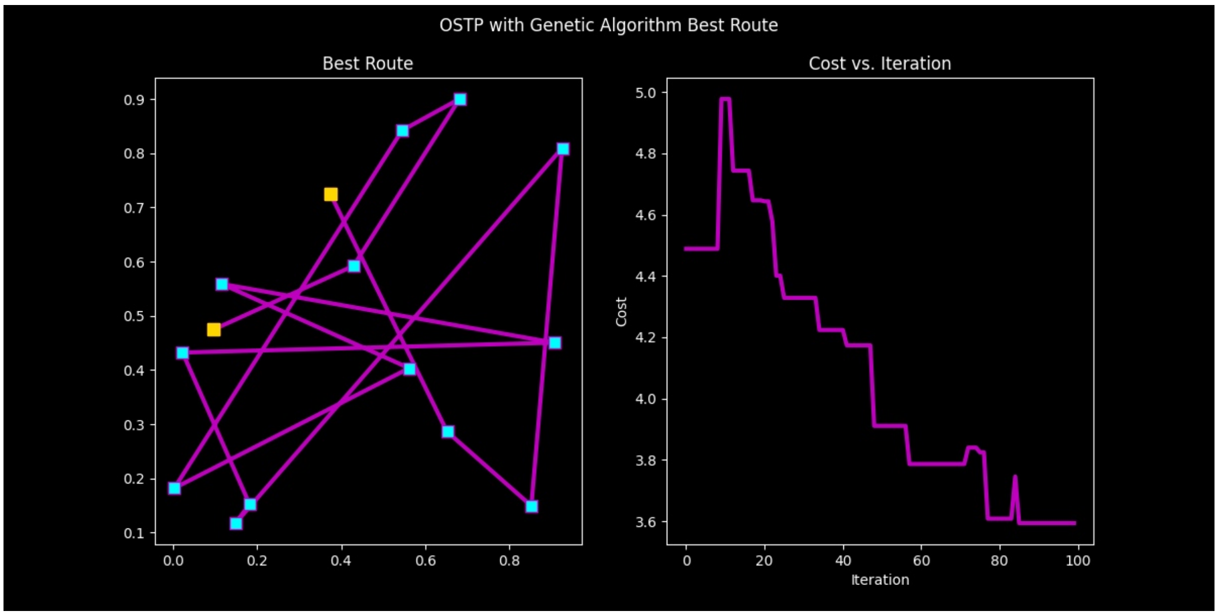
  Takes a population of routes (default size of population is 100) and performs rank selection based on the following probability distribution:

$$P(R_i) = \frac{1}{n}\left(sp - 2(sp-1)\frac{i-1}{n-1}\right) \quad ; \quad 1 \leq i \leq n \quad ; \quad 1 \leq sp \leq 2$$

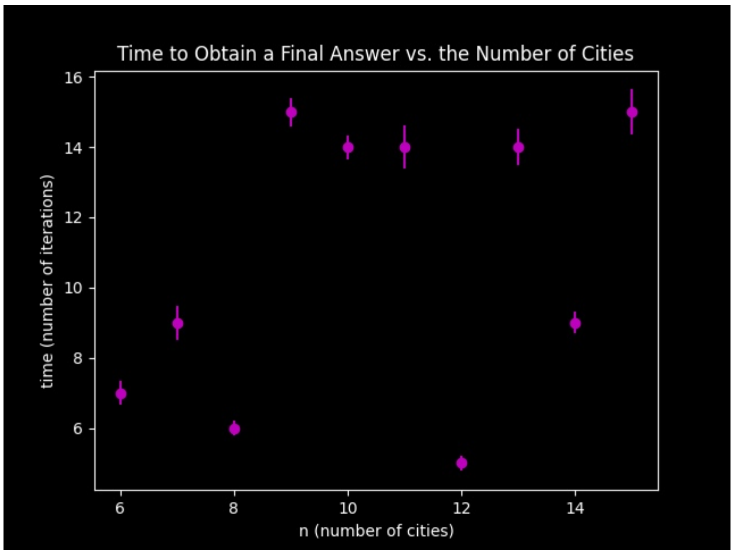
  A value of $sp = 1.4$ was used in the simulations. The function returns the top two most fitted routes in the population, as a form of elitist selection, and also another randomly chosen route, based on the aforementioned probability distribution.

- **genetic_algorithm():**
  This function takes as input n, the number of cities (points on the grid), *population_size*, the number of arrays to be kept in each generation, *sp*, selection pressure, *generations*, the number of generations (number of iterations of the genetic algorithm). It returns the best route, its cost and the cost of the best route at each iteration.

For 15 cities, population size of 100, mutation probability of 0.08, selection pressure of 1.4, the genetic algorithm was run for 100 iterations. The results are plotted as follows (Start and end points are shown in different colors than the rest.):



For values of $n \in \{6, 7, 8, ..., 15\}$, each with an ensemble size of 10, the number of iterations needed to reach a rather stable final solution was recorded. A final solution was defined as a solution that wouldn't change for 5 consecutive iterations of the algorithm. The computations are highly time-consuming, otherwise a value larger than 5 would have been chosen for this part. The resultant error bar is as follows. As evident from the plot, the higher the value of n, the more iterations should be taken to reach a rather stable solution.


Time to Obtain a Final Answer vs. the Number of Cities
time (number of iterations)
n (number of cities)