EE2703: Assignment 6 - Travelling Salesman Problem

Atreya Vedantam, EE22B004

26th October 2023

1 Approach

Given the (x, y) coordinates of n cities, the goal is to find the shortest travelling path to cover all of them. I have used simulated annealing.

Simulated annealing is based on the heat treatment process annealing. Here we initialize a random order of cities. I have defined a function called `travelDistance` which computes the total travel distance in the path specified. Then iteratively we choose two random cities and interchange their positions in the order. We check if the travel distance has now improved. If it has, we update the order and the best travel distance found thus far.

Otherwise we do not simply reject the modified order. We generate a probability with which we accept the modified order as:

$$p(\Delta E) = \exp\left(-\frac{s\Delta E}{T}\right)$$

where ΔE is the change in the travel distance, T is the temperature, and s is a factor that I have personally added (called the stochasticity) which is a measure of how much influence I want the probabilistic updates to have on the final result. This is analogous to the inverse of the Boltzmann constant present in the popular method for simulated annealing.

This iterative process is continued **as long as $\mathbf{T > 1}$**. Note that this significantly differs from the popular implementation since I have *not chosen to use a fixed number of iterations*. Part of the reason is that division by values close to zeros lead to lots of errors, especially when a code is running for as much time as this is. Hence I feel keeping the T above a fixed value and varying the decay rate to vary the number of iterations is a better choice.

1.1 `tsp`

Takes in the following arguments:

- `cities`: this is the set of n-tuples of the x and the y coordinates of the cities
- `stochasticity`: Optional and default set to 30. This is the same as adjusting the inverse of the Boltzmann constant (usually set to 1 by others, but I wanted to adjust the influence of the stochasticity of the process

1.2 `travelDistance`

Takes in the following arguments:

- `cities`: this is the set of n-tuples of the x and the y coordinates of the cities
- `cityorder`: this is the order of the cities to calculate the distance for

1.3 `relErrorCheck`

Takes in the following arguments:

- `cities`: this is the set of n-tuples of the x and the y coordinates of the cities
- `optimisedorder`: this is the optimal order found by the `tsp` function
- `initialorder`: this is the initial chosen order to compare the relative order against

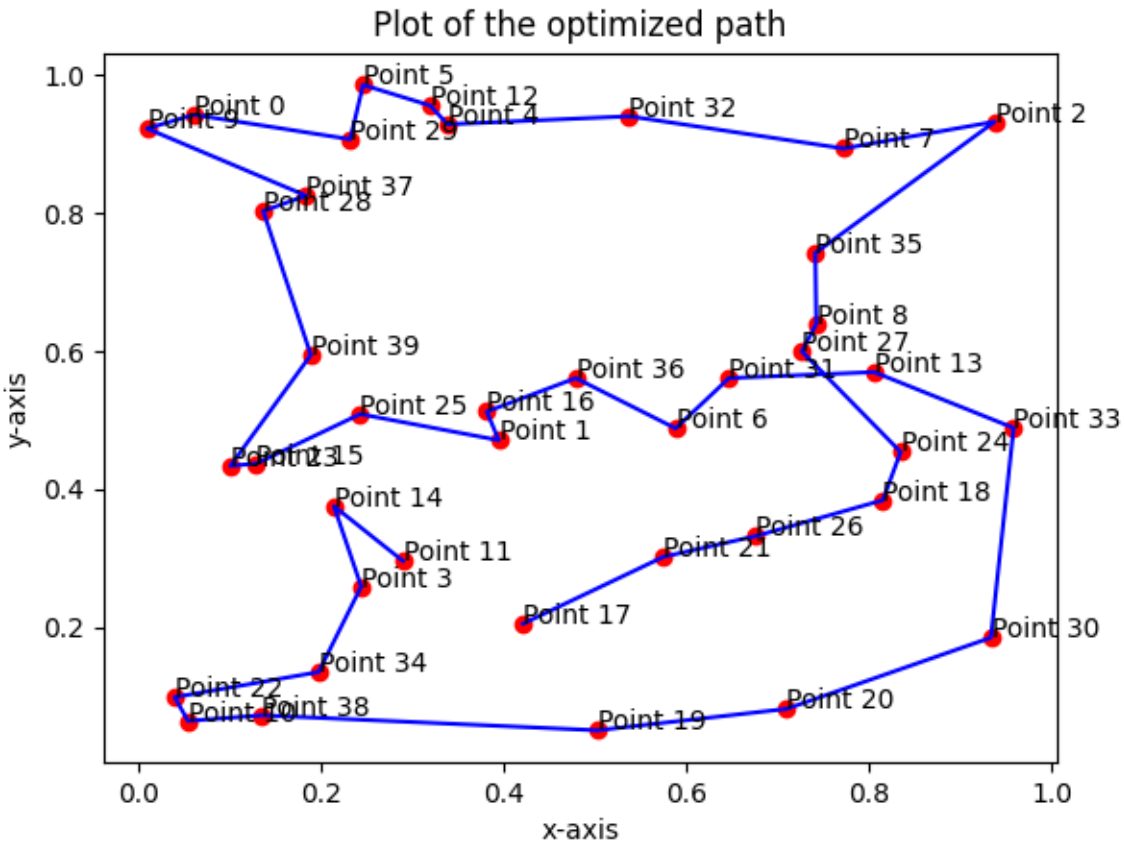

Figure 1: The optimized TSP-40 path

2 Parameters of Implementation

I have chosen $T = 1 \times 10^9$ and $k = 0.9999$ where k is the decay rate of the algorithm. I have chosen $s = 200$, a rather high number. I have experimentally observed that a larger value of s decreases the efficiency and a lower value of s seems to exhibit a great degree of variation in the answers for each run of the code. This runs the annealing algorithm for a total of 207,223 iterations. It takes 43.71 seconds to run the code.

3 Results for TSP-40

The improvement error is calculated by

$$\text{improvement error} = \frac{\text{initial distance - final distance}}{\text{initial distance}} \times 100\%$$

The optimized order is

$$\begin{bmatrix} 11 & 14 & 3 & 34 & 22 & 10 & 38 & 19 & 20 & 30 \\ 33 & 13 & 31 & 6 & 36 & 16 & 1 & 25 & 15 & 23 \\ 39 & 28 & 37 & 9 & 0 & 29 & 5 & 12 & 4 & 32 \\ 7 & 2 & 35 & 8 & 27 & 24 & 18 & 26 & 21 & 17 \end{bmatrix}$$

where the numbers corresponds to the cities numbered from 0 to 39. Note that the order has to be read row-wise. The distance covered is: **5.749831385493956** with an improvement of **71.98834406871971%**.

Please note that in the plot, the first and the last points have NOT been connected for better visibility. One can see that the plot begins at Point 11 and ends on Point 17. It is easy to visualize that the traveller has to now move from point 17 to 11.

4 How to Run the Code

Please ensure that the `filename` is set to the appropriate test case while checking. If the file name is `tsp_100.txt` then enter `filename = 'tsp_100.txt'`. Ensure that the text file is in the same directory as `simanneal.py`. Also please do note that the `distance` function which was expected in the question is written under the name `travelDistance` since the actual `distance` function is already used to calculate the distance between two cities.