ASSIGNMENT 6 : SIMULATED ANNEALING

I have used simulated annealing.

- I have defined a function `distance`. It takes in cities and city order to calculate the total distance travelled.(starting city and ending city are the same.
- I have defined the parameters temperature T and decay rate. They have to be changed according to the specific problem, for better results.
- I chose the first order to be in number sequence. And I found the distance travelled in this case and stored it as `initial_dist`.
- I defined tsp for implementing simulated annealing. My goal is to minimise the distance than the initial distance, so I tried swapping the order.
- For swapping:
  - I tried swapping the city order by random selection. I ran the code with different number of cities, even for small no. of cities, it didn't give satisfactory results. It gave me both worse and the best possibilites with equal chances. I had to change this random selection for swapping.
  - I used `np.random.permutation` under numpy. It checks for different permutations. Most of the times it gave me good results. For minimised the distance the code has to be run several times, by manual observation one has to choose the best minimised distance. The distance travelled is always better than the first chosen order.
- After the swapping the order, if the newly generated order gives me lesser distance, bestcost is updated to the new distance, cityorder is also updated. If not, using `np.random.random_sample()`, a number between (0,1) is chosen. If the probability calculated using simulated annealing is greater than the randomly generated probability, the order is accepted.
- I created a function `min_dist`. It basically runs the tsp for several times and stores the distance obtained each time. Finally it gives the best minimum distance obtained in all the runs. One can increase the no of times the program runs, but also should take care of time constraint. Running the program more times can effect the performance. But it is also necessary, if dealing with higher no. of cities say above 10.
- In the end, `improv` calculates the percentage improvement in distance covered from the initial distance to the final distance obtained.

RUNNING CODE : The code should be run from the first cell. If random city coordinates are taken, N can be varied in the 2nd cell. If file is being tested, file name should be written in the 3rd cell i.e name has to be replaced. For

better results run it several times and take the final distance corresponding to the best improvement percentage

SOLUTION : This is the best improvement I got (for better results, I have run the code for several times):

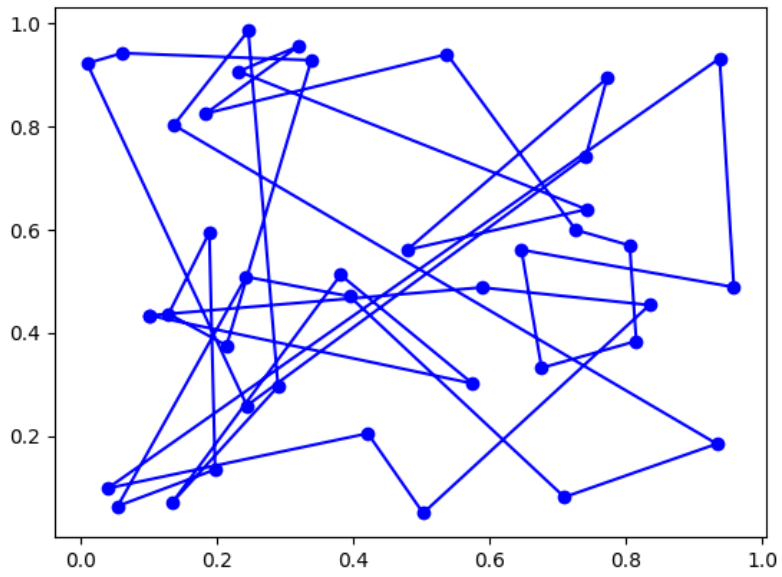

Figure 1: City Plot

Final order is : 23, 21, 16, 38, 11, 5, 28, 30, 20, 1, 25, 10, 34, 39, 15, 14, 4, 0, 9, 3, 35, 7, 36, 8 , 29, 12, 37, 32, 27, 13, 18, 26, 31, 33, 2, 22, 17, 19, 24, 6

- Initial distance covered by travelling salesman is : 20.526567224728712
- Final distance covered by travelling salesman is : 15.018126663089003
- Percentage improvemnet in the distance covered : 26.835663758738942