Spring 2022

AIX30006 Project 2 – Traveling Salesman Problem with Genetic Algorithm

Due date

Friday, May 6th, 2022 (No late submissions will be accepted)

Project description

Given that the Traveling Salesman Problem (TSP) arises in numerous applications such as vehicle routing, the purpose of this project is to exercise solving the TSP, one of the Non-deterministic Polynomial (NP) hard problems, using one of the most commonly used metaheuristic algorithms, namely Genetic algorithm. Simulation codes should be written in one of the following high-level programming languages: Python, C++, Java, or R. The main deliverables for this project are 1) a project report documenting your efforts on the project, 2) a README file that explains the overall structure of your project, and 3) simulation codes your team develops. All deliverables must be a joint effort; however, please note that one student from your group should submit materials on behalf of the entire team.

Group

Students will work in a team of two or three students with the aim of developing teamwork skills while working with other students. The entire team receives the same base grade. However, I will ask each of you to submit your own peer assessment in which you evaluate the contributions of your teammates. The purpose of this assessment is to find ways to work well together and contribute equally to the overall product.

Problem definition

You are currently working for the United Parcel Service (UPS) as an operations research analyst. Your job is to provide the shortest cycle route, which visits each delivery point exactly once and returns to the origin point, to the UPS truck drivers. Each driver is responsible for one city and you are asked to manage seven drivers for planning seven different cities (i.e., Atlanta, Boston, Cincinnati, Denver, New York, Philadelphia, and San Francisco). Please note that all edge costs (i.e., great circle distance) are symmetric and need to be calculated by the Haversine formula.

Data

Students will get real-world operational datasets that include delivery location information for each city in the United States. The datasets can be downloaded from the "Data" folder in LMS.

Deliverables

Please designate one member from your group to submit the following:

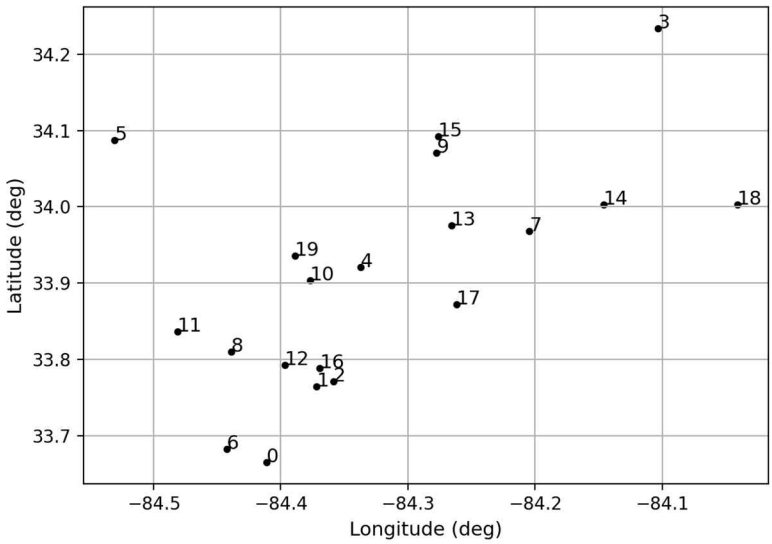
- ✧ PDF file of the report
- ✧ README file
- ✧ ZIP file of all the codes

In terms of the report, as the report will be a significant portion of your grade, I highly encourage you to put effort into producing a high-quality report. Your report should include the following items consisting of different categories that include 1) data pre-processing, 2) implementation, and 3) discussion.

To begin with, even though I provide you with seven different cities, it is important to note that you are only asked to choose three different cities for your project, meaning that you do not have to analyze all of the cities.

Data pre-processing

- ✧ Generate a plot that shows the delivery locations for all of the cities. For example, the following plot depicts the delivery locations in Atlanta. You don't have to show all of the cities but need to show at least one city in the report, which helps us see if you are able to generate the plot using your codes.



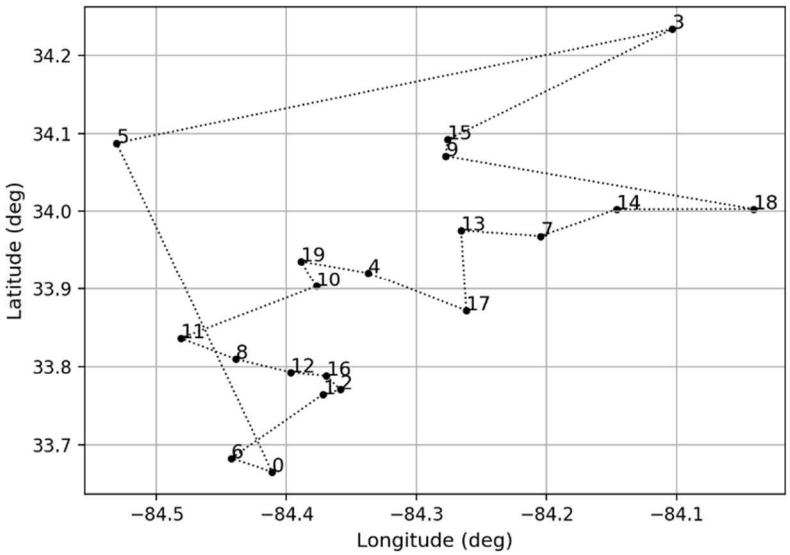
✧ Construct a distance adjacency matrix that represents great circle distances (you may need to consider using the Haversine formula to compute the distances) between cities i and j such that A_{ij} represents the distance between those two cities. The size of the matrix should be the number of cities by the number of cities (e.g., 20 by 20 matrix for 20 delivery locations). For example, the following array shows the distance adjacency matrix for the case of Cincinnati.

```
array([[    0.        ,  3281.34733522,  2733.13158647,  7494.97953992,
         3890.04737008,  4926.35973169,  5060.59287106,  4035.23474612,
         3806.77547248,  6535.14761531],
       [ 3281.34733522,     0.        ,  1967.39168999,  5047.1837725 ,
          610.6402438 ,  1880.57927625,  1989.67068157,  1534.67472862,
          559.04777493,  6742.00063607],
       [ 2733.13158647,  1967.39168999,     0.        ,  7012.26015584,
         2323.58955657,  3788.80897518,  2802.55130543,  3470.84365815,
         2447.78007218,  4880.23073389],
       [ 7494.97953992,  5047.1837725 ,  7012.26015584,     0.        ,
         4751.66934286,  3292.75244777,  5312.3445085 ,  3627.16389909,
         4579.41742644, 11627.27062934],
       [ 3890.04737008,   610.6402438 ,  2323.58955657,  4751.66934286,
            0.        ,  1476.30801265,  1482.39547347,  1515.41360832,
          273.2633999 ,  6903.10329883],
       [ 4926.35973169,  1880.57927625,  3788.80897518,  3292.75244777,
         1476.30801265,     0.        ,  2213.18415066,  1093.60233398,
         1346.89367172,  8336.03672175],
       [ 5060.59287106,  1989.67068157,  2802.55130543,  5312.3445085 ,
         1482.39547347,  2213.18415066,     0.        ,  2814.3121229 ,
         1731.76988989,  6490.63057689],
       [ 4035.23474612,  1534.67472862,  3470.84365815,  3627.16389909,
         1515.41360832,  1093.60233398,  2814.3121229 ,     0.        ,
         1252.14314109,  8276.62663965],
       [ 3806.77547248,   559.04777493,  2447.78007218,  4579.41742644,
          273.2633999 ,  1346.89367172,  1731.76988989,  1252.14314109,
            0.        ,  7109.6985993 ],
       [ 6535.14761531,  6742.00063607,  4880.23073389, 11627.27062934,
         6903.10329883,  8336.03672175,  6490.63057689,  8276.62663965,
         7109.6985993 ,     0.        ]])
```

Implementation

✧ Find a solution using the Greedy algorithm. Here, “Greedy” means that it is supposed to be designed to find a solution based on “nearest neighbor selection” from a starting point. You need to provide the shortest cycle route generated by the Greedy algorithm and a plot visualizing the optimized route. As an example, please see below.

```
The shortest cycle route by the Greedy algorithm:
[0, 6, 1, 2, 16, 12, 8, 11, 10, 19, 4, 17, 13, 7, 14, 18, 9, 15, 3, 5]
```

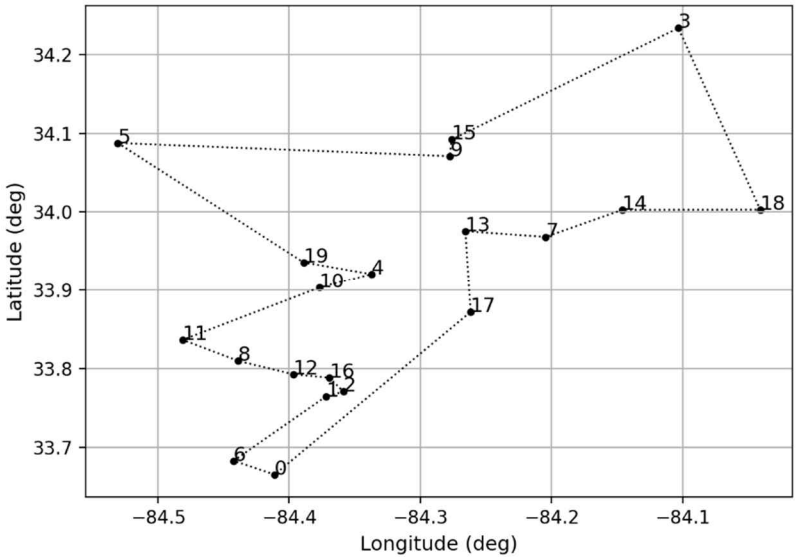


✧ Find a solution using the Genetic algorithm. You need to provide the shortest cycle route generated by the Genetic algorithm, the optimal value, and a plot visualizing the optimized route. As an example, please see below.

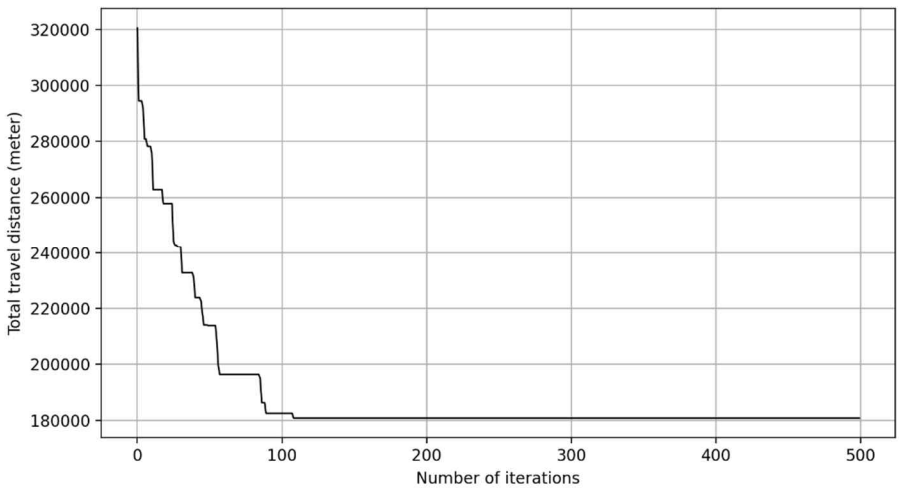
```
1 # Print the optimal route generated by the Genetic algorithm
2 optimal_route
```

```
[0, 6, 1, 2, 16, 12, 8, 11, 10, 4, 19, 5, 9, 15, 3, 18, 14, 7, 13, 17]
```

```
Generation 0: 320538.55711892486
Generation 100: 182498.07708266584
Generation 200: 180793.78346048718
Generation 300: 180793.78346048718
Generation 400: 180793.78346048718
```



- ✧ Explain the reason why the solution generated from the Greedy algorithm looks different from the solution generated by the Genetic algorithm.
- ✧ Generate a plot of convergence history showing the optimum value generated by the Genetic algorithm is converged as the number of iterations increases. As an example, please see below.



- ✧ Provide pseudo-codes for each step of the Genetic algorithm (i.e., initialization, fitness evaluation, selection, crossover, and mutation).

Discussion

- ✧ Discuss with your teammates and answer the following questions:

- Let's say that you execute the Genetic algorithm to get a solution to the given TSP problem. Do you think that it is a global optimum or not? Tell us the rationale behind your thought.
- Let's say that you execute the Genetic algorithm multiple times for the same problem. Do you think that you will always get the same solution (e.g., total travel distance is always identical)? If not, can you explain the reason why?
- Let's say that you would never be able to get a solution (i.e., it is always stuck in a local optimum) with the current set-up of the Genetic algorithm you implemented. How would you be able to address this potential issue? [Hint: you may need to come up with what user-defined parameters should be changed from the current set-up]
- Let's say that you would like to specify a stopping criterion in which the Genetic algorithm is terminated. How would you define the criterion?

Citation

Please make sure that all codes originating from other sources must be clearly documented.