Assignment6

March 6, 2024

1 Assignment

The traveling salesman problem gives you a set of city locations (x, y coordinates). Your goal is to find a route from a given starting point that visits all the cities exactly once and then returns to the origin, with the minimum total distance covered (distance is measured as Euclidean distance $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$).

You will be given a file where the first line is the number of cities N, and the next N lines give the cities as a list of x, y coordinates: for example

```
4
0.0 1.5
2.3 6.1
4.2 1.3
2.1 4.5
```

1.1 Code

Your goal is to give a sequence of numbers, for example `[0 3 2 1]` which specifies the order in which to visit the cities. Note that after the last city you will come back to the first one in the list.

You need to write a single function with the following *signature*:

```
def tsp(cities):
    return cityorder
```

where

- `cities` is a list of `(float, float)` tuples, each one representing the x- and y-coordinate of the city respectively.
- `cityorder` should be a list containing the numbers `0` to `N-1` where `N` is the number of cities, in some order. This order should ideally give the least possible distance, but since this is an NP-hard problem, your goal is actually to try and minimize the distance rather than actually finding the exact optimum.

You should also write a function

```
def distance(cities, cityorder):
    return totaldistance
```

that will compute the actual total distance required for this sequence of city visits.

1.2 Document

The document you submit should explain the approach you have used - if you decide not to use simulated annealing for some reason, that is acceptable provided you explain what you have done instead and why it is superior.

Plot the cities with the path you have specified, and output the total length of the shortest path discovered so far.

For any given input file, you should also calculate how much is the percentage improvement in the path that you see starting from a random initial point to the best possible solution you find.

1.3 Report

You need to submit a .zip file that contains all your code (either notebook or plain Python files) and a PDF file (can be generated from the notebook) that clearly documents what you have implemented and how to run your code.

[ ]:
```python
# Example Python input and plot
x_cities = np.array([0.0, 2.3, 4.2, 2.1])
y_cities = np.array([1.5, 6.1, 1.3, 4.5])
finalorder = [0, 1, 2, 3]

# Rearrange for plotting
xplot = x_cities[finalorder]
yplot = y_cities[finalorder]
xplot = np.append(xplot, xplot[0])
yplot = np.append(yplot, yplot[0])
plt.plot(xplot, yplot, 'o-')
plt.show()
```

1.4 Work expectations

- `distance` function: ~10-20m
- Understanding possible approaches to optimization - simulated annealing or otherwise - and deciding on approach: ~30-60m
- Implementing code for `tsp` optimization: ~60m
- Documentation: ~30m

Total expected time is less than 3 hours. If you find yourself spending more time than this, please contact instructor and discuss.