UNIVERSITY OF Southampton
Faculty of Social Sciences
School of Mathematical Sciences

Course: MATH6005-Introduction to Python and MATH6181-Python & Forecasting
Semester: 2021:2022
Instructor:
Vuong Phan (t.v.phan@soton.ac.uk)

March 22, 2022

Final Assignment: Part 2 (week 8)

Rules for the Final Assignment: Part 2 (week 8) solution

- You will upload, on Blackboard, one single file.
- The file name, with all functions, must be in the format: `<university username>.py`, e.g.: `fdoo1r22.py`. This file is the **Final Assignment: Part 2 (week 8)** solution that will be uploaded on Blackboard.
- All functions must follow the prototypes, i.e., they must have the name specified, have the input(s) set, and output the variables in the order specified in the prototypes.
- No partial credit will be given if the function is not working correctly, i.e., due to syntax error, neglect to follow the prototype, etc..
- Only the functions must be in the upload file; any auxiliary or debugging code must be removed of the upload file.
- Please include comments in your code to explain how it works. Fail to comment on your code will result in **10 points reduction** on your final assignment mark. Comment every line of your code.
- You can use any function/package/library, including NumPy, Pandas, etc.
- The solution must have only functions. Do not define any function within another function.
- A template solution file is available on Blackboard.

UNIVERSITY OF Southampton
Faculty of Social Sciences
School of Mathematical Sciences

The Ship Rendezvous Problem

1 Introduction

Your company has been contracted to provide a tool in Python to solve the Ship Rendezvous problem (SRP) to enable a cruise company to minimise the time it takes for the support ship to visit each of the cruise ships in the fleet.

The support ship must visit each of the n cruise ships exactly once and then return to the port (where it started). The support ship travels at a constant speed and changes direction with negligible time. Each cruise ship moves at a constant velocity (i.e. speed and direction). The objective is to minimise the **total time taken to visit all the cruise ships** (i.e. the time taken to reach the final ship).

This problem is considered in 2-dimensional (2D) Euclidean space. A problem instance is defined by specifying the starting (x, y) coordinates for all ships (including the support ship), the velocity of the support ship, and the velocity of the cruise ships.

Note that it is certain that the SRP has a finite solution if the support ship is faster than all other ships in the fleet. However, it is very likely (but not certain) that the SRP will not have a complete solution (one where all the cruise ships are visited) if one or more of the cruise ships are faster than the support ship.

2 Your Python Task

You must implement the greedy heuristic for the 2D Euclidean SRP in Python. Your program must perform the following tasks:

- Read the data from a CSV file (a sample data file is provided on Blackboard *sample_srp_data.csv*);
- Run the greedy heuristic against the problem instance to obtain a solution;
- Output a list of indexes of unvisited cruise ships, a list of indexes of the visited cruise ships, and the total time to visit all visitable cruise ships.

Greedy Heuristic for the SRP

A simple way of finding a solution to the SRP is to use a greedy heuristic. The greedy heuristic works as follows

1. For each unvisited cruise ship, calculate how long it would take the support ship to intercept it from the support ship's current position.
2. Choose the cruise ship, i, which can be reached in the shortest amount of time.
3. Visit cruise ship i and update the positions of the ships in the fleet.
4. Return to 1 if there are any unvisited cruise ships that can be reached by the support ship.

In order to make the heuristic deterministic (i.e. to guarantee the same result each time it is run on the same problem instance) you must specify how ties in Step 2 are broken. If there is a tie, the algorithm should choose to visit the ship with the smallest index (for example, if ships 5 and 7 are tied, ship 5 should be chosen in preference to ship 7).

The **Technical Appendix** (at the end of this document) provides details on how to calculate intercept times.

Function prototypes

```
import pandas as pd
def read_srp_input_data(csv_file):
    '''
    function description
    '''
    input_data=pd.read_csv(csv_file)

    return input_data
```

- The function must use the input variable `csv_file` to read the TXT file, inside of the function. You cannot use a fixed path file. If the function does not read the file using the input variable `csv_file` **no partial credit** will be given.

```
def findPath(input_data):
    '''
    function description
    '''
    unvisited_index = [] # It must be a LIST. It cannot be a tuple nor NumPy array.
    visited_path = [] # It must be a LIST. It cannot be a tuple nor NumPy array.
    total_time = 0  # it must be a float.

    return unvisited_index, visited_path, total_time
```

- The function outputs must be in the order as specified on the prototype. If the function outputs are not in the set order **no partial credit** will be given.

Example: There are 5 cruise ships on the fleet. Suppose that, using the Greedy Heuristic, the support ship can only visit 3 of them with the indexes $1, 2, 4$ (e.g. the 2 remained cruise ships with the indexes $3, 5$ cannot be visited). Remember that support ship should have index "0" and the first cruise ship has index "1" and so on. Then an example for a solution looks like:

```
[3,5], [2,4,1], 45.3
```

which mean that unvisited_index = [3,5], visited_path = [2,4,1] and total_time = 45.3. The visited_path = [2,4,1] means that the support ship will visit the cruise ship with index 2 first, and then the cruise ship with index 4, and finally the cruise ship with index 1.

Please, the debugging/testing code must not be in your solution file that will be uploaded on Blackboard.

Advice on writing the code

Make sure you follow the guidelines below when writing your code:

- You can (and are encouraged to) create new functions if needed. These must follow the good coding practices we have taught in the lectures and labs. However, your submission must include all the functions provided in the template, with the exact same names provided in the template.
- Your code should implement exactly the algorithm described above. Do not use an alternative. If you use a different algorithm (even if the algorithm solves the problem correctly and the results seem better) your assignment will be marked as incorrect.
- We will test your code against an unseen set of problem instances. We recommend that you test your algorithm and make sure that your code returns the expected solutions for different scenarios. To help you do this, you may create and test new CSV files for which you think you know the expected behaviour.
- We will only use correct input CSV files to test your code. The assignment does not ask you to include logic to handle non-numeric or incorrect CSV files. There are no extra marks available for including this functionality.

Technical Appendix

This appendix contains some technical information to help you solve the proposed problem in the assignment. Please, pay attention to these formula and specifications to ensure that your algorithm produces the correct results.

Notation

We denote the starting coordinates of the support ship by $(X_0; Y_0)$ and its speed by S. We consider the ship i in the task force, where $1 \leq i \leq n$. We denote its starting coordinates by $(x_{i0}; y_{i0})$. Its velocity is split into x-component and y-component, v_{ix} and v_{iy} , respectively. Therefore the position of ship i at time $t > 0$ is given by

$$(x_{it};\, y_{it}) = (x_{i0} + tv_{ix};\, y_{i0} + tv_{iy})$$

Note that the speed of ship i (denoted by s_i) can be obtained from its velocity with the following equation:

$$s_i = \sqrt{v_{ix}^2 + v_{iy}^2}.$$

Calculating intercept times

To calculate intercept times, it is simplest to update the position of the ships at every step, so that $(X_0; Y_0)$ represents the current coordinates of the support ship and $(x_{i0}; y_{i0})$ represents the current coordinates of ship i for $1 \leq i \leq n$. Then the time, T, taken for the support ship to move from its current position and intercept ship i is found by finding the smallest positive root of the quadratic equation:

$$aT^2 + bT + c = 0$$

where the coefficients are obtained as follows:

$$\begin{aligned} a &= v_{ix}^2 + v_{iy}^2 - S^2 \\ b &= 2\left((x_{i0} - X_0)v_{ix} + (y_{i0} - Y_0)v_{iy}\right) \\ c &= (x_{i0} - X_0)^2 + (y_{i0} - Y_0)^2. \end{aligned}$$

Input Data

The input data for each problem instance will be presented in a comma separated value (CSV) file. A correctly-formatted input file consists of N rows of data, where $N \geq 2$. Row 1 corresponds to the support ship, and the remaining rows (i.e. rows $2; ...; N$) correspond to the ships to be intercepted. Each row contains five pieces of information in the following order:

- Ship name/description (text).
- Start x-coordinate (decimal number).
- Start y-coordinate (decimal number).
- Velocity x-component (decimal number).
- Velocity y-component (decimal number).