MATH6005 Final Assignment

Ship Rendezvous Problem

1. The Problem

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

1.1. Introduction

The support ship must visit each of the *n* cruise ships exactly once. 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 speed 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 task force. However, it is very likely (but not certain) that the SRP has no solution if one or more of the cruise ships is faster than the support ship.

1.2. Your Python Task

You must implement the greedy heuristic for the 2D Euclidean SRP in a Python module. Additional information is provided in the **Technical Document** (available via Blackboard).

Note that your program must have the following components:

- Ask the user for the CSV file name and read the data – a sample data file is provided on Blackboard;
- Validate the data (recognise if there any problems with the data);
- Run the greedy heuristics; and
- Output the results to a CSV file.

Greedy Heuristic for the SRP

A simple way of finding a solution to the SRP is to use a greedy heuristic if the support ship is faster than all the cruise ships. This guarantees a solution, but it is very unlikely to be optimal.

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 task force.
4. Return to 1 if there are unvisited cruise ships.

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. The most commonly used tie-breaker is to choose the ship with the smallest index (for example, if ships 5 and 7 can be reached in the same amount of time, ship 5 should be chosen in preference to ship 7).

The **Technical Document** (available via Blackboard) provides details on how to calculate intercept times.

2. Marking Scheme

80% of the marks available for this assignment are for the Python project, the remaining 20% of the marks are available for the accompanying document.

- Marks for the Python project (80%) will be awarded on the basis of:
  - Functionality when it runs (correct results and output);
  - User-friendliness when running (robustness, data validation, error-handling);
  - Design and maintainability
  - Originality (in terms of design and/or additional functionality such as including an additional option to run a better heuristic).
- Marks for the documentation (20%) will be awarded on the basis of:
  - Accuracy;
  - Completeness; and
  - Presentation.