Lancaster University

MSCI530: Data Sourcing, Handling and Programming

Individual Coursework (60% of module)

Deadline: 20/1/2022 10AM *Michaelmas Term* Maximum Marks: 100

# 1 Coursework Description

In this coursework, you are to write a program that attempts to solve a machine delivery problem using stochastic optimisation techniques. Specifically your program will read in a problem instance from a file and will attempt to produce a solution using stochastic optimisation algorithms. Note that the optimal solutions to such intractable problems, are not currently known and so algorithms seen to quickly achieve low cost solutions are viewed more favourably.

You have been given the flexibility to structure the code in the way that you think is best following the principles of *re-usability*, *maintainability*, *information hiding* and *clarity*.

You are required to submit to Moodle a single .py file, programmed in Spyder. The name of this file should be your library number (e.g. 12345678.py, where 12345678 is an example library number).

You should also submit a document file named 12345678.docx, where 12345678 is an example library number, containing: (i) a short summary of your work listing the tasks that you have completed; and (ii) a copy of your Python code.

Please do not submit compressed files such as zip and rar files.

In case you want to import any packages apart from pandas, matplotlib, numpy, math, random, tkinter, pickle, seaborn, copy and time, you should contact the module leader first.

Please note that you will be penalised (possibly heavily) if you do not follow the instructions above.

This coursework may easily be divided up into tasks (see below) to facilitate development. If you can only do some of these tasks, then submit whatever it is that you can manage to do. You will be penalised for functionality, but you may well get quite a lot of marks in other categories.

## Problem description

There are machine requests from $n$ customers that all have to be satisfied. There is one depot location where all the machines are located. Trucks are hired to transport the machines from the depot to the customers. The number of available trucks is $\sqrt{n}$, where $n$ is the number of customers. For convenience, we always assume that $\sqrt{n}$ gives a whole number. All trucks must be allocated exactly $n/\sqrt{n}$ deliveries (locations). The route of a truck must start and end at the depot. We assume that it does not take any time to load a machine at the depot or to unload a machine at a customer. For each route, the first customer to visit must have an even ID number and the last customer to visit must have an odd ID number.

The objective is to minimise the total distance travelled by the trucks plus $\Delta$. Here $\Delta$ is the difference between the longest distance travelled by the trucks and the shortest distance travelled by the trucks (in order to be fair among the truck drivers).

In order to determine the travelled distances, integer coordinates are provided for the customer locations. The location of the depot is always $(0, 0)$. The distance between coordinates $(x_1, y_1)$ and $(x_2, y_2)$ is defined as $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, i.e., the Euclidean distance.

The problem instances, which are available in .csv format, contain the customer IDs and the $x$ and $y$ coordinates of the customer locations (all in integers). An example of a problem instance file I1.csv is shown below (note that on Moodle you will find three problem instances I1.csv, I2.csv and I3.csv).

| $ID$ | $x$ | $y$ |
|---|---|---|
| 0 | 2 | 2 |
| 1 | -2 | 1 |
| 2 | 0 | -1 |
| 3 | 3 | 1 |

In this example, we have $n = 4$ customers. The number of available trucks is $\sqrt{n} = 2$. All trucks must be allocated $n/\sqrt{n} = 2$ deliveries (locations). The locations of the first, second, third and fourth customers are (2, 2), (-2, 1), (0, -1) and (3, 1), respectively.

Using the problem instance given above, a solution which specifies the routes for the trucks can be stored as follows: [ [2, 1], [3, 0] ]. Here the route for the first truck is given as [2, 1], and the route for the second truck is given as [3, 0].

Note that all trucks depart from and must return to the depot, so this is not mentioned explicitly in this solution format. The distance travelled by the first truck is 6.064. This can be computed by calculating the straight line distance between the coordinates of the locations (i.e. distance between depot (0, 0) and customer 2 (0, -1) + distance between customer 2 (0, -1) and customer 1 (-2, 1) + distance between customer 1 (-2, 1) and depot (0, 0)). The distance travelled by the second truck in this example is 7.405. Therefore, the total distance travelled by all trucks is $6.064 + 7.405 = 13.469$ and $\Delta = 7.405 - 6.064 = 1.341$.

**Task 1:** (2 marks) Your first task is to write a Python program that asks the user for the input file name and then reads the coordinates of the customer locations from this file.

**Task 2:** (10 marks) Generate an initial solution to this problem at random. In order to obtain a good mark for this task, you will need to add some 'intelligence' to the procedure to produce better quality initial solution. Anything that improves upon the basic initialisation method is acceptable (providing it is a genuine improvement to the cost).

**Task 3:** (13 marks) Implement a **Cost** method that calculates the cost of the solution in the way described above (i.e. total distance + $\Delta$). Then implement a **Feasibility** method that counts the number of routes that do not satisfy the first constraint (i.e. the first customer to visit must have an even ID number) and the number of routes that do not satisfy the second constraint (i.e. the last customer to visit must have an odd ID number). Considering the example given above,

the first customer visited by the second truck has an odd ID and the last has an even ID, hence Feasibility method should return 2.

**Task 4:** (10 marks) Implement a method that improves the initially generated solution using hill climbing method which accepts only non-worsening moves. To do this, you will need to implement an appropriate operator, e.g. swap operator. Your program should be run for a **fixed number of iterations**. **Ask the user to input the number of iterations**. At the end of the run, your best obtained solution should be saved in a .csv file. The file specifies the routes for the trucks, and the last two rows display the Cost and the Feasibility of the solution. This format is extremely important, as I will be using my own checker software tool to automatically examine your solution and your calculated Cost and Feasibility. The example solution described above would be displayed if opened in MS Excel:

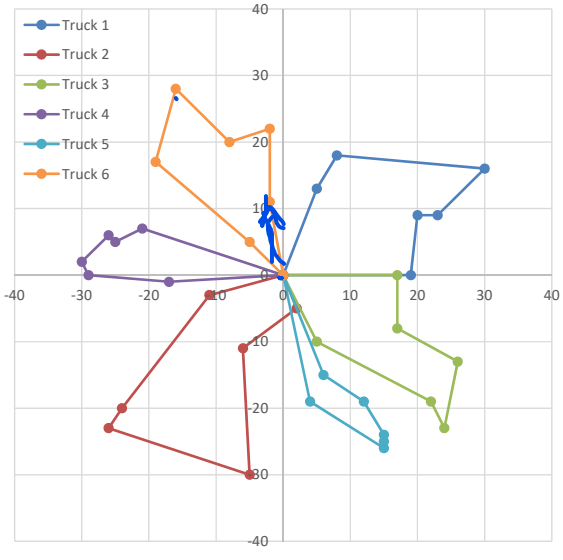
| | |
|---|---|
| 2 | 1 |
| 3 | 0 |
| 14.81 | |
| 2 | |

**Task 5:** (30 marks) Try to make your solver more efficient and powerful, by:

- Implementing simulated annealing method (you will need to research this yourselves).
- Employing more than one operator (e.g. swap and insert operators).
- When performing a move, you may notice that only few trucks are altered. Thus, you may be able to speed up your program by only re-evaluating the effected trucks.

Ultimately the goal is to minimise the Cost with no constraint violations (i.e. Feasibility = 0).

**Task 6:** (5 marks) Update your code to provide a visual representation of the best solution. Below a visual representation to a solution produced by solving the problem instance I2.csv.



**Task 7:** (5 marks) Output to the user the following information: an iterations Vs Cost line-graph that demonstrates how the Cost reduces during a run. Note that updating the graph in each iteration may make the algorithm slow, so you should update the graph every 100 iterations or so.

**Task 8:** (5 marks) Maintain a log file (using pickle package) containing the results of previous runs. In particular, for each run we should store in a legible way the problem name (e.g. "I1"), the Cost of the initial solution, the Feasibility of the initial solution, the final solution obtained (i.e. the routes), the Cost of the final solution and the Feasibility of the final solution. Other information can also be stored here if deemed appropriate.

**Task 9:** (20 marks) These marks will go to programs that are well structured (using both object oriented programming and functional programming concepts), intuitive to use (i.e. straightforward for me to run your code), generalisable (i.e. they are able to solve hidden instances such as I4.csv), flexible (i.e. provide user options via a graphical user interface) and educational (i.e. create an interactive demonstration and animation for the initial solution, improved solution and implemented operators). It is important to break your code into small, meaningful methods/functions and appropriately comment your code.

## 2 Coursework Submission

The coursework deadline is 10:00AM, Thursday 20th January, 2022. In accordance with University regulations, work submitted up to three days after the deadline will be classed as late submission (10 marks will be deducted), and after three days as a non-submission. However, if an extension is given then the rule applies from the date of the extension.

## 3 Plagiarism

According to university rules, instances of plagiarism will be treated very seriously. **Penalties** are in line with the institutional framework of the University. We will analyse the coursework using plagiarism detection software. In the days following submission, and at a prearranged time, you will be asked to demonstrate your program to me. I will use this opportunity to ask you to provide information on your program and explain parts of your code.