The Travelling Salesperson

1810ICT — Software Development Environments

School of ICT

Griffith University

Trimester 2, 2018

Part C due Sunday 6th October, Midnight

Assignment Description

The assignment for this course is centred on the travelling salesperson problem. The travelling salesperson problem (TSP) is as follows; "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city". The problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. Even though the problem is computationally difficult, a large number of heuristics and algorithms are known, so that some instances with tens of thousands of cities can be solved completely and even problems with millions of cities can be approximated within a small fraction of 1%.

You must prepare a project plan that includes an overview of the system, work-breakdown structure, activity definition and estimation and a Gantt chart for presenting scheduling & time estimation. This project plan should include broad estimates for Parts B & C of the assignment, and more specific estimates for your work on Part A. As you complete Part A, you should put the actual completion time/dates on your Gantt chart to track how close you were to your estimates. For each subsequent stage of the assignment, you should revise your project plan and continue to track your progress on your work activities.

You will then implement a TSP solver to solve some TSP problems. For the scope of this assignment, we will only use Euclidean-distance problems. A description of your solver plus the results from your tests should be presented in a brief report. All implementation must be done in Python and designed to run on a remote server, though your testing and development should be done locally to minimise load on the server.

In Part B, you will design and implement a database to store TSP problems and solutions. Your program should be modified to read files from the database, and save the solutions in the database.

In Part C, you will design and implement a graphical interface to load and display TSP problems and solutions from a centralised database.

Submission Requirements

Part C of your assignment must be submitted online via L@G under the assessment page. You are required to submit:

- A wireframe diagram of your UI layout (PNG recommended)
- A zip file containing your updated source code (.py files) that provide a GUI interface to your program as well as connect to a centralised database.
- A word document containing your updated project plan.

Part C – Graphical Interface (20% overall)

For Part C, you must design a graphical interface for your program. You will need to firstly prepare a wireframe that shows how the interface will look and operate, and then implement this interface in Python. The program you write will now connect to a centralised database with a similar structure to the one you designed in Part B. You will need to modify your code to work with the new database schema. Your program should provide graphical options to:

- Upload a new .tsp problem to the database
- Load a problem from the database (and visualise it)
- Solve a loaded problem (and visualise the solution)
- Save a solution to the database
- Load a solution from the database (and visualise it)

Project Plan (20%)

You should begin by revising your project plan to continue to track progress through your project. Specifically, a detailed WBS and activity definition for Part C is now required, and the Gantt chart will need to be updated to reflect the new information. Continue to track your actual time taken in your Gantt chart.

Wireframe (20%)

Your first task is to come up with a suitable design for your graphical interface to your TSP program. This is something that should be given some careful thought, as modifying interface components/layout after implementation can be difficult. A digital wireframe is required – use a tool like wireframe.cc to generate a professional looking wireframe that you can use to guide your UI implementation. Spend a reasonable amount of time considering the requirements of your interface so that your wireframe is accurate.

Implementation of GUI (60%)

Finally, you should implement a GUI to your TSP solver and make it connect to a central database. Modify your internal code to connect to the ICT MySQL server. To connect to this from python this you will need to install and import the mysql.connector python library available in the anaconda navigator or by typing

```
conda install mysql-connector-python
```

in an Anaconda prompt.

The database details will have been emailed to your student email address. In short:

```
user='s1234567',
password='XXXXXXX',
database='1810ICTdb',
host='mysql.ict.griffith.edu.au'
```

You username is your Griffith sNumber, and your password was emailed to you with the SQL Server details (***NOT YOUR GRIFFITH PASSWORD***). For this assignment, you may simply put your database password into your python source code. The database structure is provided in the provided schema.png file, and the SQL used to create this database is also provided in tsp.sql.

VERY IMPORTANT:

When working on 1810ICTdb please be aware that this is a shared database, and if you break it, it will break it for everyone. ***While you are developing your code, it may be wise to use your personal database (s1234567db) located on the same server.*** You may create the required tables in your personal database using the provided sql file and the phpMyAdmin module located at http://seet-dwarf.ict.griffith.edu.au/phpMyAdmin. Once you have finished development, you can just change the 'database' parameter of your sql connector to point to the 1810ICTdb shared database and everything should work the same.

You should now design a GUI that allows you to perform all the actions you used to be able to perform through the command line and a few extras. Specifically:

- Upload a new .tsp problem to the database
- Load a problem from the database (and visualise it)
- Solve a loaded problem (and visualise the solution)
- Save a solution to the database
- Load a solution from the database (and visualise it)

Visualising the loaded problems is probably best done using matplotlib. For full marks, you should incorporate it directly into your GUI. Because your GUI allows finer control over the actions a user can take, you will be able to enhance the functionality of your program. For example:

- The SOLVE command from part 2 can now be split into 3 parts - loading a problem, solving a problem and saving the solution. These should all have their own interface options so that a user could (for example) solve a problem but not save the solution.
- The FETCH command can now offer users a choice of which solution to fetch (instead of just fetching the one with the shortest tour).
- If a solution has been FETCHED, it could still be considered the 'current loaded problem' and your interface could allow the user to then click SOLVE to run your own solver on it.
- If you have implemented / would like to implement multiple solvers, you may provide an interface option to select which algorithm to use
- Any other solver parameters can now be set using the GUI.

Matplotlib's visualisation should automatically scale to display the problem, but you could also consider implementing NavigationToolbar2Wx to allow the user to pan/zoom.

Other considerations:

- Now that you are using a shared database, what happens if 2 people try to add the same problem (PRIMARY KEY???). You should check if a problem already exists before adding it
- How do you correctly format your data for database input/output? Instead of executing your generated SQL statements, print them to verify they look as expected (or test on your private database first)

Marking Guide

Part C – GUI (worth 20% of total course mark)

There are 3 components to this:

1. The updated Project Plan and Gantt Chart (20%)
2. Wireframe (20%)
3. Implementation (60%)

1) Updated Project Plan and Gantt Chart (20 marks)

The Project plan should be updated to plan for activities in Part C:

WBS (7 marks)

Should be a breakdown of all the different activities involved in completing the project. This should now be completed at a high level of detail for Part C.

Activity Definition (7 marks)

For each item in the WBS, the item should be further explained and include a time estimate that is reasonable.

Gantt chart (6 marks)

All of the items in the Activity definition should be listed in the Gantt chart with the relevant estimates and scheduling. The students should have also tracked the actual start time and time taken.

2) Wireframe (20 marks)

You should have submitted a reasonable design detailing the layout of the interface to the TSP solver. The design should show the location of all major interface elements, and look professional. There is not a requirement that the elements are itemised in the diagram, however the submitted wireframe should clearly show the obvious interface elements.

3) Implementation (60 marks)

The code should be modified to connect to the centralised database and the queries have been updated to use the schema I provided (10 marks)

Upon running the program, a GUI successfully displays and loads (10 marks – again, showing an empty window might be worth 2/10, showing a full GUI with good layout and scalability 10/10).

Each aspect of GUI functionality – uploading a new problem from a file, loading a problem from the database (and visualising it), solving a loaded problem (and visualising it) and then saving and loading solutions from the DB will be assessed. 30 marks total.

The final 10 marks are for “Quality” – does the GUI work well, is it well designed and functional, is it intuitive etc.