Delivery Vehicle Routing System

- **Due:** *11:59 pm 28/5/2025 (Wednesday of Week 12)*
- **Contributes** 50% of your final result
- **Group Assignment – Group of 4-5 students**

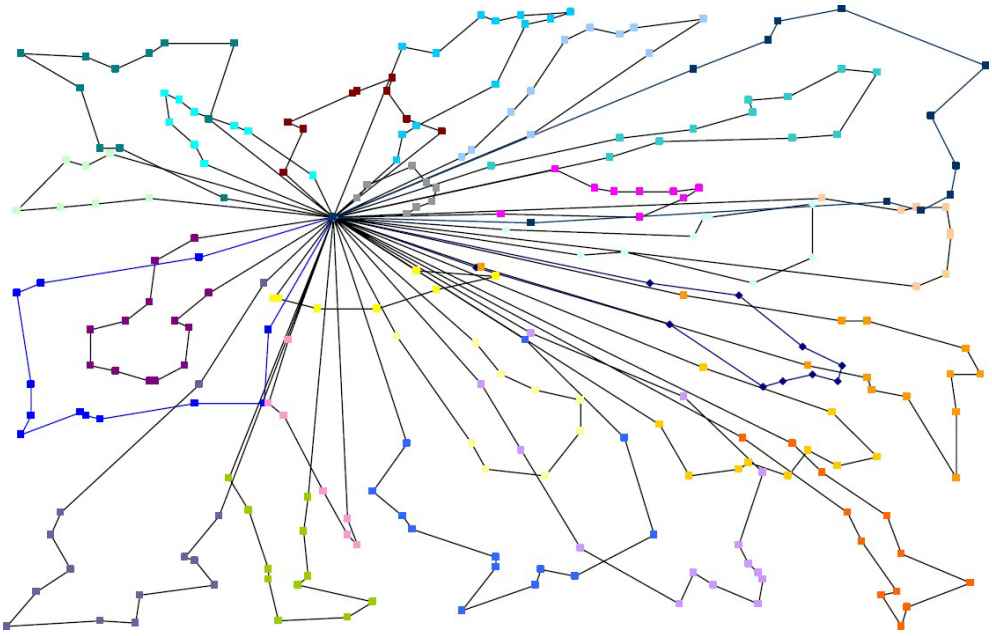
Summary

You need to implement and demonstrate a system for the Vehicle Routing Problem. The description of this problem can be found at: https://developers.google.com/optimization/routing/vrp
This is a well-known problem in both academia and industry (see also the following page: https://www.altexsoft.com/blog/business/how-to-solve-vehicle-routing-problems-route-optimization-software-and-their-apis/). We develop a solution for a couriers company with a number of delivery vehicles. There are many versions of this problem but we will start with the basic one and suggest a number of extensions to enable the student to do extra research to achieve higher marks.

The basic system will assume that all parcels to be delivered are located at the company warehouse and all delivery vehicles also depart from the company warehouse and return back to the warehouse after delivering all their parcels. It will involve the following roles:

- **Master Routing Agent (MRA)**
  - Collects capacity constraints from other agents (i.e. from delivery agents)
  - Receives as input the list of parcels to be delivered to the customers' locations
  - Produces the routes for all the delivery vehicles
  - Sends the individual routes to delivery agents
- **Delivery Agent (DA)** (***any number***)
  - Send its own capacity constraint to the MRA
  - Receives the individual schedule

A solution assigns a route to each DA so that all parcels will be delivered and the DAs finish their routes at the warehouse. The MRA aims to find the optimal solution.



Constraints:

- The MRA can use any search/optimisation technique from the unit (e.g. GA, ACO, PSO, CSP etc.) or combinations of them. Note that you may use the Google OR-Tools library as a baseline solution to test your own solution but you must implement a search/optimisation technique of your own for this project. Another powerful open-source tool that you can reference to is OptaPlanner: https://www.optaplanner.org/learn/useCases/vehicleRoutingProblem.html. Please check out their videos for some cool applications of constraint optimisation solver. Again, you may use the OptaPlanner library as a baseline solution to test your own solution but you must implement a search/optimisation technique of your own.
- The DAs and the MRA must use an appropriately defined interaction protocols for communication/coordination. Please make sure that you'll provide a sequence diagram for the implemented service calls in your report.
- The agents can use any standard content language
- A GUI will be available for the user input, parameter settings and visualisation (and a configuration file for the defaults)
- You can assume that the cost/time to go from point A to point B is calculated according to the straight line distance between A and B.

System requirements

- There are options to allow both the *automatic creation of the input list of delivery items (i.e., their locations) by taking the number of items* and *loading the input list of delivery items from a textfile*.
- **Basic version:** The capacity is specified by a number (i.e., the number of items the vehicle can carry). You can also assume that the total capacity of all vehicles is not less than the number of items to be delivered.
- **Alternative.1:** Another way of specifying the vehicle capacity. For instance, using weight: the total weight of all items to be carried by the vehicle does not exceed the vehicle's capacity.
- **Alternative.2:** System can deal with mismatches in capacities. For instance, when the total capacity of all vehicles is less than the number of items to be delivered or when the items to be delivered can be carried by some (and not all) of the delivery vehicles.
- **Extension.1:** Vehicle routing problems with time windows (VRPTWs) – see: https://developers.google.com/optimization/routing/vrptw. You'll need to do your own research to figure out how to capture the input and other necessary parameters.
- **Extension.2:** Dynamic addition of parcels and delivery agents in real time. Discuss with your tutor for specific requirements.

Project requirements

- Source code maintained on Git based VCS (Github/Bitbucket/GitLab/…). You must provide read-only access to the tutor/lecturer
- Running illustrative demo of a working prototype (please refer to **Marking Scheme** for details on functionality that needs to be implemented)
- Project report (8-10 pages) that includes the following sections
    - Cover Page (with team details) and a Table of Contents (TOC)
    - Introduction
    - Overall system architecture
    - Implemented interaction protocols,
    - Implemented search/optimization techniques,
    - Scenarios/examples to demonstrate how the system works,
    - Some critical analysis of the implementation, and

- Summary/Conclusion.
- Presentation + demo video link (10 minutes duration)

Marking Scheme

Requirements	***Mark***
Task 1: Basic interaction between DAs and MRA working fully according to a well-defined interaction protocol (with a clear sequence diagram). DAs can (a) submit their capacities according to the capacity specification, (b) MRA can receive requests from DAs and take an input list of delivery items to generate the route assignment for each DA, (c) MRA notifies DAs of their respective routes.	10
Task 2: MRA uses any optimization technique to generate the optimal route assignment.	20
Task 3: Complete **Alternative.1** OR complete **Alternative.2**; note that you only need to do one of them.	20
GUI: Route Assignment (with associated cost for each route) Visualization Dashboard	10
Project Report	10
Project Presentation (Video)	10
	80
Research Component (can be done by the whole team, a sub-team, or an individual) There are many potential extensions in this project. **Extension 1** and **Extension 2** are two examples but there are many more. Choose one and get your tutor's approval then complete it very well. You can get up to 40% for this component and your Assignment will get up to 120%	Up to 40
	120/100
You need to follow good programming practice (e.g., well-designed, well-structured codes with clear and helpful comments). Failure to do so get penalty.	**Up to -20**
You need to demonstrate the progress you make every week to your tutor. That is, if your tutor approach you and ask for the progress, you have to be able to show the tutor the progress you have made in comparison to the previous week. Failure to do so will get a penalty.	**Up to -50**

NOTE:

- Individual marks will be proportionally adjusted based on each team member's overall contribution to the project as indicated in the '*Who did what*' declaration.
- You must also provide to shlee@swinburne.edu.my read only access to your git repository within 1 week of forming teams.

Submission

- You must upload your work to Canvas by 11:59pm on 28/5/2025(Wednesday). Create a single zip file with your code and a working version of your system. Standard late penalties apply - 10% for each day late, more than 5 days late is 0%.
- The video (10 minute duration) link should be stated in the project report.