Project 1
Artificial Intelligence

Fall 2023

Distributed: Wednesday 30 August 2023

Due Part A: Wednesday 6 September 2023

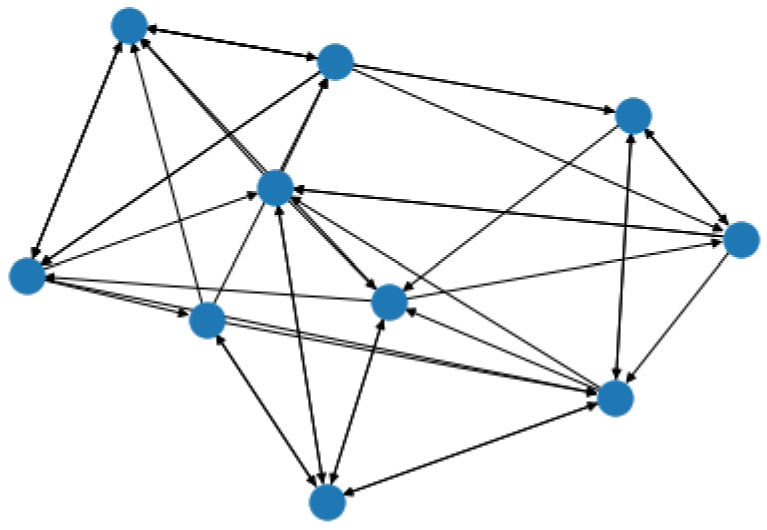
Part B: Wednesday 20 September 2023

*[Solutions to this assignment must be submitted via CANVAS prior to midnight on the due date. Submissions up to one day late will be penalized 10% and a further 10% will be applied for the next day late. Submissions will **not** be accepted if more than two days later than the due date.]*

This project may be undertaken either individually or in pairs. Please state the ***names*** of the (person or persons undertaking the project and the ***contributions*** that each has made. Only ONE submission must be made per group.

Purpose: To gain a thorough understanding of the working of an agent to assist the agent's navigation in a stochastic setting that ***mimics a real-world environment***. Thus, this requires reading through the project specification carefully to make sure that you understand its requirements. Jot down any questions/doubts that you may have and feel free to ask me questions in class or in person. Together with your partner, work out a strategy before you start coding the solution in Python. Given the limited timeframe for the project, some simplifications have been applied. In all cases, when simplifications are used, they have been pointed out explicitly. You may find it useful to compile a) a list of all functional requirements and b) a list of all assumptions before starting to code.

Environment Description: The application is a ridesharing one to be deployed in any given area of geographical coverage spanning a city. The environment is specified in the form of a graph structure as represented below (simply an example; not the entire road network). The graph that you will use consists of 100 nodes, with an average connectivity of 3 – this means setting your p value to 0.03. It is necessary that the graph that you generate is connected. If your p parameter does not result in a connected graph, then increase the p value in increments of 0.01 until it becomes connected.



Each node in the graph represents either a potential pickup or drop-off point, while edges represent roads between pairs of nodes. Edges are **bidirectional**.

All system operations such as *pickup, drop off and van parking can only take place at* ***a clock tick*** **which happens every minute**.

All *pickups, drop offs and van parking only occur at nodes, not in-between at edges. All vans are at node 0 at the start of the day.*

Scheduling

The ABC company employs a fleet of vehicles of 30 vans to service the transportation needs of its customers. The company serves its customers in a 10-hour period between 8 am and 6 pm. No pickup requests are accepted after 6 pm. Vans will continue to run after 6 pm until all the customers are dropped off, even though no pickup requests are accepted after 6pm.

Two types of scheduling occur, *pickup* and *drop off*. Firstly, the company assigns a van to a customer based on proximity of the vehicle to the customer's pickup location. This assignment of van to customer is done by examining all of the van's service queues and assigning the van whose **last** request is **closest in terms of weighted distance** to the customer's pickup location, provided that there are no more than 5 requests in that van's service queue. This is because a van's capacity is 6 (a maximum of 5 customers can be in the van at any given point in time, plus the driver). If all vans are full to capacity, then a message saying "No vans are available, try again in 15 minutes" is displayed. Once a van is assigned to a customer, a pickup request is **added** to the service queue for that van. Along with the pickup request, a drop off request for each customer in the van must also be generated.

The ordering of requests in the service queue is determined purely on the basis of minimization of weighted distance (see Tutorial 2 for details).

Van Idling

If a van is not assigned a new customer after dropping off its last one, then ***it simply parks at the drop off point and waits for the next customer pickup request***. We will assume that parking is always available.

Project Requirements

Part A (due one week after project is handed out)

Your task in this project is to implement the following requirements:

R1. Produce a pseudo code version of the algorithm needed for managing customer requests (generating and receiving customer service requests), scheduling customer pickups and drop offs. Tutorial 2 provides the foundation for this algorithm (Tutorial 2 will be posted online on Canvas and will be discussed in class). Your algorithm should follow good design principles such as modularity, have meaningful names for variables/data structures and have comments at key points in the code.

(20 marks)

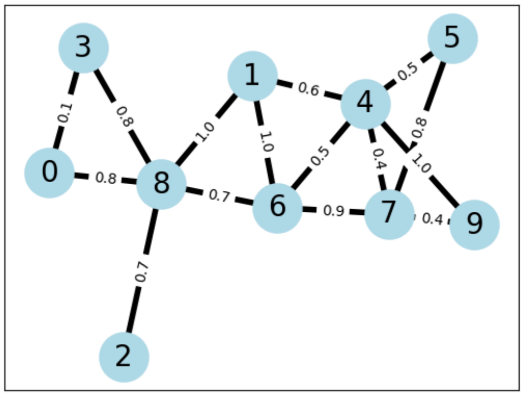
Note: Pseudo code is a high-level representation of actual code and so do not submit actual code. R1 is meant as a thinking exercise prior to coding. All pseudo code must be typewritten and submitted in a pdf document. No handwritten versions will be accepted.

Part B (due two weeks after Part A is submitted)

R2 Implement a Python program that incorporates the algorithm in R1 above on the network given below. The weights on the edges represent traffic density on the roads. A higher numeric value on a road segment means that road has a relatively higher volume of traffic when compared to other roads.

Your program must show for *each* clock tick the:

(a) contents of the service queues S1 and S2 for both vans.
(b) contents of the paths R1 and R2 taken by the vans from the start position to the position at the current clock tick.



At clock tick 1
  Pickup request at 8 for customer 1, drop off at 9.
  Pickup request at 3 for customer 2, drop off at 6.
At clock tick 2
  Pickup request at 4 for customer 3, drop off at 7.
  Pickup request at 2 for customer 4, drop off at 4.
At clock tick 3
  Pickup request at 1 for customer 5, drop off at 7.
  Pickup request at 1 for customer 6, drop off at 9.

(30 marks)

R3. To implement this requirement, you will make use of the program you developed in R2 with three changes. Firstly, you will replace the graph in R1 with a random graph of 100 nodes and a connectivity of 3. Secondly, your program will need to work with 30 vehicles, instead of just 2. Thirdly, you will need to generate 450 to 600 reservations per hour (of simulation time). Apart from these three changes the program functionality will remain the same as in the program for R2. As with R2 you will need to *populate* the service queues S and the routing queues R. Note that you will *not need* to display the contents of the queues on the console.

Your modified program should compute:

(a) the average distance traveled (over the fleet) per day when run on a road network of 100 nodes and average connectivity of 3. **(15 marks)**
(b) the average number of trips (over the fleet) per day when run on a road network of 100 nodes and average connectivity of 3. **(15 marks)**

Note that reservations occur at random points in the clock and may originate at random nodes in the graph. Likewise, destinations are also random.

R4. Now vary the fleet size to 60 vehicles and suppose that the same number of reservations occur per hour as mentioned in R3 above. What are the new values of average distance and average number of trips traveled per day taken across the entire fleet? **(10 marks)**

R5 Now run your code across a new graph having a connectivity of 4 instead of 2 whilst still having 100 nodes. What is the new value of the average distance when the booking rate given in R4 is applied? Why does it differ? **(10 marks, 5 for value, 5 for reason)**

Hand in:

For Part A

A typewritten pdf document (not an image) containing your pseud code for the algorithm required for requirement R1.

For Part B

1) A short report (between 1 and 2 pages in size) describing the approach that you took. If working in a team of size 2, a clear demarcation of contribution between team members is required.
2) Any additional assumptions that you made while undertaking this project.
3) The complete set of Python code for requirements R2, R3, R4 and R5 that you used. This needs to be submitted both as a pdf document (no images please) **and** as a publicly accessible *link*.
4) In addition to the code your pdf document must contain the answers to R3, R4 and R5.