OXFORD

# BROOKES

UNIVERSITY

School of Engineering, Computing, and Mathematics
Module COMP6032 Artificial Intelligence, Semester 1 2023/2024

# Coursework: Robo-Uber

In this project you will develop a simplified system for an automated taxi service, including both the agents (the 'taxis') and the central dispatcher. The taxis will have a map of the service area, although it should not be automatically assumed that all routes shown are passable (due to roadworks, obstructions etc). You will be asked to consider planning, search, constraint satisfaction, and reasoning in both certain and uncertain environments

## Learning Outcomes

1. Identify where AI techniques are relevant for a particular problem and apply them to develop solutions.
2. Design and construct software artefacts that employ AI techniques
3. Critically evaluate the success of employing AI approaches in a modern business context

## Introduction

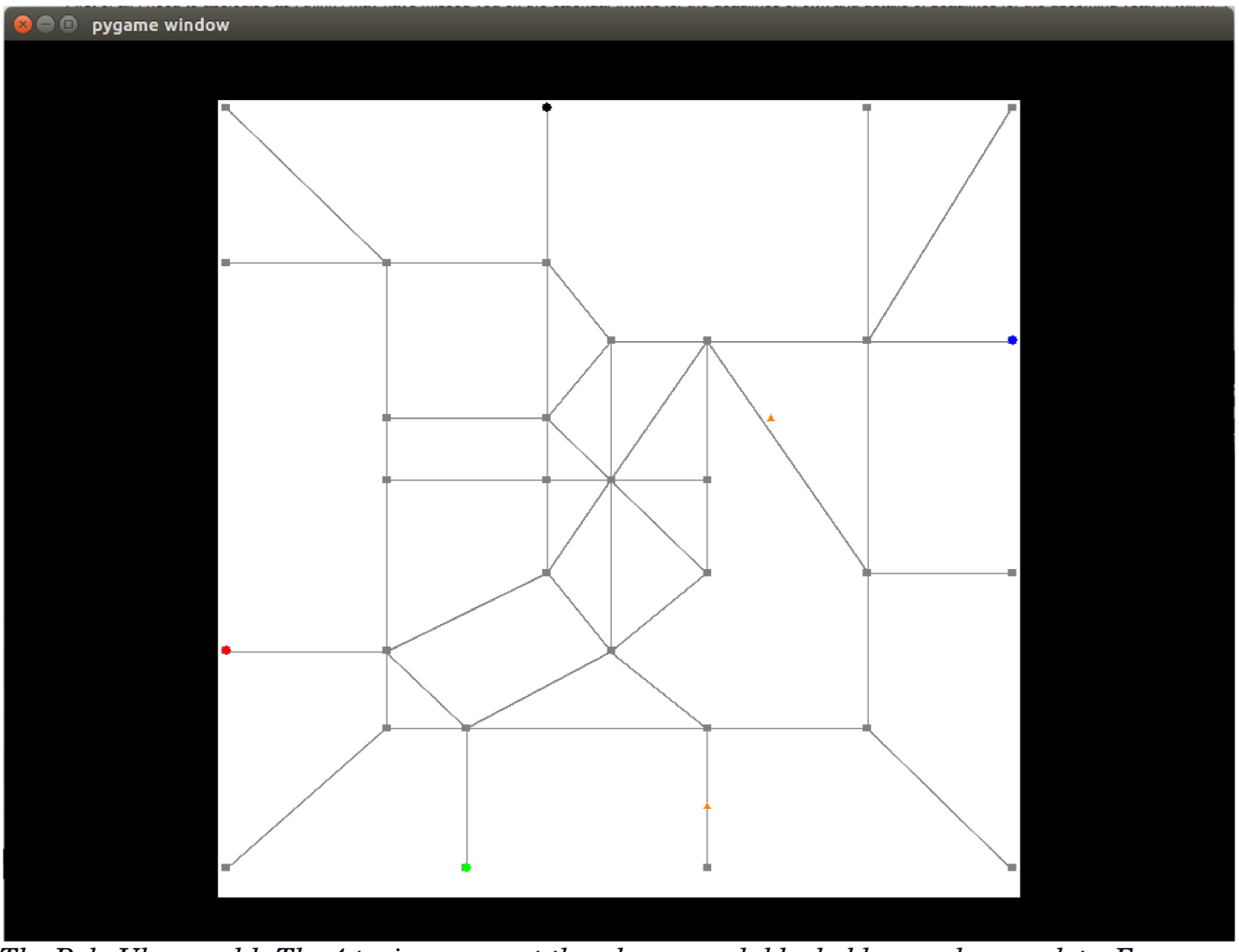
This project has 3 main components: the simulated service area: a 'net world' with marked roads, a central dispatcher which receives requests for service and bids from taxis, and schedules taxis appropriately, and each taxi itself. The service area comes pre-built, but you will be expected to design and build core components of the intelligent agents for both the taxis and for the dispatcher. You will start with some skeleton code which gives you a very primitive, deterministic system, and you will develop this into an efficient system that maximises the return to the dispatcher and the taxis.

In the environment, users of the service ('fares') appear at random and indicate a destination to be taken to. It may be assumed that they will pay the requested amount on arrival, but if a taxi does not arrive at their location within some fare-specific interval (unknown to both dispatcher and taxis, but which may be assumed to depend strongly upon quoted price), they will drop the request. Fares will also drop the request if the quoted amount (by the dispatcher) is exorbitant: over 10 times the expected time to their destination in minutes. Taxis, the dispatcher, and fares all have access to real-time information from the world which gives them a (conservative) estimate of time to destination.

Time in the environment advances in steps of minutes (these are not 'real-time' minutes: a 'minute' of environment time will elapse in a second of real time). Roads are divided into segments, each of which can only take 1 taxi in each direction. Each segment takes 1 minute to traverse when there is no traffic; there is an option to add traffic (that you can set with `trafficOn = True` in RoboUber.py) which mean some roads or intersections may become congested or blocked and require more time to traverse. Intersections may have up to 8 roads leading in or out of them; a turn is permitted in any direction, but there is a traffic light system that manages access. All of this is handled automatically in the drive function provided in the taxi agent. Some intersections can take more than 2 taxis, and may have a higher traffic capacity; this usually means also that more fares will call for a ride from these points.

It is the responsibility of the dispatcher to select which taxi gets a given fare, to determine the price for the requested journey, to issue fare requests to taxis, giving starting and final destination (along with any cancelled fares), and to coordinate bids for fares by taxis. The dispatcher collects a fixed 10% per fare transported.

The taxis, meanwhile, are responsible for route selection and for deciding whether to bid for a given fare. A taxi may bid for a fare (but are not required to bid) regardless of whether they are currently conducting another fare to a destination, but they must send their location to the dispatcher along with their fare bid. Taxis may bid on any number of fares they choose, but the dispatcher selects which taxi to assign to a given fare and the fare price. Taxis receive the price of their fare minus the 10% to the dispatcher. However, taxis also lose £1/minute whether conducting a fare or idle. Taxis start with £256; if they reach £0 they go off duty for the rest of the day. You start with 4 taxis, but you can experiment with more, or with a higher starting account balance.




*The RoboUber world. The 4 taxis appear at the edges as red, black, blue, and green dots. Fares are the orange triangles.*

## The tasks

The taxis and the dispatcher have been fitted with a very simple deterministic system for route planning, fare allocation, and fare bidding. You should not assume these work perfectly or are bug-free (this is part of the exercise)! You will have 3 tasks to complete during this coursework to

improve these functions, with opportunities for extensions for extra marks. The coursework is worth 50% of the module mark. Each task is given with its percentage of the module total.

1) (20%) In this first part you will consider the impact of path planning.
   a) Run the system over several simulated 'days' and evaluate the results. Consider the average return to each taxi and to the dispatcher, the joint return of the operation (total revenue to all parties), and the number of taxis that remain on duty as a function of the time of day. (5%)
   b) Modify the _planPath function for the taxi agent to produce a more efficient, optimum route plan and analyse its results against the previous results. You should conduct the analysis over the same time you used for the initial evaluation in 1a. Motivate your choice of path planner. (10%)
   c) For further marks, you could construct a probabilistic path planner which considers path planning with traffic estimates and plans paths according to estimated journey time. If you implement this, likewise analyse the results and evaluate the impact (if any) on fare drops (i.e, when a fare cancels because it has been waiting too long). (5%)

2) (10%) So far, you have improved the system only through changes to the taxi behaviour. Now, you will improve the dispatcher.
   Modify the _allocateFare function to schedule taxis so as to maximise the total returns at the end of the day to the taxis and the dispatcher. The crucial considerations here are how quickly the taxi can service the fare, how many bids must be in to allocate, and how to service allocation reasonably fairly. Perform an analysis of the returns and taxi schedules before and after the changes to _allocateFare. (10%)

3) (20%) In the last part you will update the bidding routines on the taxi and possibly the dispatcher.
   a) Modify the taxis' _bidOnFare function to maximise its *expected* return. Consider the deterministic time-to-location case with no traffic and the probabilistic case where there is traffic. Analyse actual results against expected results for at least 3 trials. (10%)
   b) Evaluate how successful such a system could be in commercial deployment and assess which parts of the system would need further experimental testing and live trials to modify the system for production. (5%)
   c) You could also modify the dispatcher's _costFare function to maximise the probability that each fare will be transported (i.e. minimise the likelihood that a fare will cancel). This will involve probabilistic reasoning over both the fares and the taxis. Consider how such a bidding and dispatch system might be used in a commercial environment. (5%)

## Submission Guidelines

Your analysis and evaluation should be submitted as a report of no more than 3000 words through Moodle by Friday, Week 12. All source code must be submitted through a repository to be provided by the University. Regular commits to the repository are expected and are part of the assessment criteria: clear evidence of progress on a daily or weekly basis will receive (significantly) higher marks than, e.g. no tangible evidence of progress until 2 days before the submission deadline followed by a sudden jump in commit activity. This is reflected in the rubrics, which should be consulted before submission. Any submission which cannot show evidence of substantive commits consistent with reported progress, will be capped to 40% of marks available for those components where associated code is not present in the repository.

## Feedback

Students will have the opportunity to gain formative feedback in Week 5 and Week 10. You will be asked to show your current implementation and analysis. In Week 5, only the deterministic components in Tasks 1 and 2 will be evaluated; in Week 10 the probabilistic components will also be evaluated and commented upon. Final submissions will be capped to twice the higher of the 2 marks achieved on the formative assessments, but in no case will they receive less than the formative mark. So, for example, if the first formative submission received a 47%, and the second formative submission received a 45%, the maximum mark available would be a 94% and the minimum mark would be a 45%. *Note, however, the cautions regarding plagiarism indicated below, which could result in a reduction of marks below any formative mark achieved*.

## Assessment

The assessment will consider the following aspects of your work:

- The basic functionality for each task
- The quality and originality of the implementation
- General coding practice and documentation
- Depth and insight of evaluation and analysis

This is **individual** work. Submissions will be checked for plagiarism both in the code and in the analysis. You may use the numpy and pygame external libraries, and some tools introduced as part of the practical sessions, but note in particular that you are not permitted to use third-party libraries that provide pre-written or 'black-box'-like artificial intelligence functionality. This includes automatically generated code from chatbots or IDE assistants. Any task realised with such libraries will receive **0 marks**.