COMP0037 2021/22 Coursework 1

COMP0037 Teaching Team

18th January 2022

Overview

- Coursework 01 Release Date: Tuesday, 18th January 2022; Latest Revision Date: 17th January 2022
- **Assignment Due Date: Wednesday, 16th February 12:00 (Midday UK time) 2022**
- Weighting: 40% of the module total
- **Final Submission Format**. Each group must submit *three* things:
    1. A zip file (which contains the source code implemented to tackle this coursework). The name of the zip file must be of the form `COMP0037_CW1_GROUP_g.zip`, where `g` is the letter code of your group.
    2. A report in PDF format. It will be named `COMP0037_CW1_GROUP_g.pdf`.
    3. A video which describes your answers. You may use a variety of video formats, but the video must be playable by VLC. It will be named `COMP0037_CW1_GROUP_g.[ext]`.

The total marks for each question are written in the form [X marks]. Approximately 67% of the total coursework marks are from your analysis and 33% from your coding of the algorithms. Therefore, please take time and care with your writing and analysis of solutions.

For the submitted code, you must use the notation variable names and code provided in the lectures. If you do not use these, we will consider the work a potential plagiarism case and will investigate further.

You will need to implement additional routines to support the functionality required in these questions. The code to implement this is available on Moodle. It is based upon but extends and refactors the code provided in the labs. The general areas requiring adjustment will be shown using comments.

We expect you to take care in writing your report. For example, figures and graphs need to have captions and be numbered and labelled. Equations captured by a screenshot, from the slides or elsewhere, and pasted into the document will not be accepted. When we ask you to write some code, provide an explanation in your report of what you wrote.

The final mark will be computed by summing all the marks awarded on individual questions together and dividing by the mark total.

Neither the video nor the code will be independently marked, but both must be provided before the submission deadline. Both will be checked.

Use Case

All the questions in this coursework are based on an example of a robot that automatically cleans an airport arrivals area. Airports are generally very busy places and require constant cleaning.

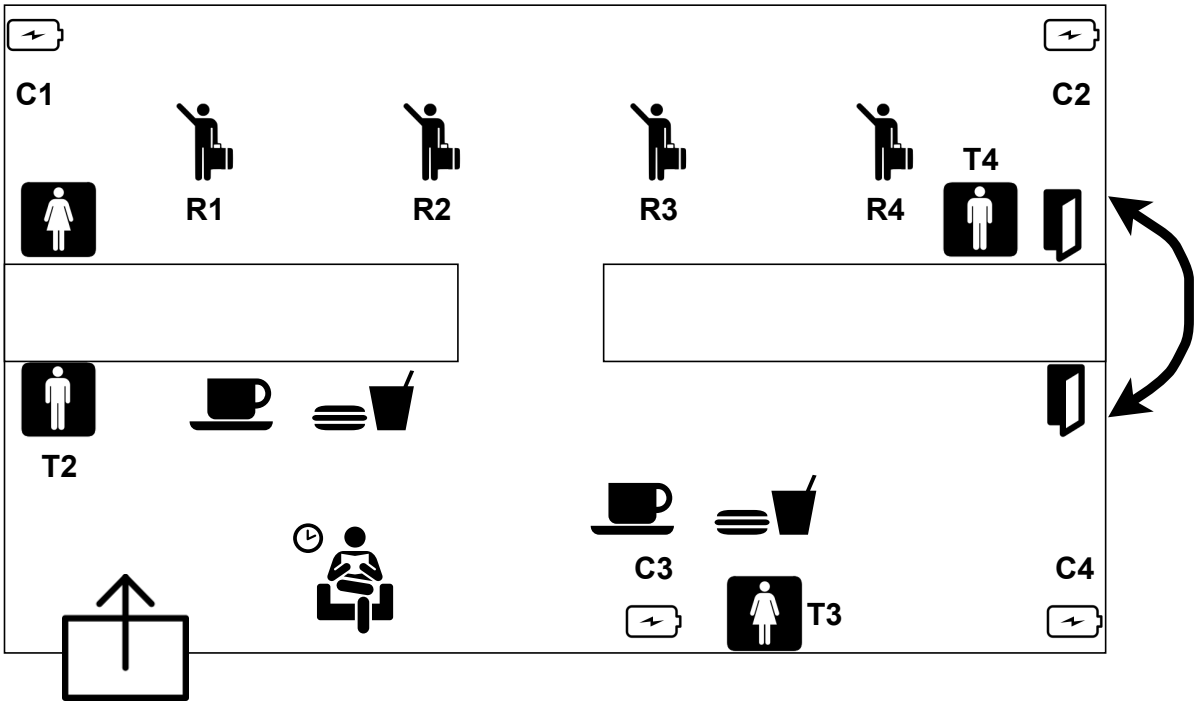

Figure 1: Overview of the arrivals area scenario. This is very loosely based on Heathrow Terminal 4. See text for description. Note this figure is not drawn to scale. The actual scenario is generated by the function `full_scenario` in the provided code (script: `scenarios`).

Figure 1 shows an overview of the environment the robot operates in. Passengers arrive at the top of the map, collect their luggage from the baggage reclaim areas (R1–R4) and pass through customs. Beyond there are various food vendors and a waiting area. Passing through the customs area is a high-cost action because of the disruption the robot would cause. Instead, the robot uses a secret door on the right-hand side of the map that only it can access. Because this door needs to be securely opened and closed, the travel costs associated with travelling through this door are much higher than driving in open space.

There are four toilet blocks (T1–T4). T1 and T4 are before customs, and T2 and T3 are after customs. The cleaning robot is powered by a battery which becomes depleted over time. Four charging stations are located throughout the terminal (C1–C4). Each charging location has a different charging rate resulting in a different charging time.

Questions

1. Although four outlets are provided for the robot to charge its battery, the different chargers have different charging rates. The charging rates are modelled as Gaussian distributions with variance 1. The true charging rates are shown in Table 1.

   The robot has been tasked with learning the best charging location. Still, the only information it can receive is visiting charging stations and measuring the charging rate it observes.

Charging Location	Mean Charging Rates (Amps)
1	4
2	4.1
3	3.9
4	4.2

Table 1: Charging information for locations.

   a. Briefly describe how this problem can be expressed as a bandit problem and specify the number of arms. Describe what the epistemic and aleatoric uncertainties are for this problem.

   [6 marks]

   b. Implement a multi-arm bandit simulation to simulate the charging rates by the different stations. Describe your code. Generate violin plots to visualise the locations using the data generated from your simulator.

   [6 marks]

   c. Define what regret is and present the formula used to calculate it. Explain why it is used to assess the performance of bandit problems. Although we can use regret for this problem, why might it not be an available tool for other problems?

   [5 marks]

d. Describe the ϵ-greedy strategy. Prove that, if there are C charging stations the robot could visit, the *long-term average probability* of visiting the optimal station is

$$1 + \left(\frac{1}{C} - 1\right)\epsilon.$$

The number of correctly pulled handles could be significantly lower than this value for very short runs. Propose an explanation for why this is the case and show how it could be modelled using the equation above.

[15 marks]

e. Extend your implementation of the bandit problem to support Monte Carlo analysis (in which you can run the whole experiment with the agent multiple times and average the rewards across all those runs).

By varying ϵ and experimenting with the number of Monte Carlo runs, demonstrate that the equation above appears to be correct.

[15 marks]

f. A number of ways to improve the performance of ϵ-greedy has been proposed. Describe one approach, implement it, and demonstrate its performance using graphs of both cumulative regret and percentage of optimal actions taken.

[15 marks]

g. Briefly describe Upper-Confidence-Bound (UCB) Action Selection, including the intuition of why this algorithm was proposed, and outline the equations used to implement it.

[5 marks]

h. Investigate the performance of the UCB for several different choices of the degree of exploration. What trend do you observe, and what do you think is the value which gives the best performance. How do you think the degree of exploration relates to the variance of the charging rates?

[10 marks]

[Question total 77 marks]

2. a. Describe the *breadth-first search* and *depth-first search* algorithms. What are the advantages and disadvantages of each?

[4 marks]

The script `compare_path_planning_algorithms.py` causes the robot to plan a path between each rubbish bin in turn in the environment.

b. Evaluate the performance of the breadth-first and depth-first planners for this case. Your analysis should include quantitative measures (e.g., the sum of all path costs and the sum of all cells visited when planning the paths) and a qualitative discussion of any unusual behaviour observed. Use screenshots to illustrate unusual behaviours and explain why they arise.

[10 marks]

c. Describe *Dijkstra's algorithm* and explain the two key changes it has compared with breadth-first search. What is the purpose of each of these modifications?

[6 marks]

d. Modify the class `DijkstraPlanner` to implement Dijkstra's algorithm and explain your implementation. Compare the results with breadth-first search with regards to path cost and the number of cells visited.

[18 marks]

Suppose a path takes a cell i into an adjacent cell j. The current l-stage additive cost is modelled as the Euclidean distance between the cells,

$$l(s_i, s_j) = |s_j - s_i| .$$

However, this does not take account of traversability constraints. These are modelled by

$$l(s_i, s_j) = \alpha(s_j) |s_j - s_i| ,$$

where $\alpha(s_j)$ is a penalty factor based on the type of j's cell. Its value is given by

$$\alpha(s_j) = \begin{cases} 5 & \text{if } s_b \text{ is part of the secret door} \\ 100 & \text{if } s_b \text{ is part of the customs area} \\ 1 & \text{otherwise} \end{cases}$$

e. Modify the implementation of `compute_transition_cost` in `airport_map` to take account of $\alpha(s_j)$. Modify the code to implement this change and explain what changes you made. Evaluate how this changes the performance of Dijkstra's algorithm.

[9 marks]

f. Describe the key ideas behind the A* algorithm. What does it mean for a heuristic to be admissible, and why is it important? Explain why the Euclidean distance is an admissible heuristic, but the squared Euclidean distance is not.

[8 marks]

g. What do you think is the *optimal heuristic* that A* could use? Could it be used in practice? Explain your answer.

[5 marks]

h. Implement the A* algorithm in the `AStarPlanner` class with the Euclidean heuristic and describe your implementation. Compare results with the earlier algorithms with regards to path cost and the number of cells visited.

[12 marks]

The A* planning algorithm with an admissible heuristic will be used for all future robot operations.

The designers need to design a recharge policy for the robot: when its battery has become sufficiently low, the robot needs to drive to a recharging station and recharge itself. This can be expressed in the policy

$$a_t = \pi\left(s_t\right),$$

where s_t is the state where the robot decides it needs to recharge and $a_t \in [1, 2, 3, 4]$ is the charging station the robot will go to. Suppose the ith charging station is at s_i and the recharge is Gaussian distributed with mean μ_i and covariance σ_i^2. The action function is

$$q_\pi(s, a) = \mu_a - l\left(s, s_a\right).$$

The values for μ_i and σ_i^2 are known. The quantities have been rescaled to match the units with the total path length. Note that the means of the charging stations differ from those in Q1.

i. Propose an algorithm to compute $\pi_*(s)$.

[6 marks]

Hint: You do not need to use policy or value iteration.

j. Implement your algorithm in the `compute_recharging_policy` script. The resulting policy must be stored in an instance of the `ChargingPolicy` class. Explain your implementation.

Display the learned policy using the `ChargingPolicyDrawer`. Discuss the results. Are they as you expect?

[12 marks]

k. Pick an inadmissible heuristic for your A* algorithm. Explain the heuristic used. Compute the new policy and display the results. Comment on whether the use of the inadmissible heuristic made a significant difference or not.

[4 marks]

l. Suppose the means and covariances for each charging station (μ_i, σ_i^2) were not known. How would you modify your algorithm to handle this case? You do not have to implement the resulting algorithm.

[6 marks]

[100 marks]