COMP0037 2024/25 Coursework 1

COMP0037 Teaching Team

Current Revision: 20250214

Overview

- Coursework 01 Release Date: Friday, 30th January 2025
- **Assignment Due Date: Monday, 24th February 16:00 (4PM UK time) 2025**
- Weighting: 40% of the module total
- **Final Submission Format**. Each group must submit *two* 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`.

    Note that a penalty of 2% will be applied if the files are submitted with the wrong name.

Mark Allocation and Further Instructions

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 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.

As discussed in Lecture 00 (Overview), we expect you to take care in writing your report. For example, figures and graphs must have captions and be numbered and labelled. Equations captured by a screenshot, from the slides or elsewhere, and pasted into the document are of low quality and are likely to cause a penalty for your marks. 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 and dividing by the mark total.

The code will not be independently marked but must be provided before the submission deadline. It might be checked, and if it does not run or produce the results presented in the report, it can impact your marks.

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.

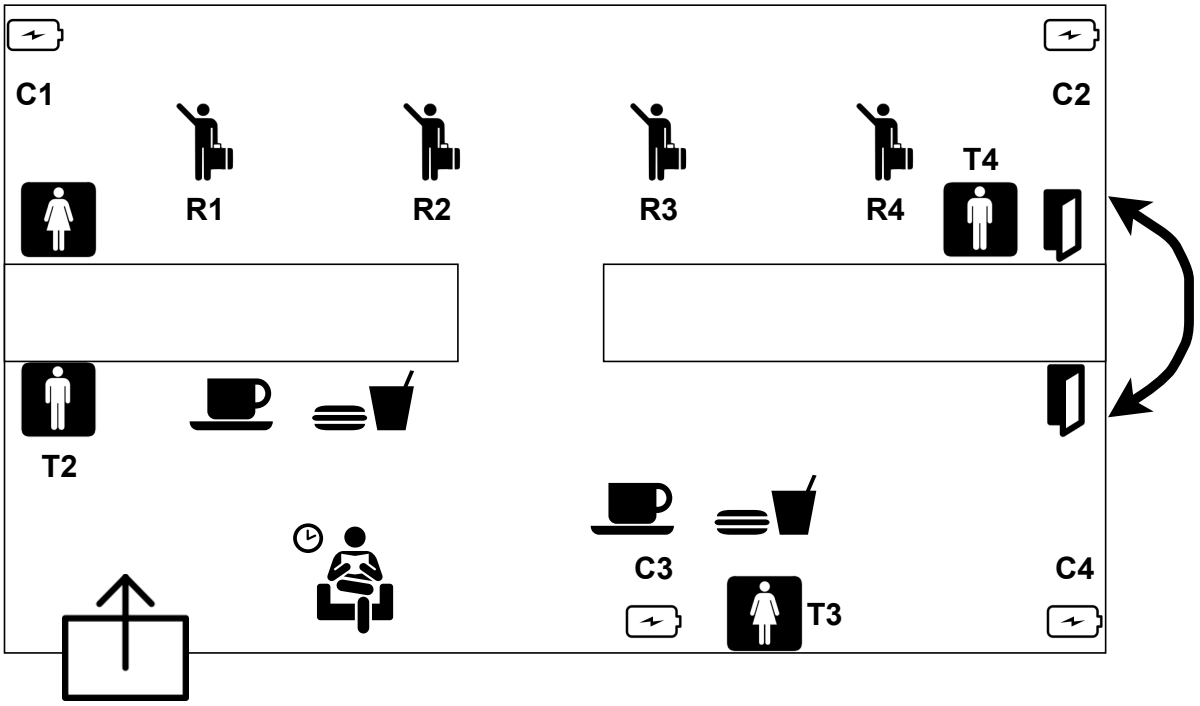

Figure 1: Overview of the arrivals area scenario. This is very loosely based on Heathrow Terminal 4. See text for description. Note this figure is not drawn to scale. The actual scenario is generated by the function `full_scenario` in the provided code (script: `scenarios`).

Figure 1 shows an overview of the environment where the robot operates. 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 the 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.

You can visualise the full airport map using the script `show_full_airport_map.py`.

Part 1

In this part, we will look at controlling the robot in a *high-level* fashion. For example, the actions the agent can take are to instruct the robot to drive to a specific location. As a result, this is modelled as a discrete finite planning problem.

1. a. Describe how to implement the *breadth-first search* and *depth-first search* algorithms in LaValle's discrete forward planner pseudocode. What advantages and disadvantages of each search type were identified in the lectures?

[6 marks]

The script `q1_b.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 unexpected behaviour observed. You should use captures of the graphs to illustrate examples of these unexpected behaviours.

[12 marks]

c. Describe how *Dijkstra's algorithm* is implemented in LaValle's discrete forward planner pseudocode. What is the purpose of each of these modifications?

[3 marks]

d. Modify `DijkstraPlanner` in `grid_search/dijkstra_planner.py` to implement Dijkstra's algorithm. Explain the modifications you made to the code and why.

Hint: To implement the planner, you will need to implement a number of methods. See `BreadthFirstPlanner` as an example. Unimplemented methods will cause the programme to raise an exception, indicating what failed and where.

[8 marks]

e. Evaluate the performance of your implementation of Dijkstra and compare it with the depth and breadth first algorithms you looked at in part (a). Your analysis should discuss both the qualitative and quantitative differences in performance with the previous algorithms.

[6 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,

$$l\left(s_i, s_j\right) = \left|s_j - s_i\right|.$$

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

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

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

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

f. Complete the implementation of the `compute_transition_cost` method in `common/airport_map.py` to take account of $\alpha\left(s_j\right)$. Modify the code to implement this change and explain what changes you made. Evaluate how this changes the performance of Dijkstra's algorithm in this scenario.

[7 marks]

[Total 43 marks]

2. a. Describe how the *A* algorithm* is implemented in LaValle's discrete forward planner pseudocode.

[2 marks]

b. What does it mean for a heuristic to be admissible, and why is it important? Illustrate your example by providing examples of admissible and inadmissible heuristics and explain why they are admissible or not.

Provide an example of a heuristic that is inadmissible but is still guaranteed to provide an optimal solution.

[7 marks]

c. In the lectures we saw examples several different admissible heuristics (Euclidean, Octal, Manhattan). However, there are many other heuristics which could be used as well. What do you think is the *best possible heuristic* that A* could use? Why is it optimal? Could it be used in practice? Explain your answer.

[5 marks]

d. Implement the A* algorithm in the `AStarPlanner` class with the Euclidean heuristic and describe your implementation.

[4 marks]

e. Evaluate the performance of your implementation of A* and compare it with the depth and breadth first algorithms you looked at in part (a). Your analysis should discuss both the qualitative and quantitative differences in performance with the previous algorithms.

Compare results with the earlier algorithms with regard to path cost and the number of cells visited.

[6 marks]

[Total 24 marks]

Part 2

The engineers realise that the robot is actually very poorly built and very poorly calibrated. As a result, when commanded to move in a given direction, it is not guaranteed to move in that direction. Rather, as illustrated in Figure 2, the robot moves in the nominal direction with a probability p or one square on either side of the nominal direction with a probability $q = 0.5(1 - p)$.

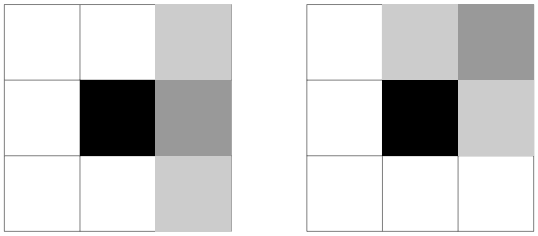

Figure 2: Examples of the errors in the process model. In both cases, the original robot position is the black square. The nominal direction of motion is the dark grey square. The errors in the robot mean that the robot will either end up in the dark grey square or the light grey square.

Because of the randomness in the robot movement, the robot needs to be analysed from a *low-level* perspective.

The traversability costs are the same as in question 2. Because the robot's motion is uncertain, it might collide with objects in the environment. If the robot collides with the luggage reclaim area, it experiences a cost of -10. If it collides with any other object, it incurs a cost of -1. The robot does not need to be recharged for the duration of the task of interest.

3. The engineers determine that this system can be modelled as a Finite Markov Decision Process (FMDP).

    a. Consider the case where the robot starts at a known start location and has to reach a known goal location. When the robot reaches the goal, the episode ends. Explain how this can be modelled as an FMDP.

    [6 marks]

The class `PolicyIterator` will implement the policy iteration algorithm. Note the provided code only implements the policy evaluation step.

b. Use the script `q3_b.py` to evaluate the policy. The policy evaluation algorithm using the initial policy and $\gamma = 1$ will not converge. Why is this the case? Suggest one way to fix this and illustrate that the approach works.

   Hint: If you want to change the parameters used in the solver, you should not have to change them directly. Rather, there is a setter/getter function you can use.

   [6 marks]

c. Modify the class `PolicyIterator` to complete the implementation of the policy improvement step. Describe your implementation.

   [12 marks]

d. Run the script `q3_d.py`. Compute the policies and value functions for $p = 1$, $p = 0.9$, $p = 0.6$ and $p = 0.3$. In each case, plot the policy and the value function for each cell. Discuss the behaviour of the policy and value function for the different values of p.

   [8 marks]

For the rest of this coursework, $p = 0.8$.

The policy iterator can be very slow because of the number of iterations required with the policy evaluation step. Therefore, the designer is interested in trying to determine ways to reduce the computational cost required but still obtain the optimal solution.

e. What parameters can you use in the provided code to adjust how often the policy evaluator runs?

   Develop a way to explore how to investigate the parameters. Explain your approach. Implement it using the script `q3_e.py` where necessary and propose a suitable set of parameters. Use empirical data to support your arguments.

   [10 marks]

f. Complete the implementation of the value iteration in the `ValueIterator` class and describe your code.

[12 marks]

g. Modify the script `q3_g.py` to compare policy iteration and value iteration on the full map and discuss the difference between the two algorithms. This includes the number of iterations required and differences in generated policy. Which algorithm would you recommend and why?

[10 marks]

[Total 64 marks]

The actual robot will be configured to operate sequentially, moving between rubbish bins and toilets in a pre-defined sequence. As the robot moves through the sequence, the engineers determine that its battery will become depleted. If the battery becomes flat, the robot will have to be rescued, which would incur a large negative cost.

After it has visited one cleaning site, the engineers propose that the robot decide to either move directly to the following cleaning site or visit a recharge station, recharge itself, and then move to the following cleaning site. The recharge stations provide Gaussian-distributed rewards.

4. Describe, but do not implement, how you would represent this as an FMDP and how policy or value iteration will be applied. Your description should be in reference to the components of an FMDP.

[12 marks]

[Total 12 marks]

[Coursework Total 143 marks]

Getting Help

You are encouraged to use the assignment discussion forum to help when you get stuck. Please check the forum regularly and try and answer each other's questions. Notifications should now be set up. We will be checking the forum as often as possible and answering any questions that have remained unanswered. Please note that if you have questions about coursework, please post them to the forum directly rather than emailing me. The reason is that all students will be able to see the reply.

Submission and Feedback

The deadline for submission is **16:00 on Wednesday 20th February, 2025**. Late penalties will start to accrue if any element of the submission (report, code) has a receipt time of 16:01 or later.