Planning and Approximate Reasoning

(MIA-MESIIA) Practical Exercise 2: PAR

Waiter Robotic Task

- The deadline for the delivery of this exercise is **November 12th, 2023**
- There is a second submission date set to **January 11st, 2024**, with a maximum grade of **8**.
- A zip file with the source codes in PDDL and a PDF file must be sent.
- The code files and the results should be exported from Visual studio code. If not, include the necessary instructions files to understand the program (documenting the *domain.pddl* and *problem.pddl* file if possible).
- A detailed documentation in PDF is required (see details below)
- The submission of the source and documentation must be done using Moodle.
- This assignment can be achieved by a **group of 2 or 3 persons**.

HANAKI is a Japanese restaurant located in Barcelona that has decided to innovate their service by employing robotic waiter, which is similar to a cat to bring the food to the customers as shown in Figure 1. These agents can plan and execute a sequence of actions to accomplish some goals (i.e., to deliver food to customers).

Figure 1: a Lucki robot (Waiter)

Figure 2 shows a map of the restaurant divided into 7 discrete areas: The buffet area itself (BTA), the anterior upper area (AUA), anterior middle area (AMA), anterior lower area (ALA), posterior upper area (PUA), posterior middle area (PMA), and posterior lower area (PLA). The robot can step directly between two adjacent areas (e.g., AUA and AMA), but it can not directly move between areas blocked by grey walls as shown in the Figure. For instance, the robot can step directly from PMA to PUA.

For simplicity, lets assume:

- Customers place orders using a mobile app, thus they do not have to go anywhere to place or accept the orders.
- Each order consists simply of a request for a plat of food, not specifying specific items from the buffet.
- The robot's movement within the building is discrete.

The setup is then as follows: the agent "robot" receives a list of customer orders, which give their name and location in the restaurant. For each order, the agent must fill plat with food from the buffet-table, then deliver the food directly to the customers table.
World's Predicates and Action

Predicates

you can assume the predicates to solve the robotic cleaner planning problem are:

- Define the predicates you will use to describe the map. Specifically, how will you establish that one location is **Adjacent** to another?
- Define how you will keep track of the **Location** that an agent, plate or customer is **At**.
- Define the predicates you will need to model a **Plate** and, in particular, whether or not it **Has Food**.
- Define the predicates you will need to model a **Customer**, and whether or not that customer has been **Served**

Using the predicates you defined and the map as depicted in Figure 2, describe an example of the initial and goal states of the problem: 1) the agent starts off at BTA, 2) There is a plate at BTA and 3) There is a single customer at PMA.

Actions

In addition, you can assume that actions, which you need to select from them to update the world state are:

- The agent can **Pick up** a plate. Note, an agent can only hold one plate at a time, so if it is currently **Holding** a plate, it will be unable to pick up another.
- The agent can **Present** a plate of food to a customer that is in the same location as the agent, after which a customer should be said to be **Served.**
- At the BTA, the agent can **Fill** a plate with food that it is **Holding** if the plate does not already **Have Food**.
- The agent can **Move** from its current location to an **Adjacent** one.

<u>Add more actions if you think that will improve the efficiency and optimality of the planning.</u>

Take particular care when defining these actions to consider what happens to the location of items that the agent is holding when it is moving.

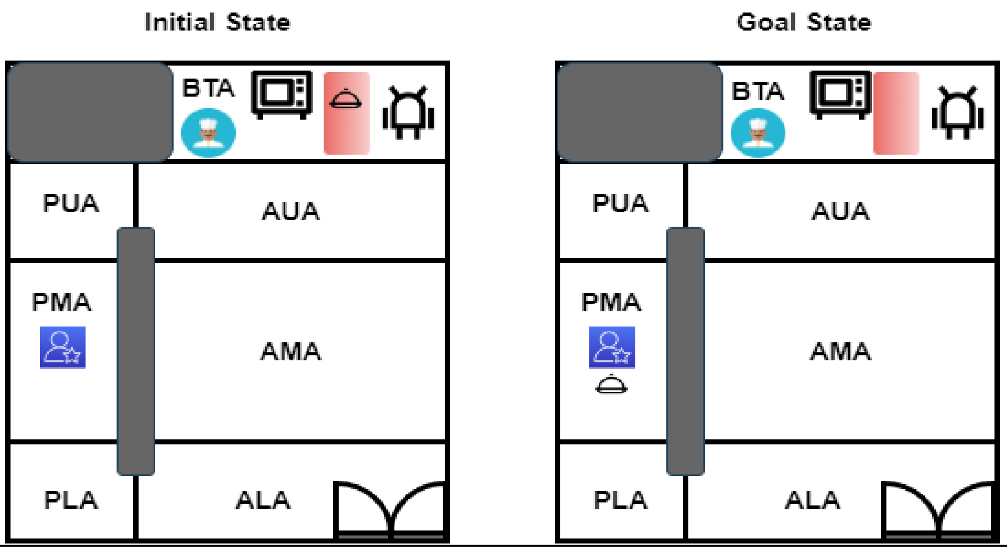

Figure 2: an example of the waiter robotic problem, (right) the initial and (left) the target states.

Modelling the planner:

- Implement the planner in PDDL to solve the problem, indicating the internal representation used to manage the preconditions, to check the applicability of the operators, etc. The <domain.pddl> should define the Robot world actions and predicates. The <problem.pddl> contains the objects, init state, and goal state. The output of your planner should be a sequence of actions that solves the given problem.
- Test your code with a set of testing cases of increasing complexity (**a minimum of 3 scenarios**; one of them is the example shown in Figure 2). For instance, you can assume the agent can start from a different area (e.g., PMA) rather than the BTA. Discuss in the document the solutions your code found for these examples, e.g., if the planner can provide an optimal plan. so if the planner fails to find a plan, there might be something wrong with your PDDL definitions.
- The algorithm execution output must be clearly printed out in a pdf text file with an explanation about your code and your representation. This file has to clearly display the states that are being generated and evaluated, and the sequence of the actions.

Documentation content:

1. Introduction to the problem
2. Analysis of the problem (objects, operators, predicates, etc.)
3. PDDL Implementation (A Domain PDDL file + 3 Problem PDDL files)
4. Testing cases and results (show the important steps to arrive to the solution, the final path as well). Analysis of the results (complexity and number of nodes generated and expanded).

Evaluation criteria:

Grade	Item
5	Implementation
2	execution of the test suit and additional tests
3	analysis (of the problem and the results obtained)
10	Total