Planning and Approximate Reasoning

(MIA-MEISISI) Practical Exercise 1: PAR

Rescue Drone Task

- The deadline for the delivery of this exercise is **October 20th, 2024**
- There is a second submission date set to **January 07st, 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.

A rescue drone is an unmanned aerial vehicle (UAV) equipped with sensors and autonomous navigation capabilities designed to assist in emergency situations by locating and rescuing isolated individuals. These drones can quickly cover difficult terrain, identify obstacles, and safely transport individuals or supplies to designated safe areas.

In this task, students will program a rescue drone to navigate a disaster site. The site is represented by a grid composed of areas, and some of these areas contain obstacles that block the drone's movement. Some areas have people stranded who need to be rescued. The drone must navigate the grid to pick up the people and transport them to a designated safe zone.

Figure 1: a rescue drone to assist in emergency situations by locating and rescuing isolated

The drone has to plan an efficient sequence of actions to rescue all stranded people while avoiding obstacles. Additionally, the safe zone has a maximum capacity for the number of people it can hold at any given time, adding a constraint that the drone needs to account for when planning.

Example: Figure 2 shows an example of the initial and target state in the state world of the rescue drone problem.

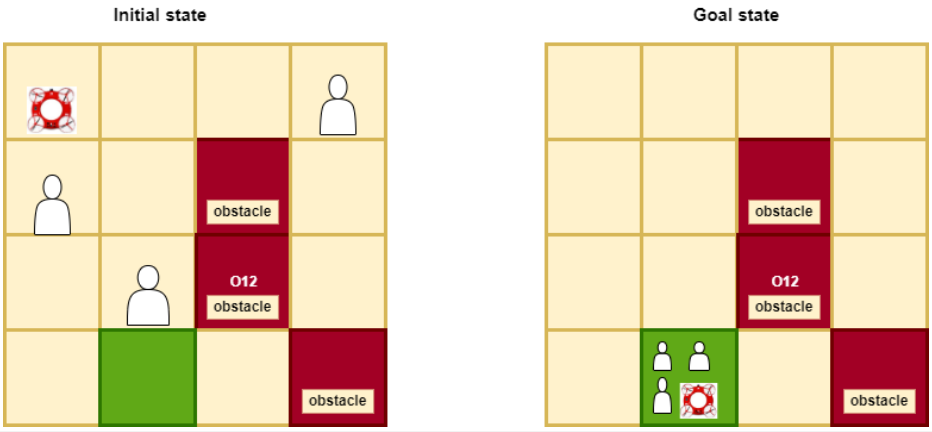

Figure 2: A 4x4 grid representing the disaster area. The drone (D) starts at (1,1), the safe zone (S) is at (4,2), obstacles (X) block certain locations, and people (P) are stranded at various positions.

In this example, the drone will have to avoid the obstacles (X), collect each person (P), and drop them off at the safe zone (S), while considering the safe zone's capacity limit.

- **Constraints**
  - **Safe Zone Location**: The safe zone is a designated area on the grid where rescued people are dropped off. The location of the safe zone is predefined for each test case.
  - **Safe Zone Capacity**: The safe zone can hold a maximum of *m* people at any given time, where m = *n-1, n* is the number of rows or columns of the grid. If the drone needs to rescue more people, it must wait until the safe zone has room again, simulating resource constraints (such as medical care or limited shelter).
- **Task**
  - Rescue people from various locations on the grid.
  - Avoid obstacles that block its path.
  - Transport people to the safe zone while respecting the maximum capacity.
  - Minimize energy consumption (distance traveled or number of actions).

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

- **Move(d1,d2):** The drone moves from location d1 to an adjacent location d2.
- **Pick-up(person,p):** The drone picks up a person stranded at location p.
- **Drop-off(person,safe_zone):** The drone drops off a person at the safe zone, as long as the capacity allows.

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

In addition, you can assume the predicates to solve the rescue drone planning problem are:

- **Drone-location(d):** The drone is at location d.
- **Person-location(person,d):** Person is stranded at location d.
- **Obstacle(d):** Location d contains an obstacle.
- **Safe-zone(d):** Location d is the designated safe zone.
- **Safe-zone-capacity(m):** The number of people currently in the safe zone is m.
- **Adjacent(d1,d2):** Locations d1 and d2 are adjacent.
- **Rescued(person):** The person has been rescued (i.e., is in the safe zone).

Change the predicates if you think that will improve the efficiency and optimality of the planning.

You have to:

- 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 drone world actions and predicates. The <problem.pddl> contains the objects, initial states, and goal states. 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 4 test cases; one of them is the example shown in Figure 2, one of them a grid 5x5, changing the capacity of the safe-zone and number of rescued people, etc.).
- Discuss in the document the solutions your code found for these examples, e.g., if the planner can provide an optimal plan. You can also use different planners available online and compare between them.
- The algorithm execution output must be clearly printed out in a pdf text file with an explanation about your code. 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, search space, operators, predicates, etc.)
3. PDDL Implementation
4. Testing cases and results (show the important steps to arrive to the solution, not only the final path).
5. 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