Planning and Approximate Reasoning

(MIA-MEIISA) Practical Exercise 1: PAR

Cleaner Robotic Task

- The deadline for the delivery of this exercise is **October 18th, 2023**
- There is a second submission date set to **January 10st, 2023**, 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.

There is a square building composed of n^2 offices located in a matrix of n rows and n columns. From each office, it is possible to move (horizontally or vertically) to the adjacent offices. Each of the offices may be clean or dirty. An automated cleaning robot, called PR2, as shown in **Figure 1** can move from one office to an adjacent office to achieve the cleaning process.

Figure 1: a robot (PR2) to perform the cleaning of the offices

Each office may be empty or contain a box (there can't be more than one box in the same office). The number of boxes in the building may be between 0 (no boxes in the all offices) and n^2-1 (i.e., less than the number of offices by 1). The PR2 can push a box from one office to an adjacent office.

The robot will start with a given initial configuration (different for each test example), with some clean and dirty offices and a box in some rooms. In the goal state, all rooms will be clean, and the final position of the boxes will be explicitly indicated.

In this practical exercise you have to design and implement (in PDDL) a planner to solve the cleaning problem that can discover how to go efficiently from the initial state of the world to the goal state.

Example: **Figure 2** shows an example of the initial and goal state in the state world of the cleaner robotic problem.

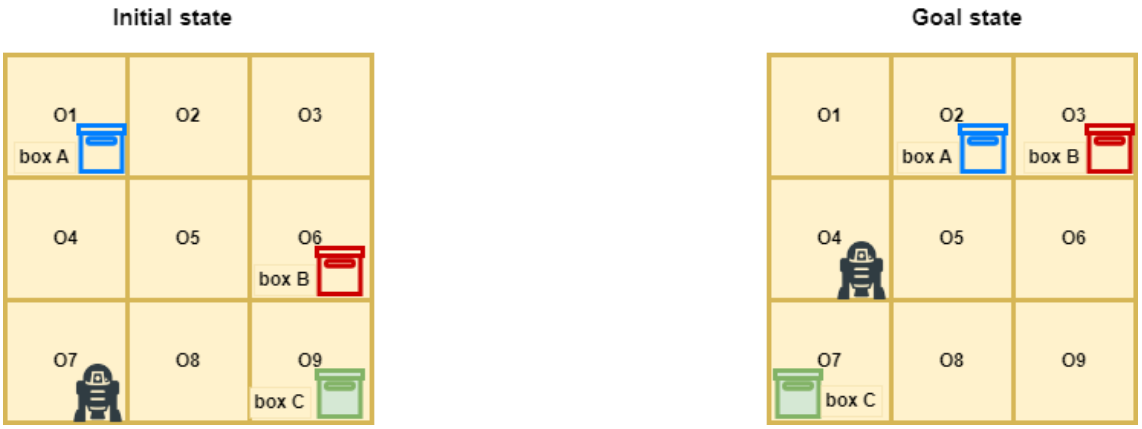

Figure 2: an example of the cleaner robotic problem, (left) the initial state, and (right) the goal state.

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

- ***Clean-office(o)***: the robot cleans office o
- ***Move(o1,o2)***: the robot moves from office o1 to office o2
- ***Push(A,o1,o2)***: the robot pushes box A from office o1 to office o2

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

In addition, you can assume the following predicates to solve the robotic cleaner planning problem:

- ***Robot-location(o)***: the robot is in office o.
- ***Box-location(A, o):*** box A is located in office o.
- ***Dirty(o):*** office o is dirty.
- ***Clean(o):*** office o is clean.
- ***Empty(o):*** there isn't any box in office o.
- ***Adjacent(o1, o2):*** offices o1 and o2 are horizontally or vertically adjacent.

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

Each student has 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 Robot world actions and predicates. The <problem.pddl> contains the objects, initial 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 4; one of them is the example shown in Figure 2) discussing in the document the solutions your code found for these examples, e.g., if the planner can provide an optimal plan.
- 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