PROJECT 1. CLEANER ROBOTIC TASK

Planning and Approximate Reasoning

Laia Font Solanich

October 2023

Contents

List of Figures

1 Introduction to the problem

The Cleaner Robotic Task is a problem involving a cleaner robot (PR2), dirty and clean offices and boxes. The building is represented as a matrix with n rows and n columns, where one of the offices can be either clean or dirty.
Additionally, offices may be empty or can contain a single box, with the number of boxes ranging from 0 to $n^2 - 1$.
The goal is for the cleaner robot to clean all the dirty offices while moving the boxes that are in the way, in the most efficient way possible.

2 Analysis of the problem

In this section we will see an analysis of the given problem. We will talk about the object, predicates, the actions, among others.

2.1 Objects

For this problem we have different objects. The first one is the cleaner robot, which is the one who has to accomplish the goal. Secondly, we have different objects the robot can move through and lastly, we have the different boxes, which the cleaner robot has to move around in order to clean a dirty office.

2.2 Search space

As for the search space, we have a matrix of n rows and n columns, giving us a square building composed of n^2 offices. This is defined by the combinations of office cleanliness, whether it is clean or dirty, the box location, the cleaner robot location, and the number of boxes, which can range from 0 to $n^2 - 1$.

2.3 Predicates

Predicates define the state of the world and the conditions that must be satisfied in it. These are the predicates used:

- Robot-location(o): Indicates the office in which the robot is in.
- Box-location(A, o): Indicates that a box A is on office o.
- Dirty(o): Indicates that office o is dirty.
- Clean(o): Indicates that office o is clean.
- Empty(o): Indicates that office o does not have any box in it.
- Adjacent(o1, o2): Identifies two offices that are horizontally or vertically adjacent.

2.4 Operators

The operators are the actions that can be taken to transition from one state to another. These represent the actions that the cleaning robot can perform. The ones chosen 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 a box from office o1 to office o2.

3 PDDL Implementation

For the PDDL implementation I create different files: 1 for the domain and different files for each problem.

3.1 Domain file

The domain file has the description of the search space and world. For this problem, the file has the predicates and actions I stated in the previous points. You also have to specify which solver you want to use, for this project we use **strips**.
The domain for this project looks like this.

```
(define (domain cleaner_robot)
    (:requirements :strips)

    (:predicates
        (robot-location ?o)
        (box-location ?b ?o)
        (dirty ?o)
        (clean ?o)
        (empty ?o)
        (adjacent ?o1 ?o2)
    )

    ; ------ ACTIONS ------
    (:action clean_office
        :parameters (?o)
        :precondition (and (robot-location ?o) (dirty ?o))
        :effect (and (not(dirty ?o)) (clean ?o))
    )

    (:action move
        :parameters (?o1 ?o2)
        :precondition (and (robot-location ?o1) (adjacent ?o1 ?o2))
        :effect (and (robot-location ?o2) (not (robot-location ?o1)))
    )
```

```
    (:action push
        :parameters (?b ?o1 ?o2)
        :precondition (and (robot-location ?o1) (box-location ?b ?o1)
            (empty ?o2) (adjacent ?o1 ?o2))
        :effect (and (box-location ?b ?o2) (not(box-location ?b ?o1))
            (not(empty ?o2)) (empty ?o1))
    )
)
```

3.2 Problem files

The problem files have the description of the world that we want to work on. In this file you have to specify the different objects you have, describe the initial state of the world and, lastly, the goal we want to achieve, the final state.

4 Testing cases and results

In this section we will see the different testing cases that were thought and the results obtained on each of them.

4.1 Case 1: Original case

For the original case we use the one given in the statement. We have a 3x3 search space, with 3 boxes and the robot. Since there is no specified amount of dirty rooms, I assume all rooms are dirty. In Figure 1, we can see the visual representation.

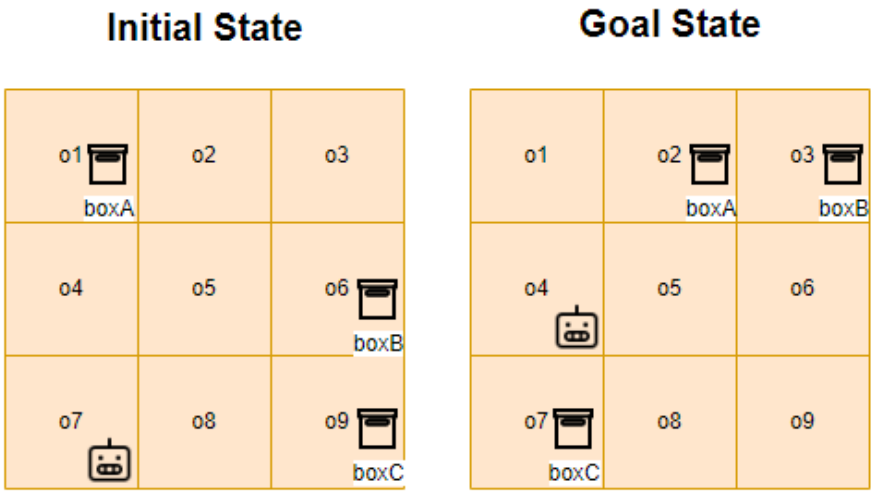

Figure 1: Visual representation for case 1 with the initial state (left) and goal state (right).

This gives us the following written problem.

```
(define (problem basic_problem) (:domain cleaner_robot)
(:objects
    o1
    o2
    o3
    o4
    o5
    o6
    o7
    o8
    o9
    bA
    bB
    bC
)

(:init
    (adjacent o1 o2) (adjacent o1 o4) (adjacent o2 o1) (adjacent o2 o3)
        (adjacent o2 o5) (adjacent o3 o2) (adjacent o3 o6) (adjacent o4
        o1) (adjacent o4 o5) (adjacent o4 o7) (adjacent o5 o2) (adjacent
        o5 o4) (adjacent o5 o6) (adjacent o5 o8) (adjacent o6 o3)
        (adjacent o6 o5) (adjacent o6 o9) (adjacent o7 o4) (adjacent o7
        o8) (adjacent o8 o5) (adjacent o8 o7) (adjacent o8 o9) (adjacent
```

```
        o9 o8) (adjacent o9 o6)
    (robot-location o7)
    (box-location bA o1) (box-location bB o6) (box-location bC o9)
    (empty o2) (empty o3) (empty o4) (empty o5) (empty o7) (empty o8)
    (dirty o1) (dirty o2) (dirty o3) (dirty o4) (dirty o5) (dirty o6)
        (dirty o7) (dirty o8) (dirty o9)
)

(:goal (and
    (clean o1) (clean o2) (clean o3) (clean o4) (clean o5) (clean o6)
        (clean o7) (clean o8) (clean o9)
    (box-location bA o2) (box-location bB o3) (box-location bC o7)
    (empty o1) (empty o4) (empty o5) (empty o6) (empty o8) (empty o9)
    (robot-location o4)
))
)
```

And the following plan, which you can see on Figure 2.

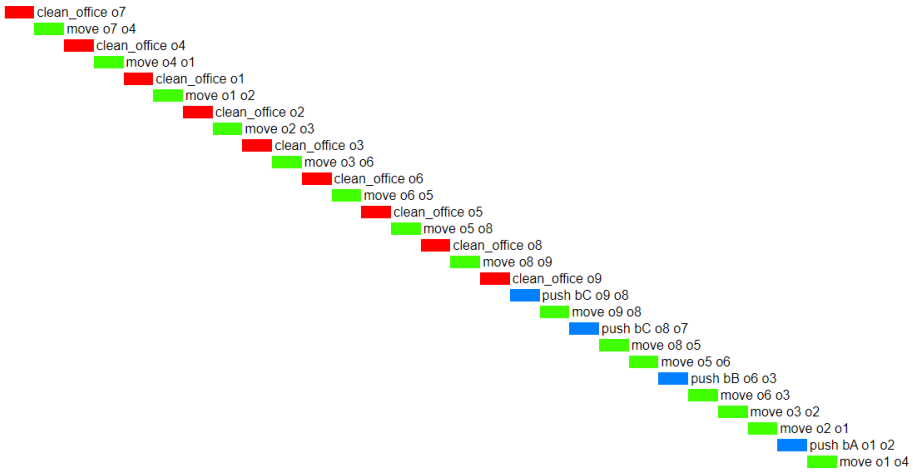

Figure 2: Plan found for case 1

4.2 Case 2: Smaller search space

In this case, I modify the search space so that it is smaller in size, which also means there are less boxes. In total, we have 4 rooms and 1 box. In Figure 3, you can see the visual representation.

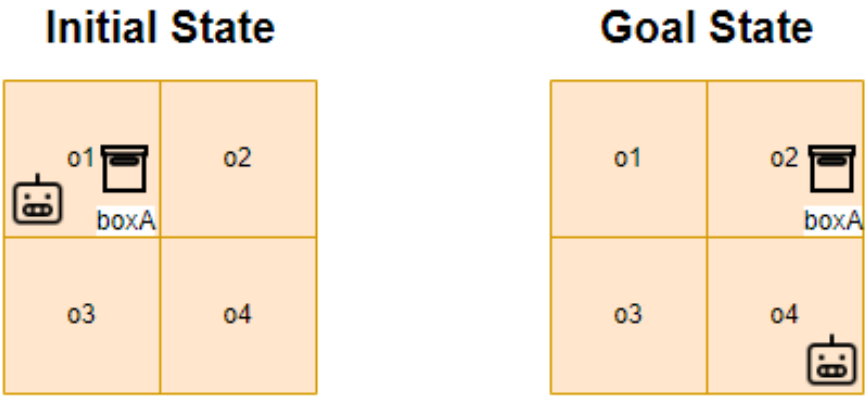

Figure 3: Visual representation for case 2 with the initial state (left) and goal state (right).

Here you can see the written problem.

```
(define (problem smaller_problem) (:domain cleaner_robot)
(:objects
    o1 o2 o3 o4
    bA
)

(:init
    (adjacent o1 o2) (adjacent o1 o3) (adjacent o2 o1) (adjacent o2 o4)
        (adjacent o3 o1) (adjacent o3 o4) (adjacent o4 o2) (adjacent o4
        o3)
    (robot-location o1)
    (box-location bA o1)
    (empty o2)
    (dirty o1) (dirty o4)
    (clean o2) (clean o3)
)

(:goal (and
    (clean o1) (clean o2) (clean o3) (clean o4)
    (box-location bA o2)
    (empty o1)
    (robot-position o4
))
)
```

In Figure 4 you can see the generated plan.

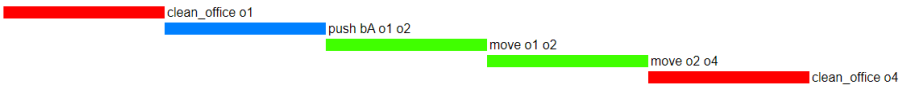

Figure 4: Plan for case 2

4.3 Case 3: Bigger search space

Contrary to last case, in case 3 I modify the search space and make it bigger, also having more boxes. The size of the building is 5x5 and there are 6 boxes. The building is 5x5 with 6 boxes and all rooms are dirty. In Figure 5 we can see the visual representation of the problem.

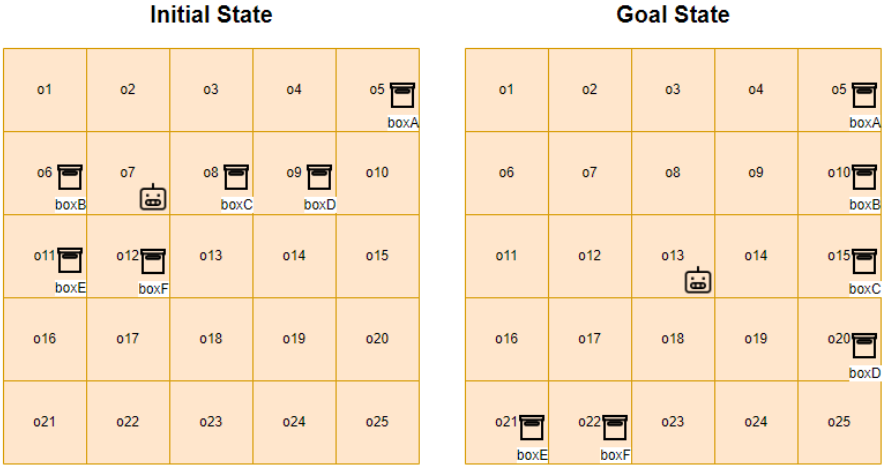

Figure 5: Visual representation for case 3 with the initial state (left) and goal state (right).

This gives us the following written problem.

```
(define (problem bigger_problem) (:domain cleaner_robot)
(:objects
    o1 o2 o3 o4 o5 o6 o7 o8 o9 o10 o11 o12 o13 o14 o15 o16 o17 o18 o19
        o20 o21 o22 o23 o24 o25
    boxA boxB boxC boxD boxE boxF
    )

(:init
    ;todo: put the initial state's facts and numeric values here
    (adjacent o1 o2) (adjacent o2 o1)
    (adjacent o1 o3) (adjacent o3 o1)
    (adjacent o1 o6) (adjacent o6 o1)
```

```
(adjacent o2 o3) (adjacent o3 o2)
(adjacent o2 o7) (adjacent o7 o2)
(adjacent o3 o4) (adjacent o4 o3)
(adjacent o3 o8) (adjacent o8 o3)
(adjacent o4 o5) (adjacent o5 o4)
(adjacent o4 o9) (adjacent o9 o4)
(adjacent o5 o10) (adjacent o10 o5)
(adjacent o6 o7) (adjacent o7 o6)
(adjacent o6 o11) (adjacent o6 o11)
(adjacent o7 o8) (adjacent o8 o7)
(adjacent o7 o12) (adjacent o12 o7)
(adjacent o8 o9) (adjacent o9 o8)
(adjacent o8 o13) (adjacent o13 o8)
(adjacent o9 o10) (adjacent o10 o9)
(adjacent o9 o14) (adjacent o14 o9)
(adjacent o10 o15) (adjacent o15 o10)
(adjacent o11 o12) (adjacent o12 o11)
(adjacent o11 o16) (adjacent o16 o11)
(adjacent o12 o13) (adjacent o13 o12)
(adjacent o12 o17) (adjacent o17 o12)
(adjacent o13 o14) (adjacent o14 o13)
(adjacent o13 o18) (adjacent o18 o13)
(adjacent o14 o15) (adjacent o15 o14)
(adjacent o14 o19) (adjacent o19 o14)
(adjacent o15 o20) (adjacent o20 o15)
(adjacent o16 o17) (adjacent o17 o16)
(adjacent o16 o21) (adjacent o21 o16)
(adjacent o17 o18) (adjacent o18 o17)
(adjacent o17 o22) (adjacent o22 o17)
(adjacent o18 o19) (adjacent o19 o18)
(adjacent o18 o23) (adjacent o23 o18)
(adjacent o19 o20) (adjacent o20 o19)
(adjacent o19 o24) (adjacent o24 o19)
(adjacent o20 o25) (adjacent o25 o20)
(adjacent o21 o22) (adjacent o22 o21)
(adjacent o22 o23) (adjacent o23 o22)
(adjacent o23 o24) (adjacent o24 o23)
(adjacent o24 o25) (adjacent o25 o24)
(dirty o1) (dirty o2) (dirty o3) (dirty o4) (dirty o5) (dirty o6)
    (dirty o7) (dirty o8) (dirty o9) (dirty o10) (dirty o11) (dirty
    o13) (dirty o12) (dirty o14) (dirty o15) (dirty o16) (dirty o17)
    (dirty o18) (dirty o19) (dirty o20) (dirty o21) (dirty o22)
    (dirty o23) (dirty o24) (dirty o25)
(robot-location o7)
(empty o1) (empty o2) (empty o3) (empty o4) (empty o7) (empty o10)
    (empty o13) (empty o14) (empty o15) (empty o16) (empty o17)
    (empty o18) (empty o19) (empty o20) (empty o21) (empty o22)
    (empty o23) (empty o24) (empty o25)
(box-location boxA o5)
(box-location boxB o6)
```

```
    (box-location boxC o8)
    (box-location boxD o9)
    (box-location boxE o11)
    (box-location boxF o12)
)

(:goal (and
    ;todo: put the goal condition here
    (clean o1) (clean o2) (clean o3) (clean o4) (clean o5) (clean o6)
        (clean o7) (clean o8) (clean o9) (clean o10) (clean o11) (clean
        o12) (clean o13) (clean o14) (clean o15) (clean o16) (clean o17)
        (clean o18) (clean o19) (clean o20) (clean o21) (clean o22)
        (clean o23) (clean o24) (clean o25)
    (empty o1) (empty o2) (empty o3) (empty o4) (empty o6) (empty o7)
        (empty o8) (empty o9) (empty o11) (empty o12) (empty o13) (empty
        o14) (empty o16) (empty o17) (empty o18) (empty o19) (empty o23)
        (empty o24) (empty o25)
    (box-location boxA o5)
    (box-location boxB o10)
    (box-location boxC o15)
    (box-location boxD o20)
    (box-location boxE o21)
    (box-location boxF o22)
    (robot-location o13)
))
)
```

And in Figure 6 you can see the generated plan.

Figure 6: Plan for case 3

4.4 Case 4: Modified original case

For this case, I modify the original case to be unsolvable for the planner. To make the problem unsolvable, I added on the goal that 2 boxes should be on the same office, which cannot be achieved. In Figure 7 you can see the visual representation.

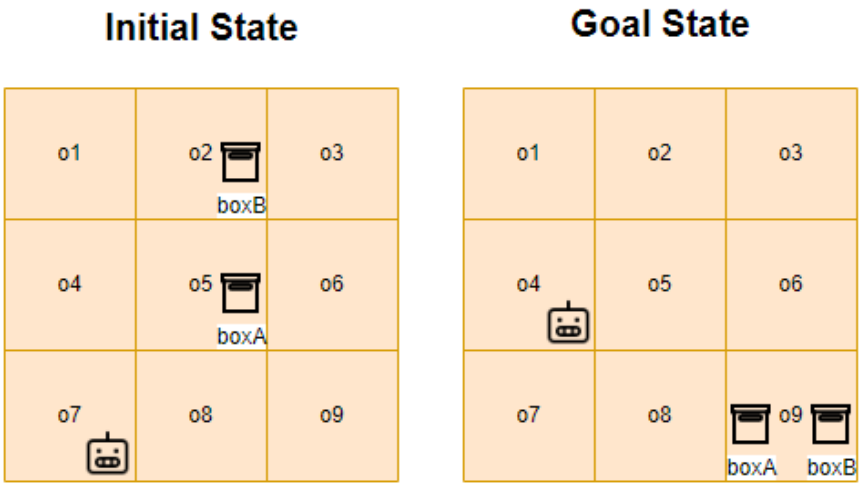

Figure 7: Visual representation for case 4 with the initial state (left) and goal state (right).

```
(define (problem modified_problem) (:domain cleaner_robot)
(:objects
    o1
    o2
    o3
    o4
    o5
    o6
    o7
    o8
    o9
    bA
    bB
)

(:init
    (adjacent o1 o2) (adjacent o1 o4)
    (adjacent o2 o1) (adjacent o2 o3) (adjacent o2 o5)
    (adjacent o3 o2) (adjacent o3 o6)
```

```
    (adjacent o4 o1) (adjacent o4 o5) (adjacent o4 o7)
    (adjacent o5 o2) (adjacent o5 o4) (adjacent o5 o6) (adjacent o5 o8)
    (adjacent o6 o3) (adjacent o6 o5) (adjacent o6 o9)
    (adjacent o7 o4) (adjacent o7 o8)
    (adjacent o8 o5) (adjacent o8 o7) (adjacent o8 o9)
    (adjacent o9 o8) (adjacent o9 o6)
    (robot-location o7)
    (box-location bA o5)
    (box-location bB o2)
    (empty o1) (empty o2) (empty o3) (empty o4) (empty o6) (empty o7)
        (empty o8) (empty o9)
    (dirty o1) (dirty o2) (dirty o3) (dirty o4) (dirty o5) (dirty o6)
        (dirty o7) (dirty o8) (dirty o9)
)

(:goal (and
    (clean o1) (clean o2) (clean o3) (clean o4) (clean o5) (clean o6)
        (clean o7) (clean o8) (clean o9)
    (box-location bA o9)
    (box-location bB o9)
    (empty o1) (empty o2) (empty o3) (empty o4) (empty o5) (empty o6)
        (empty o7) (empty o8)
    (robot-location o4)
))
)
```

4.5 Case 5: Modified domain

Lastly, I create another domain to see how different preconditions could affect the results obtained on previous plans. The preconditions that I changed are the following. The visual reference is the same as in case 1, which you can see in Figure 1

- Clean-office: A room must be empty in order to be cleaned.
- Push: A room must be clean in order to push a box into.

This is the modified domain.

```
    (define (domain cleaner_robot)
    (:requirements :strips)

    (:predicates
        (robot-location ?o)
        (box-location ?b ?o)
        (dirty ?o)
        (clean ?o)
        (empty ?o)
        (adjacent ?o1 ?o2)
```

```
    )

    ; ------ ACTIONS ------
    (:action clean_office
        :parameters (?o)
        :precondition (and (robot-location ?o) (dirty ?o) (empty ?o))
        :effect (and (not(dirty ?o)) (clean ?o))
    )

    (:action move
        :parameters (?o1 ?o2)
        :precondition (and (robot-location ?o1) (adjacent ?o1 ?o2))
        :effect (and (robot-location ?o2) (not (robot-location ?o1)))
    )

    (:action push
        :parameters (?b ?o1 ?o2)
        :precondition (and (robot-location ?o1) (box-location ?b ?o1)
            (clean ?o2) (empty ?o2) (adjacent ?o1 ?o2))
        :effect (and (box-location ?b ?o2) (not(box-location ?b ?o1))
            (not(empty ?o2)) (empty ?o1))
    )
)
```

And in Figure 8 you can see the plan obtained when applied to the problem used in case 1.

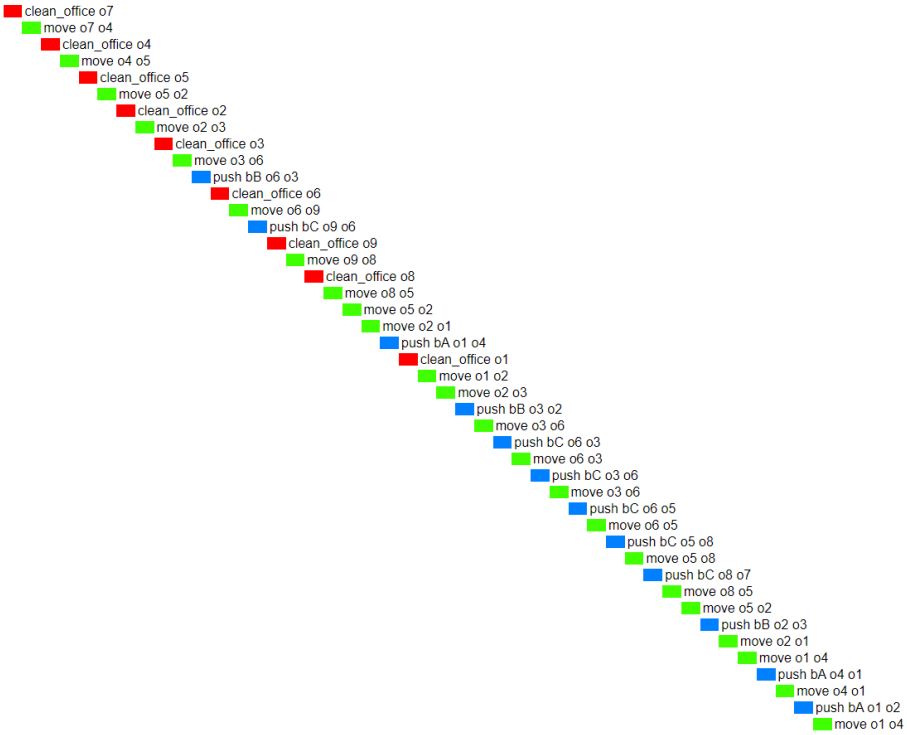

Figure 8: Plan generated for case 5

5 Analysis of the results

Looking at the results obtained in the previous point, we can see many relevant factors of how a plan can vary from another. Looking at Figure 9, we have a plot with the different data that I payed attention to when generating the plans.

- Seconds to Generate: Tells us how long it took to generate the plan.
- Nodes Generated: The quantity of nodes that where generated in the process.
- Nodes Expanded: How many nodes has the algorithm expanded to in the process.

It is clear that the larger the problem is, the longer it takes. What surprised me is case 4, where not having an answer made it took longer compared to the other cases, that is because it had to expand to almost all nodes, when the other cases, did not have to since they found a plan before having to explore all nodes.

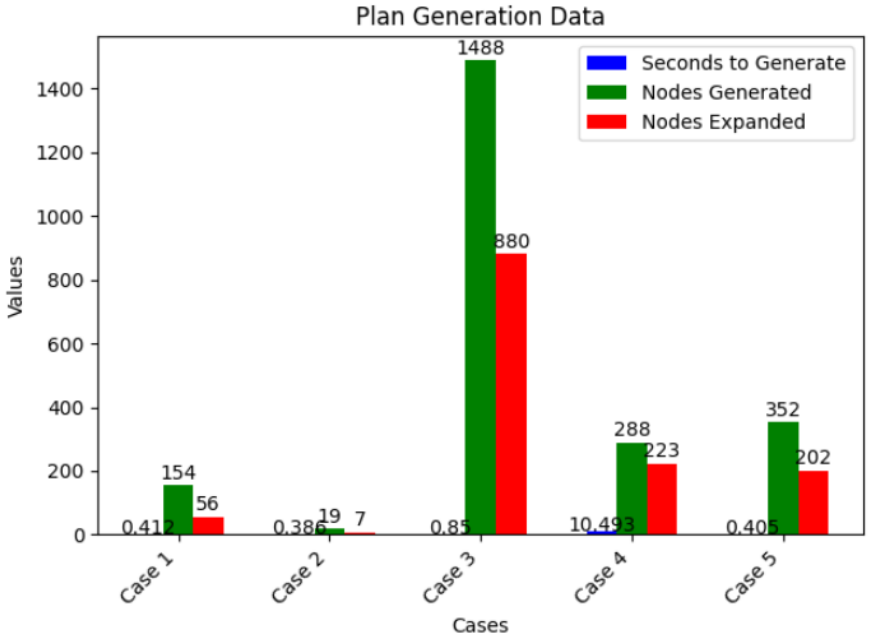

Figure 9: Plotting of the plan generation

On Figure 10, we can see a comparison between the nodes generated and the explored nodes. From this plot we can extract that the most efficient case is the first one. This is because the relation between generated and explored nodes is the lowest between all other cases. This can again be influenced by the size of the building of the problem and the amount of actions that the agent

has to accomplish to solve the problem. We can also see that the least efficient is case 4, from which we had to almost check all nodes to see that there was no possible solution.

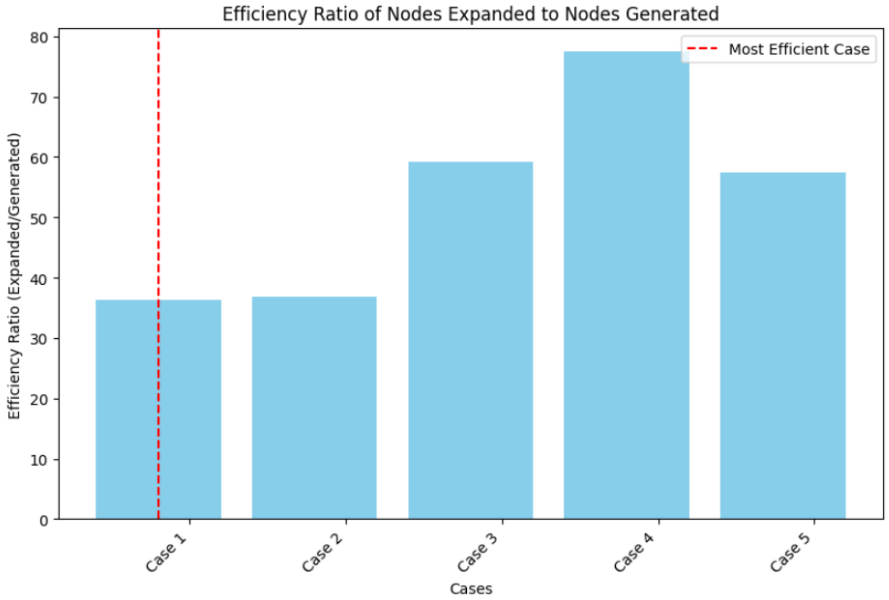

Figure 10: Plotting of the efficiency ratio

Taking into account the results obtained from the plotting, if I add the time it took to generate a plan for each case, we can see in Figure 11 that case 1 is still the most efficient, since it did not take a long time to generate the plan and efficiency ratio was the best, as we saw on the last figure.

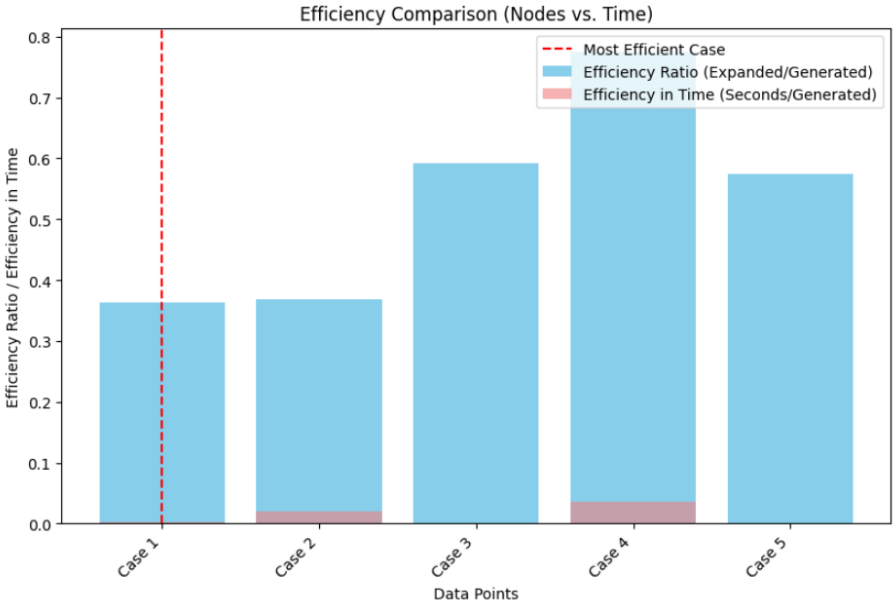

Figure 11: Plotting of the efficiency related to time

6 Conclusion

I can conclude that the optimal solutions for each of the cases was found, meaning that the domain and the problem were described correctly. The cases proposed show relevant results from which I can extract the following conclusions.

First, not all domains are the same. With a small change on an action, creating a different predicate or even putting the predicates in one order or another, can influence the result obtained by the planner. The same can be said about the problem, depending on the way you describe the world the result plan can vary.

Secondly, you have to describe everything concisely, to be sure you do not write extra statements in the problem, making the process take longer.

Lastly, the longer the problem in size and the more tasks the agent, in this case the cleaner robot, has to accomplish, the longer list of actions you will have. I have also realized that if you make a syntax error, the interpreter does not always warn you.