ARTIFICIAL INTELLIGENCE FOR ROBOTICS 2

ASSIGNMENT 2

June 2022

Basit Akram - s5161322

Durga Varun Gangesetti – s5058219

Rakesh Motamarri – s5058220

Soundarya Pallanti – s4807289

1. Introduction

This document is a report on our solution of the second assignment of the Artificial intelligence 2 course. The initial files are provided by Professor Anthony Thomas and the planner are provided through our github repositories https://github.com/popftif/popf-tif. The aim of this assignment is to familiarize with the modelling integrated task and motion planning. The report is divided into two sections: one regarding the PDDL files and one regarding the implementation in cpp.

The first thing to do is to install the popf-tif planner with the instructions given in the above-mentioned link and make sure that the planner is working correctly. Then extract the domain and module folders given along with assignment.

The domain folder consists of the PDDL domain file which of actions and a corresponding problem file where a robot has to visit four regions. The WAYPOINT.TXT file contains the geometric waypoint. The REGION_POSES file contains the mapping from a region to its corresponding waypoint. The LANDMARK.TXT contain landmarks to localize the robot.

The module contains the motion planning files. The CMakeLists.txt is the cmake used for linking the directories to the library. ExternalSolver.cpp is the definition of the interface used for the semantic attachments. ExternalSolver.h is the declaration of the interface used for the semantic attachments. VisitSolver.cpp is the definition of the class implementing the interface. VistSolver.h is the declaration of the class implementing the interface. buildInstructions.txt is the script used to build the library. main.cpp is the main using the solver, can be used for a standalone test executable.

2. PDDL

Initially, the PDDL domain simply implemented the durative action goto-region, which handled both robot movement and cost computation. The PDDL domain file has been changed, with the addition of a new durative operation named localize to calculate the cost of the robot going from one region to another. According to this, the goto-region has also been updated to manage the robot's movement now only.

The (dummy) function, which is generated in CPP scripts and connected to the PDDL, is used to update the function (act-cost), which reflects the cost of a movement. As a result, the planner can devise a strategy that reduces the overall cost of the activity. Two fluents were added to make everything work:

- **(moved ?From ?to - region)** is set to true as the end result of the goto-region durative action, and it is also a precondition for the action localization. The primary goal of this predicate is to initiate the localize action ONLY once the movement has ceased.
- **(localized)** that is a prerequisite for the goto-region action to be set to false as soon as the goto-region action begins as a start effect. It is then set back to true because of the operation localize. This predicate's major aim is to avoid parallelism in action execution and to allow robot mobility ONLY once the robot has located himself. The Problem file has been updated by include the predicate (localized) in the init section, indicating that the robot starts from a known point and may begin moving.

3. CPP

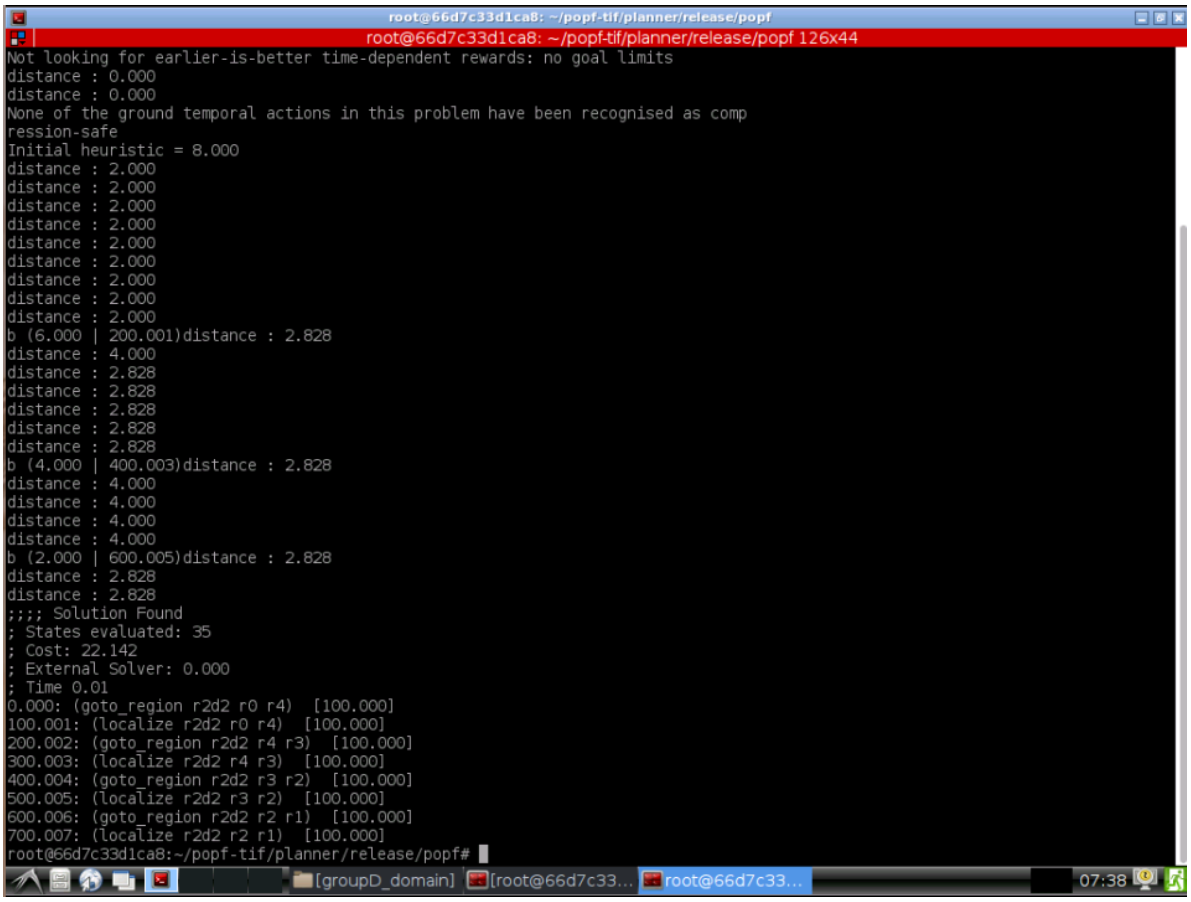
The basic task of the assignment was to compute the Euclidean distance between two regions. We implemented one function in the VisitSolver.CPP file to accomplish this. Furthermore, the VisitSolver.h class has been modified by adding one double-valued variable 'dist' to store the distances calculated at each step. It is the cost of a certain motion between two regions.

- **void Euclidean_dis(string from, string to)** This member function computes the Euclidean distance between the two regions the robot has travelled between by taking the initial and goal arguments from the CallExternelSolver() member function. It extracts the first two values from the given 'waypoints.txt' file and uses them in the Euclidean_dis() member function. The formula to calculate distance is:

$$\mathbf{Dist} = \sqrt{\ } \mathbf{[(x2 - x1)2 + (y2 - y1)2]}$$

4. Conclusion

The results obtained from the planner are as follows. The distance travelled by the robot is calculated using the formula mentioned above.


root@66d7c33d1ca8: ~/popf-tif/planner/release/popf
root@66d7c33d1ca8: ~/popf-tif/planner/release/popf 126x44
[groupD_domain]
[root@66d7c33...
root@66d7c33...
07:38

```
Not looking for earlier-is-better time-dependent rewards: no goal limits
distance : 0.000
distance : 0.000
None of the ground temporal actions in this problem have been recognised as comp
ression-safe
Initial heuristic = 8.000
distance : 2.000
distance : 2.000
distance : 2.000
distance : 2.000
distance : 2.000
distance : 2.000
distance : 2.000
distance : 2.000
distance : 2.000
b (6.000 | 200.001)distance : 2.828
distance : 4.000
distance : 2.828
distance : 2.828
distance : 2.828
distance : 2.828
distance : 2.828
b (4.000 | 400.003)distance : 2.828
distance : 4.000
distance : 4.000
distance : 4.000
distance : 4.000
b (2.000 | 600.005)distance : 2.828
distance : 2.828
distance : 2.828
;;;; Solution Found
; States evaluated: 35
; Cost: 22.142
; External Solver: 0.000
; Time 0.01
0.000: (goto_region r2d2 r0 r4)  [100.000]
100.001: (localize r2d2 r0 r4)  [100.000]
200.002: (goto_region r2d2 r4 r3)  [100.000]
300.003: (localize r2d2 r4 r3)  [100.000]
400.004: (goto_region r2d2 r3 r2)  [100.000]
500.005: (localize r2d2 r3 r2)  [100.000]
600.006: (goto_region r2d2 r2 r1)  [100.000]
700.007: (localize r2d2 r2 r1)  [100.000]
root@66d7c33d1ca8:~/popf-tif/planner/release/popf#
```