Artificial Intelligence for Robotics II

Assignment 2 report

Matteo Maragliano, Mattia Musumeci, Davide Leo Parisi,
Daniele Martino Parisi, Sara Sgambato

27 June 2022

Scenario

Abstract

The **Planning Domain Definition Language (PDDL)** is an attempt to standardise Artificial Intelligence planning languages. But, in the real world context, it is sometimes necessary to compute complex non-linear equations and the syntax that PDDL provides is not suited enough to express these computations because it allows only basic operations between numeric fluents.
To overcome this problem, a generic PDDL planner can be connected to a **specialised advisor** which equips the planner with the ability to carry out sophisticated mathematical computations.
This solution is referred to as **Semantic Attachment in PDDL** and, more precisely, it allows to evaluate fluents using externally specified functions. Obviously, it is necessary to implement an interface between the planner and the specialised advisor.

Objective

The aim of this assignment is to familiarize with modeling **Task Motion Planning (TMP)** problems. By definition, TMP is the problem of planning for a robot that operates in environments containing large number of objects, taking actions to make the robot move trough the world and changes the state of objects. Therefore, these problems contain elements of discrete task planning, discrete-continuous mathematical planning and continuous motion planning. The vast majority of these problems intrinsically contain complex equations that need to be evaluated to obtain a suitable solution.
There are many different approaches to Task Motion Planning problems and, in this assignment, we discuss one of the possible solutions which is Semantic Attachment in PDDL.

Problem definition

As model for the problem we consider a "coffee shop" scenario in which a single waiter robot needs to serve orders to the correct tables in an optimal way, such as keeping into account the distance travelled. This requires choosing a certain sequence of tables to deliver the orders to. Thus, deciding the discrete sequence of table visits require reasoning about the distance travelled by the robot.

Realization

In this context, we define the environment the robot can explore as divided into **regions** each containing one or more **waypoints** which contain information about the position and orientation in which the robot need to be to deliver an order. Notice that, only for sake of brevity of the assignment, we consider a problem in which there are only five regions and each region is characterized only by one waypoint.
We consider that the robot starting at a certain region in the environment.
The robot has to navigate through five different regions. In practice, the robot can be anywhere around the table while delivering the orders.

To solve this problem we used the **popf-tif** planner which can be found at this repository. To adapt the planner to our problem, we modified the external module, which is written in $C++$ code, to obtain the cost of the path (defined as Euclidean distance).

Implementation

The goal was to modify the value of the variable which takes into account the cost of the plan by considering the distance between the positions of waypoints. We are given two text files:

- *region_poses*: which associates each region to its waypoints.
  Given the fact that in our problem each region was associated with only one waypoint, this file simply specifies that region r[i] is associated to waypoint w[i].

- *waypoint.txt*: which specifies the posture of the *i*-th waypoint in the region.
  The posture is composed of a position (x, y) and a orientation θ into the form $\begin{bmatrix} x & y & \theta \end{bmatrix}$.

The function implemented takes the value of the start region and the end region of the path step and implements the computation of the Euclidean distance between this two waypoints.
The Euclidean distance used does not keep into account the posture of the robot, as stated above the robot can be anywhere around the waypoint. The resultant formula is:

$$dist = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$$

Notice that this formula takes into account only two dimensional points because the robot is operating in a 2D environment. With this implementation we succeed in updating the total cost of a path through a value that depends on the chosen waypoints.

Results

Given the fact that the total cost of a plan is composed only of the sum of the single path steps, it is possible to demonstrate in a simple way that the implementation is working as expected.
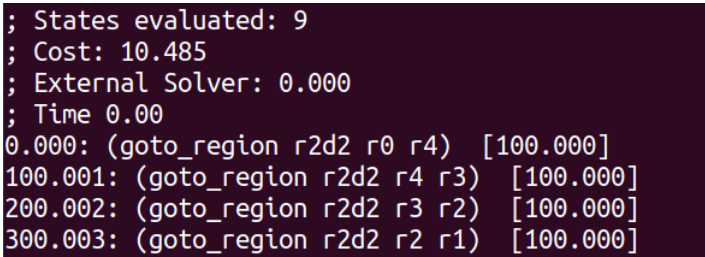
The repository of this assignment which can be found at the following link:

<div align="center">

https://github.com/mmatteo-hub/AIRO2_Assignment2

</div>

The result can be obtained by running the two scripts:

1. *build.sh* : helper script to guide the installation and building of the necessary tools.

2. *run.sh* : runs the planner and prints the solution in console.

Which is the following:


```
; States evaluated: 9
; Cost: 10.485
; External Solver: 0.000
; Time 0.00
0.000: (goto_region r2d2 r0 r4)  [100.000]
100.001: (goto_region r2d2 r4 r3)  [100.000]
200.002: (goto_region r2d2 r3 r2)  [100.000]
300.003: (goto_region r2d2 r2 r1)  [100.000]
```

Therefore, we can observe that the cost obtained for the plan is **10.485**.

Remember that we have the following the following environment:

$r0 \rightarrow wp0 = [0, 0, 0]$

$r1 \rightarrow wp1 = [2, 0, 0]$

$r2 \rightarrow wp2 = [0, 2, 1.57]$

$r3 \rightarrow wp3 = [-2, 0, 3.14]$

$r4 \rightarrow wp4 = [0, -2, -1.57]$

Therefore, the total cost can be manually computed as:

$$cost = dist(wp0, wp4) + dist(wp4, wp3) + dist(wp3, wp2) + dist(wp2, wp1) \approx$$

$$\approx 2 + 2.828 + 2.828 + 2.828 \approx 10.485$$

Which is exactly the result found by the planner.