UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY,
BIOENGINEERING, ROBOTICS AND SYSTEM ENGINEERING

ARTIFICIAL INTELLIGENCE FOR ROBOTICS 2

Second Assignment

AI Planning

Authors:

Borelli Simone - s4662264
Carlini Massimo - s4678445
Forni Matteo - s4673648
Gavagna Veronica - s5487110

Professors:

Fulvio Mastrogiovanni
Anthony Thomas

June 23, 2023

1 Abrstract

The following report for the AIRO2 course will describe how we have efficiently modelled, as much as possible, a robot task for managing the collection of assignments in specific regions.
An interface is implemented for communication between the PDDL planner and the specialized advisor. The research explores Task Motion Planning (TMP) in a 2D environment, where a robot collects any two assignment reports from four student groups and delivers them to a submission desk. Regions and waypoints create a roadmap and the objective is to minimize motion costs by selecting efficient paths with the lower cost. The higher the value of the number of connections for each waypoint (k), the higher the probability of having direct paths between the regions corresponding to the lowest possible cost.

2 Introduction

In order to standardize planning languages in the field of Artificial Intelligence, the Planning Domain Definition Language (PDDL) was introduced. However, practical situations often involve the computation of complex non-linear equations that cannot be adequately expressed using PDDL's limited syntax, which only allows basic operations between numeric fluents.

To overcome this limitation, a solution known as Semantic Attachment in PDDL integrates a generic PDDL planner with a specialized advisor. This integration empowers the planner to perform advanced mathematical computations by evaluating fluents using externally specified functions. To achieve this, an interface between the planner and the specialized advisor needs to be implemented.

The objective of this assignment is to familiarize ourselves with Task Motion Planning (TMP) problems and their modelling. TMP involves planning for a robot in environments with multiple objects, requiring the robot to navigate and manipulate objects to accomplish tasks. These problems encompass discrete task planning, discrete-continuous mathematical planning, and continuous motion planning. Solving TMP problems often entails handling complex equations to find appropriate solutions.

Among the various approaches to TMP problems, we will explore one specific solution: Semantic Attachment in PDDL.

3 Assignment Scenario

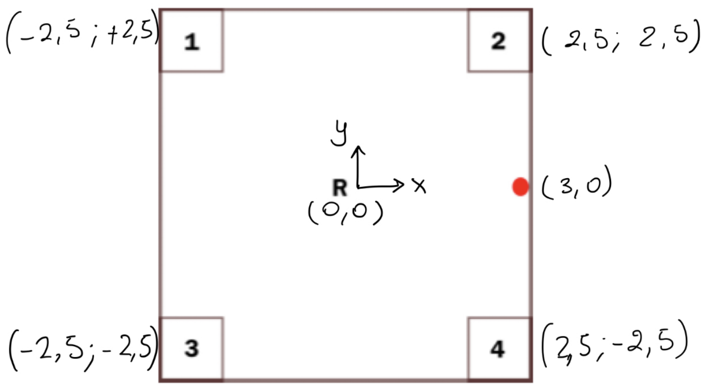


Figure 1: Assignment scenario

In the given 2D environment (Figure 1), Robot R is assigned the task of collecting two assignment reports from four student groups and delivering them to the submission desk (highlighted in red).
The environment measures 6m×6m, with R initially positioned at (0,0) in region 0, and the submission desk located at (3,0) in region 5. Each region is associated with a specific 1m×1m waypoint (x, y). Additionally, 24 waypoints are randomly sampled outside the regions and connected to form edges. The maximum number of connections for each waypoint, denoted as parameter k, can be set by the user.
The connection of sampled waypoints, region waypoints, R's initial position, and the submission desk creates a node-based roadmap. Robot R's objective is to collect any 2 assignment reports, minimizing motion cost, calculated by summing the Euclidean distances between traversed waypoints.

4 Implementation

Within this scenario, we define the exploration environment for the robot as divided into regions, each containing one or more waypoints that provide information about the necessary position and orientation for order delivery. It is important to note that, for the sake of assignment brevity, we consider a simplified case with only six regions, each characterized by a single waypoint.

The robot's initial location is within a specific region of the environment. The objective is for the robot to navigate through the five distinct regions, regardless of r0 which is the initial position while having the freedom to move anywhere around the table during the delivery process.

To address this challenge, we used the popf-tif planner. In order to tailor the planner to our specific problem, we made modifications to Visitsolver and the main files, which are written in C++ code. These modifications enabled us to calculate the path cost (defined as the sum of Euclidean distances between nodes), generate 24 randomly positioned waypoints, and connect them according to the specifications outlined in the assignment.

Our goal is to calculate the cost of possible paths using the Dijkstra algorithm and find the minimum cost path for the robot to move from one region to another. The following files gather the necessary data for this task:

- *waypoint.txt*: This file contains the 24 randomly generated waypoints, plus the six shown in Figure 1. Hence in the file, there are 30 waypoints. The ones that are generated randomly are used for constructing the graph through which the robot will find the optimal path. Each waypoint consists of coordinates (x, y) and an orientation theta (always set to 0 as it is not necessary for calculating the Euclidean distance).
- *region_poses*: The six initial regions, including the starting point, delivery region, and four corner regions of the scenario, are associated with specific waypoints as shown in Figure 1.

The implemented functions serve to generate the random 24 waypoints. Subsequently, we construct the graph using adjacency and distance matrices. In the distance matrix, we collect the Euclidean distance values between a node and its connected nodes. To calculate the Euclidean distance, we refer to the following formula, considering that we are working in a 2D scenario and using the x and y coordinates:

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

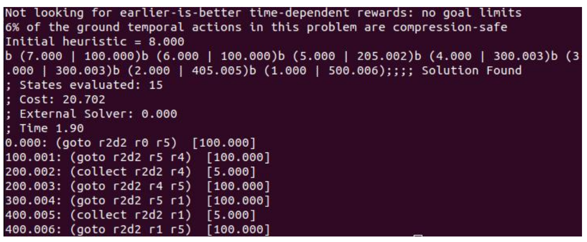
Once the distances are calculated, we take the input of the start region and end region, compute the total cost for each possible path, and select the path with the minimum cost. The found value is then assigned to the variable representing the cost and transmitted to the PDDL.

Additionally, to accomplish the objective of collecting the assignments, a durative action has been added to the domain file.

5 Results

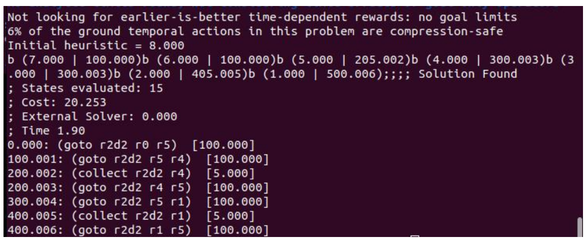
The repository of this assignment can be found at the following link:
`https://github.com/VeronicaG24/assignment2_AIRO2.git`

The following results are given by setting the parameter k = 7 in the first case and k = 15 in the second case:



```
Not looking for earlier-is-better time-dependent rewards: no goal limits
6% of the ground temporal actions in this problem are compression-safe
Initial heuristic = 8.000
b (7.000 | 100.000)b (6.000 | 100.000)b (5.000 | 205.002)b (4.000 | 300.003)b (3
.000 | 300.003)b (2.000 | 405.005)b (1.000 | 500.006);;;; Solution Found
; States evaluated: 15
; Cost: 20.702
; External Solver: 0.000
; Time 1.90
0.000: (goto r2d2 r0 r5)  [100.000]
100.001: (goto r2d2 r5 r4)  [100.000]
200.002: (collect r2d2 r4)  [5.000]
200.003: (goto r2d2 r4 r5)  [100.000]
300.004: (goto r2d2 r5 r1)  [100.000]
400.005: (collect r2d2 r1)  [5.000]
400.006: (goto r2d2 r1 r5)  [100.000]
```

(a) k = 7



```
Not looking for earlier-is-better time-dependent rewards: no goal limits
6% of the ground temporal actions in this problem are compression-safe
Initial heuristic = 8.000
b (7.000 | 100.000)b (6.000 | 100.000)b (5.000 | 205.002)b (4.000 | 300.003)b (3
.000 | 300.003)b (2.000 | 405.005)b (1.000 | 500.006);;;; Solution Found
; States evaluated: 15
; Cost: 20.253
; External Solver: 0.000
; Time 1.90
0.000: (goto r2d2 r0 r5)  [100.000]
100.001: (goto r2d2 r5 r4)  [100.000]
200.002: (collect r2d2 r4)  [5.000]
200.003: (goto r2d2 r4 r5)  [100.000]
300.004: (goto r2d2 r5 r1)  [100.000]
400.005: (collect r2d2 r1)  [5.000]
400.006: (goto r2d2 r1 r5)  [100.000]
```

(b) k = 15

Figure 2: Results showing the cost

Therefore, we can observe that the cost obtained for the plan is 20.702 in the first case and 20.253 in the second case. Considering that the overall cost of a plan solely consists of the cumulative sum of individual path steps, the more the value of k is increased, the higher the probability of having a direct connection between the waypoints associated with the regions, resulting in a cost equal to that of the direct path.