Artificial Intelligence II: Second Assignment

Students Claudio Tomaiuolo (5630055)
Nicholas Attolino (5656048)
Teodoro Lima (5452613)

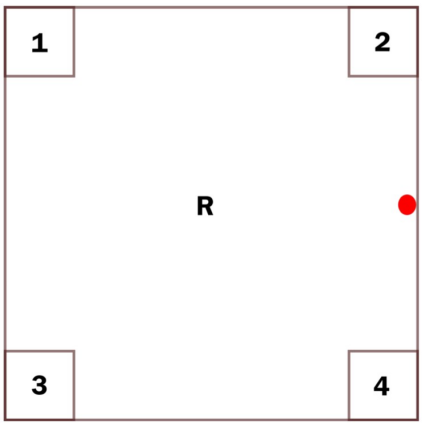
Abstract

In this paper, we analysed the efficiency of the Task Motion Planning (TMP), which is the integration of task planning and motion planning. It links the planning in discrete space and continuous space. Our work has the task of calculating the best path to collect 2 assignment reports from the regions and deliver them to the submission desk. The path calculation is performed by an external C++ solver within which an A* algorithm is used; in addition, way-points have been defined through which the robot must pass to reach the goal.

Introduction

Robotics and Artificial Intelligence are among the current issues that are posing the great challenges for our times and society. The future of health, work and mobility are closely interwoven with the development of robotics and AI[1]. Developments in the field of high-tech are already changing various environments and scenarios, such as education, industry, farming, as well as services like banking. There have been robots that have played chess and won. There have been artificial intelligences programmed to read the emotions on a stranger's face, to make decisions on the financial markets, to generate medical diagnoses and to reason based on common sense. They have often succeeded, beyond the very hopes of their engineers[2].

Environment and constraints



There are four student groups, each assigned to regions 1-4, working on their assignments. Robot R is responsible for collecting the assignment reports and delivering them to the submission desk (red in figure). The dimension of the environment is 6m×6m. R is initially at (0,0) and the submission desk is at (3,0). For the sake of simplicity, assume each region is of size 1m×1m and associate a single way-point (x,y) to each region. Randomly sample 24 way-points (x, y) from the environment, outside the regions. The sampled way-points are connected to form edges. This results in a roadmap with way-points as nodes. Finally, the 4 way-points associated with each region, the initial location of R, and the submission desk are to be connected to the nearest nodes in the roadmap.

Problems to solve

The objective for robot R is to collect any 2 assignment reports and bring them to the submission desk while minimizing its motion cost. The motion cost is the length of the path covered by the robot. The path length is computed by adding the Euclidean distance between the way-points (edge costs) traversed by the robot.

Implementation

To solve the task, we used the **popf-tif** planner which allows us to link C++ code with PDDL. Most of the work has been done on the **VisitSolver** class and on the PDDL files. First, we assigned a way-point to each region (from wp0 to wp5), then we generated 24 random way-points, within the range of the arena width and height. These way-points are generated excluding the areas of the 4 tables, the initial position and the goal points as possible locations. In addition, each way-point has the list of way-points connected to the same. The distance between two way-points is identified with the Euclidean distance equation and finally, we computed the shortest path to go from one region to another using the **A*** algorithm, which is one of the most efficient to find the shortest path between two nodes. Note that the orientation angle of the robot is not considered for the assignment.

A* algorithm

This algorithm can be divided into the following phases:

1. FIrst, the planner calls the external module;
2. The external module takes as input the starting region of the robot and those in which the robot is to go;
3. Translation occurs through the functions *parseRegions* and *parseWaypointconn*;
4. Once the start and finish way-points are obtained, it starts the *A** algorithm that explores the tree of all possible nodes, and thanks to the heuristic function **h**, which in the module is the second variable contained for each way-point in the *open* or *close* table, we are able to obtain an optimal path at minimum cost;
5. Finally, it returns the path, which is plotted and written to the *path.txt* file; after which it returns the control to the PDDL, which repeats these steps until the solution is reached.

Conclusion

```
path found
wp1
wp22
wp2
b (3.000 | 300.003)path found
wp5
wp15
wp23
wp2
b (2.000 | 400.005)path found
wp2
wp23
wp15
wp5
b (1.000 | 500.006);;;; Solution Found
; States evaluated: 15
; Cost: 41.265
; External Solver: 0.000
; Time 0.01
0.000: (goto_region r2d2 r0 r5)  [100.000]
100.001: (goto_region r2d2 r5 r1)  [100.000]
200.002: (take_assignment r2d2 r1)  [0.001]
200.003: (goto_region r2d2 r1 r5)  [100.000]
300.004: (goto_region r2d2 r5 r2)  [100.000]
400.005: (take_assignment r2d2 r2)  [0.001]
```

The image above shows the result obtained. In this work, it is analysed the efficiency of a Task Motion Planning used to solve the task, having identified the A* algorithm as the best for calculating the optimal path, found in 0.01 seconds.

References

[1] Henry A Kissinger, Eric Schmidt, Daniel Huttenlocher. (2021) *The Age of AI: And Our Human Future*. John Murray Press.

[2] Pascal Chabot (2017). *ChatBot le robot. Drame philosophique en quatre questions et cinq actes*. Presses Universitaires de France/Humensis.