**Artificial Intelligence for Robotics 2 – Assignment 2**

Rivi Alice, ID: 5135011

Scamarcia Angelica, ID: 5290802

Vaselli Alessandro, ID: 5320822

This coursework makes use of the popf-tif planner, which u can find more details about here - <https://github.com/popftif/popf-tif>.  
  
**Introduction**

The aim of this assignment is to get familiar with modelling integrated task and motion planning problems. Let’s consider a scenario where a waiter robot needs to serve the orders in an optimal way in a bar (example, minimizing the distance travelled). This might require choosing a certain sequence of tables to deliver the orders. Thus, deciding the discrete sequence of table visits require reasoning about the distance traversed by the robot.

For this assignment we were required to compute the Euclidean distance travelled between two regions for the engine to be able to use it as the metric for the optimization of the plan.

The project is organized in 3 folders:

* **popf-tif** is the planner folder
* **visits\_domain** contains the pddl files and other useful files for localization, that is, the *region\_pose*s file contain the mapping from a region to its corresponding way point, the *waypoint.txt* file contain the geometric way-point locations for the four regions to visit as well as the starting region and the landmark.txt file contains landmarks to localize the robot
* **visits\_module** contains the external module files useful for the needed computation.

**dom1.pddl**

The functions triggered, act-cost, dummy are the ones that involves the external module or semantic attachment part. (more infos about them in the external modules .cpp files)

The action *goto\_region* (which was already given) models only the fact that the robot has reached a particular region, it doesn’t compute the distance. We defined therefore the *compute\_cost* action which caculates the cost, that is in fact, the distance travelled. We used then the *triggered* function to communicate the start and goal region to the external module (since computing the distance requires of course start and goal locations).

Predicates

* + *robot\_in* – robot in a certain location
  + *visited*  - robot has visited a certain region.
  + *path* - to have a track of the path travelled.

Functions

* + *drive* which returns T if the value is grater then 1, or F if it’s not. (T = true, F = false)

*PERCHE’ DRIVE E’ MESSO MAGGIORE DI 0 E NON > 1????*

* + *act-cost* which represents the distance traversed by the robot (cost)
  + *triggered* which is used to trigger the path performed by the robot, modifying the value while the robot is moving NON SONO SICURO DI QUESTO.
  + *dummy* is a variable used to update the cost each time the motion is completed from one region to another.  
    (dummy is given a value of 2, but uncommenting the lines in the corresponding .cpp file it can be randomized)

Actions

* + *goto\_region* – already provided durative action that makes the robot travel from one region to another
  + *compute\_cost –* additional durative action to compute the cost when drive is set to

A brief description of predicates, functions, actions, metric and goals is also to be found directly on the .pddl files.

To have a better understanding of how the PDDL code works please take a look at *prob1.pddl* and *dom1.pddl* files

**Running**

**NON HO IDEA SE QUELLO CHE C’E’ SCRITTO QUA SOTTO SIA GIUSTO O MENO, MAGARI ALI NE SAI DI PIU’ DATO CHE SEI STATA TE A FAR PARTIRE IL TUTTO**

The planning used is popf-tif, which needs to be installed together with his package.The package uses the Armadillo tools to optimize matrix computations.

The semantic attachment library can be built going into the directory:

visits/visits\_module/src

Then should be executed the following command:

./buildInstructions.txt

Once the building process completes, the planner can be executed moving the executable file popf3-clp in the visits/visits\_domain and executing the command:

./popf3-clp -x -n -t10 dom.pddl prob.pddl ../visits\_module/build/libVisits.so region\_pose

**Results and Conclusions**

This is how we decided to implement the Euclidean distance computation in the external module file *VisitSolver.cpp*

**Text

Description automatically generated**

And this is an overview on the action we developed to compute the cost

**Graphical user interface, text, application

Description automatically generated**

**NON SONO SICURO CHE IL COSTO SIA GIUSTO, POI DOVREMMO PROVARE A CAMBIARE LE DISTANZE, NEL SENSO, METTERNE 2 VICINISSIME E 2 LONTANISSIME, IN MODO DA ESSERE SICURI CHE EFFETTIVAMENTE OTTIMIZZI IL COSTO**

**This is the output of the terminal that shows how the planner decided to approach the problem:**

**Immagine che contiene testo

Descrizione generata automaticamente**