Introduction to Artificial Intelligence (CS470): Assignment 5

Deadline:

Wednesday 24th May, 2023

Setup

Prerequisites

This assignment requires Ubuntu 20.04 and ROS Noetic to be installed. If you have not installed it yet, please click on the links and follow the steps for further installation. Moreover, you have to have the following (For more details, please refer to our assignment 5 repository):

1. Dependencies for TurtleBot3 packages
2. Simulation packages

Start with your workspace

You will create your own workspace for the assignment. Please copy the `assignment_5` folder into your workspace assignment 5 and build the packages.

1 A* algorithm [65 pts]

You will implement a representative informed search algorithm, A*, to find the shortest path leveraging heuristics. The performance varies depending on the heuristics. You will compare the performance to Dijkstra algorithm that uses a path cost only. Please, fill your code in the blank section following the "`PLACE YOUR CODE HERE`" comments in the `py_astar_planner/src/py_astar_planner/astar.py` file following the sub-problems below.

1.1 A* algorithm implementation [50 pts]

A* algorithm expands the search tree by selecting a node n with the lowest value of an evaluation function $f(n)$, which is the sum of a path cost $c(n)$ and heuristics $h(n)$ (i.e., $f(n) = c(n) + h(n)$) (See details on Algorithm 1). As a testbed of the implemented A* algorithm, you will use a simple path-finding problem in a 20×20 size *grid* map where the goal and initial states will be given. You need to fill in the `astar_planning()` function and run the `astar.py` file.

In your report, you must

- describe (justify within 100 words) your selected heuristics function for this grid-based search problem,
- count the explored nodes at the end of the search (start=[3,3] and goal=[4,18]),
- plot the explored nodes at the end of the search (same as above), and
- plot the obtained paths from a start to a goal. You have to show the results of two scenarios: (i) start=[2, 1], goal=[19, 19] (ii) start=[2, 18], goal=[18, 7].

1.2 Comparison of A-star vs. Dijkstra algorithms [15 pts]

Dijkstra algorithm is another method for finding the shortest path but it uses a heuristic identically equal to 0. You need to fill in the `dijkstra_planning()` function and run the `dijkstra.py` file. Note

Algorithm 1: A* algorithm pseudo code

Input: $G := (V, E)$: a graph, $d : V \times V \to \mathbb{R}$: a distance measure, h: Heuristics, n_{start}: a start node, n_{goal}: a goal node

Output: A *path* from a start node to a goal node.

```
openset := {n_start} // A frontier container.
closedset = {} // Nodes have been visited.
while openset is not empty do
    n_cur := arg min_{n∈openset}  g(n) + h(n) // g is a path cost from n_start
    if n_cur = n_goal then
        break // We found the optimal path from start to goal.
    Remove n_cur from openset and add n_cur to closedset
    for n ∈ successors of n_cur do
        Ignore n if n ∈ closedset
        x := g(n_cur) + d(n_cur, n) // cost − to − n
        Add n with x to openset // update the openset if n is already in.
path = Backtracking(n_cur)
```

that you can obtain the Dijkstra's algorithm by simply removing heuristics from A* code. In your report, you must

- count the explored nodes at the end of the search (same as Problem 1.1),
- plot the explored nodes at the end of the search (same as Problem 1.1),
- plot the obtained path from a start to a goal. You have to consider the same scenarios given in Problem 1.1 above, and
- describe the difference between Dijkstra and A-star methods based on the results.

2 Running A* on Turtlebot3 [35 pts]

The A* has been used for various AI applications including navigation. In this problem, you will adopt the result of the Problem 1.1 for the navigation problem given a simulated mobile robot, Turtlebot3. Throughout the Gazebo simulator, you command your Turtlebot3 finds a global collision-free path from a current location to a goal via RViZ. Then, you can observe a computed path from your A* implementations and the robot's tracking performance. In your report, you must

- attach **three** different obstacle-avoidance paths captured from RViZ, and
- attach a sequence of screen captures that show your robot is tracking a computed path

Note that you can select challenging start-goal configurations as you want for the captures.

Submission Guide

Submission Requirement

Write a report comparing the search algorithms and discussing the performance by attaching the plots in a PDF file. Generate a zip file of your code (i.e., `astar.py` and `dijkstra.py`) and report, then save your zip file as `cs470_yourname_studentID.zip`. Please submit the `.zip` file via KLMS.

Please make sure that the submitted python files have been saved.

Academic Integrity Policy

This is homework for each student to do individually. Discussions with other students are encouraged, but you should write your own code and answers. Collaboration on code development is prohibited.

There will be given no points in the following cases:

- Plagiarism detection
- Peer cheating
- The incompleteness of the code
- The code does not work