Artificial Intelligence Laboratory 2: A*(A star) search algorithm

DT8012 (HT14) Halmstad University
13th of November 2014

Introduction

This lab is designed to introduce you to search algorithms, using one of the most common ones, namely A*. You will implement the A* algorithm, and use it in two applications:

- Path planning: find the shortest path from agent's current position to the goal.
- Poker player: given an opponent with known strategy, find a sequence of actions that maximize your winnings.

Implementation of A*, like many similar graph algorithms, is quite simple in a high level programming language such as Python. Hint: the easiest way to represent a graph is by using lists to store all children of any given node.

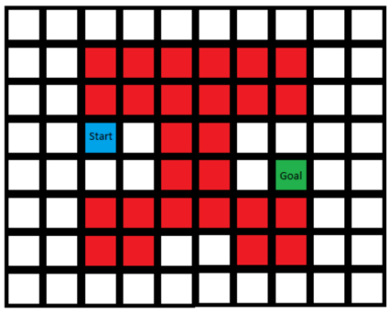
Task 1: A* path planning

In this task you will implement the A* algorithm and use it for path planning problem. The task is to go from a start point to a goal point, under the following conditions:

- Do not cross obstacles
- Possible moves are ↑, →, ↓, ←.

The map is represented as a 2D matrix. Here is an example of how it can look like:

0	0	0	0	0	0	0	0	0	0
0	0	-1	-1	-1	-1	-1	-1	0	0
0	0	-1	-1	-1	-1	-1	-1	0	0
0	0	-2	0	-1	-1	0	0	0	0
0	0	0	0	-1	-1	0	-3	0	0
0	0	-1	-1	-1	-1	-1	-1	0	0
0	0	-1	-1	0	0	-1	-1	0	0
0	0	0	0	0	0	0	0	0	0



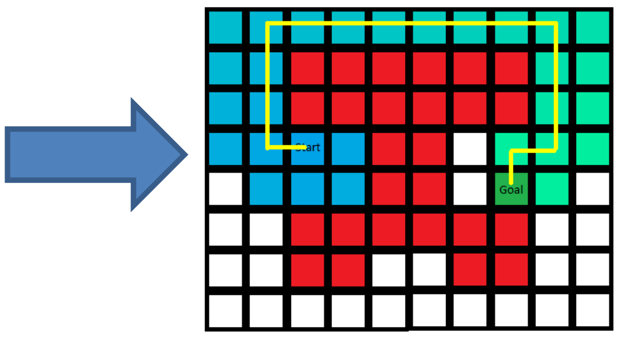
The 0s represent free cells, the -1s represent obstacles (or impassable cells), -2 represents the starting position and -3 represents the goal. In the example, agent has to move from (3, 4) to (8, 5), as shown on the right side figure.

In the **`lab2.zip`** file you can find a library in python that does the following:

- Generates a map (as shown above).
- Plots the result of A* algorithm, coloring expanded nodes (cells) according to their value, and draws the solution found (as shown in figure below).

Given the map, start point and goal point the task is to implement A* to find the (optimal) solution. Use the provided plotting function to analyze the behavior of your algorithm. In particular, make sure that you plot at least the following as the values of the cells: value of the heuristic *h(x)*, cost of moving from starting point to this cell *g(x)*, expansion order (which nodes in the graph were evaluated first), and total cost *f(x)* (movement cost + heuristic value),

0	4	5	6	7	8	9	10	11	12
4	3	-1	-1	-1	-1	-1	-1	12	13
3	2	-1	-1	-1	-1	-1	-1	13	14
2	1	-2	1	-1	-1	0	15	14	15
0	2	1	2	-1	-1	0	-3	15	0
0	0	-1	-1	-1	-1	-1	-1	0	0
0	0	-1	-1	0	0	-1	-1	0	0
0	0	0	0	0	0	0	0	0	0

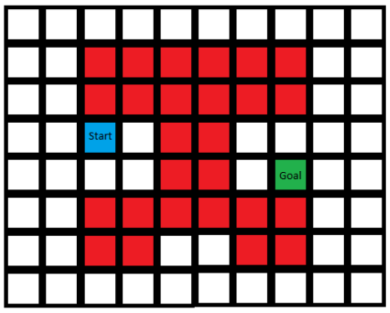


Environment setup

The only required software tool is Python. To get you started with Python, if needed, we recommend Codeacademy and PyCharm.

Task 1a:
Implement A* that finds the shortest path, if it exists, from start point to end point. Try several different heuristics (at least Euclidean and Manhattan distances) and compare how well do they work. Which visualizations show you the effect of heuristic most clearly?
Test your algorithm on a number of random size maps, with randomly placed obstacles as well as a start and goal points.

Task 1b:
The map generation function can also ensure that obstacles are always blocking the middle of the map, i.e. that they are placed in such a way that you can only pass from left half of the map to the right half of the map near the top or bottom edge. In this setup, the starting point is always on the left side, and the goal is always on the right side.
An example of how such a map could look like:



Your task here is to come up with a new, specialized heuristic, that is better adapted to this particular situation, and allows A* to find the optimal path faster.
As before, use the visualization to analyze the effect of the chosen heuristic.

Task 2: Poker Bidding

Introduction

In the `lab2.zip` file you can find a library implementing a deterministic strategy for a poker agent. It takes player’s actions as arguments and returns agent’s response. Have a look at this function and understand the strategy.

Task 1a:
Given complete information (regarding agent’s cards, how many cards they will discard and what will be their draws… and thus what actions the agent will make), use your A* implementation to come up with a plan that maximizes your winnings (for a single hand).
Explore a few suitable heuristics for this problem.

Task 2b:
Given complete information, as before, use your A* implementation to maximizes your winnings over 10 (random but known up-front) hands.

Grading criteria

- Grade 5: Complete all the tasks above (within the 2 weeks deadline).
- Grade 4: Complete one of the task above, and partially the other task (within the 2 weeks deadline).
- Grade 3: Complete at least one of the tasks above.