15-887 Homework 2
v2

Assigned on: 10/05/16
Due date: 10/19/16

1 Moving Target

Problem Description

In this problem a point robot has to catch a moving target. Both the robot and the target live in a ***N*x*N* 4-connected 2D gridworld**, as depicted in Figure 1. This means they can only move up, down, left or right. Additionally, both the robot and the target can also decide to remain in the current position. However, neither the robot nor the target can move through the outer walls of the grid.

Each cell in the gridworld is associated with the cost of visiting it. This cost is an integer that ranges from 1 to 1,000. Note that when the robot chooses to remain in the same cell, it will pay the cost of the visit again.

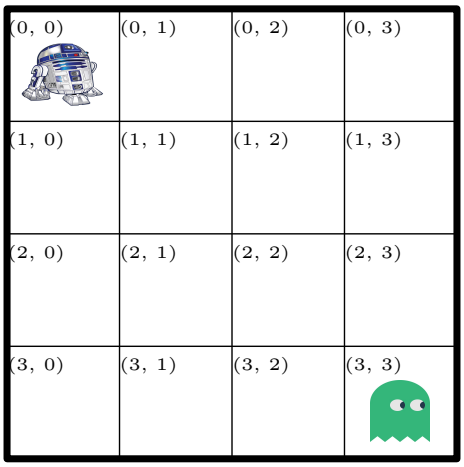

Figure 1: The robot and the target in the 2d gridworld.

The robot knows in advance the predicted trajectory of the moving target, as a sequence of positions in the grid (for example: (3,3), (2,3), (2,2)). The first element of this trajectory is the initial position of the target at time step 0. Both the robot and the target move at the speed of one cell per timestep.

Note: After the last cell on its trajectory, the target disappears. So, if the given target's trajectory is of length 40, then at timestep = 40 (assuming timesteps begin with 0) the object disappears and the robot can no longer catch it.

In this assignment you will be writing two planners to help the robot catching the moving target. The task of the planners is to generate the path the robot should follow in order to catch

the target. In other words, **this path should make sure that the robot arrives at one of the cells visited by the moving target either before or at the exact same timestep the target gets there**.

Implementation Details

We should be able to run your planners with the following commands:
./moving-target-a <problem file> and *./moving-target-b <problem file>*, for the planner of questions a) and b), respectively.

The input file will be of the format:
N
<width/height of the grid>
R
<initial position of the robot (timestep 0)>
T
<position of the target at timestep 0>
<position of the target at timestep 1>
...
B
<costs of the cells on the first row of the grid>
<costs of the cells on the second row of the grid>
...
<costs of the cells on the Nth row of the grid>

Check the links below for some examples

- **Problem 0** `http://www.cs.cmu.edu/~mmv/planning/homework/Homework2/problem_0.txt`
- **Problem 1** `http://www.cs.cmu.edu/~mmv/planning/homework/Homework2/problem_1.txt`

Note that all positions are given in the format *row, column* and start in 0.

Your planner must be able to read and interpret these problem files, compute the plan and output the solution path. The path planned should be outputted to the screen (printed) in the format:
<cost of path>
x_0, y_0
x_1, y_1
...
where the first line contains the cost of the path, and the following lines indicate the positions (row, column) the robot should follow, starting at timestep 0.

Note: You can expect N to be on the order of 1,000 or so.

a) [25 pts]

Planner Write a planner that allows the robot to catch the moving target. Specifically, the planner must be able to compute the **least cost path that allows the robot to catch the moving target**. Note that your grade will depend on the quality of the planner.

Explanation Provide an explanation for the approach you used for this planner.

Summary Run Problem 1 (listed above) and present a summary on the performance of the planner in terms of *execution time*, *cost of the path* and *number of states expanded during the search.*

b) [25 pts]

Planner This time, write a planner that is **computationally faster, at the expense of optimality of the solution path computed**. Your planner should be able to run the examples provided in a few seconds. Make sure there is a significant difference in terms of the execution time and number of states expanded during the search, when compared with the previous approach. Again, note that your grade will depend on the quality of the planner.

Explanation Provide an explanation for the approach you used for this planner.

Summary Run Problem 1 (listed above) and present a summary on the performance of the planner in terms of *execution time*, *cost of the path* and *number of states expanded during the search*.

Comparison Briefly discuss and compare the performance achieved by both planners. In the discussion, refer possible scenarios where you expect each planner to perform worse.

2 Planning with Heuristics

2.1

Suppose you have two consistent heuristic functions: h_1 and h_2.

a) [10pts]

Prove that $h(s) = \min(h_1(s), h_2(s))$ for all states s in the graph is also a consistent heuristic function. (Someone asked this question in class. So, credits for this question go to this student.)

b) [10pts]

Prove that $h(s) = max(h_1(s), h_2(s))$ for all states s in the graph is also a consistent heuristic function.

2.2 [10pts]

If you print out the f-values of states that get expanded by A* with consistent heuristics in the order of their expansions then this sequence will have (mark ONE that is true):

a. The same f-values across all the states

b. Monotonically non-decreasing sequence (e.g., $f(s) \leq f(s')$ if s is expanded before s')

c. Monotonically increasing sequence (e.g., $f(s) < f(s')$ if s is expanded before s')

d. Monotonically non-increasing sequence (e.g., $f(s) \geq f(s')$ if s is expanded before s')

e. Monotonically decreasing sequence (e.g., $f(s) > f(s')$ if s is expanded before s')

f. None of the above

2.3 [10pts]

If you print out the f-values of states that get expanded by **weighted** A* with consistent heuristics inflated by $w > 1$ in the order of their expansions then this sequence will have (mark ONE that is true):

a. The same f-values across all the states

b. Monotonically non-decreasing sequence (e.g., $f(s) \leq f(s')$ if s is expanded before s')

c. Monotonically increasing sequence (e.g., $f(s) < f(s')$ if s is expanded before s')

d. Monotonically non-increasing sequence (e.g., $f(s) \geq f(s')$ if s is expanded before s')

e. Monotonically decreasing sequence (e.g., $f(s) > f(s')$ if s is expanded before s')

f. None of the above

2.4 [10pts]

Suppose you are running MHA* with N inadmissible heuristics $h_1 \dots h_N$ and one admissible and consistent heuristic h_0 for the Anchor search in MHA*. Mark ALL that are true:

a. MHA* is guaranteed to perform more expansions than A* with heuristic h_0

b. MHA* is guaranteed to perform no more expansions than A* with heuristic h_0

c. MHA* is guaranteed to perform no more expansions than weighted A* with heuristic h_0 inflated by w_1 used by MHA*

d. MHA* is guaranteed to perform no more expansions than two times # of expansions performed by weighted A* with heuristic h_0 inflated by w_1 used by MHA*

e. None of the above