Matthew Konstantinou

9/18/2014

Homework 1

3.9)

1. [image]
2. It is a good idea to keep track of repeated states because any repeated state has a limited number of paths. This is not true in between complete “loops,” a.k.a. the boat going from the left side, to the right, then back to the left.
3. People have a hard time solving the problem because visualizing and keeping track of all possible paths is difficult without help. In code, we are able to keep a hard copy of all states previously visited, as well as quickly analyze all other paths to find the most optimal.

3.2)

1. State space: Five possible states: facing north, facing south, facing east, facing west, and moving forward (one space), and standing still.

The total size of the state space is 6(W\*L), where W is the width, and L is length of the maze. This is true because, assuming the robot moves one step at a time at one unit per step and that after each step it analyzes its new position, it can be in any of the four directions, as well as either moving or stopped.

1. Point of clarification, the robot also has to be able to turn around at a dead end.

The state space is: 4i + 2(W\*L – i), where I is the number of intersections. This is because we only need to turn when at an intersection of two or more corridors, and in any other corridor, we can be facing one direction and moving

1. In a corridor, at an intersection, moving forward, or turning. We don't need to keep track of the orientation because we only ever move forward
2. We assume that the robot knows the state it is currently in, it knows where it has been in the past, and it can only move in one direction

3.3)

1. Each city (i) is a node with a series of neighbors (j). Each city, i, is a distance D(i,j) from its neighbor, j. The goal is to place two searches on the map that will find each other in the shortest amount of distance. At each new city, the searches share their current state, then individualy choose a path to travel.

State Space: the state space is any (i,j), where i is the city of state 1 and j is the city of state 2

Goal: Any (i,j), where i=j

Actions: travel to an adjacent state

1. Best case scenario, both searches pick the shortest paths and meet at the center, meaning that heuristic function iii is admissible
2. Yes, if there are only two cities. Each search would swap places indefinitely
3. Yes. Say you start at city A with a distance of 2 to city B and a distance of 10 to city C. You pick city B, which is distance 13 to city C. It is less expensive, in this case, to travel back to city A, then straight to city C (total distance 14), than it is to travel straight from D to C (total cost 15).

3.18)

1. A tree in which the left-most node at depth d is the goal node