CSI535 – Introduction to A.I. Assignment #1: Direction Finding Using A*

Due: Friday 03/10/05
Worth: 20% of Final Grade
Late Policy: You lose one full grade for each week (including partial weeks) you are late.

Read the instructions carefully, ask questions if you have any doubts.

The A* algorithm is a very useful A.I. technique for direction finding in applications such as robot navigation and computer game character movement (many advancements to A* have been created by the gaming industry). Direction finding involves making the optimal series of actions to move from a start to goal point.

You will be using an environment that consists of a 20 x 20 cell universe where your agent takes up exactly one cell. Your agent can make only one of four movements (back, forward, right, left one block). You will begin by implementing A* to navigate this universe in its most basic form and then make changes to handle progressively difficult versions of the environment. Email your code to davidson@cs.albany.edu. Note that the better your admissible your heuristic the more points you get.

Question 1). Basic version of environment (20 points)

Assume that your agent can only see one cell in all moveable directions. Implement a program to determine the optimum series of actions (shortest distance) for the following map using A*. The heart represents the goal state, the smiley face the start state, the black blocks are borders that cannot be crossed, all non-filled blocks can be passed through with equal cost. Furthermore, the goal cell has a transmitter that can provide the Euclidean distance from the agent’s current position to the goal and the number of barriers crossed. For example the agent in its starting position will receive the information (Distance: 12, Number of Barriers to Cross 4). Note your code will have to mimic this transmitter.

Define the state space, constraints, actions and goal test. Carefully define your admissible heuristic under all conditions and your f and g functions. Draw the first two levels of your data structure (nominate if you intend to use a graph or tree) showing f, g and h values. Present your results by showing the f(n)=g(n)+h(n) values for each node in the map your implementation evaluates. State the number of nodes your agent expands. Present the pseudo code to your algorithm.

Question 2). Environment with elevation (25 points)
All information presented in question 1 still holds except now the environment has a topography to it. In this question you will be using the map below. Some cells are “higher” than others, your agent can climb from one elevation to another at a fixed cost dependent only on the elevation of the destination cell but not on the elevation of the cell the agent is moving from (this is a simplification). As before, the black cells are impossible to go through, the dark gray cells can be traversed through at cost 10, the light gray cells the cost is 5, traversing through all remaining cells cost 1.

Define the changes (if any) to your space state space, constraints, actions and goal test. Carefully define your admissible heuristic under all conditions and your $f$ and $g$ functions. Present your pseudo code. Present your results by showing the $f(n)=g(n)+h(n)$ values for each entry in the following map that your agent investigates. Note the non-uniform cost aspect of A* can be used to handle direction planning to avoid oppositions, pits, collect as much reward as possible etc.

Question 3. Environment with topography and partial access to the environment (25 points)

Now imagine that your agent has unlimited 360 degree vision but the cells with non-zero height block his/her entire view beyond that cell. Define the state space, constraints, actions and goal test. Carefully define your admissible heuristic under all conditions and your $f$ and $g$ functions. Present your pseudo code. Present your results by showing the $f(n)=g(n)+h(n)$ values for each entry in the above map that your agent investigates. Does your agent find the same trail as in question 2?

Question 4. A* With Hints/Clues? (30 points)

You should change your code from question 2 to answer this question.

A* is indeed optimal and optimally efficient as proven in class. However the memory requirements are great as every node, $n$, $f(n) < C^*$ is expanded where $C^*$ is the optimal cost. One option is to prune the fringe is using apriori hints/clues on what cells are on or not on the optimal path. Consider four types of hints: Existential hints (a cell must be on the optimal path), Absence hints (a cell must not be on the optimal path), separate hints (if a cell is on the optimal path then another is not and vice versa), link hints (if a cell is (is not) on the optimal path then another is (is not) on the optimal path).

Assume that you have access to the complete map when setting the hints but the A* algorithm does not. Specify at least two methods of generating different hint types. For the topographical map above list your hints. Change your A* implementation to prune the fringe based on these hints. Present your changed pseudo code. Present your results by showing the $f(n)=g(n)+h(n)$ values for each entry in the
above map that your agent investigates and indicate those cells that are pruned from the fringe. Does your agent investigate fewer nodes than in question 2?

Suggestion: When creating hints you may wish to consider that there are four types of terrain (flat, medium, high and infinite) and specify hints with respect to these terrain types.