Upload your answers document in BlackBoard (Paperless Homework)

Question 1 (Chapter 3, 15 points)

Define in your own words the following terms: state, state space, search tree, search node, goal, action, transition model, and branching factor.

State – A set of sensor data that corresponds to the current iteration that the agent is currently taking or occupying.

State Space – The set of all configurations that an environment and goal setting could possibly achieve.

Search Tree – A hierarchical structure where the nodes correspond to different states. These nodes can be searched in different ways depending on the algorithm used.

Search Node – A part of the search tree that corresponds to a particular state of our environment.

Goal – The solution to our problem. What our algorithm is trying to find in our search tree.

Action – Looking at the agent's current state, the algorithm decides on an action to take. This is the action that moves it to the next state. One action moves the agent from one state to only one next state.

Transition Model – Is a description of what each action does. A mapping or function of the current state and the action that should be taken.

Branching Factor – The number of children that each node contains in a search tree. This number can be consistent or can vary between nodes, depending on the situation.

Question 2 (Chapter 3, 20 points)

Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

(a) Formulate this problem. How large is the state space?

If the robot moves in squares (similar to Wumpus World), we will then set the initial state to be the middle of the maze. The robot can move in the direction it is facing, or rotate to face a different direction. Because there are 4 possible directions for the robot to face, the size of the state space is 4n.

(b) In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?

Now the robot can only turn when it reaches a corridor (represented as i). The state space now becomes 4i. The robot will move when it is not at an intersection (changes directions only at i). Therefore, there are n-I non-intersection spaces. The total state space then becomes 4i + 2(n-i)

Question 3 (Chapter 4, 25 points)

Define in your own words the following algorithms:

(a) **Local beam search** – Search algorithm that randomly selects k states to begin. Then it determines the successors of those states and sees if any are the goal state. If not, it selects the k best successors and continues to search. Considers multiple routes at once.
(b) **Simulated annealing algorithm** – Algorithm doesn't always improve. We need to explore the state space to find an optimum that may be better. This is similar to the hill climbing algorithm, but with occasionally chooses a bad move to hopefully get out of a local minimum.
(c) **Genetic algorithm** – The algorithm mimics human biology and genetics. Successor states are created by combining two parent states. A fitness function determines which sections of the parent states are worth passing to their children. The state is mutated over iterations to get closer to the goal.

Question 4 (Chapter 5, 20 points)

Discuss how well the standard approach to game playing would apply to games such as tennis, pool, and croquet, which take place in a continuous physical state space.

- The standard approach utilizes the minimax, evaluation function, and alpha-beta pruning. It would simply be infeasible at best, but most likely impossible. We cannot search continuous space with methods that separate the state space into discretized chunks. Also, our evaluation functions would need to be modified to handle continuous data.

Question 5 (Chapter 5, 20 points)

Prove the following assertion: For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will never be lower than the utility obtained playing agains an optimal MIN.

- Consider the tree where MIN is selecting terminal nodes. If MIN is suboptimal, then the nodes they select will be greater than or equal to the values it would have selected if playing optimally. Thus, the value that MAX picks can only be increased by MIN playing sub optimally.

Can you come up with a game tree in which MAX can do still better using a suboptimal strategy against a suboptimal MIN?

- I can picture a game of chess where both players are playing sub optimally. I believe this is how most games of chess are played. Player one may play a move which leaves a piece undefended. Player two will not grab the free piece (because they are also playing sub optimally) but they will still end up winning the game.