ASSIGNMENT 1

1. Agents and Environments

1. Robocup

a. What is the appropriate PEAS specification for an agent that solves the problem and what are the features of the environment?

i. Performance Measure(P):

- To detect the other robot soccer players correctly when they are close enough.
- To detect the robots that belong to the same team(cooperation) or not(conflict) accurately.
- To locate the ball and the goal exactly to take necessary action.
- To run fast and shoot the ball correctly towards the goal to maximize the number of wins of their own team.
- To defend the goals from the opponent team to minimize their chances of winning the game.

ii. Environment(E):

- Soccer field or ground.
- Same team and opponent team players.
- Supervisors, referees, scorekeepers and the audience.

iii. Actuators(A):

- Robot body and legs to run and shoot the ball.
- Navigators to move from current location to the desired location.
- Transmitters to communicate with the same team robots players.
- Display screen to communicate with referees, scorekeepers and other players politely.

iv. Sensors(S):

- Motion and Touch Sensors to detect the nearby robots.
- Cameras for 360 degrees view of the ground.
- Receivers to receive live game information like ball passes, assists and goal strategy from the same team robot players.

v. Environment Features:

- Partially Observable - Each player robot's sensors provide it with all the information required to play its game and strategy of its own team but it is not fully aware of what other opponent team player robots are thinking about the game so it is not fully observable.
- Multi Agent - We have two player robots teams here. So one tries to minimize the performance of the other inorder to maximize its own performance and win the game. Hence it is a competitive multi agent environment.
- NonDeterministic - In this case the environment is non-deterministic because the next state of the environment is not completely determined by the current state and action taken by a robot player. For example, a goal shot from a team 'A' robot player may lead to a goal miss due to team 'B' goalkeeper which was instead expected to be a successful goal by team 'A'.
- Sequential - It is a sequential environment because the decision taken at a current state could affect all future decisions. For example, an incorrect ball pass by team 'A' robot player may leave the ball in wrong hands i.e team 'B' player.
- Dynamic - This is a dynamic environment because each robot player and the other robot players (including the same and opponent team) keep moving while the robot with the ball in its current state thinks about what to do next.
- Continuous - Robots playing soccer keep moving continuously all over the field at different speeds to achieve their individual goals and each robot's actions are also continuous, for example capturing data from camera sensors, navigating continuously from one place to another.
- Known - This is a known environment because, even though we are not sure about the other team player's game strategy, we can still come up with a probability of outcomes for the actions performed by a player.

b. What is the least powerful agent that can solve the problem, and why? - utility based

The least powerful agent that can solve this problem is a utility-based agent. Because, here the performance measure is to maximise the

number of goals. And to maximize the goals, we need to assign the scores to any given sequence of environment states, so it can easily differentiate between more and less desirable ways of getting more wins. So, we design a utility function which is essentially an internationalization of performance measures. With that said, the internal utility function and external environment are in agreement with each other which will help the agent to choose actions to maximize its utility (happiness i.e maximize the number of goals).

2. Deep Sea Drone

a. What is the appropriate PEAS specification for an agent that solves the problem and what are the features of the environment?

i. Performance Measure(P):

- Accuracy in the salinity and temperature values recorded at a given set of locations.
- Image and video quality recorded underwater.
- Proper motion of the drone both underwater and above water surface to ensure agent safety.

ii. Environment(E):

- Ocean, water and the other sea animals.

iii. Actuators(A):

- Diver to dive into a pre-specified depth and booster to return to the water surface.
- Steering to change direction, accelerator to move forward and a brake to hold at a specific position.
- Transmitter to transmit data once the drone is returned to the surface.

iv. Sensors(S):

- Depth sensor to determine the depth of the water.
- GPS to lock the position in water.

- Sensors to measure salinity and temperature information at a set location underwater.
- Accelerometer to check the speed under water.
- Collision detection sensor to avoid collisions with underwater animals.

v. Environment Features:

- Fully Observable - The agent’s sensors give it access to the complete state of the environment at each point in time. At each point in time it knows the depth, location, speed and the likeliness of a collision with sea animals which make it completely aware of the environment. Hence this is fully observable.
- Single Agent - Here the agent does all the tasks by itself without depending on any other agent’s actions. So it is clearly a single agent environment.
- Deterministic - This environment is deterministic because the next state of the environment is completely dependent on the current state and the action taken. And also the environment is fully observable and has full knowledge of the environment. For example, In order to transmit the data after returning to the water surface the agent has to successfully dive in, capture the data and then return back to the surface which makes it dependent on the actions performed in the previous state.
- Sequential - It is a sequential environment because the decision performed at a current state could affect all future decisions. For example, if the agent moves forward even after knowing the collision possibility may lead the agent to a collision and self destruction underwater.
- Dynamic - This is a dynamic environment because the animal life underwater and ocean currents keep moving at different speeds while the navigating algorithm of the drone keeps deciding on what to do next.
- Continuous - The sea drone and all the water animals keep moving continuously at different speeds and the drone's actions are also continuous, for example capturing information continuously underwater which include either depth, location, salinity or temperature.

- Known - This is a known environment because the agent knows the outcomes of all its actions and has a clear idea of all the actions to be performed in order to achieve its goal.

b. What is the least powerful agent that can solve the problem, and why?

The least powerful agent that can solve this problem is a goal-based agent. Because, here the deep sea drone has to achieve a specific goal, i.e to transmit salinity and temperature information of a specific location underwater by diving into the ocean. It's only goal is to go, record and get the data for us. So we design a goal based agent with a model and we specify the destination as the goal to be reached.

3. Intelligent Monitor

a. What is the appropriate PEAS specification for an agent that solves the problem and what are the features of the environment?

i. Performance Measure(P):
- To detect the moving targets accurately.
- To maintain a good video quality of all the recordings.
- To maintain good coverage all over the storehouse.

ii. Environment(E):
- Moving targets, unmovable objects and the storehouse.

iii. Actuators(A):
- Video transmitter to transmit the video recording data to some storage device when a moving target is detected.
- Display monitors to display the live recording of a moving target.

iv. Sensors(S):
- Motion sensors to detect a moving target.
- Camera to record videos.

v. Environment Features:

- Fully Observable - The agent has full coverage all over the house and can easily identify a moving target with motion sensors. Hence the agent has full knowledge of the environment and its actions, it is a fully observable environment.
- Single Agent - This is a single agent environment because the monitor by itself does all the video recordings whenever one or more moving targets appears. And there is no other agent in the environment that affects the monitor's performance in terms of video quality provided as output.
- Deterministic - In this case the environment is deterministic because the next state of the environment is completely determined by the current state and the action performed. For example, whenever one or more moving targets are detected the recording starts.
- Episodic - It is an episodic environment because the agent performs the same action regardless of its previous episodes. For example, when a moving target is detected a video recording is done and if a moving target is detected again the same process starts again which does not have anything to do with the previous recording.
- Static - This is a static environment because the monitor need not keep looking at the world while it is recording video of a specific target. If some other target moves, it will also be captured in the same video recording.
- Continuous - Monitor is continuously awake with the motion detection sensor to detect a moving target. And also the number of intermediate steps are not finite as they may vary based on the unique number of moving targets detected. So this is a continuous environment.
- Known - This is a known environment because the outcomes for all the actions are known. For example, considering all the possible outcomes, whenever a moving target is detected a video is recorded otherwise the sensor stays on standby mode.

b. What is the least powerful agent that can solve the problem, and why?

The least powerful agent that can solve this problem is a model-based agent. Because, here the agent doesn't strictly emphasize on achieving a goal. It only does a job of recording whenever a moving target is detected and does seem to bother even if there is a small amount of data loss in the video recordings. So, this can be achieved by designing a model-based agent, where the agent is trained to detect the moving targets based on some training data. Once the training is done well, the agent can be placed in the real world.

4. Intelligent Infrastructure Management

a. What is the appropriate PEAS specification for an agent that solves the problem and what are the features of the environment?

i. Performance Measure(P):

- To minimize the power usage and provide fast user responses.
- To perform proper classification on used and most unused resources to make effective use of them.

ii. Environment(E):

- Virtual machines, Network kits and E-books.
- Intelligent Agent(Can be a robot/computer in this case) and a software to manage the effective usage of resources.

iii. Actuators(A):

- Display screen showing the used and available resources.
- Can also be a website where each user can login and check the availability of resources.
- A chatbot to provide faster responses to user queries.

iv. Sensors(S):

- Sensor to measure the power consumption of the resources allocated.
- Sensors to check all the available resources in the store(Can be a software or automated bots)

v. Environment Features:

- Fully Observable - The agent has full control over the availability of all the resources. It can also estimate the power consumption and consequences of allocating a resource in terms of availability to the other students. Hence the agent has full knowledge of the environment and its actions, it is a fully observable environment.
- Single Agent - This is a single agent environment because the agent by itself does effective resource allocation whenever there is a request. And there is no other agent in the environment that affects its performance in terms of power consumption, faster user responses and resource allocation.
- Deterministic - In this case the environment is deterministic because the next state of the environment is completely determined by the current state and the action performed. For example, whenever there is a request a resource is allocated and marked as unavailable for others.
- Sequential - It is a sequential environment because the current decision could affect all the future decisions. For example, if a request is placed for machine 'A' then it is allocated and marked as unavailable for all the other requests requesting the same machine 'A', which results in affect in the agent's decision due to its previous decision or action.
- Dynamic - This is a dynamic environment because the resources constantly keep allocating or deallocating. So based on the availability of the resources at each point of time the agent has to make a decision to allocate further or not.
- Continuous - Agent is continuously keeping track of the availability of the resources to serve the incoming requests. And also the number of intermediate steps are not finite as they may vary based on the unique number of requests placed. So this is a continuous environment.
- Known - This is a known environment because the outcomes for all the actions are known. For example, considering all the possible outcomes, whenever a request is placed the agent can allocate a resource and mark it as unavailable. Similarly once a resource is deallocated the agent can mark it as available. And also for all the resources allocated the agent can maintain something like a grace period within which the resource should be

returned so that it can estimate the future availability of such resources.

b. What is the least powerful agent that can solve the problem, and why?

The least powerful agent that can solve this problem is a utility-based agent. Because, here the performance measure is to minimize the power usage, provide fast user responses and to do an effective resource allocation. In order to achieve this, we need to assign the scores to any given sequence of environment states, so it can easily differentiate between more and less desirable ways of performing effective resource allocation. So, we design a utility function which gives scores to all types of resource allocations. Based on the scores allocated to the outcomes, the agent will choose actions to maximize its utility (happiness i.e perform effective resource allocation at minimum cost).

2. Search

1. Define an admissible or consistent heuristic for the maze using one of the techniques in class and justify that here. Identify any limitations of your algorithm.

Among all the techniques discussed in the class, for this maze Manhattan distance is admissible or consistent heuristic.

Below is the code block used to implement it:

```
def calculateHeuristicValue(startNode, endNode):
        startRow = startNode.rowIndex
        startColumn = startNode.columnIndex
        endRow = endNode.rowIndex
        endColumn = endNode.columnIndex

        return abs(startRow - endRow) + abs(startColumn - endColumn)
```

Justification: According to triangle inequality, In order for a heuristic to be admissible to the search problem, the estimated cost must always be lower than or equal to the actual cost of reaching the goal state.

h(n) <= c(S,P) + h(P) and h(G) = 0
h(n) is the consistent heuristic function
S is the start point
P is any descendant of S
G is goal node
c(S,P) is cost from start node to the node P
h(P) is the estimated distance from node P to the goal

By calculating Manhattan distance from a particular node to the goal we are specifying the minimum distance to be travelled in order to reach the goal (which includes the distance in all Up, Right, Bottom and Left directions). And if the heuristic function, h always underestimates the true cost (h(n) is smaller than h*(n)), then A* is guaranteed to find an optimal solution

Limitations: Manhattan distance is being used as an estimate of the actual distance but it is not the same as the actual one. Hence it is consistent but might not give us the exact shortest path all the time but reaches almost close to it. And when it comes to diagonal distances, Manhattan distance is not at all advisable.

2. **Implement a maze solver using state-space search that implements DFS, Best First, and A* search with your chosen heuristic. Your code must take a single argument specifying the maze file and must output an answer file in the same format supplied. The dimensions of the puzzle and other information must be read from the file itself. Note that your code will be tested on other unseen mazes.** - Check the search_algorithms.zip file for implementation of all the above mentioned algorithms.

3. **Use the implemented solvers to solve each maze, including the example mazes. For each maze provide the following information:**

    a. **The shortest path**

        i. **4x4 maze**

            1. DFS - (0,0)->(2,0)->(2,1)->(3,1).
            2. Best First Search - (0,0)->(2,0)->(2,1)->(3,1).
            3. A* Search - (0,0)->(2,0)->(2,1)->(3,1).

        ii. **6x6 maze**

            1. DFS - (0,0)->(5,0)->(3,0)->(3,3)->(0,3)->(0,1)->(3,1)->(3,5)->(3,2)->(2,2)->(2,1)->(2,5)->(4,5)->(4,1)->(1,1)->(1,4)->(3,4)->(5,4)->(2,4)->(2,0)->(1,0)->(1,3)->(5,3)->(5,5).
            2. Best First Search - (0,0)->(0,5)->(4,5)->(4,1)->(1,1)->(1,4)->(3,4)->(5,4)->(2,4)->(2,0)->(1,0)->(1,3)->(5,3)->(5,5).
            3. A* Search - (0,0)->(0,5)->(4,5)->(4,1)->(1,1)->(1,4)->(3,4)->(5,4)->(2,4)->(2,0)->(1,0)->(1,3)->(5,3)->(5,5).

        iii. **8x8 maze**

            1. DFS - (0,0)->(0,7)->(0,6)->(0,3)->(6,3)->(6,2)->(2,2)->(2,0)->(6,0)->(3,0)->(3,5)->(3,3)->(4,3)->(4,5).
            2. Best First Search - (0,0)->(0,7)->(1,7)->(2,7)->(2,2)->(2,4)->(6,4)->(6,5)->(4,5).
            3. A* Search - (0,0)->(0,7)->(0,6)->(0,3)->(6,3)->(6,4)->(6,5)->(4,5).

        iv. **12ax12a maze**

            1. DFS - (0,0)->(2,0)->(4,0)->(3,0)->(1,0)->(7,0)->(6,0)->(6,6)->(6,1)->(3,1)->(3,7)->(3,3)->(2,3)->(5,3)->(5,1).
            2. Best First Search - (0,0)->(0,2)->(4,0)->(4,1)->(7,1)->(2,1)->(5,1).

3. A* Search -
   (0,0)->(0,2)->(6,2)->(6,4)->(6,3)->(5,3)->(5,1).

v. 12bx12b maze

1. DFS -
   (0,0)->(6,0)->(4,0)->(11,0)->(3,0)->(3,5)->(8,5)->(1,5)->(10,5)->(10,4)->(2,4)->(5,4).
2. Best First Search -
   (0,0)->(6,0)->(4,0)->(4,7)->(4,1)->(5,1)->(5,6)->(5,3)->(5,4).
3. A* Search -
   (0,0)->(6,0)->(4,0)->(4,7)->(4,1)->(5,1)->(5,6)->(5,3)->(5,4).

b. The number of states expanded by each algorithm when finding the path.

i. 4x4 maze

1. DFS - 12
2. Best First Search - 4
3. A* Search - 5

ii. 6x6 maze

1. DFS - 26
2. Best First Search - 35
3. A* Search - 37

iii. 8x8 maze

1. DFS - 31
2. Best First Search - 11
3. A* Search - 19

iv. 12ax12a maze

1. DFS - 15
2. Best First Search - 9
3. A* Search - 23

v. 12bx12b maze

1. DFS - 103
2. Best First Search - 14
3. A* Search - 17

c. **The total number of unique paths through the graph for 6*6.** - The total number of unique paths through the graph for 6*6 maze are **4**. The are as follows:
    i. (0, 0)->(5, 0)->(3, 0)->(3, 3)->(0, 3)->(2, 3)->(2, 5)->(4, 5)->(4, 1)->(1, 1)->(1, 4)->(3, 4)->(5, 4)->(2, 4)->(2, 0)->(1, 0)->(1, 3)->(5, 3)->(5, 5).
    ii. (0, 0)->(5, 0)->(5, 2)->(0, 2)->(3, 2)->(3, 3)->(0, 3)->(2, 3)->(2, 5)->(4, 5)->(4, 1)->(1, 1)->(1, 4)->(3, 4)->(5, 4)->(2, 4)->(2, 0)->(1, 0)->(1, 3)->(5, 3)->(5, 5).
    iii. (0, 0)->(0, 5)->(4, 5)->(4, 1)->(1, 1)->(1, 4)->(3, 4)->(5, 4)->(2, 4)->(2, 0)->(1, 0)->(1, 3)->(5, 3)->(5, 5).
    iv. (0, 0)->(5, 0)->(3, 0)->(3, 3)->(0, 3)->(0, 1)->(3, 1)->(3, 5)->(3, 2)->(2, 2)->(2, 1)->(2, 5)->(4, 5)->(4, 1)->(1, 1)->(1, 4)->(3, 4)->(5, 4)->(2, 4)->(2, 0)->(1, 0)->(1, 3)->(5, 3)->(5, 5).

d. **The number of states expanded in finding the paths.** - The total number of states expanded in finding the above mentioned unique paths are **99**. States expanded for each corresponding path are as follows:
    i. (0, 0)->(5, 0)->(3, 0)->(3, 3)->(0, 3)->(2, 3)->(2, 5)->(4, 5)->(4, 1)->(1, 1)->(1, 4)->(3, 4)->(5, 4)->(2, 4)->(2, 0)->(1, 0)->(1, 3)->(5, 3)->(5, 5) = 21.
    ii. (0, 0)->(5, 0)->(5, 2)->(0, 2)->(3, 2)->(3, 3)->(0, 3)->(2, 3)->(2, 5)->(4, 5)->(4, 1)->(1, 1)->(1, 4)->(3, 4)->(5, 4)->(2, 4)->(2, 0)->(1, 0)->(1, 3)->(5, 3)->(5, 5) = 26.
    iii. (0, 0)->(0, 5)->(4, 5)->(4, 1)->(1, 1)->(1, 4)->(3, 4)->(5, 4)->(2, 4)->(2, 0)->(1, 0)->(1, 3)->(5, 3)->(5, 5) = 24.
    iv. (0, 0)->(5, 0)->(3, 0)->(3, 3)->(0, 3)->(0, 1)->(3, 1)->(3, 5)->(3, 2)->(2, 2)->(2, 1)->(2, 5)->(4, 5)->(4, 1)->(1, 1)->(1, 4)->(3, 4)->(5, 4)->(2, 4)->(2, 0)->(1, 0)->(1, 3)->(5, 3)->(5, 5) = 28.

Total number of unique paths = 21 + 26 + 24 + 28 = 99.