Name:- Aditya Ahuja
Rollno- 02 Class:- D15C

AIDS (Assignment-1) 04/03

1. What is AI? Considering the COVID-19 pandemic situation, how did AI help to survive and renovate our way of life with different applications?

→ AI (Artifical intelligence) is the replication of human intelligence in machines, enabling them to learn, reason, solve problems, and make decisions without direct human intervention. AI played a vital role in manging the COVID-19 crisis by detecting outbreaks early through data analysis accelerating drug and vaccine development, automating medical diagnoses with imaging and predictive models, supporting telemedicine via AI chatbots, optimizing supply chains for essential goods, and enhancing remote work and online education through AI-driven automation, making daily life more adaptable and efficient.

2. What are AI Agents terminology? Explain with examples

→ AI agents are systems that perceive their environment, process information, and take actions to achieve specific goals.

1. Agent - An entity that interacts with environment (eg self-driving car)

2. Environment - The external system in which the agent operates (eg. traffic conditions for a self-driving car)

3. Perception: Data collected from sensors (eg: a robot's camera detecting obstacles)

4. Actuators - Components that execute actions (eg - robotic arms in manufacturing)

5. Rationality - The ability to make decisions based on available information (eg: AI trading bots adjusting stock investments)

6. Types of Agents:

Simple Reflex Agents - Act based on current perception

Model Based Agents - Use internal models to predict future states (eg - GPS navigation systems)

Goal Based Agents - Take actions to achieve a specific goal eg (Chess playing AI)

Utility Based Agents - Optimize outcomes for maximum benefit (eg recommendation systems in e-commerce)

Learning Agents - Improve performance over time using past experience (eg self learning robot in warehouse)

3. How is the AI technique used to solve 8-puzzle problem?

-> The 8-puzzle problem is solved using AI search techniques that explore possible moves to reach the goal state. Common AI techniques include:

1. Uninformed Search:

- Breadth First Search (BFS): Explores all possible moves level by level but is inefficient for large problems.

- Depth-First Search (DFS):- Explores one path deeply before backtracking but may get stuck in loops.

2. Informed search (Heuristic Based):
- Best-first Search (Greedy Algorithm): Selects moves based on heuristic values like misplaced tiles.

A* Algorithm uses heuristic function which is
$f(n) = g(n) + h(n)$

9. What is PEAS descriptor?

-> PEAS (Performance measure, Environment, Actuators, Sensors) is a frame work used to define the components of an AI agent by specifying how it interacts with its environment and evaluate its success.

1. Taxi Driver
Performance Measure: Safety, speed, fuel efficiency
Environment: Roads, traffic, pedestrians, weather conditions
Actuators: steering, acceleration, brakes, turn signals
Sensors: GPS, cameras, speedometer, LIDAR, fuel gauge

2. Medical Diagonsis System
Performance measure: Accuracy of diagnosis, patient recovery rate
Environment: Patient data, medical records, symptoms, lab reports
Actuators: Display diagnosis, prescribe medication, recommend tests
Sensors: Patient input, test results, doctor's note, medical imaging

3. A music Composer
Performance Measure: Musical quality, originality, listener preference
Environment: Musical genres, user preferences, historical compositions
Actuators: Generating notes, melodies, instrument selection
Sensors: User feedback, music database, emotional tone analysis

4. An aircraft Autolander
Performance measure: Safe landing, smooth touchdown, passenger comfort
Environment: Weather conditions, runway, air traffic
Actuators: Flaps, landing gear, brakes, engine thrust control
Sensors: Altimeter, GPS, wind speed sensors, radar, cameras

5. An Essay Evaluator
Performance measure: Grammar Accuracy, coherence, relevance
Environment: Essays, writing, rules, predefined grading criteria
Actuators: Score assignment, grammar suggestions, feedback generation
Sensors: Text input, grammar and plagiarism checkers, semantic analysis

5. Categorize a shopping bot for an offline bookstore according to each of six dimensions (fully / partially observable, deterministic / stochastic, episodic / sequential, static / dynamic, discrete / continuous, single / multi agent)

=> A shopping bot for an offline book store can be categorized accordingly to the six AI environment dimension as follows

1. Observability: Partially observable The bot may not have full knowledge of stock availability customer preferences or real time changes in book store.

2. Deterministic vs stochastic: Stochastic. Book availability, customer behavior, and price changes introduce randomness, making the environment unpredictable
3. Episodic vs Sequential: Sequential - Each decision (eg. recommending a book) affects future interactions, meaning past actions influence subsequent outcomes
4. Discrete vs Continuous: Discrete: The bot operates with distinct choices, such as recommending a book, or processing a purchase
5. Single Agent vs Multi Agent: Multi-Agent: The bot interacts with customers, bookstore staff, and possibly other AI systems (eg. inventory management) making it a multi-agent system

6. Differentiate Between Model Based and Utility Based agent

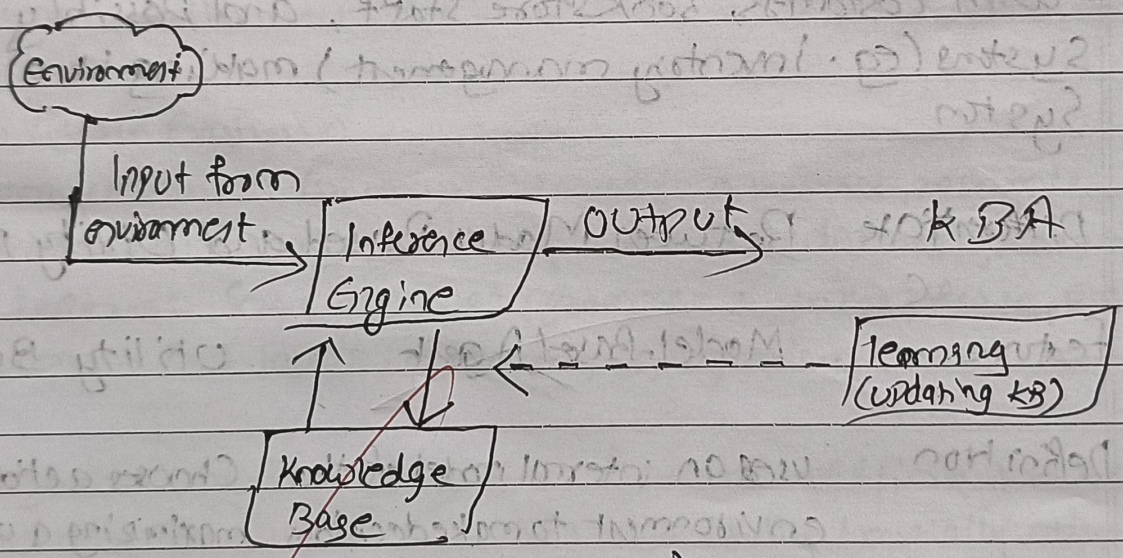
Feature	Model-Based Agent	Utility Based Agent
Definition	used an internal model of the environment to make decisions	Chooses actions based on maximizing a utility function
Decision making	Predicts future states using the model before acting	Evaluates multiple possible actions and selects the best based on utility
Goal-oriented	Yes, but focuses on how the environment works	yes, but aims to achieve the best possible outcome

Optimality — May not always take the best possible action, only a feasible one — Always selects the action that provides the highest

Example — A self driving car using a traffic model to predict congestion — A stock trading AI selecting the stock with highest expected profit.

07 Explain the architecture of a knowledge based agent and learning Agent

→ Architecture of a knowledge-Based agent

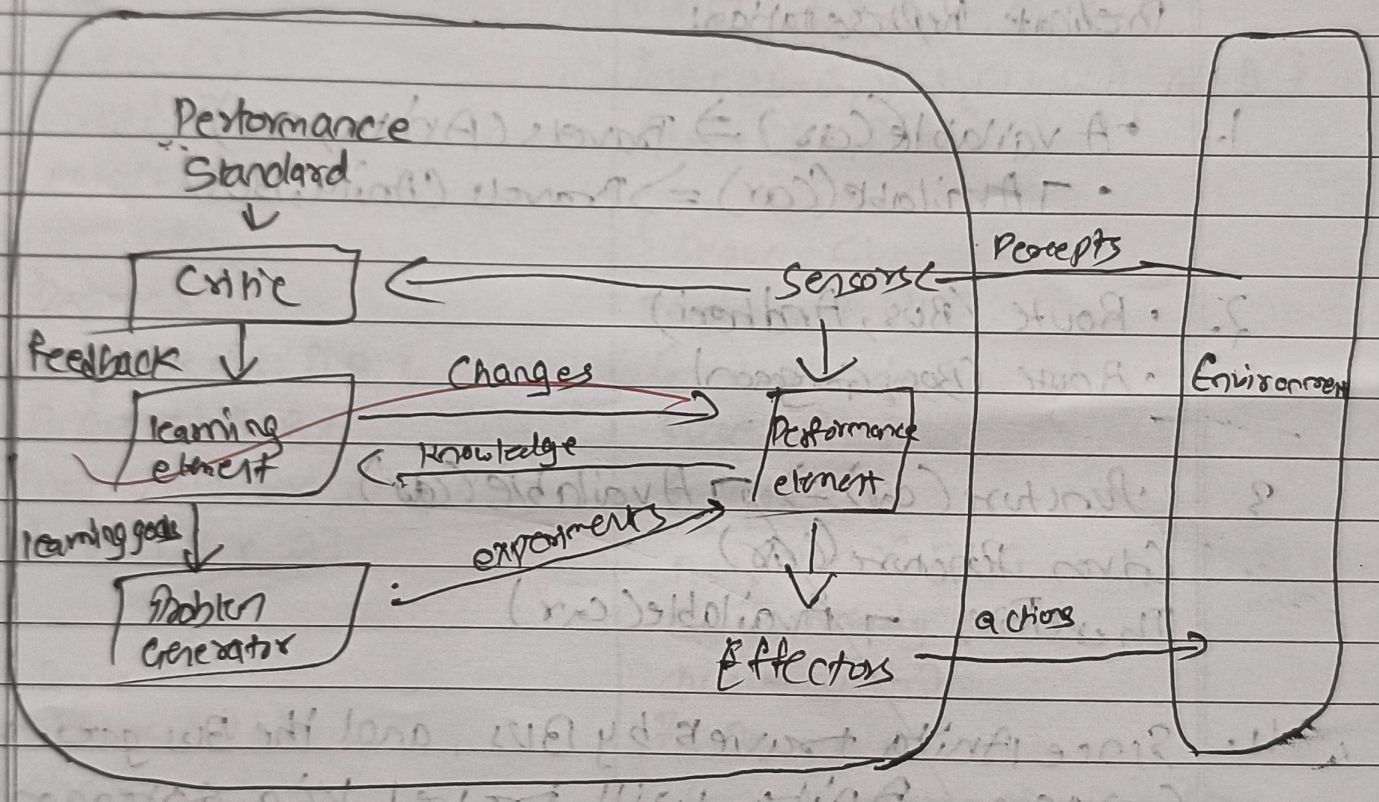


A knowledge-Based Agent (KBA) uses stored knowledge to make decisions and reason about the environment. Its architecture consists of

1. Knowledge Base (KB) - stores facts, rules, and background knowledge in a structured format
2. Inference Engine - Applies logical reasoning to derive conclusions from knowledge Base

3. Perception Module: Gathers input from sensors or external sources
4. Action Module: Executes actions based on derived conclusions
5. Learning Mechanism: Updates the knowledge base with new information

Architecture of a learning Agent



1. Learning Agent: learns from interactions and update knowledge
2. Performance Agent: Uses the learned knowledge to make decisions
3. Critic: Evaluates the agent's performance by comparing actions with desired outcomes
4. Problem Generator: Suggests new exploratory actions to improve learning.

9. Convert the following to predicates:

a) Anita travels by car if available otherwise travels by bus
b) Bus goes via Andheri and Goregaon
c) Car has puncture so is not available

Anita
Will travel ~~thri~~ via Goregaon!

Predicate Representation:

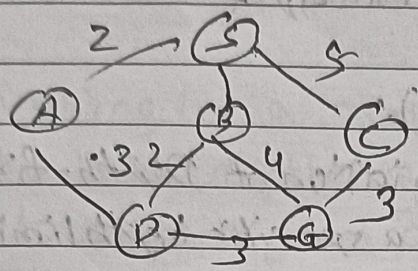
1. • Available(car) ⇒ Travels(Anita, car)
   • ¬ Available(car) ⇒ Travels(Anita, Bus)

2. • Route(Bus, Andheri)
   • Route(Bus, Goregaon)

3. • Puncture(car) ⇒ ¬ Available(car)
   Given Puncture(car)
   Therefore, ¬ Available(car)

4. Since Anita travels by Bus, and the Bus goes via Goregaon, Anita will travel via Goregaon

10 find the route from S to G using Bfs



1. Start:
Queue [S]
visited {S}
parent: {}

2. Dequeue S:
Neighbors of S: A, B, C
Enqueue: A, B, C
Parent: {A: S, B: S, C: S, D: A}
Queue: [B, C, D]

3. Dequeue A:
neighbors of A: D
Enqueue D
Visited {S, A, B, C, D}
Parent {A: S, B: S, C: S, D: A}
Queue [B, C, D]

4. Dequeue B:
neighbors of B: D, G
D is already visited
Enqueue G

visited {S, A, B, C, D, G}
Parent {A: S, B: S, C: S, D: A, G: B}
Queue: [C, D, G]

5. Dequeue C:
neighbors of D: G
G is already visited
Queue: [G]

6. Dequeue D:
neighbors of D: G
G is already visited
Queue: [G]

7. Dequeue G:
G is the destination we stop

path is S → B → G

Q11 What do you mean by depth limited search? Explain iterative deepening search with example

Depth-limited search (DLS):
Depth limited search is a variation of Depth-first search where the search is restricted to a specific depth limit 'l'. If a goal is not found within this limit, the search terminates, it prevents infinite loops in deep or infinite state spaces but risks failing to find a solution if 'l' is too low

Example:
If l=2 DFS explores node only up to depth 2

Iterative Deepening Search:
IDS combines the space efficiency of DFS and completeness of BFS by repeatedly running DLS with increasing depth limits (l=0, 1, 2, ... until the goal is found

Example: (Searching for F in the given graph)

DLS with l=0 only checks S, goal not found
DLS with l=1 expands S, explores A, B, C goal not found
DLS with l=2 expands A, B, C explores D, G goal found at depth 2

Advantage: Guarantees finding the shortest path while using less memory than BFS

12 Explain Hill Climbing and its drawbacks in detail with example. Also State limitations of steepest ascent hill climbing

⇒ Hill Climbing is a local search algorithm that continously moves toward the best neighboring state with a higher heuristic value, aiming to reach a global optimum

Steps:

1. Start from an initial state
2. Evaluate neighboring states and move to the one with highest heuristic value
3. Repeat until no better neighbor exists (local maximum)

Example:

Imagine a mountain climbing scenario where a hike moves uphill based on steepest slope. if they reach a peak that is not the highest (global maximum) they might get stucked

limitations of Steepest-Ascent Hill Climbing:

1. Slow progress in Narrow Ridges: if the optimal path requires small incremental changes, the algorithm struggles:
2. More likely to Get Stuck in local maxima: Choosing only the steepest path can lead to premature convergence.
3. Inefficient in large search spaces: Evaluating all neighbors increases computation time.

Soln

Random Restart: start from different points.

Simulated Annealing: Occasionally accept worse moves to escape local optima.

Q13 Explain Simulated Annealing and write its algorithm

Simulated Annealing is a probabilistic optimization algorithm inspired by annealing process in metallurgy. It helps in finding a global optimum by allowing occasional moves to worse solutions to escape local optima.

Algorithm:

1. Initialize the current state and temperature T.
2. Repeat until stopping conditions:
   - Select a random neighbor state
   - Compute energy difference $\Delta E = E_{new} - E_{current}$
   - if $\Delta E < 0$, accept the new state
   - Else, accept it with probability $e^{-\Delta E / T}$ (Metropolis criterion)
   - Reduce the temperature T (cooling schedule)
3. Return the best found solution

Q14 Explain A* Algorithm with an example

→ The A* Algorithm is an informed search algorithm that finds the optimal path by considering both the cost to reach a node $g(n)$ and the estimated cost to the goal $h(n)$ using the formula

$$f(n) = g(n) + h(n)$$

where

$g(n)$ = actual cost from start node to n

$h(n)$ = estimated cost from n to goal (heuristic)

Q15 Explain Minimax algorithm and draw game tree for tic tac toe

The minimax algorithm is used in adversarial search (eg two player games) to determine the optimal move by assuming both players play optimally
- maximizer (eg. X) tries to get the highest score
- minimizer (eg. O) tries to reduce the score

Algorithm:
1. Generate the game tree upto depth limit
2. Assign heuristic values to leaf nodes
3. Back propagate values:
   - maximizer picks the maximum value
   - minimizer picks the minimum value
4. The root node selects the best move based on propagated values

Game tree:
A minimax tree for Tic-Tac-Toe would show all possible board states, evaluating the best move at each step

Q16 Explain Alpha-beta pruning Algorithm for adversarial search with an example

→ Alpha beta pruning optimizes Minimax by skipping unnecessary branches, reducing computations. It uses two values

- Alpha (α): Best score maximizer can achieve
- Beta (β): Best score minimizer can achieve

Algorithm:
1. Perform Minimax search
2. prune branches where
   - If maximizer's best (α) $\geq$ minimer's best (β), further evaluation is skipped
   - If minimizer's best (β) $\leq$ maximizer's best (α), further evaluation is skipped.
3. This reduces time complexity without affecting the final decision

Example:
In a game tree, if a branch has already provided a worse outcome than the best known move, it is ignored to save time

Q17 Explain Wumpus World Environment giving its PEAS description, Explain how percept sequence is generated ?

=> The Wumpus world is a grid-based AI environment where an agent explores a cave while avoiding hazards like pits and wumpus monster

PEAS Descriptor:

- Performance Measure: Reaching the gold safely, minimizing steps
- Environment: A 4x4 grid with pits, gold and wumpus
- Actuators: move forward, turn, grab, shoot, climb
- Sensors: perceive stench (wumpus nearby), breeze (pit nearby), glitter (gold found)

Percept Sequence Generation:

1. Agent starts at (1, 1) sensing its surroundings
2. If stench is detected the wumpus is nearby
3. If breeze is detected a pit is nearby
4. The agent infers safe paths and navigates towards the gold while avoiding hazards.

Q18 Solve the following Crypto-arithmetic problem: send + more = money

→

	SEND	9 5 6 7
+	MORE	+ 1 0 8 5
M	ONEY	1 0 6 5 2

S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, Y = 2

Q19 Consider the following axioms:

1. Represent these axioms in First order predicate logic
1. $\forall x$ (Graduating(x) $\Rightarrow$ Happy(x))
2. $\forall x$ (Happy(x) $\rightarrow$ Smiling(x))
3. $\exists x$ (Graduating(x))

2. Convert each formula to clause form:
1. $\neg$ Graduating(x) $\vee$ Happy(x)
2. $\neg$ Happy(x) $\vee$ Smiling(x)
3. Graduating(A) (assuming "A" is the individual who is graduating)

3. Prove that "Is Someone Smiling?" using resolution technique:
- From 3: Graduating(A).
- Using 1: Happy(A) (Modus Ponens)
- Using 2: Smiling(A)

Thus Someone is Smiling.

Resolution Tree:

1. Graduating(A) (Given)
2. Graduating(A) $\rightarrow$ Happy(A) $\rightarrow$ Happy(A)
3. Happy(A) $\rightarrow$ Smiling(A) $\rightarrow$ Smiling(A)

Q20 Explain Modus Ponens with a suitable example

→ Modus Ponens (Law of Detachment) is a rule of inference stating: $P \rightarrow Q, P \Rightarrow Q$

Example:
1. If it rains, the ground gets wet (Premise: Rain → wet ground)
2. It is raining (Premise: Rain)
3. Conclusion: The ground is wet (Applying Modus Ponens)

Q21 Explain forward Chaining and backward Chaining algorithm with an example

→ forward chaining:
- Data-driven inference: Starts from known facts and applies rules to reach a goal
- Used in expert systems (eg. medical diagnosis)

Example:

1. fact: "sore throat"
2. Rule: "If sore throat → infection"
3. New fact: "Infection"
4. Rule: "If infection → need antibiotics"
5. Conclusion: "Need antibiotics"

Backward Chaining:

- Goal-driven inference: starts from the goal and works backward to find supporting facts
- Used in AI reasoning and theorem proving

Example:

Goal: Does the patient need antibiotics?

1. Check: Does the patient have an infection?

2. Check: Does the patient have a sore throat?

3 If both hold, conclude "Need antibiotics"

This reduces unnecessary computations by only exploring relevant facts.