UNIT I

1. FUNDAMENTALS

Intelligent Agents - Agents and Environment - Good Behavior - The Nature of Environments - Structure of Agents - Problem Solving - Problem Solving Agents - Example Problems - Searching for Solutions - Uniformed Search Strategies - Avoiding Repeated States - Searching with Partial Information

1.1. Introduction to AI

Artificial Intelligence (AI)

AI is that the learning of the way to create machines performs actions that, at the instant, human perform best.

It leads **4** important levels.

a) Machines that feel similar to peoples.
b) Machines that perform similar to peoples.
c) Machines that feel logically.
d) Machines that perform logically.

Performing Humanly: The Turing Assessment Approach

To conduct this assessment, we want 2 individuals and also the machine to be evaluated. One human being plays the responsibility of the questioner, who is in an exceedingly separate area from the compute and also the different human being. The questioner will raise queries of either the human being or the computer by writing queries and receiving written responses. However, the questioner is aware of them just as A and B and targets to work out that the human being is and that are the machine. The target of the machine is to fool the questioner into believing that's the human being. If the machine achieve at this, in that case we will complete that the machine is performing humanly. However program a machine to give the assessment gives lots to perform on, to posses the subsequent abilities

- **Natural Language Processing:** is to permit it to interact well in English.
- **Automated Reasoning:** is to apply the saved data to reply queries and to get latest conclusions.
- **Knowledge Representation:** is to save data supplied before or for the duration of the investigation.

- **Machine Learning:** is to undertake to latest assets and to study from practice, example etc.,

Total Turing Assessment (TTA): The assessment which contains a video signal in order that the investigator should assess the emotional capabilities of the machine. To go through the TTA, the machine requires:

- **Computer Vision:** is to recognize entities
- **Robotics:** to shift them.

Sensible Humanly: The Cognitive-modeling Method

To build a machine application to assume similar to a people, initial it needs the information regarding the particular workings of human mind.

Once finishing the study regarding human mind it is potential to precise the speculation as a machine application.

If the application‘s ip/op and temporal order attitude equals with the people attitude then we will state that the program‘s method is functioning like a human mind.

Example: General Problem Solver (GPS)

Thinking Rationally: The Laws of Thinking Approach

The correct thinking introduced the idea of logic.

Example: Phani is a student of 3rd year CSIT.

All students are excellent in 3rd year in CSE. Phani is an excellent student.

Performing Rationally: The Rational Agent Method

Performing Rationally denotes, to attain one‘s goal specified one‘s idea. Within the earlier topic laws of thought approach, accurate reasoning is chosen, conclusion is derived, but the agent acts on the conclusion defined the task of acting rationally. The learning of rational agent has **2** advantages:

1. Correct reasoning is chosen and applied.
2. It focuses on scientific improvement instead of different strategies.

The History of AI

1943:	Pitts & McCulloch: Boolean circuit of design of mind
1950:	Turing's assessing machines and brilliance
1950s:	Near the beginning AI applications, which includes Samuel's-checkers application, Gelernter's Geometry Engine, Simon & Newell's Logic Theorist
1956:	Dartmouth summit: AI accepted
1965:	Robinson's entire set of rules for logical analysis
1960 – 74:	AI find out computational difficulty Neural network studies nearly disappears
1969 – 79:	Near the beginning improvement of knowledge based methods
1980 – 88:	Specialist systems enterprise boom
1988 – 93:	Specialist systems enterprise busts: AI winter
1985 – 95:	Neural Networks come back to reputation
1988:	Reappearance of probability; common enhance in technical path Nouvelle AI; GA's, A Life, and soft computing
1995:	Agents
2003:	Human being level AI rear on the plan

1.2. Intelligent Agents

1.3. Environments and Agents

An **agent** is some thing that should be regarded as appearing up on that environment through **actuators** and recognizing its **environment** over **sensors** given away in Figure.

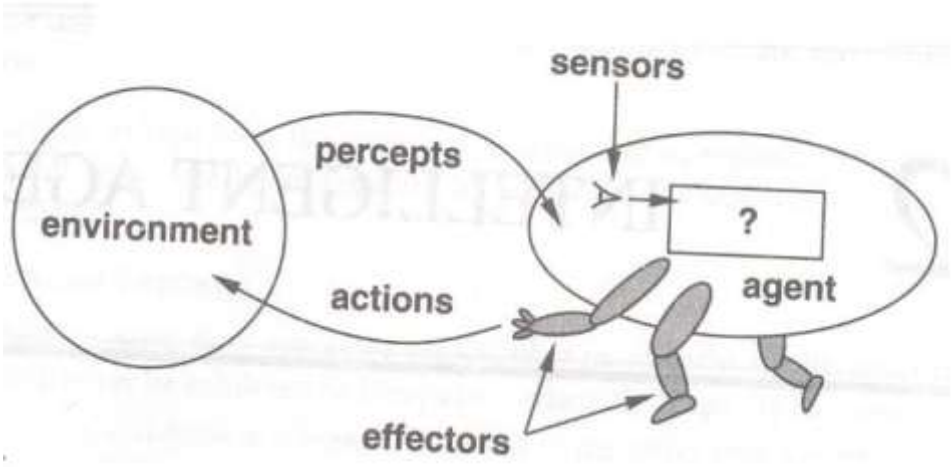

Figure: Agents connect with environment through effectors and sensors

The different categories of agent are:

- **Robotic Agent:** This agent contains infrared and cameras variety selectors for the sensors and different machines for the selectors.
- **Human Agent**: This agent contains ears, eyes and additional organs for sensors and mouth, hands, legs and additional human parts for selectors.
- **Generic Agent:** A preferred shape of an agent who connect with the environment.
- **Software Agent:** This agent contains programmed bit series as its rules and procedures.

The agent work for an agent indicates the activity extract by the agent in reaction to any rule order. It is a hypothetic mathematical explanation.

Inside, the agent work for an artificial-agent could be applied by using an agent application. An agent application is a task, which appliance the agent mapping from rules to activities. It is an actual implementation operation on the agent structural design.

1.4. Excellent Behavior: The Idea of Rationality

An agent has to act as a Rational Agent. This agent is one which performs the proper thing that is the proper actions will cause the agent to be mainly doing well within the environment.

Performance Measures

A performance measures represents the principle for achievement of an agent‘s behavior. As a standard regulation, it is improved to plan performance-measures in line with what one really desires within the environment, quite as per to how one imagines the agent could behave.

Rationality

What is a rational at all specified instance depends on 4 items:

- The **performance-measure** that describes the principle of **fulfillment**.
- The agent‘s earlier **awareness** of the environment.
- The **movements,** which the agent should execute.
- The agent‘s rule **collection** to date.

This directs to a description of a rational-agent (perfect rational agent)

- *For all potential rule series, a rational-agent need to choose an activity, which is anticipated to boost its presentation measure, specified the proof furnished by way of the rule series and anything fixed expertise the agent contain, that is the task of rational agent is to improve the performance measure depends on percept sequence.*

Omniscience, Autonomy, and Learning

An omniscient agent identifies the real final results of its performances and may work appropriately; but all-knowing is not possible actually.

A rational agent moreover to collect data, but also to study as a great deal as viable from what it perceives. The agent's preliminary arrangement should return some previous awareness of the environment, but because the agent gain knowledge this will be changed and improved.

Successful agents break up the function of measures the agent task into 3 dissimilar phases: while the agent is planned, a few of the calculation is achieved via its developers; while it is considering into account on its subsequent work, the agent performs higher working out; and as it study from familiarity, it performs higher working out to determine how to change its performance.

A rational-agent shall be independent – it could study what it is able to catch up on incomplete or wrong earlier information.

1.5. The Nature of Environments

Stating the Work Environment

The work environment specification consists of the outside environment, the presentation measure, the sensors, and the actuators.

In planning an agent, the primary step need to forever be to state the work environment as entirely as doable. Work environments are distinctive as a PEAS (Performance, Environment, Actuators, Sensors) or PAGE (Percept, Action, Goal, Environment) description, both methods are same.

The following table describes some of the agent types and the basic PEAS description.

Table: Patterns of Agent Categories and their PEAS Explanation

Agent category	Actuators	Environment	Sensors	Performance computation
Taxi-Driver	Accelerator, steering, signal, brake, display, horn	Roads and Further traffic, person on foot, clients	Speedometer, GPS, cameras, sonar, odometer, engine sensors, accelerometer, keyboard	Fast, legal, safe, comfortable trip, maximize profits
Health diagnosis system	Show queries, checkup, diagnoses, Recommendation, treatments	Staff, Hospital, patient,	Keyboard enter the symptoms, results, patient‘s responds	Lawsuits, Well patient, minimize costs
Satellite picture investigation method	Show group of picture	Down link from tracking satellite	Color pixel arrangements	Exact picture group
Refinery controller	Pumps, heaters, valves, displays	Operators, refinery,	Pressure, temperature, chemical sensors	Yield, maximize purity, safety
Part picking robot	Hand and linked arm	Bins, Conveyor belt with parts;	Camera, Joint angle sensors	proportion of parts in right bins
Interactive English trainer	Show work outs, Instructions, modifications	Assessing agency, group of students	Entry with Keyboard	Increase student‘s marks on assessment

Some **Software Robots** or **Software Agents** or **Soft bots** present in rich, and unlimited domains. So,

- Actions are done in real time.
- Action is performed in the complex environment.
- It is designed to scan online news origin and display the procedures with ordinary language processing.

Properties of Work Environments

1. Completely Recognizable vs. Partly Recognizable (or) Accessible vs. Inaccessible

If an agent‘s sensors deliver it gain to the whole condition of the environment at every end in the moment, next we state that the work environment is completely recognizable. Completely observable environments are suitable due to the fact the agent unneeded to continue any inside status to preserve path of the universe.

An environment is probably in partly observable because of inaccurate sensors and noisy or because elements of the condition are just lost from the sensor records.

2. Deterministic vs. Non-deterministic (or) Stochastic

If the subsequent condition of the environment is absolutely decided via the present condition and the operation achieved by way of the agent, next the environment is **deterministic**; or else, it is **stochastic**. In belief, an agent may not concern regarding doubt in a deterministic environment, **completely recognizable**. If the environment is **partly recognizable**, however, next it should arrive like stochastic.

3. Episodic vs. Non-episodic (or) Sequential

In an episodic environment, the agent‘s knowledge is split into atomic episodes. Each episode contains of its private percepts and activities and it does not rely on the preceding episode.

In sequential-environments, the current choice could control all expectations of decisions.

Eg: Taxi driving and Chess.

Episodic-environments are easier than sequential-environments for the reason that the agent could not required thinking in advance.

4. Dynamic vs. Static

If the environment is not changing for agent‘s action then the environment is static for that agent otherwise it is **dynamic**.

If the environment does not modify for some instance, then it modifies due to agent‘s action is called **semi-dynamic environment**.

E.g: Taxi driving is **dynamic**. **Chess** is **semi dynamic**. **Crossword puzzles** are **static**.

5. Continuous Vs. Discrete

If the environment has clearly defined percepts, finite number of distinct, and actions then the environment is **discrete**. E.g.: **Chess**

If the environment varies continuously with variety of value the environment is **continuous**. E.g: **Taxi driving**.

6. Single Agent vs. Multi Agent

The following Table indexes the properties of a count of known environments.

Task Environment	Episodic	Deterministic	Agents	Observable	Discrete	Static
Crossword puzzle	Sequential	Deterministic	Single	Fully	Discrete	Static
Taxi driving	Sequential	Stochastic	Multi	Partially	Continuous	Dynamic
Chess with a clock	Sequential	Strategic	Multi	Fully	Discrete	Semi
Medical diagnosis	Sequential	Stochastic	Single	Partially	Continuous	Dynamic
Backgammon	Sequential	Stochastic	Multi	Fully	Discrete	Static
Refinery controller	Sequential	Stochastic	Single	Partially	Continuous	Dynamic
Image analysis	Episodic	Deterministic	Single	Fully	Continuous	Semi
Part-picking robot	Episodic	Stochastic	Single	Partially	Continuous	Dynamic
Poker	Sequential	Strategic	Multi	Partially	Discrete	Static
Interactive English tutor	Sequential	Stochastic	Multi	Partially	Discrete	Dynamic

1.6. The Structure of Agents

An intelligent agent is a grouping of Architecture and Agent Program.

Intelligent Agent = Architecture + Agent Program

Agent Program is a task, which performs the agent mapping from rules to procedures. Their presence a range of primary agent program plan, reverse the type of records prepared explicit and utilized within the choice procedure. The proposes range in compactness, flexibility and efficiency. The suitable propose of the agent program relies upon the character of the environment.

Architecture is a computing tool utilized to run the agent program.

Four kinds of agent programs are there to execute the mapping assignment. They are:

1. Simple reflex agents
2. Utility based agents
3. Goal based agents
4. Model based reflex agents

We then clarify in common expressions **how to change** all these into **learning-agents**.

1. Simple Reflex Agents

The easiest kind of agent is the simple-reflex agent. It reacts directly to percepts i.E. those agent pick task on the source of the current rule, avoiding remaining rule record.

An agent explains in relation to how the situation – action policy permit the agent to build the connectivity from rule to operation.

Condition action rule: if condition then action

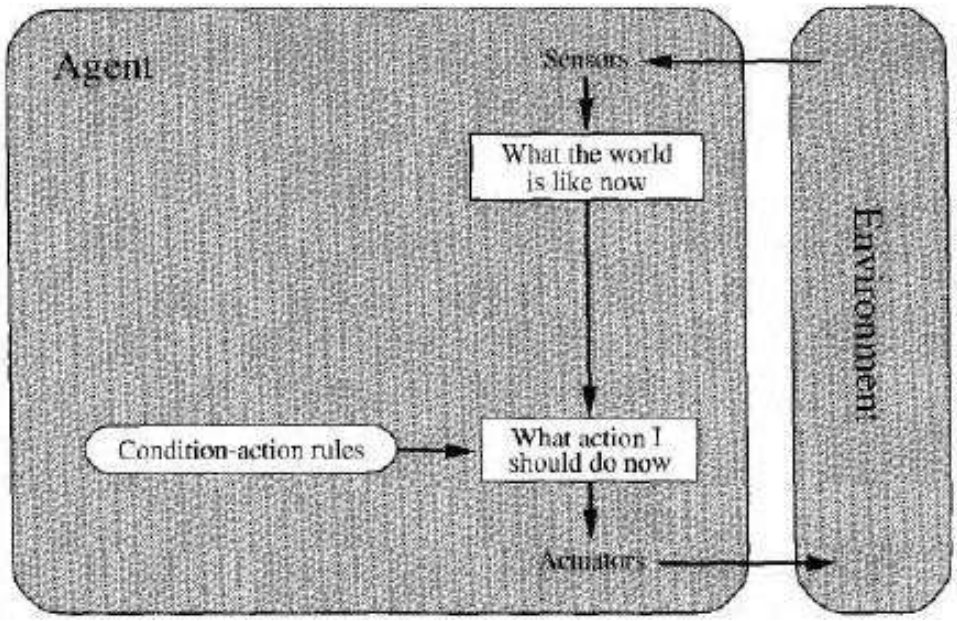

Figure: Schematic Figure of a Simple Reflex Agent

Diagram: specifies the formation of this common plan in schematic type, presenting how the state – activity rules permit the agent to build the link from rule to operation. In the above schematic figure, the shapes are described as:

Oval – to symbolize the background data in the procedure.

Rectangle – to indicate the present internal state of the agent's decision procedure.

The agent-program, that is also easy, is exposed in the following statements.

function SIMPLE REFLEX AGENT (percept) **returns** an action

static: rules, a set of condition-action

rules state ? INTERPRET –

INPUT (percept) rule ? RULE –

MATCH(state, rules)

action ? RULE – ACTION[rule]

return action

INTERRUPT INPUT - Is the job of creates an absorbed explanation of the present status from the rule.

RULE MATCH - Is the job of returns the 1st regulation in the group of regulations that matching with the specified status explanation.

RULE ACTION - The preferred rule is performed as action of the specified percept.

The agent in diagram shall functions just -*if the right choice should be prepared on the source of just the present rule – which is, just if the environment is completely observable*.

Example: Health diagnosis system

If the sufferer has reddish brown bad skin **then** begin the medication for measles.

2. Utility based Agents (Utility Mentions to the Quality of being Helpful)

An agent creates a target state with great quality manners (utility), if higher than one series contains to attain the target condition next the series with safer, greater reliable, faster and cheap than further to be selected.

A **utility function** maps a situation (or series of situations) onto an actual count, which explains the connected degree of gladness. The utility function can be utilized for two special cases: **First**, while there are incompatible targets, only a few of which should be attained (for example, safety and speed), the utility function denotes the precise transaction. **Second**, whilst the agent targets for several targets, nothing can be attained with certainty, then the achievement may be weighted up towards the significance of the goals.

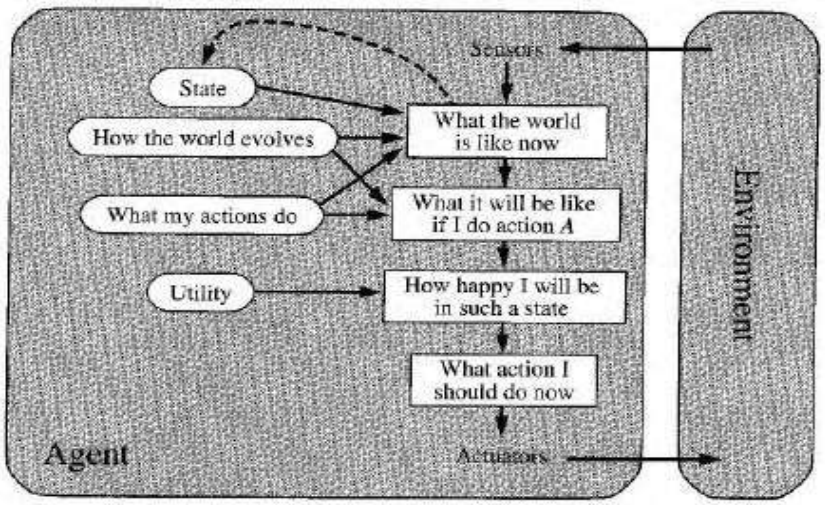

Figure: A Model based, Utility based Agent

Learning Agents

The **learning** function permits the agent to perform in primarily unfamiliar environments and to grow to be high qualified than its early awareness.

A **learning** agent should be separated into **4 theoretical elements**, as exposed in Figure.

Learning element - This is answerable for preparing developments. This makes use of the comments from the commentator on how the agent is performing and resolves how the action elements have to be changed to do well in forthcoming.

Performance element - which is answerable for opting external actions and it is equal to agent: this receives in rules and choose on operations.

Critic - It informs the learning-element how fine the agent is performing with specific to a permanent execution level.

Problem generator - This is accountable for signifying works to be able to lead to latest and informative knowledge.

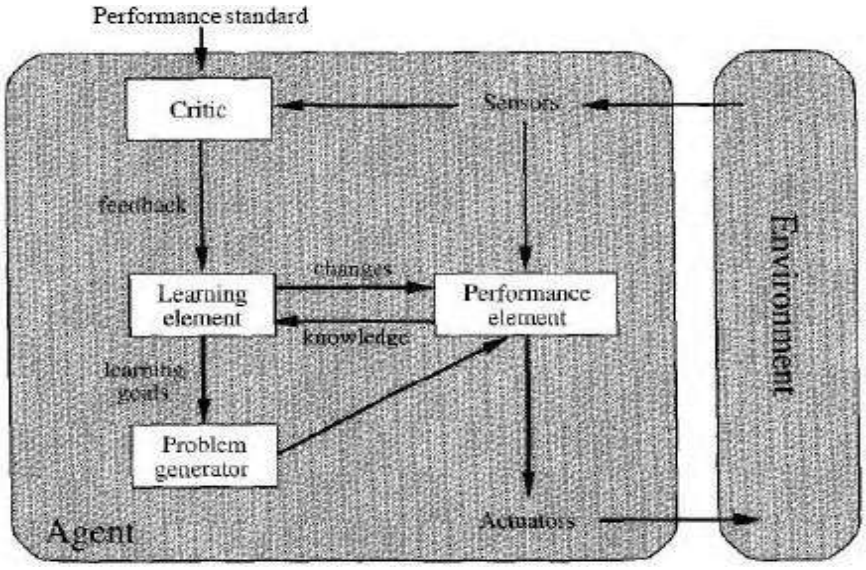

Figure: A Normal Version of Learning Agents

In review, agents contain a range of elements, and those may be presented in several methods inside the agent program, thus there seems to be high range between studying ways. **Learning** in intellectual agents might be outlining as an action of variation of all factors of the agent to carry the elements into nearer settlement with the existing elements record, there by developing on the whole action of the agent (All agents can enhance their overall performance through **learning**).

3. Goal based Agents

An agent recognizes the explanation of present status and also wishes some kind of **goal** records that illustrates conditions which might be desirable. The action equivalent with the present status is selected depend on the **goal** situation.

The goal based agent is high flexible for greater than one destination. After recognizing one destination, the latest destination is stated, goal based agent is started to provide with a latest behavior. **Planning** and **Search** are the sub-fields of AI dedicated to discovery work series that attain the agent‘s goals.

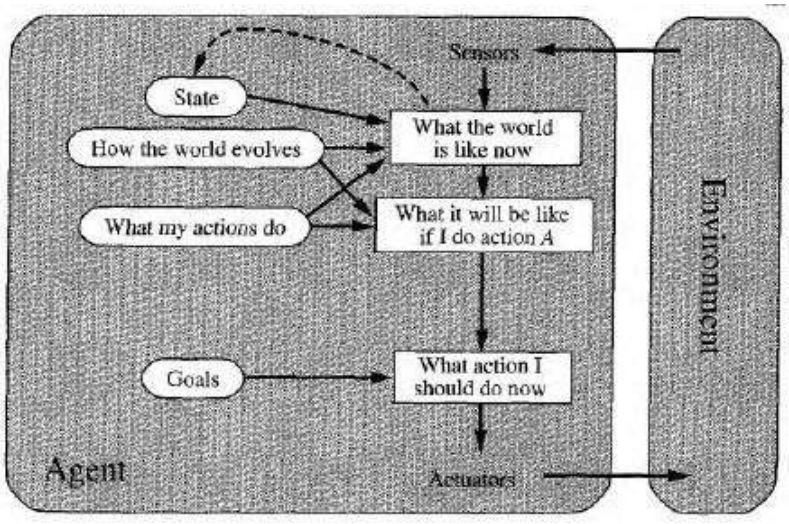

Figure: A Goal based, Model based Agents

The goal based agent arrives fewer competent, it is high adjustable for the reason that the information that supports its choice is signified clearly and should be changed. The goal based agent‘s manners should simply be modified to go to a dissimilar position.

4. Model based Reflex Agents

The only mode to deal with partly recognizability is for the agent “*to be track of the piece of the universe it cannot observe at present*”. That is, the agent that mixes the current rule with the previous internal status to make updated explanation of the current status.

The current rule is mixed with the previous internal situation and it gains a latest current situation is up to date in the situation explanation is also. This updation needs **2 types of understanding** inside the agent program. **First**, we need a little knowledge regarding how the globe evolves separately of the agent. **Second**, we need a little knowledge regarding how the agent‘s personal actions have an effect on the world.

The above 2 information applied in easy Boolean-circuits or in entire methodical hypothesis is referred as a **model** of the universe. An agent, which makes use of this kind of model is called **model based-agent.**

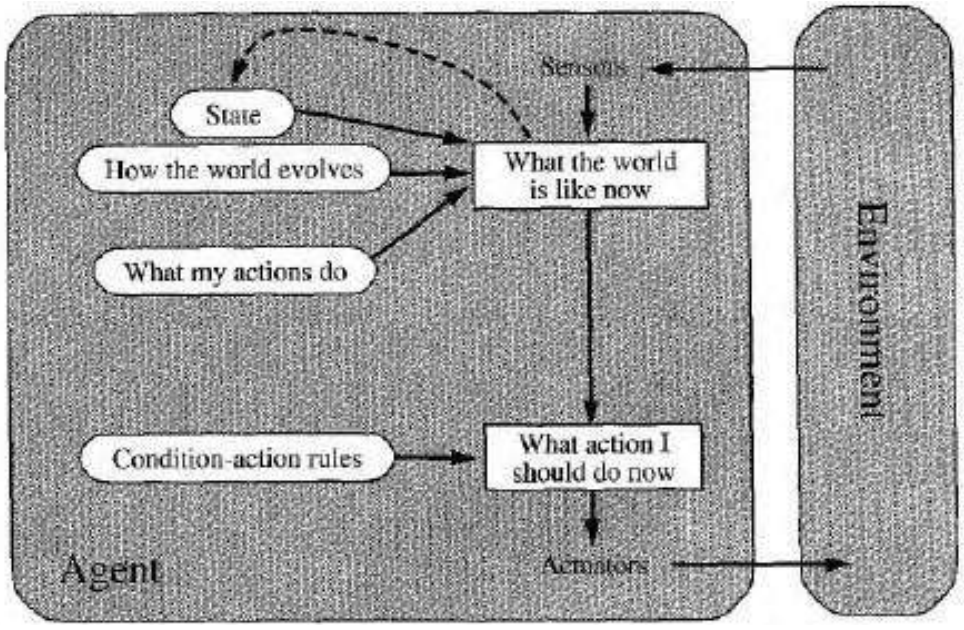

Figure: A Model-based Reflex-agent

The figure demonstrates the formation of the reflex-agent with internal condition, displaying how the present rule id jointed with the previous internal condition to produce the upgraded explanation of the present condition. The agent program is exposed in figure.

function REFLEX AGENT WITH STATE (*percept*) **returns** an action

static: *state*, an explanation of the present world state

rules, a set of condition action rules

action, the latest action, primarily not

state ? UPDATE STATE(state, action, percept)

rule ? RULE MATCH(state, rules)

action ? RULE ACTION[rule]

return action

Figure: A model-based Reflex-agent

It remains catch of the present status of the globe by applying an internal design. It next selects an operation in the similar method as the reflex-agent.

UPDATE STATE - This is answerable for producing the latest internal state information by joining percept and present state information.

1.7. Solving Problems by Searching

Problem Solving Agents

It is one form of goal-based agent, in which the agent chooses what perform by selecting series of operations that result in ideal states. If the agent known the meaning of problem, it's quite simple to create a search procedure for discovering solutions, which recommends that problem solving agent need to be an intelligent agent to increase the overall performance measure.

The series of steps performed through the intelligent agent to increase the performance measure:

1. **Goal formulation** - primarily based on the agent‘s execution assess and the present state, is the initial step of problem-solving.

2. **Problem formulation** – it is the procedure of identifying what states and measures to deal, specified a goal.

3. **Search** - An agent with numerous quick alternatives of unfamiliar value should determine what to perform by using initial investigating dissimilar possible series of operations, which guide to states of identified value, and next selecting in fine series. This procedure of looking for this sort of series is known as search.

4. **Solution** - Search algorithm gets a problem as entry and arrival a result in the structure of an operation series.

5. **Execution phase** - formerly a result is found, the activities it suggests can be passed out.

The figure exposed as an easy problem-solving agent. It initially defines a problem and goal, inspect for a series of operations that universe resolve the problem, after which performs the operations individually at a time. When that is finished, it prepares one more startsover and goal.

```
function SIMPLE PROBLEM SOLVING AGENT (percept) returns an action
  inputs: per, a percept
  static: ser, an action series, primarily empty
          stat, some information of the present world state
          gl, a goal, primarily null
          prob, a problem design

  stat ?  UPDATE STATE (state, percept)
  if ser is empty then do
          gl ?  FORMULATE GOAL(stat)
          prob ?  FORMULATE PROBLEM(stat, gl)
          ser ?  SEARCH(prob)
  action ?
  FIRST(ser) ser ?
  REST(ser)
  return action
```

Figure: An Easy Problem Solving Agent

Note: First - 1st action in the series

Rest - returns the rest

Search - selecting the top one from the series of actions

Formulate problem - series of actions and situation that guide to target state

Update state - 1st state is enforced to subsequent state to reach the target state.

Well Described Problems and Solutions

A Problem should be described properly by four elements:

- P**rimary state** is the agent begins.
- Information of the available **actions (act)** offered to the agent. The general formulation utilizes a **successor (success) function**. Specified a specific state '**x**', **SUCCESSOR_FN (x)** gives a group of **< act, success >** structured pairs, accessible from **x** by any one action.

- The preliminary state and successor_function absolutely outline the **state-space** of the problem the group of each state available from the primary state. The state-space figures a diagram in which the joints are states and the arch among the joints and operations. A **path** inside the state area is a series of states joined by way of chain of operations.
- The **goal test** that regulates regardless if a specified state is a goal-state. If greater than one goal state be present, then we can test whether anyone of the goal-state is arrived at or not.
- A **path cost** utility, which allocates a numerical cost to every path. The cost of a path should be explained as the addition of the costs of all specific operations beside the path. The **step-cost** of getting action(*act)* to move from '*x*' to state '*y*' is indicated by *c(x, act, y).*

The previous elements outline a problem should be collected jointly into a separate records formation that is taken as entry to a problem-solving algorithm. A **result** to a problem is a path from the primary state to a goal state. The result value is calculated by the path-cost operation, and an **optimal-solution** contains the minimum path-cost amongst each results.

Formulating Problems

We derive a formulation of the problem in provisos of the **primary state**, **successor-function**, **goal-test**, and **path-cost**.

The procedure of eliminating feature from a depiction is known as **abstraction**. In extension to abstracting the state information, we should abstract the operations by itself.

data type PROBLEM

Components: Primary State, Successor Function, Goal Test, Path Cost.

1.8. Example Problems

Toy Problems

i. The Eight-puzzle Problem

The eight-puzzle contains of a 3×3 board with 8 valued tiles and an empty space as displayed in figure 3.2. A tile next to the empty space will move into the space. The entity is to arrive at a described target state. The principle formulation is:

- **States**: State information indicates the empty in 1 of the 9 boxes and the position of all of the 8 tiles.

- **Primary state**: Every state may be selected as the primary state.
- **Successor function**: It produces the permissible states that outcome from attempting the 4 operations (empty shifts Down, Up, Right and Left).
- **Goal test**: It tests regardless if the state fits the goal arrangement.
- **Path cost**: Every step costs one, so the steps count in the path is the length of the path (path-cost).

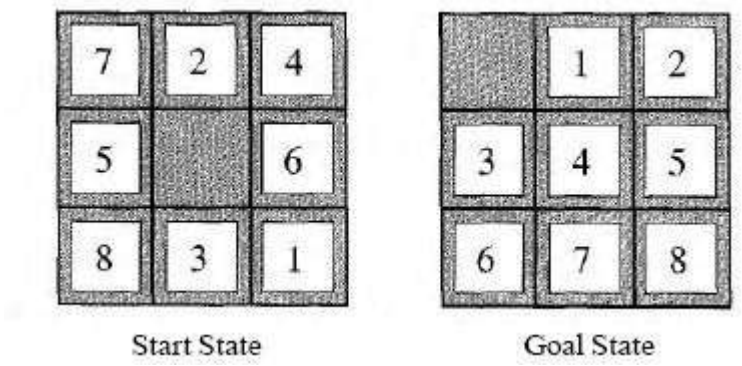

Figure: A Distinctive Instance of the Eight – puzzle

ii. The 8 – Queen's Problem

The purpose of the 8 – queen's-problem is to put 8-queens on a chess-board such that any queen does not attacks further in the same diagonal, column or row exposed in figure. There are **2 major types of formulation**. An **incremental_formulation** associates operative, which increase the state information, opening with a blank state; for the eight – queen's problem, this denotes that every operation include a queen to the state.

The incremental formulation is as follows:

- **States**: Any adjustment of zero to eight-queens on the board is a state.
- **Primary state**: The board does not contain any queen.
- **Successor function**: Any blank square inserted by a queen.
- **Goal test**: Eight-queens are at the board, no one attacked.

In this, we contain 3 × 1014 probable series to examine.

A **whole state formulation** begins with every eight queens at the board and shifts them on all sides. In each case, the path-cost is not important because just the number of last states.

A best formulation could disallow insertion a queen in any squared box, which is previously attacked:

- **States**: Adjustment of n-queens (0≤n≤8), one consistent with column within the left-most 'n' columns, and no queen attacking a different state.
- **Successor function**: Insert a queen to every squared box inside the left-most blank column such that it isn't always attacked by means of further queen.

This formulation decreases the eight-queen's state area from 3 ×1014 to simply 2,057 and results are simple to find.

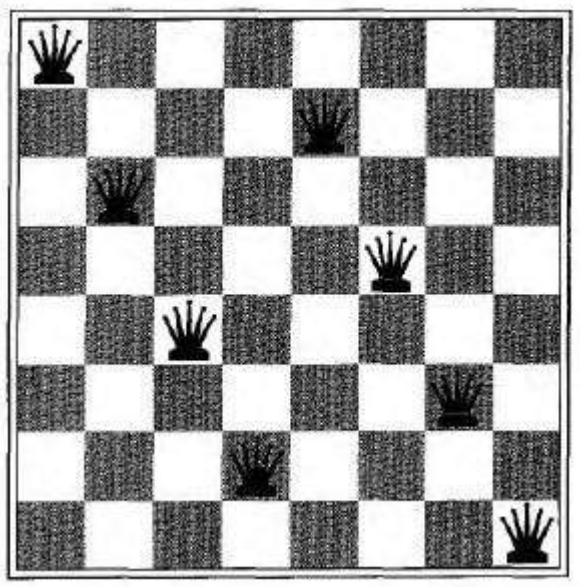

Figure: The 8 – Queen's Problem

Real-world Problems

Name of the problems	Applications
i) Route finding	Airline journey planning system, armed operations preparation and Routing in networks.
ii) Touring and traveling salesperson problem	Shortest path tour.
iii) VLSI layout	Cell layout, channel routing
iv) Robot navigation	Route finding problem in continuous space.
v) Automated assembly ordering	Assembly of composite objects by Robot. E.g. Electric motors.

1.9. Searching for Solutions

We need to solve the formulated problems are completed by a search method throughout the state space that utilize a **search-tree** that is produced by the primary state and the successor-function that jointly describe the state-space. The root of the search-tree is a **search-node** matching to the primary state.

We keep on selecting, checking, and increasing upto either a result is observed or no states to be extended. The option of which state to develop is set by using the **search-plan**. The common search-tree algorithm is explained in the figure as follows:

function TREE SEARCH (problem, plan) **returns** a result, or failure
initialize the search tree by using the primary state of problem
loop do
if there are not having any candidates for growth **then return** failure
select a leaf node for growth according to plan
if the node holds a target state **then return** the matching solution
else enlarge the node and include the resulting nodes to the search tree

Figure: Explanation of the general search-tree algorithm

The nodes in the search tree are defined using **five components** in data structure. They are

1. **STATE**: the state location to whichever the node is correlate;
2. **PARENT NODE**: the node in the search-tree that produced another node;
3. **ACTION**: the action, which is executed to the root to produce the node;
4. **PATH COST**: the cost, generally indicated by g (n) of the path from the primary state to the node, as denoted by the parent indicator.
5. **DEPTH**: the steps count next to the path from the primary state.

The **difference among** states and nodes. A **state** correlates to the design of the world. A **node** is recording records structure applied to signify the search-tree.

To signify the group of nodes which were produced but till now not extended – this grouping is known as **fringe**. Every component of the fringe is a **leaf-node**, that is, a node without successors inside the tree. The demonstration of the fringe could be a group of nodes. The search method next could be a task that chooses the upcoming node to be accelerated from this group. It can be computationally costly, because the approach task may have to take a view at each component of the group to select the better one. Otherwise, the grouping of nodes is executed as a **queue** representation. The **queue actions** are:

- **MAKE QUEUE** (element…) – makes a queue with the specified elements.
- **EMPTY ?**(queue) – if no elements are in the queue, then it returns true.
- **FIRST**(queue) – it returns the initial element in the queue.
- **REMOVE_FIRST**(queue) – it returns initial element and eliminates that element from the queue.

- **INSERT**(element, queue) – adds an element within the queue and the resultant queue.
- **INSERT_ALL**(elements, queue) – adds a group of elements within the queue and returns the resultant queue.

The formal method of the common tree search algorithm is exposed in the bellow figure:

```
function TREE-SEARCH(problem, fringe) returns a result, or failure
    fringe ← INSERT(MAKE NODE (INITIAL STATE [problem] ), fringe )
    loop do
        if EMPTY ? ( fringe ) then return failure
        node ← REMOVE FIRST ( fringe )
        if GOAL TEST [ problem ] tested to STATE [ node ] achieves
            then return SOLUTION ( node )
        fringe ← INSERT ALL ( EXPAND ( node, problem ), fringe )

function EXPAND ( node, problem ) returns a set of nodes
    successors ← the unfilled set
    foreach < action, result > in SUCCESSOR-FN [ problem ] ( STATE [ node ] )
    do
        s ← a fresh NODE
        STATE[s] ← result
        PARENT NODE[s] ←
        node ACTION[s]
        action
        PATH COST[s] ← PATH COST [ node ] + STEP COST ( node, action, s )
        DEPTH[s] ← DEPTH [node] + 1
        inserts to successors
    return successors
```

Figure: The General Tree Search Algorithm

Evaluating Problem Solving Performance

The result of a problem_solving is any of fail or a result. We can assess an algorithm‘s actions in **4** methods:

- **Optimality**: If more than one way exists to derive the solution then the best one is selected.
- **Completeness**: The method definitely to discover a result when there is one.
- **Time complexity**: The Time obtained to run a result.
- **Space complexity**: Memory required executing the search.

In Artificial Intelligence, wherein the chart is pictured completely by means of the primary state and successor_function and is often endless, **difficulty** is disclosed in terms of 3 measures '**m'** is the highest length of any path in the state-space; '**d'** is the depth of the slight target node; and **b** is the branching component.

Definition of branching factor (b): The nodes count that is connected to every node in the search-tree. It is used to find space and time complexity of the search strategy.

1.10. Search Strategies

The Search Strategies are classified into **Informed search** (or) **Heuristic search** and **Uninformed search** (or) **Blind search.**

Informed Search vs. Uninformed Search

S. No	**Informed search(Heuristic search)**	**Uninformed search(Blind search)**
1.	The path-cost from the present state to target state is measured, to choose the least path-cost as the further state.	Path-cost from the present state to target state (or) no information regarding the steps count.
2.	Less effective in search method	More effective
3.	Problem to be clarified with the specified information	Additional information can be added as assumption to solve the problem
4.	E.g. 1. Depth limited search 2. Breadth first search 3. Uniform cost search 4. Depth first search 5. Interactive deepening search 6. Bi-directional search	E.g. a) Best first search b) Greedy search c) A* search

Breadth First Search (BFS)

BFS is an easy method wherein the root node is improved initially; next every incomers of the root node are extended further, afterward their incomers, and so on. Commonly, every node is extended at a specified depth within the search-tree previous to all nodes at the further level are extended.

Simple calling TREE_SEARCH (problem, FIFO-QUEUE ( )) outcomes in a breadth first search. The bellow figure displays the movement of the searching on a simple-binary tree.

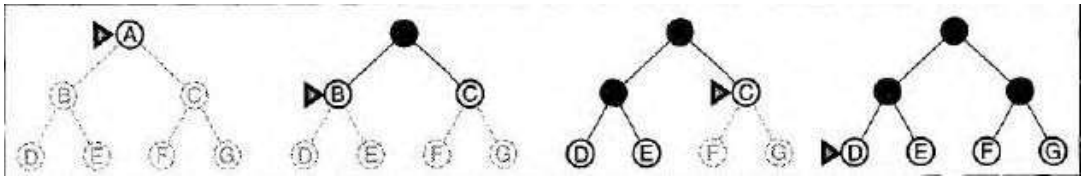

Figure: BFS on Simple-binary-tree

Time complexity is the total generated

nodes count is

$$b1+b2+b3+\ldots\ldots+bd+bd+1-b= O (bd +1)$$

Each node that is created should present in storage. The space complexity is same as the time complexity.

Advantages: Guaranteed to discover the single result at the shallowest depth point.

Disadvantages

1. The memory compulsion is a larger problem for BFS than the implementation moment.
2. Exponential complexity search problems can be unsolved by unaware systems for at all but only suitable for minimum instances problem (i.e.) (number of levels to be minimum (or) branching factor to be minimum)

Depth First Search (DFS)

It for all time extends the indepth node in the present fringe of the search-tree. If deadend take place, back tracking will be performed to the further quick before node for nodes to be extended.

Advantages: If greater than one result contains (or) levels count is more, next DFS is better because searching will be performed just in a little section of the total space.

Disadvantages: There is no guaranty to discover the solution.

Uniform - Cost Search (UCS)

The root node is extended initially, and then the next node also to be extended is chosen as the minimum cost-node on the fringe alternatively the minimum depth node i.e.

g(n) =path cost, breadth first search is equal to UCS when

g (n) =DEPTH (n).

Advantages: Guaranteed to discover the single result at lowest path cost.

Disadvantages: Only appropriate for minimum instances problem.

Depth Limited Search (DLS)

The boundless trees problem would be improved by giving DFS with decided depth limit '**L'**. i.e., at depth **L** nodes are considered, if they won't contain successors. This strategy is known as **depth limited search (DLS)**. The highest level of depth depends upon the states count.

Bidirectional Search (BDS)

It is an approach that it explores both the ways simultaneously i.e. frontward from the primary state and toward the back from the target, and ends when the 2 explores convene within the central point. It is applied by containing one or together of the explores verify every node earlier than it is extended to observe, if it is in the fringe of the order search-tree; if so, a result had been discovered.

Interactive Deepening Search (IDS)

It is a common approach that avoids the issue of selecting the finest depth limitation by attempting all available depth limitations. It joins the advantages of DFS and BFS.

1.11. Avoiding Repeated States

The repeated states can be avoided using **three** different ways. They are:

1. Does not return to the state we just approached from i.e. avoid any incomer, which is similar state of the node's parent.
2. Does not generate path with rotations i.e. sidestep any incomer of a node, which is similar of the node's family.
3. Does not create any state, which was continually created in advance.

1.12. Searching with Partial Information

When the information of the states or actions is unfinished about the environment, then only partial information is known to the agent. This incompleteness directs to three different problem categories:

1. **Sensor less problems** (**important problems**): If the agent does not having sensors at all, next it would be so many available primary states, and every action could lead to so many available successor states.

E.g. the Vacuum world problem

2. **Exploration problems**: Whenever the operations of the environment and the states are unidentified, the agent can find them. It may be considered as an severe case on contingency problems.
3. **Contingency Problems**: If the location is incompletely recognizable or if the operations are unsure, next the agent's rules give the latest record after every operation. Every probable rule describes a possibility, which should be designed for. A problem is known as adversarial, if the ambiguity is occurred by the operations of some other agent.

Question Bank

Unit - I

Part - A

1. Describe AI.
2. Describe Agent and Agent Function.
3. What is Turing Test?
4. How to evaluate the performance of an agent?
5. Write Short notes on Autonomy.
6. Give the architecture of agent in an environment.
7. What is an agent type and write the corresponding PAGE description.
8. Describe ideal Rational Agent.
9. Explain Agent Program.
10. What is Mapping. Give with an Example.
11. Explain Ideal Mapping with an example.
12. Describe Environment Program.
13. Explain the Problem Solving Agent.
14. What is meant by operators?
15. Explain the four different types of problem with one example.
16. Define State Space.
17. Define Path.
18. Define Path Cost.
19. Explain the steps in simple problem solving technique.
20. Give example for Artificial Intelligence problems.
21. Give example for real world problems in Artificial Intelligence.
22. What is the Search Tree? Explain with an example.

23. Describe Frontier or Fringe.
24. Differentiate heuristic search with blind search.
25. Describe the following terms for eight-Puzzle Problem: States, Operator, Goal-Test and Path-Cost.
26. List the steps to evaluate the performance of search strategies.
27. Write an informal explanation for the general tree search algorithm.
28. Why the problem formulation has to follow the goal formulation?
29. Differentiate Depth-First-Search, Breadth-First-Search with Best-First-Search.
30. Describe branching factor.
31. Explain the use of Heuristic Functions?

Part - B

1. Describe and Solve the bellow Problems:
   a) Missionaries and Cannibals Problem b) Water Jug Problem
2. Describe and solve the specified cryptographic mathematical problem utilizing the bellow explanations:
   i) Operator ii) State iii) Goal Test iv) Solution Process v) Initial State
3. Explain in brief about the Environment Programs with an example.
4. Describe the Vacuum World Problem and sketch the related state set-space presentations.
5. Explain in brief regarding blind-search methods with a sample example.
6. Explain the bellow uninformed search approaches with examples.
   a) Uniform Cost Search
   b) Depth first Search
   c) Depth Limited Search
   d) Breadth First Search
   e) Bidirectional Search