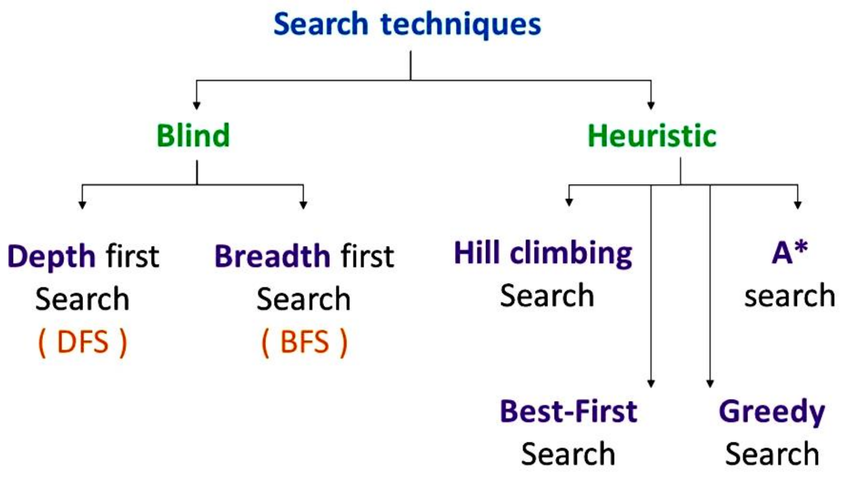
Unit –II

SOLVING PROBLEMS BY SEARCHING

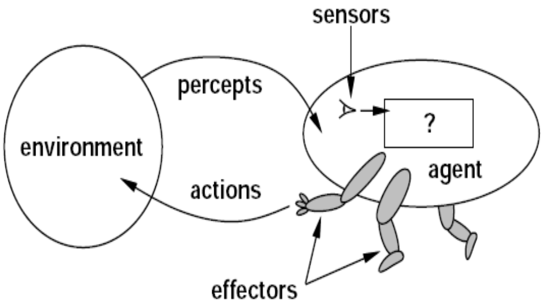
2.1 SEARCHING FOR SOLUTIONS/ WHAT IS SEARCHING?

- Searching is the most commonly used technique of problem solving in AI.
- Problem solving is a process of generating solutions from observed data.
- A problem is characterized by
  - a set of goals,
  - a set of objects, and
  - a set of operations.
- A problem consists of five parts:
  1. Initial state
  2. A set of actions
  3. A transition model describing the results of those actions,
  4. A goal test function
  5. A path cost function.
- The environment of the problem is represented by a state space.
- A path through the state space from the initial state to a goal state is a solution.
- **Types of search algorithm**
  - Uninformed search(Blind search)
  - Informed search (Heuristic search)

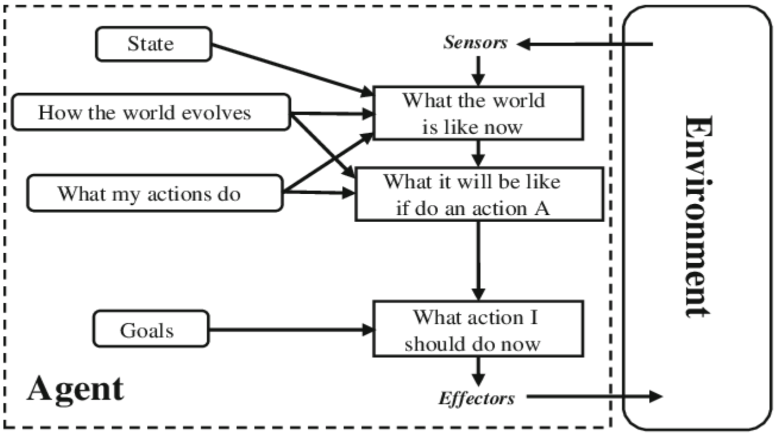


2.2 PROBLEM SOLVING AGENTS

- Definition: "An agent can be anything which perceives its environment through sensors and act upon that environment through actuators or effectors."



- In simple, an agent senses the environment and takes actions autonomously in order to achieve goals.
- An agent runs in the cycle of perceiving, thinking and acting.
- Problem-solving agents or rational agents in AI mostly use search strategies or algorithms to solve a specific problem and provide the best result.
- Problem-solving agents are the goal-based agents.
- **Goal Based agent architecture:**

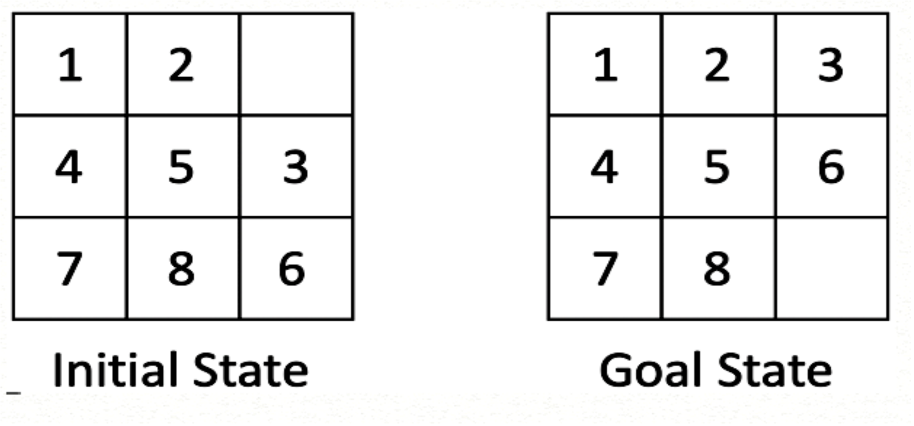

State
Sensors
How the world evolves
What the world is like now
What my actions do
What it will be like if do an action A
Goals
What action I should do now
Agent
Effectors
Environment

- These kinds of agents take decisions based on how far they are currently from their **goal** (which describes **desirable situations**).
- They choose an action, so that they can achieve the goal.
- Their every action is intended to reduce its distance from the goal.
- These considerations of different scenario are called **searching and planning**, which makes an agent proactive.
- The goal-based agent's behavior can easily be changed.

2.3 EXAMPLE PROBLEMS

- The problem-solving approach has been applied to a vast array of task environments
- Some of them are:
  - **8-puzzle problem**
  - **8-queen problem**
  - **Vacuum cleaner world**
  - **Route finding problem**
  - **Touring problem**
  - **Travelling sales person problem**
  - **Robot navigation**
  - **Automatic assembly sequencing**
- For all the above problems, the standard formulation is as follows:

  8-puzzle problem



1	2	
4	5	3
7	8	6

Initial State

1	2	3
4	5	6
7	8	

Goal State

- **States**: Location of tiles and the blank in one of the nine squares.
- **Initial state**: Any state can be designated as the initial state.
- **Actions**: Movements of the blank space *Left*, *Right*, *Up*, or *Down*.
- **Transition model**: Given a state and action, this returns the resulting state; applying the movements in the initial state to reach goal state.
- **Goal test**: This checks whether the state matches the goal configuration or not.
- **Path cost**: Each step costs 1, so the path cost is the number of steps in the path.

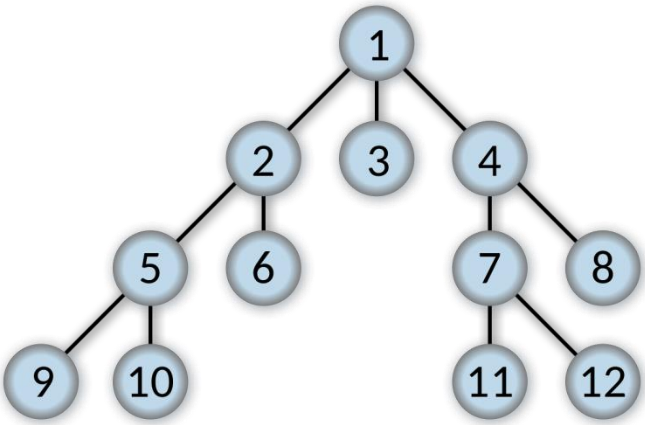
2.4 UNINFORMED SEARCH STRATEGIES (BLIND SEARCH)

- A search strategy is defined by picking the order of node expansion.
- Uninformed search strategies use only the information available in the problem definition.
- Uninformed search algorithms do not have additional information about state or search space, so it is also called **blind search.**
- This does not require information to perform search.
- This does not contain any domain knowledge such as closeness, location of the goal.
- Following are the various types of uninformed search algorithms:

1. **Breadth-first Search (BFS)**
2. **Depth-first Search (DFS)**
3. **Depth-limited Search**
4. **Uniform- cost search**
5. **Iterative deepening search**
6. **Bidirectional Search**

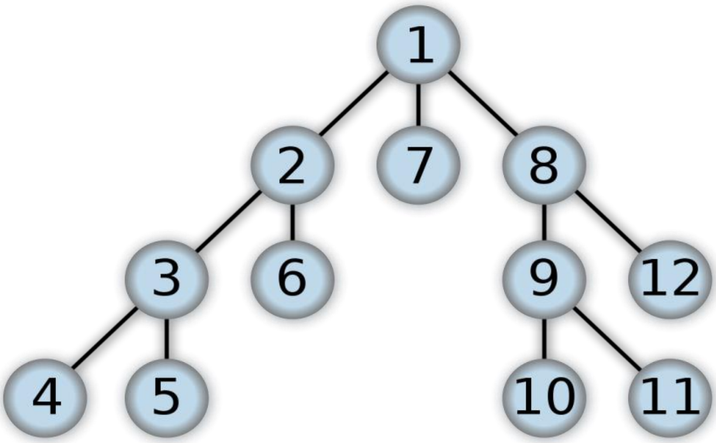
1. Breadth-first Search (BFS)

- ✓ BFS is the most common search strategy for traversing a tree or graph.
- ✓ This algorithm searches breadth-wise in a tree or graph.
- ✓ BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- ✓ The BFS algorithm is an example of a general-graph search algorithm.
- ✓ BFS uses **FIFO queue** data structure.
- ✓ BFS needs lots of time if the solution is far away from the root node.

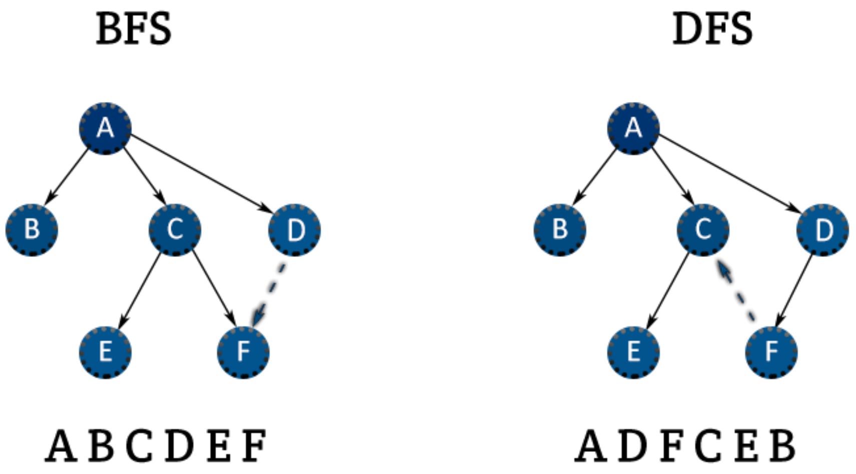


2. Depth-first Search (DFS)

- ✓ DFS is a recursive algorithm for traversing a tree or graph data structure.
- ✓ It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- ✓ DFS uses **LIFO queue** data structure.
- ✓ DFS uses a stack data structure for its implementation.
- ✓ The process of the DFS algorithm is similar to the BFS algorithm.
- ✓ DFS requires very less memory and takes less time to reach to the goal node than BFS algorithm



<u>BFS Vs DFS</u>



3. Depth-Limited Search Algorithm

- ✓ This is similar to DFS with a predetermined limit.
- ✓ This can solve the drawback of the infinite path in the Depth-first search.
- ✓ Depth-limited search is Memory efficient.
- ✓ Depth-limited search also has a disadvantage of incompleteness.
- ✓ It may not be optimal if the problem has more than one solution.

4. Uniform-cost Search Algorithm

- ✓ This is equivalent to BFS algorithm if the path cost of all edges is the same.
- ✓ This algorithm comes into play when a different cost is available for each edge.
- ✓ The primary goal is to find a path to the goal node which has the lowest cumulative cost.
- ✓ It can be used to solve any graph/tree where the optimal cost is in demand.
- ✓ This algorithm may be stuck in an infinite loop.

5. Iterative deepening Search

- ✓ This is a combination of DFS and BFS algorithms.
- ✓ This combines the benefits of BFS's fast search and DFS's memory efficiency.
- ✓ This search algorithm finds out the best depth limit.
- ✓ This is useful when search space is large, and depth of goal node is unknown.
- ✓ The main drawback is that it repeats all the work of the previous phase.

6. Bidirectional Search Algorithm

- ✓ Bidirectional search can use search techniques such as BFS, DFS etc.
- ✓ This search replaces one single search graph with two small sub-graphs.
- ✓ The search stops when these two graphs intersect each other.
- ✓ Bidirectional search is fast.
- ✓ Bidirectional search requires less memory

2.5 INFORMED (HEURISTIC) SEARCH STRATEGIES

- ➢ The informed search algorithm is more useful for large search space.
- ➢ Informed search algorithm uses the idea of heuristic, so it is also called Heuristic search.
- ➢ Informed Search algorithms have information on the goal state which helps in more efficient searching.
- ➢ Following are the various types of informed search algorithms:
    1. Hill climbing search
    2. Travelling salesman search
    3. 8-puzzle search
    4. N-queen search
    5. Best first search
    6. Greedy Best first search
    7. A* search

1.Hill climbing search

- ✓ Hill Climbing is a heuristic search used for mathematical optimization problems in the field of Artificial Intelligence.
- ✓ Given a large set of inputs and a good heuristic function, it tries to find a sufficiently good solution to the problem.

2.Travelling salesman search

- ✓ The traveling salesman problem (TSP) is an algorithmic problem tasked with finding the shortest route between a set of points and locations that must be visited.
- ✓ Focused on optimization, TSP is often used in computer science to find the most efficient route for data to travel between various nodes.

3. 8 puzzle search

- ✓ An 8 puzzle is a simple game consisting of a 3 x 3 grid (containing 9 squares). One of the squares is empty.
- ✓ The object is to move to squares around into different positions and having the numbers displayed in the "goal state".

4. N queen search

- ✓ The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.
- ✓ The chess queens can attack in any direction as horizontal, vertical, horizontal and diagonal way.

5. Best first search

- ✓ Best-first search is a class of search algorithms, which explore a graph by expanding the most promising node chosen according to a specified rule.
- ✓ The idea of Best First Search is to use an evaluation function to decide which adjacent is most promising and then explore.

6. Greedy Best first search

- ✓ It is the combination of BFS and DFS algorithms.
- ✓ Greedy best-first search algorithm always selects the path which appears best at that moment.

7. A* Search

- ✓ A * algorithm is a searching algorithm that searches for the shortest path between the initial and the final state.
- ✓ A* search is a combination of lowest-cost-first and best-first searches.
- ✓ This considers both path cost and heuristic information in its selection.

2.6 HEURISTIC FUNCTIONS

- ➢ Heuristic is a function which is used in Informed Search, and it finds the most promising path.
- ➢ A heuristic function is a shortcut for solving a problem when there are no exact solutions for it or the time to obtain the solution is too long.
- ➢ Heuristic search –
  - Tries to optimize a problem using heuristic function.
  - Tries to solve problem in minimum steps/cost.
- ➢ Heuristic function –
  - It is a function "**h(n)**" that gives an estimation on the cost of getting from node "n" to the goal state.
  - It helps in selecting optimal node for expansion.
- ➢ The performance of heuristic search algorithms depends on the quality of the heuristic function.

2.7 LOCAL SEARCH ALGORITHMS AND OPTIMIZATION PROBLEMS

- ➢ In computer science, local search is a heuristic method for solving computationally hard optimization problems.
- ➢ Optimization mainly focuses on minimizing cost and maximizing the value.
- ➢ Local search algorithms move from solution to solution in the search space by applying local changes, until an optimal solution is found.
- ➢ Local search use single current state and move neighboring states.
- ➢ **Example : Hill climbing search,,8 queens problem.**

2.8 SEARCHING WITH NON DETERMINISTIC ACTIONS

- In majority of the applications, it is assumed that the environment is fully observable and deterministic and that the agent knows what the effects of each action are.
- Therefore, the agent can calculate exactly which state results from any sequence of actions and always knows which state it is in.
- Its percepts provide no new information after each action.
- When the environment is either partially observable or non-deterministic, percepts become useful.
- When the environment is nondeterministic, percepts tell the agent which of the possible outcomes of its actions has actually occurred.
- **Example : Vaccum World.**

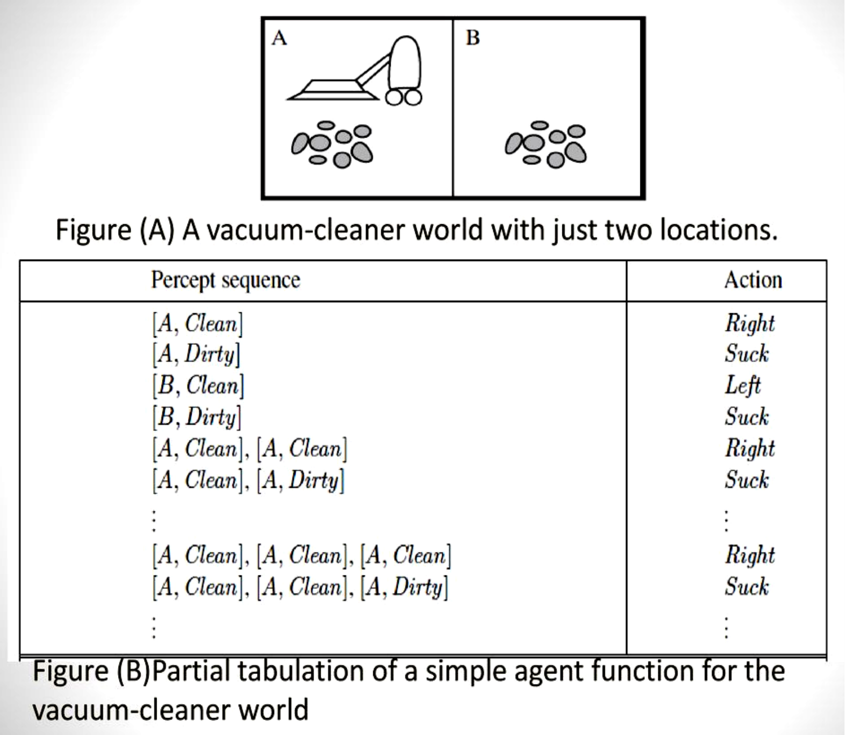


Figure (A) A vacuum-cleaner world with just two locations.

Percept sequence	Action
[A, Clean]	*Right*
[A, Dirty]	*Suck*
[B, Clean]	*Left*
[B, Dirty]	*Suck*
[A, Clean], [A, Clean]	*Right*
[A, Clean], [A, Dirty]	*Suck*
⋮	⋮
[A, Clean], [A, Clean], [A, Clean]	*Right*
[A, Clean], [A, Clean], [A, Dirty]	*Suck*
⋮	⋮

Figure (B)Partial tabulation of a simple agent function for the vacuum-cleaner world

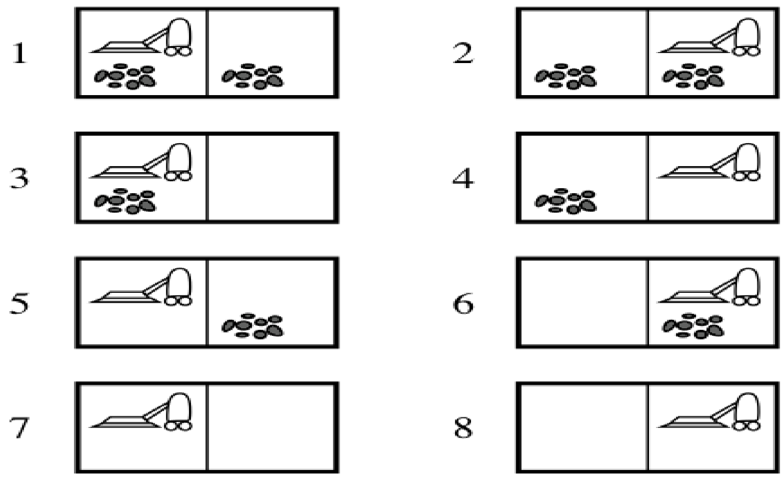


Fig (C). 8 possible states of vacuum world; states 7 and 8 are goal states.

2.9 SEARCHING WITH PARTIAL OBSERVATIONS

- In this the agent does not always know its state.
- Instead, it maintains a belief state: a set of possible states.
- **Example:** A robot can be used to build a map of the environment. It will have sensor that allow it to "see" the world.
- There can be 3 types of incompleteness lead to problems
  - 1. Sensorless problems – the agent has no sensors.
  - 2. Contingency problems – environment is partially observable or actions are uncertain.
  - 3. Exploration problems – both state & actions of the environment are unknown.

2.10 ONLINE SEARCH AGENTS AND UNKNOWN ENVIRONMENTS

- Online search is a necessary idea for unknown environments, where the agent does not know what states exist or what its actions do.
- **Example: Roomba Vacuum cleaner**
- The agents using OFFLINE SEARCH algorithms will compute a complete solution before setting foot in the real world and then execute the solution.
- But ONLINE SEARCH agent provides computation and action: first it takes an action, then it observes the environment and computes the next action.

Offline Search	**Online Search**
Knows the "map" of the situation	Doesn't Know the "map" of the situation.
Basically finds the shortest path knowing the whole layout of the situation.	Has to explore and find out where to go, and then determine the shortest path.
An offline algorithm knows all about its input data the moment it is invoked.	An online algorithm can get parts or all of its input data while it is running.
Works like a GPS navigation system	Works like a Roomba Vacuum cleaner.

- Online search is a good idea in dynamic domains.
- Online search is also helpful in nondeterministic domains.