# Assignment-01

1. Compare and contrast uniformed search strategies (such as Breadth-First search, Uniform cost search, Depth-First search) with informed search strategies interms of efficiency, completeness, and optimality.

**Ans:** Uninformed search strategies, like Breadth-First search (BFS), Depth-First Search (DFS) and Uniform cost search (UCS), explore the search space without any prior knowledge of the solution's location. Informed search strategies, on the other hand, utilize a heuristic function to guide their search, potentially leading to more efficient and optimal solutions, especially in complex problem spaces.

**Uninformed Search strategies:**

**Breadth-First Search (BFS):**

Explores the search space level by level, guarenteeing the shortest path in an unweighted graph but potentially requiring significant memory.

**Depth-First Search (DFS):**

Explores as deep as possible along each branch before backtracking, generally more memory-efficient than BFS but may not find the shortest path and can get stuck in infinite loops in cyclic graphs.

**Uniform Cost Search (UCS):**

Expands nodes based on their cumulative cost, guaranteeing the optimal path in weighted graphs,

but can be less efficient than BFS or DFS in some cases.

## Informed Search Strategies:

Informed Search strategies algorithms utilize a heuristic function to estimate the cost from a given node to the goal, guiding the search towards promising paths.

| Feature | Uninformed Search | Informed Search |
|---|---|---|
| Efficiency | Generally less efficient, especially in large (or) complex spaces | Generally more efficient, potentially finding solutions faster. |
| Completeness | BFS is complete (finds a solution if it exists). DFS can be incomplete (can get trapped in infinite loops). | A* is complete if the heuristic is admissible (never overestimates the cost to the goal) |
| Optimality | BFS guarantees the shortest path in unweighted graphs, UCS guarantees the shortest path in unweighted graphs | A* guarantees the optimal path if the heuristic is admissible. |
| Memory usage | BFS can be memory-intensive, DFS can be more memory-efficient. | Memory usage varies depending on the specific algorithm and the search space. |
| Guidance | No guidance, relies on systematic exploration | Uses a heuristic function a guide the search. |

8. Explain the working principles of hill-climbing search and simulated annealing search. Illustrate with an example how simulated annealing overcomes the limitation of hill-climbing.

Hill-climbing search:

It is a local search Algorithm which continuously moves in the direction increasingly elevation/value to find the peak of the mountain (or) best solution of problem. It start with an arbitrary solution and, in each iteration; explores its neighbours. If a neighbour provides a better value for the objective function, the algorithm moves to that neighbor and repeats the process. It stops when no neighboring solution yields a better objective function value, indicating a local optimum has been reached. This approach is greedy as it always moves to the best available neighbor.

Algorithm:-

Step-1: Evaluate the initial state, if it is goal state then return success and stop.

Step-2: Loop until a solution is found or there is no new operator left to apply.

Step-3: select and apply an operator to the current state.

Step-4: Check new state:

1. If it is goal state, then return success and quit.
2. Else if it is better than the current state then assign new state as a current state.
3. Else if not better than the current state, then return to step 2.

Step-5: Exit.

## Simulated Annealing Search:-

It is a probabilistic optimization algorithm inspired by the metallurgical annealing process. In this, the algorithm starts with an initial solution and a "temperature parameter. It explores neighboring solutions and, similar to hill-climbing, accepts better solutions. However, it also accepts worse solutions with a certain probability. This probabilistic acceptance allows the algorithm to escape local optima and explore a broader search space.

## Algorithm:-

Step-1: Evaluate the initial state, if it is goal state then return success and stop.

Step-2: Loop Until a solution is found or there is no new operator left to apply

Step-3: Select and apply an operator to the current state.

step-4: Check new state:

1) If it is goal state, then return success and quit

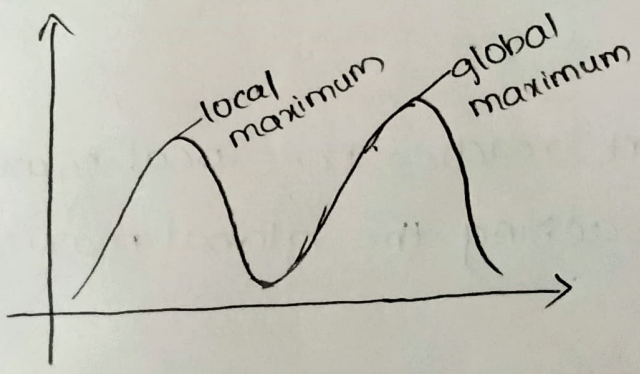
2) Else if it is better than the current state then assign new state as a current state.

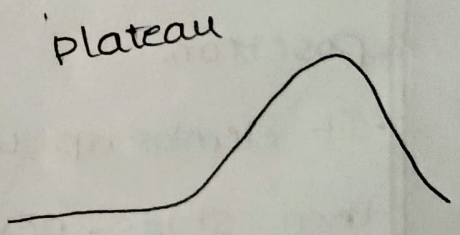
3) Else if not better than the current state, then return to step 2

step-5: Exit

## Limitations of Hill-climbing:-

- At each step, it moves to the neighbor with the best improvement.
- If it reaches a local maximum (or plateau), it gets stuck, even if a better global maximum exists elsewhere.




## How Simulated Annealing overcomes limitations of Hill-climbing:-

Simulated Annealing is inspired by the process of cooling metals:

- It sometimes accepts worse moves (downhill) with

a probability that decreases as "temperature" decreases.

- This allows it to escape local maxima early on and hopefully converge to the global maximum.

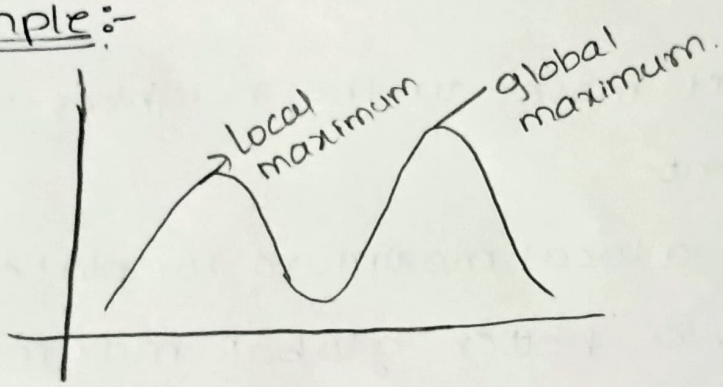
The acceptance probability of a worse move is:

$$P = e^{-\Delta E/T}$$

$\Delta E \rightarrow$ drop in value

$T \rightarrow$ current "Temperature".

Example:-




- The hill climbing algorithm starts at the leftmost position.
- It climbs up until it reaches the local maximum, then stops - never reaching the global maximum.

3. Design a strategy for solving a CSP like sudoku using Constraint Propagation and backtracking. Highlight how constraint propagation can reduce the search space.

## Problem formulation:-

- **Variables:-** Each empty cell in the Sudoku grid is a variable.
- **Domain:-** The domain for each variable is the set of possible digits $\{1, 2, \ldots, 9\}$.
- **Constraints:-**

**Row Constraint:** Each digit appears exactly once in each row.

**Column constraints:** Each digit appears exactly once in each column.

**Block Constraint:** Each digit appears exacly once in each 3x3 sub-grid.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | 2 | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 9 |
| | | | | 8 | | | 7 | 5 |

## Constraint Propagation:-

→ Initially, for each empty cell, its domain contains all digits 1-9

→ When a value is assigned to a cell (either initially given or during the search), propagate

this assignment related cells.

- For example, if a cell is assigned with the value '5', then '5' is removed from the domains of all other unassigned cells in that row, column, and 3x3 subgrid.
- This process continues iteratively until no more domain reductions are possible.

Backtracking:-

→ Choose an unassigned value. Heuristic like Minimum Remaining Values can be used to select the variable with smallest domain, potentially leading to earlier detection of inconsistencies.

→ Iterate through the values in the chosen variable's domain.

→ For each value, tentatively assign it to the variable. Then, perform constraint propagation to update the domains of related variables

→ If after the propagation, any variable's domain becomes empty, the current assignment is inconsistent. Backtracking to the previous decision point and try a different value.

→ If the assignment is inconsistent, recursively call the solver for the next unassigned variable.

→ If all variables are assigned with values without inconsistency, a solution is found.

Reduction of Search Space Using Constraint Propagation:-

Constraint Propagation significantly reduces the search space by eliminating inconsistent values from variable domains before backtracking begins. This proactively prunes branches that would otherwise lead to dead ends during the search, making the backtracking process more efficient.

4. Analyze the advantages and limitations of backtracking and local search strategies in solving Constraint Satisfaction Problems (CSPs). In what types of problems is each strategy more suitable?

Backtracking:-

→ Advantages:-

- Completeness: Backtracking guarantees finding a solution if one exists by systematically exploring the entire search space.
- Simplicity: It's relatively easy to implement and understand.
- Flexibility: It can handle various constraint types and problem structures.

→ Limitations:-

- Inefficiency for large problems: The exponential nature of the search space can make it slow for

complex CSPs. with many variables and tight constraints.

- Redundancy: without proper optimization, it might explore the same invalid states multiple times.

Local Search:-

Advantages:

- Efficiency for large problems: It explores the search space locally, making it suitable for large, complex problems where systemic methods like backtracking struggling.
- Low memory usage: Typically requires constant memory, making it suitable for very large problem.

Limitations:-

- Not complete: It might not find a solution even if one exists, and it can get stuck in local optima.
- Can be sensitive to initial state: The final solution can be depend on the initial assignment of variables.

Suitable Problems on Backtracking:-

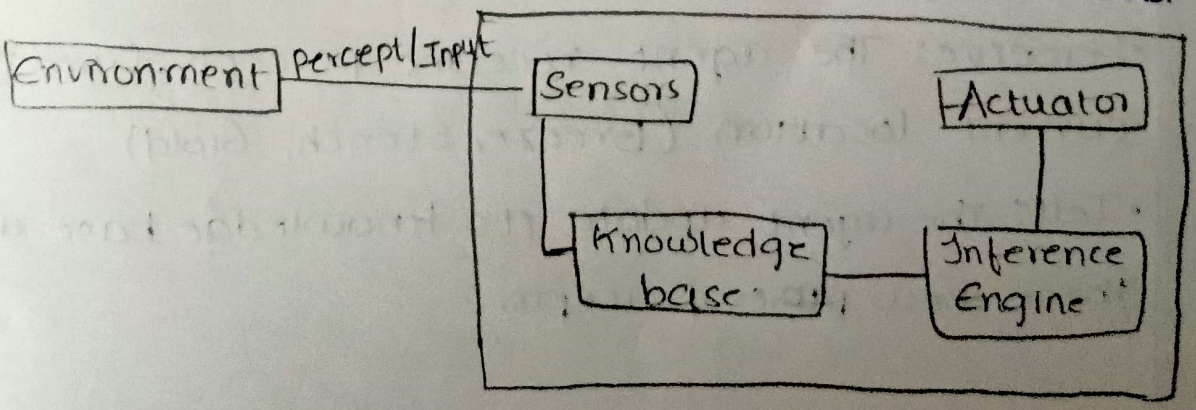
- Problems with a relatively small variables and constraints, where completeness is crucial, and finding the optimal solution is not the primary concern.
- Examples: Crossword puzzles, Sudoku.

Suitable Problems (local search):

- Problems with a very large number of variables and constraints, where finding an approximate solution quickly is more important than finding the optimal solution.
- Problems with continuous variables or infinite state spaces.
- Examples: Scheduling problems, routing problems, and some optimization problems where an approximate solution is acceptable.

5. Explain the structure and behaviour of a knowledge based agent in the context of the wumpus world. How does it use logical inference to make decisions? Explain in detail with an example.

Ans: A knowledge-based agent in the wumpus world is an intelligent agent that uses a knowledge base to store information about its environment and employs logical inference to make decisions.



Structure:-

Knowledge Base is a central component hrding a collection of logical sentences (facts and rules) representing the agent's understanding of the Wumpus world. This includes:

- Facts about the environment: e.g., "There is no breeze in (1,1)", "Agent is at (1,1)".
- Rules about the Wumpus world physics: e.g., "If there is a pit in (x,y), then there is a breeze in all adjacent squares;" "If there is a Wumpus in (x,y), then there is a stench in all adjacent squares."
- Agent's goals: eg., "find the gold", "Exit safely".

Inference Engine: This component applies logical inference rules (e.g., Modus Ponens, resolution) to the knowledge base to derive new conclusions or determine if a preposition is true or false.

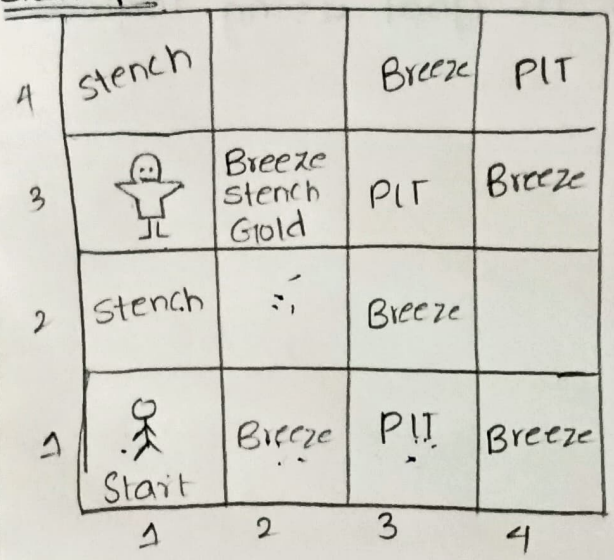
Behaviour:-

The agent operate in loop:

- Perceive: The agent receives percepts from its current location (Breeze, Stench, Gold)
- Tell: The agent update its knowledge base with these new percepts.

- Ask: The agent queries the knowledge base to determine safe moves, locate hazards, or identify the gold.
- Infer and Decide: The Inference engine uses logical reasoning to deduce new information and decide the best action (move, shoot, grabgold, climbout)
- Act: The agent executes the chosen action.

Example:-




Consider an agent starting at (1,1) in the Wumpus world.

Agent at (1,1). No percepts (no Breeze, no Stench)

KB: At(Agent, 1,1), -Breeze(1,1), -Stench(1,1)

Inference: Since there is no Breeze or Stench in (1,1) the agent can infer (1,2) and (2,1)

Move to (2,1):

Agent moves to (2,1) and perceives a Breeze.

KB update: At(Agent, 2,1), Breeze(2,1)

• Inference:

Since Breeze(2,1) is true and (1,1) is known to be OK, the pit must be either in (2,2) or (3,1). The agent can infer Pit(2,2) or Pit(3,1).

This iterative process of perceiving, updating the KB, and inferring new knowledge allows the agent to navigate the Wumpus world, avoid dangers, and achieve its goal using logical deduction.