Local Search and Optimization: Easy Explanation

Local search and optimization are important concepts in AI, especially when dealing with large problems where traditional search methods (like BFS and DFS) are inefficient. Instead of looking at all possible paths, local search **only keeps track of the current state** and tries to find a better solution by moving to its **neighboring states**.

Local Search

- **How does it work?**
  - Keeps track of **only one** current state.
  - Moves **only to neighboring states**.
  - **Ignores paths** (does not keep a full path history like A*).
- **Advantages of Local Search:**
  - Uses **very little memory**.
  - Can often find **good solutions in large or infinite state spaces**.
- **When is it used?**
  - Best for **"pure optimization"** problems where the goal is to find the **maximum or minimum** of an objective function.
  - Example: **Traveling Salesperson Problem (TSP), N-Queens problem, Exam Scheduling, VLSI design**.

Local Search in Optimization Problems

- Used for **very large problems**.
- Returns a **good but not necessarily optimal** solution.
- Example: **Traveling Salesperson Problem (TSP)**:
  - The state space consists of **"complete"** configurations (e.g., all possible city tours).
  - The goal is to find a **near-optimal** solution **satisfying constraints**.

Example: 4-Queens Problem

- **Goal:** Place 4 queens on a 4×4 chessboard without attacking each other.
- **State Space:** Every configuration where each column has exactly one queen.
- **State Transition:** Move a queen to a different row in its column.
- **Local Search Approach:**
  - Start with a **random configuration**.
  - Move the queens around to **reduce conflicts**.
  - Repeat until a valid solution is found.

Hill Climbing Algorithm

A **hill-climbing algorithm** is a **local search algorithm** that continuously moves **upward (increasing)** until the **best solution** is attained. The algorithm stops when the **peak is reached**.

How It Works

- The algorithm has a **node** that consists of two parts: **state and value**.
- It starts with a **non-optimal state** (the hill's base) and **improves** this state until a **precondition** is met.
- The **heuristic function** is used as the basis for determining this precondition.
- The process of **continuous improvement** of the current state is called **climbing**, which explains why the algorithm is known as **hill climbing**.

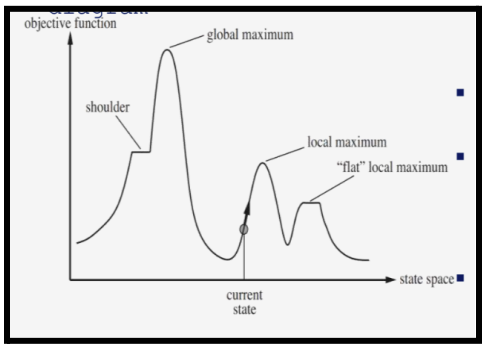
Key Features of Hill Climbing

1. **Greedy Approach** – Always moves towards a better solution.
2. **No Backtracking** – Does not revisit previous states.
3. **Feedback Mechanism** – Uses feedback to decide the **direction of movement**.
4. **Incremental Change** – Improves the **current solution** step by step.

Problems in Hill Climbing Algorithm

Hill climbing suffers from several challenges that can prevent it from finding the best solution. These include **local maxima, plateaus, and ridges**.

1. **Local Maximum**
    - A **local maximum** is a solution that is **better than neighboring solutions** but **not the best possible solution** (global maximum).
    - **Problem:** Once hill climbing reaches a **local maximum**, it stops, even though a **better solution** may exist elsewhere.

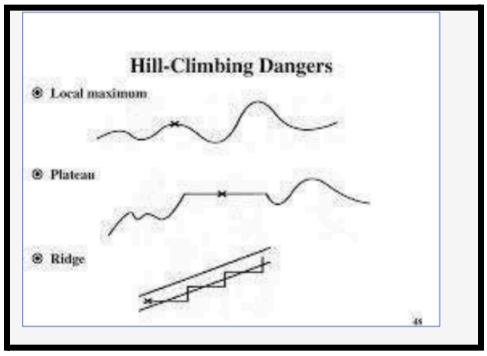


2. **Current State**
    - Refers to the **present state** of the algorithm during the search process.
3. **Flat Local Maximum (Plateau)**
    - A **flat region** where **all neighboring solutions have the same value**.
    - **Problem:** The algorithm **cannot determine the best direction** to move.
4. **Shoulder**
    - A **plateau with an upward edge**, which may eventually lead to a **better solution**.

5. **Global Maximum**

    - The **best possible solution** in the entire search space.

<u>Challenges in Hill Climbing</u>



- **Local Maximum:**
    - When the algorithm reaches a **local maximum**, it stops because all **neighboring states have lower values**.
    - The **greedy approach** prevents the algorithm from moving to a **worse state** temporarily to find a better one later.
- **Plateau:**
    - In this situation, the **neighboring states have the same value**, making it difficult for the algorithm to decide on the best direction.
- **Ridge:**
    - A **ridge is a narrow peak** where **small moves lead downward**.
    - The algorithm **fails to climb higher** because it cannot make large jumps.

<u>Simulated Annealing</u>

Simulated Annealing Algorithm

Simulated annealing is a search algorithm that helps **escape local maxima** by allowing some **"bad" moves** initially but gradually decreasing their probability over time.

Key Idea

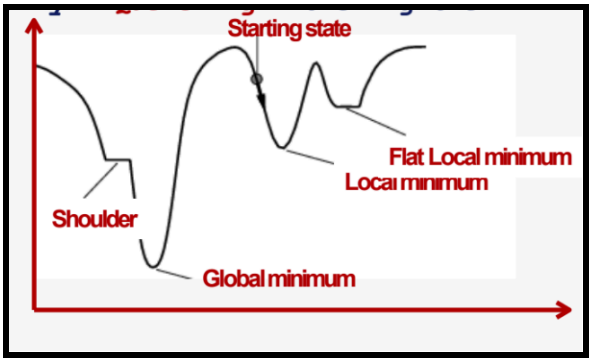
- The probability of making a **bad move** is controlled by a parameter called **temperature (T)**.
- **Higher temperatures** → Allow **more bad moves**, increasing exploration.
- **Lower temperatures** → Reduce bad moves, focusing on finding an optimal solution.
- **Annealing:** The process of **gradually lowering the temperature**.
- **Quenching:** The process of **rapidly lowering the temperature**.

How Simulated Annealing Works

- The algorithm starts with a **high temperature** to allow exploration.
- Gradually, the temperature **decreases**, reducing the probability of moving to worse states.
- The goal is to **escape local minima** and **move toward the global minimum**.

Probability of Movement

- If a new state has **lower energy (better solution)**, it is always accepted.
- If a new state has **higher energy (worse solution)**, it may still be accepted with some probability, given by: $P=e^{-\Delta E/TP}$ where **ΔE** is the difference in energy (cost), and **T** is the temperature.
- If **E1 > E2**, moving from **A to C** is exponentially **more probable** than moving from **A to B**.



Properties of Simulated Annealing

1. If **T decreases slowly enough**, simulated annealing will find the **global optimum** with probability **approaching 1**.
2. In practice, a **temperature schedule** is used to balance speed and accuracy, often settling for a **sub-optimal solution**.
3. Simulated annealing works **very well in real-world applications**, including:
    - **VLSI layout** (chip design).
    - **Airline scheduling**.
    - **Optimization problems in AI**.

Local Beam Search

- Unlike **hill climbing**, local beam search **maintains multiple states at the same time**.
- **How does it work?**
    - Start with **k random states**.
    - At each step, generate **all successors**.
    - Keep the **k best successors** and repeat.
    - Stops when a **goal state** is found.

Genetic Algorithms

A **biologically inspired** optimization method.

- **How does it work?**
    - **Generate an initial population** of random solutions.
    - **Evaluate** solutions using a **fitness function**.
    - **Select** the best individuals for reproduction.
    - **Crossover** – Combine two parents to create new solutions.
    - **Mutation** – Randomly modify a small part of a solution.
    - **Repeat** until an optimal solution is found.
- **Applications:**

- **AI, Machine Learning, Robotics, Game AI, Evolutionary Computing**.

Comparison of Local Search Algorithms

- **Local search** is efficient for **large optimization problems**.
- **Hill climbing** is simple but has **local maxima issues**.
- **Simulated annealing** helps escape **local maxima**.
- **Local beam search** and **genetic algorithms** are more advanced but need **more resources**.

Comparison of Local Search Algorithms

Algorithm	Key Idea	Strength	Weakness
Hill Climbing	Moves towards the best neighbor	Fast, simple	Stuck in local maxima
Simulated Annealing	Allows bad moves initially	Escapes local maxima	Slower than hill climbing
Local Beam Search	Keeps multiple states	More robust	Needs more memory
Genetic Algorithm	Evolves solutions over generations	Very flexible	Computationally expensive