MANJARA CHARITABLE TRUST
RAJIV GANDHI INSTITUTE OF TECHNOLOGY, MUMBAI

EXPERIMENT 5

ROLLNO:A608

DATE:

AIM : Implementation of Hill climbing

THEORY:

Introduction

Hill Climbing is a heuristic optimization algorithm used for mathematical and artificial intelligence problems. It is a local search algorithm that iteratively moves towards the optimal solution by selecting the neighboring state with the highest value. This algorithm is widely used in function optimization, robotics, and artificial intelligence applications such as game playing and machine learning.

Theory Behind Hill Climbing

Hill Climbing operates on the principle of iterative improvement, making it a **greedy** algorithm. The goal is to find the best possible solution by incrementally adjusting the current state and evaluating the objective function. The algorithm terminates when it reaches a peak where no further improvements are possible.

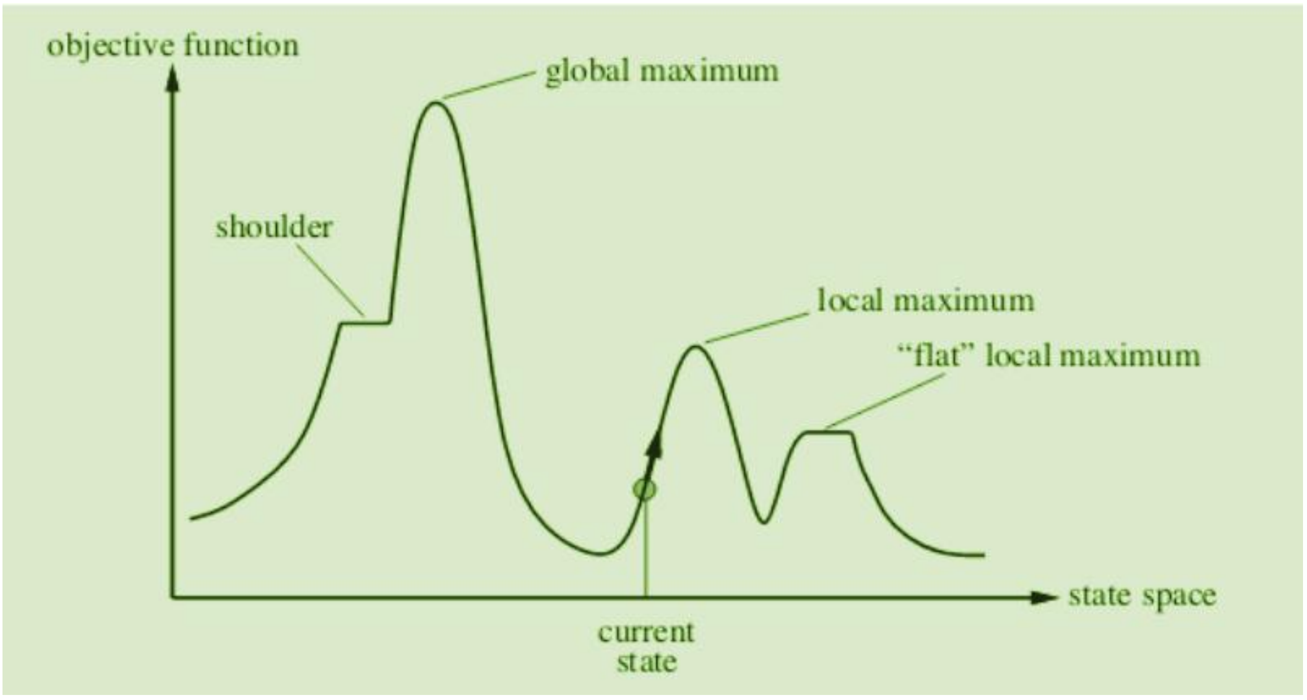
Types of Hill Climbing

1. **Simple Hill Climbing**: Evaluates a neighboring state and moves if it is better than the current state.
2. **Steepest-Ascent Hill Climbing**: Examines all possible neighboring states and selects the one with the highest improvement.
3. **Stochastic Hill Climbing**: Randomly chooses a neighbor and moves if it improves the objective function.

Algorithm Implementation

Below is a step-by-step breakdown of the Hill Climbing algorithm:

1. **Initialize** the algorithm with a random or predefined state.
2. **Evaluate** the objective function at the current state.
3. **Generate** neighboring states.
4. **Select** the best neighboring state.
5. **Move** to the new state if it improves the objective function.
6. **Repeat** until no improvement is found.



Advantages and Disadvantages

Advantages:

1. Simple and easy to implement.
2. Requires less computational power compared to other algorithms.
3. Works well for problems with smooth and unimodal functions.

Disadvantages:

1. **Local Maxima**: The algorithm can get stuck at a suboptimal peak.
2. **Plateaus**: If there is no significant change in neighboring states, the algorithm may stagnate.
3. **Ridges**: The algorithm may struggle to move in certain directions due to steep or flat slopes.

Applications of Hill Climbing

1. **Function Optimization**: Used to find optimal parameters in mathematical models.
2. **Pathfinding**: Helps robots and AI agents navigate through optimal routes.
3. **Game AI**: Used in strategy games for decision-making.
4. **Machine Learning**: Applied in feature selection and hyperparameter tuning.

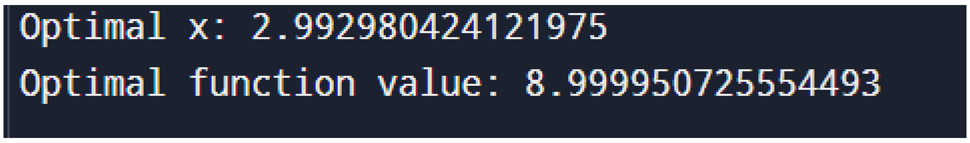
Code Implementation:

```
import random

# Objective function: A simple quadratic function with a maximum at x = 3

def objective_function(x):
    return -1 * (x - 3) ** 2 + 9
# Function to generate neighbors: Small perturbations around the current state
def generate_neighbors(x, step_size=0.1, num_neighbors=10):
    return [x + random.uniform(-step_size, step_size) for _ in range(num_neighbors)]
# Hill Climbing algorithm
def hill_climbing(f, x0, step_size=0.1, max_iterations=1000):
    x = x0  # Initial solution
    for _ in range(max_iterations):
        neighbors = generate_neighbors(x, step_size)
        best_neighbor = max(neighbors, key=f)
        if f(best_neighbor) <= f(x):  # If no improvement, return current solution
            return x
        x = best_neighbor  # Move to the best neighbor
    return x
# Example usage
if __name__ == "__main__":
    initial_guess = 0  # Starting point
    optimal_x = hill_climbing(objective_function, initial_guess)
    print("Optimal x:", optimal_x)
    print("Optimal function value:", objective_function(optimal_x))
```

Output:



```
Optimal x: 2.992980424121975
Optimal function value: 8.999950725554493
```

Conclusion

Hill Climbing is a fundamental algorithm in optimization that provides a straightforward approach to finding local optima. While it has its limitations, modifications such as **random restarts** and **simulated annealing** help mitigate these issues. It remains a powerful tool in artificial intelligence and computational problem-solving.