Name: Jigar Siddhpura **SAPID:** 60004200155

DIV: C/C2 **Branch:** Computer Engineering

AI EXPERIMENT 4 - Hill Climbing

Aim: Implement Hill Climbing Algorithm

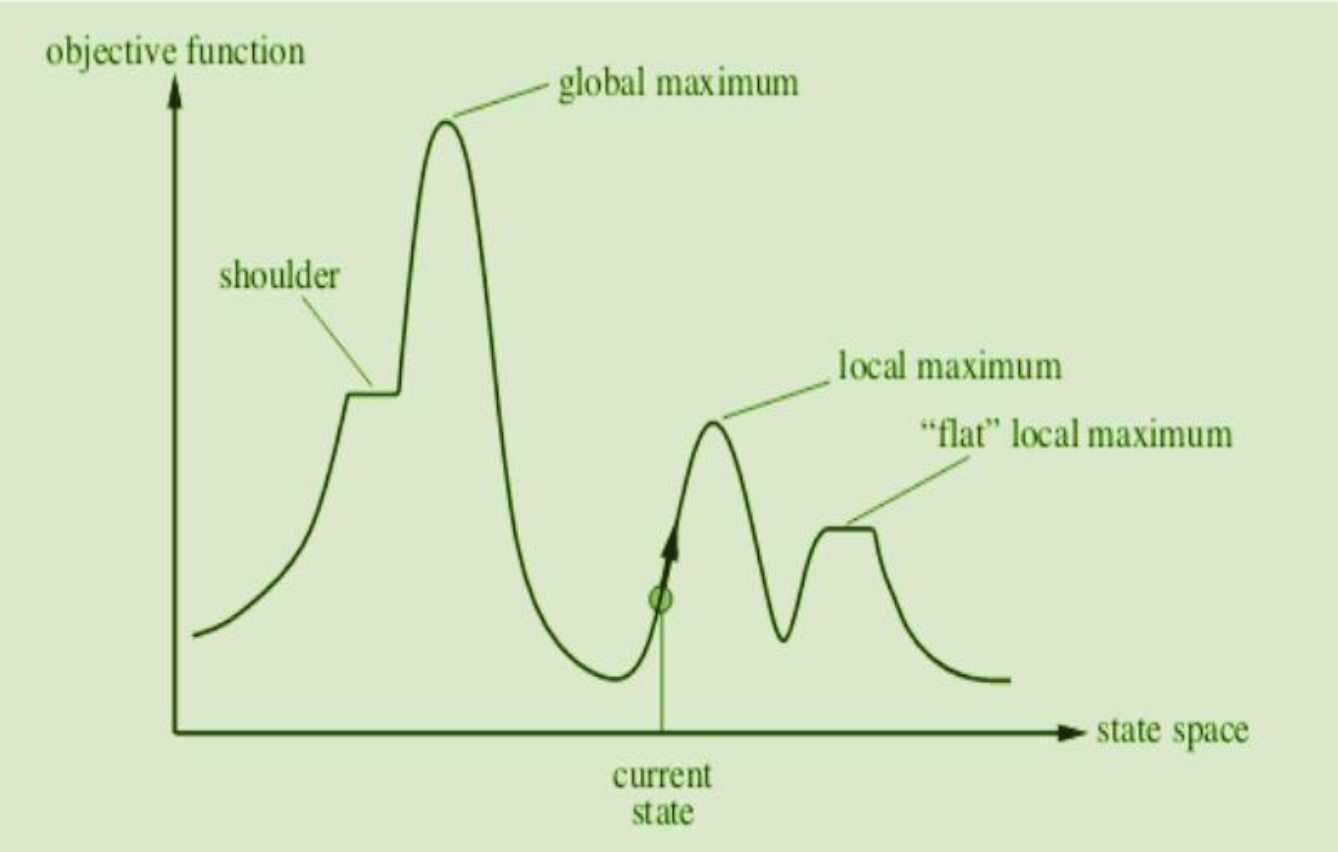
Theory:

Hill climbing algorithm is a local search algorithm which continuously moves in the direction of increasing elevation/value to find the peak of the mountain or best solution to the problem. It terminates when it reaches a peak value where no neighbour has a higher value. Hill climbing algorithm is a technique which is used for optimizing the mathematical problems. One of the widely discussed examples of Hill climbing algorithm is **Traveling-salesman Problem** in which we need to minimize the distance travelled by the salesman.

It is also called **greedy local search** as it only looks to its good immediate neighbour state and not beyond that. A node of hill climbing algorithm has two components which are state and value. Hill Climbing is mostly used when a good heuristic is available. In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state.

Features:

1. *Generate and Test variant*: Hill Climbing is the variant of Generate and Test method. The Generate and Test method produce feedback which helps to decide which direction to move in the search space.
2. *Greedy approach:* Hill-climbing algorithm search moves in the direction which optimizes the cost.
3. *No backtracking:* It does not backtrack the search space, as it does not remember the previous state



Disadvantages of hill climbing algorithm :

1. *Local Maximum*: A local maximum is a peak state in the landscape which is better than each of its neighbouring states, but there is another state also present which is higher than the local maximum

2. *Plateau*: A plateau is the flat area of the search space in which all the neighbour states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area

3. *Ridges*: A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

Code :

```
import copy

visited_states: list = []

def generate_child_state(current_state :list, prev_heuristic :int, goal_state :list) -> list|int :
    """Generates a child state by moving an element from one peg to another."""

    global visited_states
    state_copy = copy.deepcopy(current_state)

    for i in range(len(state_copy)):
        temp_state = copy.deepcopy(state_copy)

        if len(temp_state[i]) > 0:

            # .pop remove last element from the list and returns it *
            element = temp_state[i].pop()

            for j in range(len(temp_state)):
                new_state = copy.deepcopy(temp_state)

                if j != i:
                    new_state[j] = new_state[j] + [element]
                    current_heuristic = calculate_heuristic(
                        new_state, goal_state)

                    if current_heuristic > prev_heuristic:
                        child_state = copy.deepcopy(new_state)
                        return child_state

    return 0

def calculate_heuristic(current_state :list, goal_state :list) -> int:
    """Calculates the heuristic value for the given state compared to the goal state."""

    goal_positions = goal_state[3]
    heuristic_value = 0

    for i in range(len(current_state)):
        check_values :list = current_state[i]

        if len(check_values) > 0:
            for j in range(len(check_values)):
                if check_values[j] != goal_positions[j]:
                    heuristic_value -= j
                else:
                    heuristic_value += j

    print(f"Heuristic value for {current_state} is {heuristic_value}")
    return heuristic_value

def solve_puzzle(initial_state, goal_state):
    global visited_states

    if initial_state == goal_state:
        print(f"Solution found! {goal_state}\n")
        Return

    # create deepcopy to prevent changes in the NESTED structure of the ORIGINAL list
    current_state = copy.deepcopy(initial_state)

    while True:
```

```
        visited_states.append(copy.deepcopy(current_state))
        print(f"Current State: {current_state}")


        prev_heuristic = calculate_heuristic(current_state, goal_state)
        child = generate_child_state(current_state, prev_heuristic, goal_state)

        if child == 0:
            print(
                f"No better heuristic value is obtained, declaring this as the goal state - {current_state}\n"
            )
            Return

        print(f"Child chosen for exploration: {child}\n")
        current_state = copy.deepcopy(child)


def main():
    global visited_states
    initial_state = [[], [], [], ['B', 'C', 'D', 'A']]
    goal_state = [[], [], [], ['A', 'B', 'C', 'D']]
    solve_puzzle(initial_state, goal_state)

if __name__ == "__main__":
    main()
```

Output :

```
PS D:\SEM 5\AI\EXPERIMENTS> python -u "d:\SEM 5\AI\EXPERIMENTS\hill_climbing.py"
Current State: [[], [], [], ['B', 'C', 'D', 'A']]
Heuristic value for [[], [], [], ['B', 'C', 'D', 'A']] is -6
Heuristic value for [['A'], [], [], ['B', 'C', 'D']] is -3
Child chosen for exploration: [['A'], [], [], ['B', 'C', 'D']]

Current State: [['A'], [], [], ['B', 'C', 'D']]
Heuristic value for [['A'], [], [], ['B', 'C', 'D']] is -3
Heuristic value for [[], ['A'], [], ['B', 'C', 'D']] is -3
Heuristic value for [[], [], ['A'], ['B', 'C', 'D']] is -3
Heuristic value for [[], [], [], ['B', 'C', 'D', 'A']] is -6
Heuristic value for [['A', 'D'], [], [], ['B', 'C']] is -2
Child chosen for exploration: [['A', 'D'], [], [], ['B', 'C']]

Current State: [['A', 'D'], [], [], ['B', 'C']]
Heuristic value for [['A', 'D'], [], [], ['B', 'C']] is -2
Heuristic value for [['A'], ['D'], [], ['B', 'C']] is -1
Child chosen for exploration: [['A'], ['D'], [], ['B', 'C']]

Current State: [['A'], ['D'], [], ['B', 'C']]
Heuristic value for [['A'], ['D'], [], ['B', 'C']] is -1
Heuristic value for [[], ['D', 'A'], [], ['B', 'C']] is -2
Heuristic value for [[], ['D'], ['A'], ['B', 'C']] is -1
Heuristic value for [[], ['D'], [], ['B', 'C', 'A']] is -3
Heuristic value for [['A', 'D'], [], [], ['B', 'C']] is -2
Heuristic value for [['A'], [], ['D'], ['B', 'C']] is -1
Heuristic value for [['A'], [], [], ['B', 'C', 'D']] is -3
Heuristic value for [['A', 'C'], ['D'], [], ['B']] is -1
Heuristic value for [['A'], ['D', 'C'], [], ['B']] is -1
Heuristic value for [['A'], ['D'], ['C'], ['B']] is 0
Child chosen for exploration: [['A'], ['D'], ['C'], ['B']]
```

```
Current State: [['A'], ['D'], ['C'], ['B']]
Heuristic value for [['A'], ['D'], ['C'], ['B']] is 0
Heuristic value for [[], ['D', 'A'], ['C'], ['B']] is -1
Heuristic value for [[], ['D'], ['C', 'A'], ['B']] is -1
Heuristic value for [[], ['D'], ['C'], ['B', 'A']] is -1
Heuristic value for [['A', 'D'], [], ['C'], ['B']] is -1
Heuristic value for [['A'], [], ['C', 'D'], ['B']] is -1
Heuristic value for [['A'], [], ['C'], ['B', 'D']] is -1
Heuristic value for [['A', 'C'], ['D'], [], ['B']] is -1
Heuristic value for [['A'], ['D', 'C'], [], ['B']] is -1
Heuristic value for [['A'], ['D'], [], ['B', 'C']] is -1
Heuristic value for [['A', 'B'], ['D'], ['C'], []] is 1
Child chosen for exploration: [['A', 'B'], ['D'], ['C'], []]

Current State: [['A', 'B'], ['D'], ['C'], []]
Heuristic value for [['A', 'B'], ['D'], ['C'], []] is 1
Heuristic value for [['A'], ['D', 'B'], ['C'], []] is 1
Heuristic value for [['A'], ['D'], ['C', 'B'], []] is 1
Heuristic value for [['A'], ['D'], ['C'], ['B']] is 0
Heuristic value for [['A', 'B', 'D'], [], ['C'], []] is -1
Heuristic value for [['A', 'B'], [], ['C', 'D'], []] is 0
Heuristic value for [['A', 'B'], [], ['C'], ['D']] is 1
Heuristic value for [['A', 'B', 'C'], ['D'], [], []] is 3
Child chosen for exploration: [['A', 'B', 'C'], ['D'], [], []]

Current State: [['A', 'B', 'C'], ['D'], [], []]
Heuristic value for [['A', 'B', 'C'], ['D'], [], []] is 3
Heuristic value for [['A', 'B'], ['D', 'C'], [], []] is 0
Heuristic value for [['A', 'B'], ['D'], ['C'], []] is 1
Heuristic value for [['A', 'B'], ['D'], [], ['C']] is 1
Heuristic value for [['A', 'B', 'C', 'D'], [], [], []] is 6
Child chosen for exploration: [['A', 'B', 'C', 'D'], [], [], []]

Current State: [['A', 'B', 'C', 'D'], [], [], []]
Heuristic value for [['A', 'B', 'C', 'D'], [], [], []] is 6
Heuristic value for [['A', 'B', 'C'], ['D'], [], []] is 3
Heuristic value for [['A', 'B', 'C'], [], ['D'], []] is 3
Heuristic value for [['A', 'B', 'C'], [], [], ['D']] is 3
No better heuristic value is obtained, declaring this as the goal state - [['A', 'B', 'C', 'D'], [], [], []]

PS D:\SEM 5\AI\EXPERIMENTS>
```

Conclusion :

Thus we successfully studied and implemented Hill-Climbing Search, and solved the block world problem with this algorithm.