Fr. Conceicao Rodrigues College of Engineering
Fr. Agnel Ashram, Bandstand, Bandra (W), Mumbai - 400050

Department of Computer Engineering
Academic Term II: 23-24

Class: B.E (Computer), Sem – VI **Subject Name: Artificial Intelligence**

Student Name: Siddhesh Pradhan **Roll No: 9632**

Practical No:	**7**
Title:	Block World Problem solving by hill climbing approach
Date of Performance:	
Date of Submission:	

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Marks
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Logic/Algorithm Complexity analysis (03)	03(Correct)	02(Partial)	01 (Tried)	
3	Coding Standards (03): Comments/indention/Naming conventions Test Cases /Output	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Assignment (03)	03(done well)	2 (Partially Correct)	1(submitted)	
Total					

Signature of the Teacher:

Experiment No: 7

Title: Block world problem solving by Hill Climbing method

Objective: To study the solution for block word problem by Hill Climbing approach

Theory:

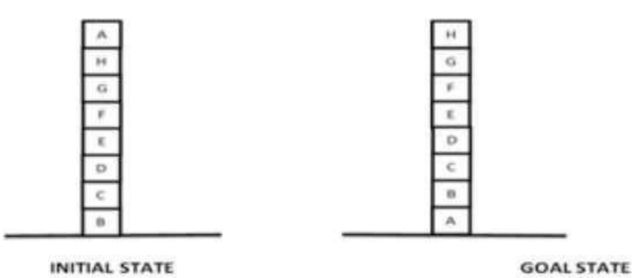
SIMPLE HILL CLIMBING

1. Evaluate the initial state. If it is also a goal state, then return it and quit. Otherwise, continue with the initial state as the current state.
2. Loop until a solution is found or until there are no new operators left to apply in the current state:
    a) Select an operator that has not yet been applied to the current state and apply it to produce a new state
    b) Evaluate the new state,

i. If it is a goal state, then return it and quit.

ii. If it is not a goal state but it is better than the current state, then make it the current state. iii. If it is not better than the current state, then continue in the loop.

Consider the blocks world problem. Assume the same operators (i.e., pick up one block and put it on the table; pick up one block and put it on another one) suppose it uses the following heuristic

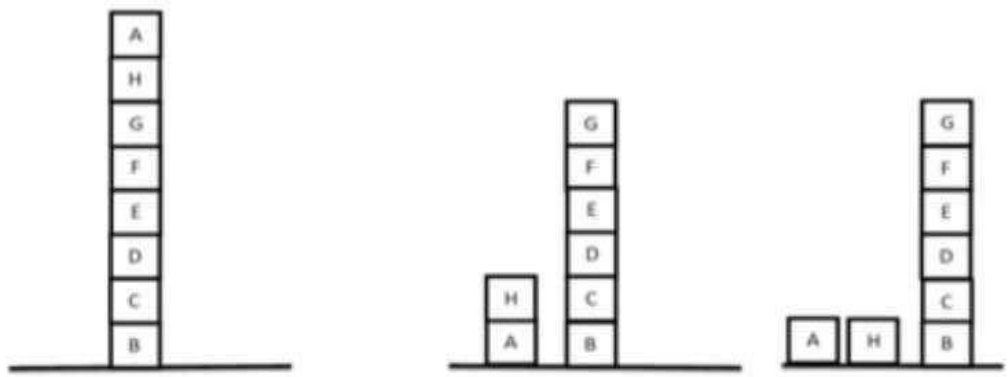


HILL CLIMBING PROBLEM

Local: Add one point for every block that is resting on the thing it is supposed to resting on.

Subtract one point for every block that is sitting on the wrong thing.

Using this function, the goal state has a score of 8. The initial state has a score of 4 (since it gets one point added for blocks C, D, E, F, G and H and one point subtracted for blocks A and B).



THREE POSSIBLE STATES

There is only one move from the initial state, namely to move block A to the table. That produces a state with a score of 6. The hill-climbing procedure will accept that move. From the new state, there are three possible moves, leading to the three states.

These states have the score: (a) 4, (b) 4, and (c) 4. Hill climbing will halt because all these states have lower scores than the current state. The process has reached a local maximum that is not the global maximum.

CODE:

```
class BlockWorld:
    def __init__(self, initial_state, goal_state):
        self.initial_state = initial_state
        self.goal_state = goal_state

    def evaluate_state(self, state):
        score = 0
        for block, resting_place in state.items():
            if block == resting_place:
                score += 1
            else:
                score -= 1
        return score

    def generate_next_states(self, current_state):
        next_states = []
        for block, resting_place in current_state.items():
            if resting_place != "table":
                # Move block to the table
                next_state = current_state.copy()
                next_state[block] = "table"
                next_states.append(next_state)
            for other_block in current_state.keys():
                if other_block != block:
                    # Move block on top of other_block
                    next_state = current_state.copy()
                    next_state[block] = other_block
                    next_states.append(next_state)
        return next_states

    def hill_climbing(self):
        current_state = self.initial_state
        current_score = self.evaluate_state(current_state)

        while True:
            print("Current state:", current_state)
            print("Current score:", current_score)

            if current_state == self.goal_state:
                print("Goal state reached!")
                break
```

```
            next_states = self.generate_next_states(current_state)
            best_next_state = None
            best_next_score = current_score

            for next_state in next_states:
                next_score = self.evaluate_state(next_state)
                if next_score > best_next_score:
                    best_next_state = next_state
                    best_next_score = next_score

            if best_next_state is None or best_next_score <= current_score:
                print("Local maximum reached. Stopping.")
                break

            current_state = best_next_state
            current_score = best_next_score

        print("Final state:", current_state)
        print("Final score:", current_score)


def main():
    initial_state = {'A': 'table', 'B': 'table', 'C': 'C', 'D': 'D', 'E': 'E', 'F':
'F', 'G': 'G', 'H': 'H'}
    goal_state = {'A': 'A', 'B': 'B', 'C': 'C', 'D': 'D', 'E': 'E', 'F': 'F', 'G':
'G', 'H': 'H'}
    block_world = BlockWorld(initial_state, goal_state)
    block_world.hill_climbing()


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

OUTPUT:

```
(base) PS C:\Users\Siddhesh\Desktop> python expt7.py
Current state: {'A': 'table', 'B': 'table', 'C': 'C', 'D': 'D', 'E': 'E', 'F': 'F', 'G': 'G', 'H':
'H'}
Current score: 4
Local maximum reached. Stopping.
Final state: {'A': 'table', 'B': 'table', 'C': 'C', 'D': 'D', 'E': 'E', 'F': 'F', 'G': 'G', 'H': 'H
'}
Final score: 4
```

Post Lab Questions:

1. What are the advantages and disadvantages of state space search?
2. What are the advantages and disadvantages of the Hill Climbing approach?
3. Describe variations of Hill Climbing approach
4. Solve the Block World problem by using the STRIPS method.

Postlab

1>

Advantage	Disadvantage
- Allows syitematic exploration of possible states and transitions	- Complexity increases exponentially with problem size
- Can find optimal solutions for problem with well-defined state and transition rules	- May get stuck in local optimal or search spaces with infinite loops.
- useful for modeling and solving a wide range of problems in AI, including search and planning	- Requires careful design and implentation to ensure efficiency and correctness.

2>

simple and easy to implement.	prone to getting stuck in local optima, especially in rugged search spaces
- Iterative improvement leads to quick convergence	- cannot guarantee finding the global optimum
- suitable for problems with a continuous search space	- sensitive to intial starting points.

=>3 Simple hill Climbling: Iteratively makes small improvements to the current solution.

Steepest Ascent hill: Considering all neighbours states and selects the one with the highest improvement.

Random Restart hill climbing: Randomly restart the search from different initial states to escape local optimal.

Simulated Annealing: Introduces randomness to escape local optimal by allowing uphill moves with a decreasing probability.