Group : **14**

CS 312: Artificial Intelligence Laboratory 2

Task 2: Heuristic Search Algorithms

Team Members

- **180010029** [Rupesh Kalantre]
- **180010031** [Shagun Bera]

1.(a) The State space consists of all valid moves , which in this case is all possible arrangements of stacking ‘B’ blocks in maximum of ‘C’ columns.

(b) The start and goal node is generated randomly using shuffle() and rand(), srand() is called once so that the function generates different states everytime

(c) **MoveGen()**
Generate the possible moves and return the best move according to the heuristic . Mark the state as visited (in BFS)

GoalTest()
Compare each block and return whether the current state is the goal state or not

2. Heuristic Functions Considered

(i) **Position based** : If the position of a block in the current state is the same as its position in the goal state , higher is the heuristic value of this current state.

In Hill Climbing

(ii) **Manhattan Distance based** : The position of i^{th} block in the current state is compared to the position of that block in goal state. The absolute distance between its coordinates is used to calculate the heuristic . The closer the block is to its position in Goal State , more is its Heuristic value.

(iii) **Score based** : If two columns in current and goal state match , irrespective of its position , the score is increased . If the block is in the same position as its goal state , the score is increased . If the sum of blocks in the column is the same in current and goal state , the score is increased. Basically , if the current state is similar to the goal state , its score will be higher .

In Best First Search

(ii) **Order based** : The columns in current state and goal state are compared from bottom to top. The score is more if more blocks are in the correct order from bottom in each stack according to goal state.

3. Best First Search analysis and observations :

What is Best First Search ?

This is an informed searching algorithm in which we use priority queue instead of a simple queue to explore the state with high priority first rather than a state which has low priority based on a heuristic function value.

Pseudo code :

```
bool bfs(State start) {
    priority_queue<State, vector<State>> P;
    P.push(start);
    v.push_back(start);
    while(P.size()) {
        State cur = P.top();
        P.pop();

        cur.print();
        no_of_states_explored ++;

        if(GoalTest(cur)) {
            cur.print();
            cout << "Goal state reached in " << no_of_states_explored << " moves\n";
            return 1;
        }

        for(State& nextState: MoveGen(cur))
            P.push(nextState);
    }
    return 0;
}
```

Analysis & **Observations** :

- ➔ Best First Search will always find the path to Goal state if it's achievable from our starting state.
  - ◆ In case our heuristic function is not efficient, we will end up searching for all possible states. (which will take more time)
- ➔ Suppose there are total '**N**' / O($\mathbf{b^d}$) states possible from our starting state as we visit each state once, our worst case time complexity for this algorithm will be roughly O(**N*log(N)**) (**logN** factor comes from priority queue).
- ➔ Worst case space complexity will be of order of total number of states possible : O(**N**)

➔ Performance of this Best For search algorithm depends on heuristic function. Good heuristic function will lead us to goal state very early compared to normal Breadth First Search.

4. Comparison between BFS and Hill-climbing search algorithm :

1) States explored :

Hill-Climbing : After we reach a local optimal solution, we stop exploring further. So most of the time we end up in a local optimal solution which is not the goal state.

BFS : We explore the states until we find our goal state in BFS. So, we end up exploring more states compared to hill-climbing algo.

2) Time-Taken :

Hill-Climbing : This algorithm stops very early compared to BFS, as we end our search for goal state after reaching local optima.

BFS : This algorithm can take significant time depending on the heuristics used. If our heuristic is bad, it takes a significant amount of time to reach goal state.

3) Reaching the optimal solution :

Hill-Climbing : If start state and goal state are considerably similar and heuristics are good, we may end up in global optima (our goal state), else our search terminates at local optima.

BFS : We will always reach our goal state using BFS. because even if our heuristics are bad, we will endup exploring all states. And if our heuristics are good, we will get to goal state early.