CS 214: Artificial Intelligence Lab Assignment 2 - Report

Tanishq Trivedi (210010056), Prajwal Biradar(210020037)

January 18, 2023

1 Abstract

Blocks World Domain Game starts with an initial state consisting of a fixed number of blocks arranged in three stacks, and we can move only the top blocks of the stacks. We must achieve a goal state that is a particular arrangement of blocks by moving these blocks. This is a planning problem where we know the goal state beforehand, and the path to the Goal state is more important. Hill Climbing has been used to solve this problem.

2 Domain Description

2.1 State Space

A state space is a collection of elements in which each element is a state, and a state is a description of the world in which a problem solver or agent operates. Here, the state space is all the possible configurations of the blocks in the three stacks.

2.2 Start State & Goal State

The start and goal states will be given in the input file. An example is given below.

`Initial state:`

$[C, B, A]$

$[D]$

$[E, A]$

`Goal state:`

$[C, B, D, A]$

$[\,]$

$[E]$

2.3 Pseudo Codes

2.3.1 MoveGen Function

The moveGen function takes the current state and goal state as input and produces the neighbours of the current state. Then, it returns the state whose heuristic value is maximum. The pseudo-code for the moveGen function is present below:

Algorithm 1 moveGen Function

```
procedure moveGen(current state, goal state)
    nextStates ← ()                                ▷ Initialize nextStates
    for neighbour n of current state do
        nextStates.append(n)
    end for
    sort(nextStates)      ▷ Sort the states according to their heuristic values
    x ← nextStates.pop()          ▷ Pop the state with max heuristic value
    return x
end procedure
```

2.3.2 GoalTest Function

This function checks whether the current state is the goal state or not.

Algorithm 2 GoalTest Function

```
procedure GoalTest(current state, goal state)
    if current state == goal state then
        return true
    else
        return false
    end if
end procedure
```

3 Heuristic Functions

3.1 Heuristic function 1

This function calculates the heuristic value of a state by adding the height of a particular block to the total value if it's in the correct position and subtracting the height of a particular block from the total value if it's in the incorrect position.

For Example:-

Let the current state be

$[C, B]$

$[D]$

$[E, A]$

and the goal state be

$[C, B, D, A]$

$[\,]$

$[E]$

Here, the heuristic value is equal to 1 + 2 - 1 + 1 - 2 = 1

3.2 Heuristic function 2

This function adds 1000 to the total if a block is in the correct position. Also, if a block is in the correct stack but not in the right position, then 100 is added to the total.

Consider the current state and the goal state in the previous example.
The heuristic value returned by this function is equal to
1000 + 1000 + 1000 = 3000

4 Hill Climbing

4.1 States Explored

Each state has a maximum of six neighbours. This is because there are three stacks, and the blocks at the top of each stack can be moved to only two other stacks. The total number of states explored depends on the heuristic function. A better heuristic function will result in lesser states being explored.

4.2 Time Taken

The time taken for the agent to reach the goal state is proportional to the length of the steepest gradient ascent route, i.e. the number of iterations required. If the path length is finite, the agent will travel along this path and reach the goal state. Hence, the time complexity of the Hill Climbing Algorithm is linear.

4.3 Optimal Solution

The optimal solution is the state which is reachable from the start state and has the highest heuristic value, i.e. the peak of the hill. This isn't always the goal state, and this is Hill Climbing's biggest flaw.

5 Conclusion

The Hill Climbing Algorithm is undoubtedly very useful. It also requires a constant amount of space to be executed. However, it doesn't always result in the goal state being reached.