AI Lab 2 Report | G6

Shreyas Sathe | 180010033
Janhavi Dadhania | 180010012

Introduction:

Blocks World Domain Game starts with an initial state consisting of a fixed number of blocks arranged in 3 stacks(as asked in this specific assignment) and we can move only top blocks of the stacks and we have to achieve a goal state that is a particular arrangement of blocks by moving these blocks. Blocks World is a planning problem where we know goal state beforehand and path to Goal state is more important.

Initial state: ([F, B, E], [A, D], [C])

State space: { ([F, B], [A, D], [C, E]) , ([F, B], [A, D, E], [C]), ([F, B, E, D], [A], [C]), ([F, B, E], [A], [C, D]), ([F, B, E, C], [A, D], [ ]), ([F, B, E], [A, D, C], [ ])

Goal State: ([B, D, A], [C, F, E], [ ])

Pseudocode:

```
MoveGen( arrangement ):
    children = [ ]
    candidate_moves = { (a, b) : block on top of stack 'a' moves to stack 'b' }
    for move in candidate_moves:
        temp = deepcopy(arrangement.design)
        if temp[move[0]]:
            block = temp[move[0]].pop()
            temp[move[1]].append(block)
            result = candidate_design(temp, design, move[0], move[1])
            children.append(result)
    return children
```

```
goalTest( current, destination):
    if not current or not destination :
        return false
    If current.design == destination.design:
        return true
    return false
```

Heuristic Functions considered:

1	**Out of Place Heuristic**
	If position of the block is not same as in goal state, 1 is added to the cost **Pseudo code:** for each block in state_space: If block above exist: If block above is not block above it in goal state: Cost += 1 Else : If block above exist in goalState: Cost += 1 If block below exist: If block below is not block below it in goal state: Cost += 1 Else : If block below exist in goalState: Cost += 1
2	**Different penalty for top and bottom**
	If the block(or clear) at top is not as in the goal state add 1 to the cost function. If the bottom block(or table) is not as in the goal state add 1 to the cost function. **Pseudo Code:** For each block in state_space:

```
        If block at bottom exist and not equal to block at bottom in goal
state:
                Cost += 1
         If block at top exist and not equal to block at top in goal state:
                Cost += 1
        If table at bottom and goal state has block at bottom:
                Cost += 1
        If clear at top and goal state has block at top:
                Cost += 1
```

Best First Search Analysis and Observation:

It is quite evident that heuristics will not matter in case of 3 blocks since maximum stacks that can be used is equal to 3. All the blocks can be placed in one stack and any goal arrangement can be achieved.

<table>
<tr><td>3</td><td>[3, 2, 1]
[]
[]</td><td>[1, 3, 2]
[]
[]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>4</td></tr><tr><td>2</td><td>4</td></tr></table></td></tr>
<tr><td>3</td><td>[3, 2, 1]
[]
[]</td><td>[1, 2, 3]
[]
[]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>3</td></tr><tr><td>2</td><td>3</td></tr></table></td></tr>
<tr><td>3</td><td>[3, 2, 1]
[]
[]</td><td>[1]
[2]
[3]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>2</td></tr><tr><td>2</td><td>2</td></tr></table></td></tr>
</table>

<table>
<tr><td>5</td><td>[3,5,2]
[]
[1,4]</td><td>[1,2,3]
[4,5]
[]</td><td><table><tr><th>Heuristic</th><th>Moves</th></tr><tr><td>1</td><td>4</td></tr><tr><td>2</td><td>4</td></tr></table></td></tr>
<tr><td>5</td><td>[3,5,2]
[]
[1,4]</td><td>[1,2]
[5,4]
[3]</td><td><table><tr><th>Heuristic</th><th>Moves</th></tr><tr><td>1</td><td>5</td></tr><tr><td>2</td><td>5</td></tr></table></td></tr>
<tr><td>5</td><td>[3,5,2]
[]
[1,4]</td><td>[2,5,3]
[4,1]
[]</td><td><table><tr><th>Heuristic</th><th>Moves</th></tr><tr><td>1</td><td>5</td></tr><tr><td>2</td><td>5</td></tr></table></td></tr>
</table>

Second, heuristic seems better than the first one.

<table>
<tr><th>Number of Blocks</th><th>Goal</th><th>Initial State</th><th>Number of Moves</th></tr>
<tr><td>6</td><td>[4, 2]
[6, 1, 5, 3]
[]</td><td>[3]
[2, 6, 1, 4]
[5]</td><td><table><tr><th>Heuristic</th><th>Moves</th></tr><tr><td>1</td><td>15</td></tr><tr><td>2</td><td>11</td></tr></table></td></tr>
</table>

<table>
<tr><td>6</td><td>[4, 2]
[6, 1, 5, 3]
[]</td><td>[3, 5, 4]
[2, 6, 1]
[]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>17</td></tr><tr><td>2</td><td>18</td></tr></table></td></tr>
<tr><td>6</td><td>[4, 2]
[6, 1, 5, 3]
[]</td><td>[]
[2, 6, 1]
[4, 5, 3]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>9</td></tr><tr><td>2</td><td>10</td></tr></table></td></tr>
<tr><td>6</td><td>[4, 2]
[6, 1, 5, 3]
[]</td><td>[1, 2, 3, 4, 5, 6]
[]
[]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>17</td></tr><tr><td>2</td><td>12</td></tr></table></td></tr>
<tr><td>6</td><td>[4, 2]
[6, 1, 5, 3]
[]</td><td>[6, 5, 4, 2, 1, 3]
[]
[]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>23</td></tr><tr><td>2</td><td>12</td></tr></table></td></tr>
</table>

<table>
<tr><td>8</td><td>[1,2,3,4,5,6,7,8]
[]
[]</td><td>[8,4,5]
[2,6,1]
[3,7]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>45</td></tr><tr><td>2</td><td>38</td></tr></table></td></tr>
<tr><td>8</td><td>[1,2,3,4,5,6,7,8]
[]</td><td>[3,2,1]
[8,6,5]</td><td></td></tr>
</table>

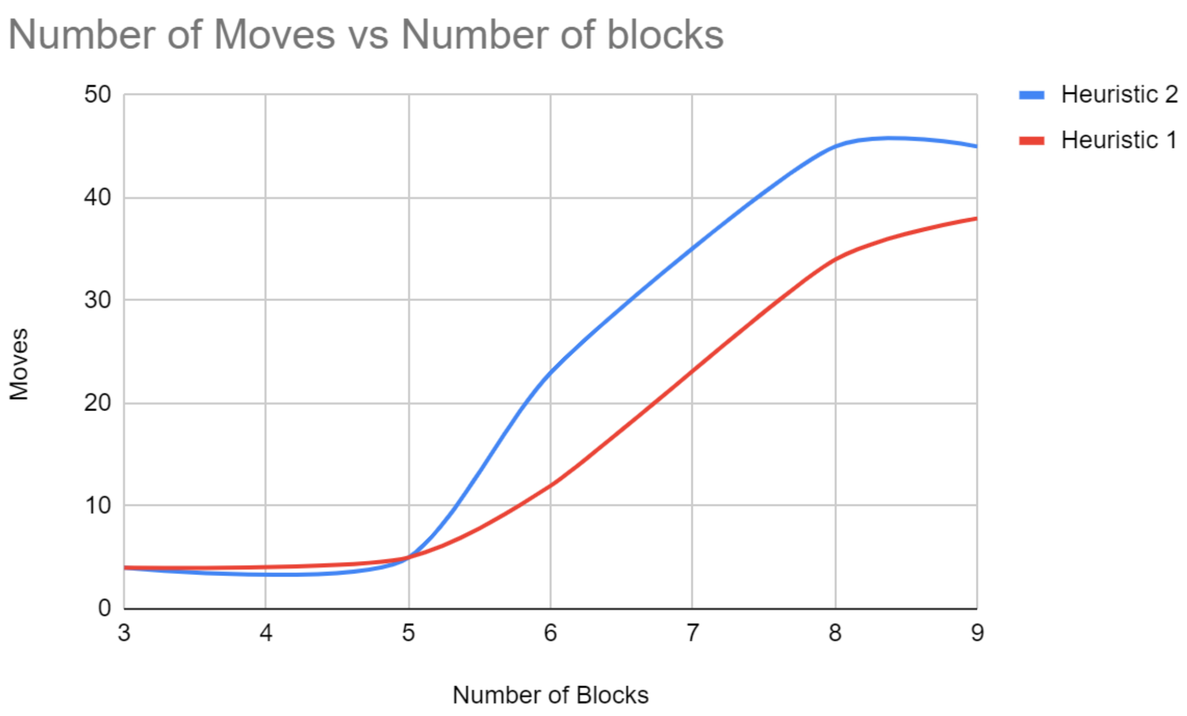
<table>
<tr><td></td><td>[]</td><td>[4,7]</td><td><table><tr><th>Heuristic</th><th>Moves</th></tr><tr><td>1</td><td>28</td></tr><tr><td>2</td><td>18</td></tr></table></td></tr>
<tr><td>8</td><td>[1,2,3,4,5,6,7,8]
[]
[]</td><td>[3,2,1]
[5,6,8]
[4,7]</td><td><table><tr><th>Heuristic</th><th>Moves</th></tr><tr><td>1</td><td>37</td></tr><tr><td>2</td><td>41</td></tr></table></td></tr>
<tr><td>8</td><td>[1,2,3,4,5,6,7,8]
[]
[]</td><td>[3,4,2,1]
[7,5,6,8]
[]</td><td><table><tr><th>Heuristic</th><th>Moves</th></tr><tr><td>1</td><td>36</td></tr><tr><td>2</td><td>17</td></tr></table></td></tr>
<tr><td>8</td><td>[1,2,3,4,5,6,7,8]
[]
[]</td><td>[3,4,7,5,6,8,2,1]
[]
[]</td><td><table><tr><th>Heuristic</th><th>Moves</th></tr><tr><td>1</td><td>24</td></tr><tr><td>2</td><td>29</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td>[1,2,3,4,5,6,7,8]
[]
[]</td><td>[8,2,1]
[4,7,5]
[3,6]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>25</td></tr><tr><td>2</td><td>34</td></tr></table></td></tr>
</table>

<table>
<tr><td>9</td><td>[1,2,3]
[7,8,9]
[4,5,6]</td><td>[1,2,3,4,5,6,7,8,9]
[]
[]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>10</td></tr><tr><td>2</td><td>14</td></tr></table></td></tr>
<tr><td>9</td><td>[1,2,3]
[7,8,9]
[4,5,6]</td><td>[1,3,4]
[2,6,7]
[9,8,5]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>26</td></tr><tr><td>2</td><td>35</td></tr></table></td></tr>
<tr><td>9</td><td>[1,2,3]
[7,8,9]
[4,5,6]</td><td>[1,3,4]
[2]
[9,8,5,7,6]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>26</td></tr><tr><td>2</td><td>25</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td>[1,2,3]
[7,8,9]
[4,5,6]</td><td>[1,2,3]
[7,8]
[4,5,6,9]</td><td><table><tr><td>Heuristic</td><td>Moves</td></tr><tr><td>1</td><td>45</td></tr><tr><td>2</td><td>38</td></tr></table></td></tr>
</table>

Plots :



It can be observed from the plot that heuristic 2 performs better than heuristic 1 as the number of blocks increases.

Hill Climbing and Best First Search Comparison

Initial State	Goal State	Steps taken (BFS)	Time (steps)	States explored (BFS)	States Explored	Reaching Optimal Solution

[3, 2, 1] [] []	[1, 3, 2] [] []	**4**	**6**	**13**	**7**	Reached
[1,2,3] [4,5] []	[3,5,2] [] [1,4]	**4**	**-**	**480**	**8**	Not Reached
[3] [2, 6, 1, 4] [5]	[4, 2] [6, 1, 5, 3] []	**11**	**4**	**124**	**4**	Not Reached
[1,2,3,4,5,6,7,8] [] []	[8,4,5] [2,6,1] [3,7]	**38**	**43**	**658**	**43**	Not Reached
[1,3,4] [2] [9,8,5,7,6]	[1,2,3] [7,8,9] [4,5,6]	**25**	**40**	**981**	**40**	Not reached

Observation:

1. Best First Search explored much more states compared to hill climbing.
2. Time taken (when optimal solution is reached) is very less compared to BFS.
3. As the total number of blocks increases, in most of the cases goal state is not reached in hill climbing.