Performance Metrics

5 Block : : 3 stacks

Original State:
1 | B
2 | C E
3 | A D

Heuristic	Goal Test	Max Queue Size	Mean Solution Path Length
Heuristic used	12	24	10
Simple heuristic(h_0)	378	337	10

==========

6 Block : : 3 stacks

Original State:
1 | E C
2 | B
3 | A D F

Heuristic	Goal Test	Max Queue Size	Mean Solution Path Length
Heuristic used	23	43	16
Simple heuristic(h_0)	4514	3379	14

==========

10 Block : : 5 stacks

Original State:
1 | D
2 | E F I J
3 | B G
4 | C H
5 | A

Heuristic	Goal Test	Max Queue Size	Mean Solution Path Length
Heuristic used	80	976	19
Simple heuristic(h_0)	Failure*	Failure*	Failure*

==========

12 Block : : 6 stacks

Original State:
1 | L
2 | H F I
3 | B C
4 | G E
5 | A J
6 | K D

Heuristic	Goal Test	Max Queue Size	Mean Solution Path Length
Heuristic used	91	1759	19
Simple heuristic(h_0)	Failure*	Failure*	Failure*

==========

12 Block : : 5 stacks

Original State:
1 | G K D
2 | H I C
3 | L F B E
4 |
5 | A J

Heuristic	Goal Test	Max Queue Size	Mean Solution Path Length
Heuristic used	450	3835	25
Simple heuristic(h_0)	Failure*	Failure*	Failure*

==========

20 Block : : 10 stacks

Original State:
1 | M
2 | P R H
3 | L B E
4 | N I
5 | J
6 | D S
7 | A K T
8 | Q
9 | F C
10 | O G

Heuristic	Goal Test	Max Queue Size	Mean Solution Path Length
Heuristic used	1130	69782	31
Simple heuristic(h_0)	Failure*	Failure*	Failure*

==========

PS : *Goal Test limited to 15000 test.

How heuristics works?

The heuristics used here, takes into account the following three factors.

1. ***blocksOutOfPlace*** : keeps count of all the blocks that are out of place.
2. ***stepsToGetOutOfStack*** : steps required by each block to get out of its current stack.
3. ***stepsToPutItInCorrectPosn*** : steps required to put each block in its correct position considering that the blocks before it are in its correct position (using *positionInGoalNode* + *blocksOutOfPlace*) where *positionInGoalNode* is the correct position of the block.

First while finding the number of blocks that are out of place, we maintain a list containing the blocks that are in place. Now, we find the *stepsToGetOutOfStack* for each block and also the *stepsToPutItInCorrectPosn* for each block ,considering the factor of *blocksOutOfPlace* + *positionInGoalNode.* Also, if a block is already in place (by checking the list) then we skip this process for that block. The final heuristic is taken as sum of *stepsToGetOutOfStack and stepsToPutItInCorrectPosn* of all the blocks and the distance of a state from the original state.

```
heur += stepsToGetOutOfStack + stepsToPutItInCorrectPosn;
heur + = getDepth()
```

Whether it is admissible or not?

The heuristic that I have used here is not admissible and it overestimates the cost to reach the goal state.

For e.g:

1 | C H
2 | E F G I
3 |
4 | A B D J
5 |

The steps required for the problem to solve is 21, but at a stage, *iter=140, f=g+h=24, depth =18*, it shows the heuristic value as 24, which is an overestimate.

But **with more iterations, the heuristic becomes admissible.**

Scalability

To see how scalable the performance of the heuristic is, first we keep the block constant and vary the number of stack and then we keep the number of stack constant and very the blocks.

1. Keeping the **number of block constant**(10) and check for the following state:

Original State :
1 | C I
2 | B D F H
3 | G
4 | E J
5 | A

No. of stacks	Goal Test	Max Queue Size	Solution Path Length
5	72	859	19
6	20	343	16
7	19	471	17
8	18	589	17
9	17	670	16
10	17	765	16

The heuristic here **scales as expected**. With the increase in number of stacks the problem becomes easy to solve taking **less** goal tests and **shorter** path length.

2. Now we keep the number of stacks same and vary the number of blocks. We keep 5 stacks and keep increasing the blocks from 5 to 10.

Original State :
1 | C I
2 | B D F H
3 | G
4 | E J
5 | A

No. of blocks	Goal Test	Max Queue Size	Solution Path Length
5	8	74	7
6	11	110	10
7	12	122	11
8	15	160	13
9	34	424	16
10	72	859	19

The heuristic **scales as expected**. With the increase in number of blocks the problems becomes harder to solve and takes **greater** goal test and **longer** solution path.