CSCE 625 Programming Assignment #1

Jacob Fenger

October 4, 2017

Contents

1 Running the Program

To run the program:

1. Compile blocksworld.cpp (I used g++ version 4.8.4). E.g. 'g++ blocksworld.cpp'.
2. To run the program, you must specificy the stack and block number as command line arguments (E.g. './a.out 3 5' for 3 stacks and 5 blocks).
3. The output should now output information regarding the state of the program. If a path is found to the goal state, it should show it as well as the number of iterations and the priority queue size. If no path is found, or the number of iterations exceeds 5000, then it will output a message regarding failure.

2 Example Traces

The following shows an example trace of running with a stack and block size of 3 and 5 respectively.

```
Starting Search....
ITR: 0 - QUEUE: 0 - F: 48
```

```
ITR: 1 - QUEUE: 3 - F: 32
ITR: 2 - QUEUE: 6 - F: 21
ITR: 3 - QUEUE: 9 - F: 18
ITR: 4 - QUEUE: 12 - F: 19
ITR: 5 - QUEUE: 15 - F: 17
ITR: 6 - QUEUE: 16 - F: 18
ITR: 7 - QUEUE: 19 - F: 13
ITR: 8 - QUEUE: 22 - F: 11
ITR: 9 - QUEUE: 23 - F: 9
START:
0 | A C D
1 | E
2 | B

0 | A C
1 | E
2 | B D

0 | A
1 | E C
2 | B D

0 | A
1 | E C D
2 | B

0 | A B
1 | E C D
2 |

0 | A B
1 | E C
2 | D

0 | A B C
1 | E
2 | D

0 | A B C D
1 | E
2 |

0 | A B C D E
1 |
2 |
```

```
Final number of iterations: 9
Max. queue size: 23
```

3 Heuristic Description

My heurestic gives the worst penalty when blocks are not in the correct spots in the final stack. The intuition here is that the algorithm will be more likely to remove blocks from the final stack if they are not in the correct position so that it will allow the correct blocks to be placed there. The problem with doing just this is that the search will be more likely to place blocks into their correct spots even if the block below isn't in the correct position. This leads into the next component of my heuristic.

Blocks can be placed on top of lower valued blocks (Such as a B on top of an A) which will lead to more moves to get the lower block into the goal state. My heuristic penalizes states which have the lower block below other blocks when it is not in the goal state. This will ensure that states with lower blocks on top will be more likely to be chosen since less moves are required to reach the goal state in those situations.

3.1 Admissability

This heuristic is not admissable because the cost it estimates tends to always be higher than the actual cost of reaching the goal. In my opinion, it is very hard to find an admissable heuristic in this problem without using a more extensive search algorithm.

4 Performance Metrics

Below is a table showing the average number of iterations, queue size, and path length for solutions of different stack and block sizes. Each problem space was ran ten times and averages were computed.

	3-5	**6-10**	**3-7**	**3-10**	**5-12**
Iterations	22.9	27.67	106.6	528.8	64.4
Path Length	9.9	15.3	18.9	27	22
Queue Size	48.9	442.3	241.3	1367.8	716.2
Failures	0	4	0	0	5

Table 1: Average iteration, average queue size, average path length, and number of failures out of 10 executions for solutions with stack-block sizes (3-5 represents three stacks, five blocks).

5 Results Discussion

My heuristic worked fine for cases where the stack size was smaller regardless of the number of blocks. As we scale the number of stacks up, I ended up with several failures depending on the starting state (I consider it a failure once my search hits 5000 iterations). This is because the number of neighbor states is much larger each iteration and my heuristic does not do much to account for a lot of empty stacks.

I was able to solve 3 stacks and 12 blocks in a reasonable time for most starting states which is the largest I tested. The queue grew at mostly a linear rate when compared to the number of iterations.

When it comes to finding optimal solution, my program sometimes found it but only on the most simple starting states. My heuristic sometimes provided the same value for all the neighbors which causes my program to take a non-optimal route.

To improve my heuristic, I would need to find a way to make it a priority to place blocks with a lower value on top of blocks of a higher value at all times.