Kunal Purohit

011877606

CS 420

Project 1 Report

For my project, I created two main class objects; the PuzzleBoard class and the Handle class. The PuzzleBoard class represents a puzzle’s state. This class also contains the parent state that the current puzzle is derived from, as well as the path cost / heuristic cost of the puzzle relative to the original puzzle state and the goal state. The PuzzleBoard also calculates a hash value to identify itself, which is essentially the puzzle values laid all out on one line sequentially. The Handle class takes care of the priority queue that holds the PuzzleBoard and the operations to pop off the top PuzzleBoard and add its children nodes if necessary. This class also handles a Hash Map that tracks the visited PuzzleBoards. An instance of this class has methods that allow for it to generate random puzzle boards and run both heuristic functions and then output the values to a csv file.

Both heuristic functions were able to easily find the optimal path in efficient time. For a significant number of cases , both functions derived the function in the same number of nodes with very similar execution times. These cases occur more frequently in cases where the depth is very low(3 to 7). However, for the other cases, it was clear that the Manhattan distance function performed better that the Misplaced function.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Initial | Mis. Path Cost | Mis. # Nodes | Mis.Time | Man. Path Cost | Man. # Nodes | Man.Time |
| 123576408 | 13 | 13 | 382250 | 13 | 12 | 322147 |
| 243160758 | 13 | 22 | 401902 | 13 | 19 | 737460 |
| 123046758 | 13 | 8 | 72564 | 13 | 8 | 69440 |
| 413256708 | 13 | 27 | 340693 | 13 | 20 | 261189 |
| 123485760 | 13 | 11 | 100347 | 13 | 9 | 84347 |
| 253016478 | 15 | 25 | 308361 | 15 | 18 | 236000 |
| 102483765 | 15 | 27 | 254910 | 15 | 14 | 150902 |
| 123046758 | 15 | 8 | 58963 | 15 | 8 | 73047 |
| 162403758 | 15 | 21 | 129319 | 15 | 17 | 94320 |
| 123576408 | 15 | 13 | 167256 | 15 | 12 | 100363 |
| 236105478 | 15 | 15 | 106543 | 15 | 15 | 86524 |
| 412583706 | 17 | 14 | 71712 | 17 | 14 | 72676 |
| 136428750 | 19 | 14 | 275417 | 19 | 12 | 192723 |
| 102463758 | 19 | 11 | 125470 | 19 | 11 | 118195 |
| 123508467 | 19 | 46 | 277977 | 19 | 22 | 129702 |
| 236048175 | 23 | 91 | 1227947 | 23 | 30 | 538983 |

This small snippet of the data retrieved shows cases where both functions resulted in different number of nodes. For all cases, the Manhattan distance performs better than the Misplaces Distance. In terms of number of nodes generated, the difference is clearly exponential; cases where the path cost/depth is low, the difference is very miniscule, while depths of greater amounts result in bigger differences. In terms of the execution time, the Manhattan functions clearly performs better as well. This is usually by a factor of two, but in general, the execution time of the Manhattan distance is shorter.

When we evaluate possible causes of each functions performance, I believe this is due to the inherent significance and representation of each heuristic. For the misplaced function, while it is efficient in identifying the changes that need to be made, it does not take into account the necessary steps/actions that will be needed to execute and get the optimal solution. Moreover, this can cause the function to get into cases that may be optimal in terms of the function, but not optimal in terms of the actions that will need to be made to achieve those states, resulting in more nodes being generated. This is where the Manhattan Distance functions provides more information; the function takes into account the necessary distance of each individual piece and bases future steps off of that evaluation.