Solving the 8-Puzzle

Shahin Ahmadi and **DuBose Tuller**
{shahmadi,dutuller}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

The 8-puzzle is a sliding game consisting of numbered square tiles in a 3-by-3 grid with one tile absent. We implemented the A* search algorithm to solve the 8-puzzle and assessed the impact of three different heuristic functions on search performance. The Manhattan Distance heuristic resulted in the most efficient searching. We concluded heuristics designed based on less relaxed versions of the 8-puzzle lead to more efficient A* search performance.

1 Introduction

A* is an informed search algorithm that uses a heuristic function to find the optimal path to a goal state in a graph. The choice of heuristic function can significantly impact the efficiency of A*. A well-designed heuristic function can greatly reduce the number of nodes that need to be expanded in the search process, leading to faster runtime. Previous research by Russell and Norvig (2003) has shown that solving the 8-puzzle with a heuristic based on the Manhattan distance between each tile and its goal position is more efficient than one based on the count of misplaced tiles. Here, we assess the impact of heuristic quality on A* performance by replicating Russell and Norvig's experiment and comparing the outcomes. We also examine and compare the effect of the checkerboard misplaced heuristic on the A* performance in solving the 8-puzzle.

2 Background

The 8-puzzle is a simple sliding game consisting of 8 numbered square tiles and an empty tile placed randomly on a 3-by-3 grid. The game involves rearranging the numbered tiles within the frame by sliding them one at a time into the empty tile, with the goal of arranging the numbered tiles in ascending numerical order with the empty tile in the upper-left corner of the frame.

To solve the 8-puzzle, we used A* algorithm with three different heuristic functions:

- Manhattan Distance (h_{MD}): calculates the sum of the Manhattan distance between each tile and its correct position in the goal state.
- Misplaced Tiles (h_{MT}): finds the number of tiles that are not in the correct position.
- Checkerboard Misplaced (h_{CM}): divides the frame into two sub-graphs composed of tiles connected by diagonals and determines the minimum number of steps between a tile's current and target position, where the tile can move to any location in the *other* sub-graph.

All three heuristics are admissible and derived by creating a relaxed version of the 8-puzzle's rules that a computer can solve without searching. Heuristics designed based on "less" relaxed problems produce estimates that are closer to the actual search cost (Hansson, Mayer, and Yung 2011).

3 Experiments

We generated random 8-puzzle board states by starting from the goal state and making a specified number of random moves. Another way to achieve this would be to generate random permutations of tiles. However, we reasoned the second approach would also generate unsolvable board states and thus can be computationally more expensive. We performed A* search on randomized board states with each of the three heuristics. We evaluated each heuristic by measuring the search cost (N) and effective branching factor (b^*). N is measured by the total number of board states generated instead of the number of explored board states, as the latter approach underestimates the computational work. b^* estimates the average branching factor of a tree containing only the searched board states and is calculated with the following formula (Russell and Norvig 2003):

$$N + 1 = 1 + b^* + (b^*)^2 + \cdots + (b^*)^d$$

where d is the depth of the goal state (length of the optimal path). Note that b^* cannot be calculated analytically and is instead estimated to a desired level of precision using techniques such as Newton's Method.

We excluded the empty tile in our implementation of h_{MD} and h_{MT} to maintain admissibility. We tested each heuristic against 10,000 board states generated by 100 random moves. We noticed that board states with an optimal path length of 2-10 were not generated nearly as often, so we supplemented our data by running an additional 1000 board states using 20 random moves. When searching, we did not allow for direct backtracking to the first previous board state to avoid redundant checking. Our implementation of A* entered an infinite loop when given an unsolvable board state. Finally, we grouped the search costs by their optimal distance and averaged them within each group. Though the groups did not have the same number of trials, they were all sufficiently large to be representative.

4 Results

The table below summarizes the average search cost of each heuristic in conjunction with A* at a given depth. Each heuristic's relative performance can be measured by their b^* values, with lower values corresponding to lower search costs (Russell and Norvig 2003).

	Average Search Cost			Branching Factor		
d	h_{MD}	h_{CM}	h_{MT}	h_{MD}	h_{CM}	h_{MT}
2	5	5	6	1.79	1.79	2.00
4	8	8	9	1.30	1.30	1.35
6	14	15	15	1.25	1.27	1.27
8	21	23	34	1.21	1.23	1.32
10	35	46	73	1.22	1.27	1.35
12	63	97	179	1.24	1.30	1.39
14	124	236	486	1.27	1.34	1.43
16	275	600	1,302	1.29	1.38	1.46
18	558	1,463	3,596	1.31	1.40	1.48
20	1,127	3,742	9,975	1.32	1.42	1.50
22	2,310	10,062	28,358	1.34	1.44	1.52
24	5,024	27,265	80,559	1.35	1.46	1.53
26	11,409	76,691	234,271	1.36	1.48	1.55
28	27,664	215,331	666,423	1.38	1.49	1.56
30	73,215	644,093	2,075,996	1.39	1.50	1.57

Figure 1: Average Search Costs and Effective Branching Factor by Heuristic Method

h_{MD} outperforms the other two heuristics in all search depths and is followed by h_{CM}, with its b^* values falling roughly halfway between those of h_{MD} and h_{MT}. As shown in Figure 2, at larger d, small perturbations in b^* have a large effect on search cost and can make a dramatic difference in real-world performance. For example, at $d = 30$, the h_{MT} heuristic took on average almost 30 times longer to run than h_{MD}.

After the problems become decently complex ($d \geq 6$), the algorithm must, on average, search a higher number of neighboring board states. This increasing b^* effect was also observed in Russell and Norvig's experiment but happened for a slightly higher $d \geq 8$ (Russell and Norvig 2003). Our implementations of h_{MD} and h_{MT} outperform Russell and Norvig's at lower depths (h_{MD}: $d < 10$; h_{MT}: $d \leq 14$) but do worse at higher depths (See Figure 3 and 4). Though Russell and Norvig's results do not include $d = 26, 28, 30$, we expect the trend to be the same.

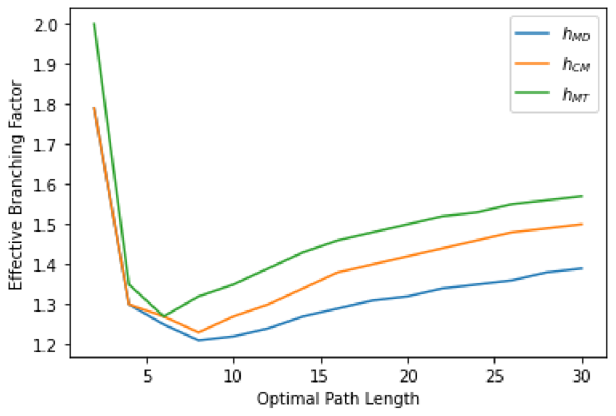

Figure 2: Effective Branching Factors by Heuristic

	Average Search Cost		Branching Factor	
d	h_{MD}	h_{MT}	h_{MD}	h_{MT}
2	6	6	1.79	1.79
4	12	13	1.45	1.48
6	18	20	1.30	1.34
8	25	39	1.24	1.33
10	39	93	1.22	1.38
12	73	227	1.24	1.42
14	113	539	1.23	1.44
16	211	1301	1.25	1.45
18	363	3056	1.26	1.46
20	676	7276	1.27	1.47
22	1219	18094	1.28	1.48
24	1641	39135	1.26	1.48

Figure 3: Average Search Costs and Effective Branching Factor by Heuristic Method from Original Study (Russell and Norvig 2003)

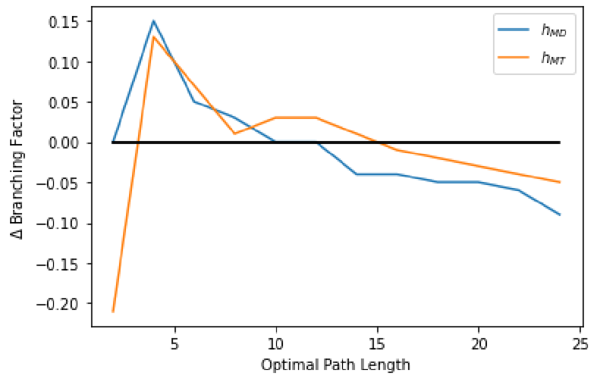

Figure 4: Comparing our results to the original study. Positive values reflect higher performance by our algorithm.

5 Conclusions

In this paper, we assessed the impact of heuristic function on the performance of A* by solving the 8-puzzle utilizing Manhattan Distance, Misplaced Tiles, and Checkerboard Misplaced heuristics. Our study confirms Russell and Norwig's results, as patterns of search cost with respect to depth are consistent. Moreover, the performance of the additional Checkerboard method falls in between the two from their study. We conclude that consistent heuristics that more accurately estimate the true number of steps to the goal state result in better A* search performance. Future research can examine other heuristics and apply h_{MD}, h_{MT}, and h_{CM} to boards of larger sizes. Additionally, analyzing versions of the game where multiple goal states exist (e.g., considering all boards with ascending tiles valid regardless of the empty tile's position on the board) can provide a deeper understanding of the impact of heuristic quality on A* performance.

References

Hansson, O.; Mayer, A. E.; and Yung, M. 2011. Generating admissible heuristics by criticizing solutions to relaxed models. *Department of Computer Science, Columbia University* 4–7.

Russell, S. J., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Pearson Education.