Solving the 8-Puzzle Implementing the A* Algorithm: An Investigation into the Impact of Heuristic Quality on Search Performance

Ilina Navani and **Katie Kowalyshyn** and **Oma Hameed**
{ilnavani,kakowalyshyn,omhameed}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

In this paper we implemented the A* algorithm and investigated the impact of heuristic quality on search performance using the 8-puzzle. Specifically, we looked at two heuristics and compared the search costs and the effective branching factors for each heuristic with the A* algorithm. Our data was averaged over 10,000 instances of the 8-puzzle. We found that in comparison to the results of Russell and Norvig, our search costs when using the first heuristic were generally lower; our search costs when using the second heuristic were generally higher; and our effective branching factors for both heuristics were overall higher. Finally, within our experiments, the second heuristic performed better than the first, for both search costs and effective branching factors.

1 Introduction

In engaging with search algorithms in an attempt to find the most efficient one, the A* algorithm quickly became widely used in graph traversal and path finding. First described in 1968, the core characteristics of A* are the frontier – a priority queue – and the heuristics, such that while recursively traversing a tree, nodes are popped along the path with the lowest search cost, thus guaranteeing A* to find an optimal solution. To detect this optimal path, we use heuristics that estimate the cost of reaching the goal state from each node state, all of which are admissible and consistent as proved by Russell and Norvig (Russell and Norvig 2003).

Our implementation of A* tests two heuristics, keeping track of the total number of nodes generated and the effective branching factor, and ultimately compares their results to see which performs better. By our definition, the total number of nodes generated is the number of nodes added to the priority queue, that is, the total number of legal board states explored throughout the algorithm. In contrast, Russell and Norvig do not define what total nodes generated means in their experiment. In our results section, we discuss the discrepancies this produces. By testing our algorithm on 10,000 randomly generated boards with solutions at varying depths, our goal was to determine which heuristic yields a more efficient search performance: the Hamming heuristic (h_1) based on the number of misplaced tiles or the Manhattan heuristic (h_2) based on the sum of distances of tiles from goal positions. We hypothesize that given Russell and Norvig's experimental results, and the nature of the Manhattan heuristic in calculating exact distances of tiles from the goal, that h_2 will have a better search performance than h_1.

When investigating this question, we referenced two papers, "Artificial Intelligence: a Modern Approach" by Russell & Norvig (Russell and Norvig 2003) and "Comparing the Hamming and Manhattan Heuristics in Solving the 8—Puzzle by A* Algorithm" by Siaw Chong Lee and Tyan Her See (Lee and See 2021). Both papers compared the A* algorithm using the Hamming and Manhattan heuristics, just as we did. Both papers also concluded that the Manhattan Heuristic performed significantly better than the Hamming Heuristic. Specifically, Lee and See conclude that h_2 dominates h_1 based on the lower computing time required by h_2 to solve the puzzle (Lee and See 2021). Comparatively, our paper and Russell and Norvig's paper tested the heuristics based on search cost and the effective branching factor, and arrived at the same conclusion. Eventually, after analyzing and visualizing our experimental results, we concluded that h_2 outperforms h_1 in search performance, given the drastically significant differences in number of nodes generated, especially as solution depth increases. Our numbers followed similar trends to that of Russell and Norvig, that is, the number of nodes increases exponentially at larger depths, specifically for h_1. The effective branching factor was fairly consistent, albeit higher than the values produced by Russell and Norvig's.

In the remainder of this paper, we will provide background on the 8-puzzle and our algorithm, describe our experimental setup, and analyze our results. We will also provide justifications for these results, and propose probable reasons for differences observed in comparison to Russell and Norvig's findings.

2 Background

The 8-puzzle is a presented as board with seven tiles and one blank space, with the goal being to arrange the numbers on the tiles in increasing order, from left to right and top to bottom. An example of such a board is shown in Figure 1 as both an unsolved puzzle and a solved puzzle. The

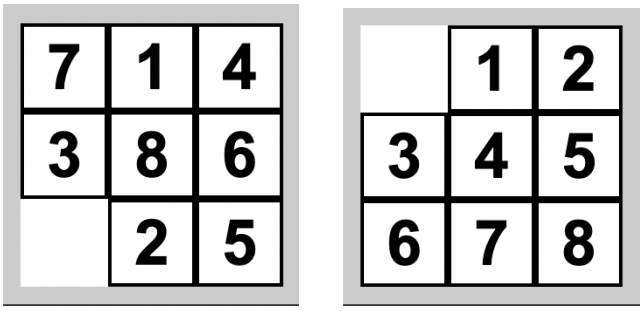

Figure 1: An example of an unsolved 8-puzzle (left) and a solved 8-puzzle (right).

A* algorithm takes in arbitrary 8-puzzles as inputs and solves them after having traversed an optimal path of legal, intermediate board states, each of which is represented as nodes in the frontier. It is important to note however that not all initial 8-puzzles are solvable. Therefore, when generating random instances of 8-puzzles, we first need to check to see if they are solvable or not. A solvable board is defined as one that has an even number of inversions, wherein a pair of tiles form an inversion if the values on the tiles are in reverse chronological order of their appearance in the solution. For example, given the initial board shown in Figure 1, the total number of inversions is 14: {(7, 1), (7, 4), (7, 3), (7, 6), (7, 2), (7, 5), (4, 3), (4, 2), (3, 2), (8, 6), (8, 2), (8, 5), (6, 2), (6, 5)}. Given that the number of inversions is even, we can conclude that our puzzle is indeed solvable.

Another measure to be mindful of is the effective branching factor, b^*. According to Russell and Norvig's definitions, for a problem with the total number of nodes generated by A* as N, and a solution depth d, b^* can be characterized by the following equation:

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

In our calculations, we used Wolfram Alpha to provide us with an approximation of the above equation, which we used to calculate b^* for each iteration of A*. Therefore, for our results, b^* is calculated as:

$$b^* = N^{(1/d)}$$

Overall, our approach was consistent with the standard implementation of the A* algorithm, as well as Russell and Norvig's descriptions, the details of which we further elaborate on in our experimental setup.

3 Experiments

Our experiments were designed to compare the impact of the two different heuristics on search costs and effective branching factors using the A* algorithm. We used the programming language Python to write all of our code. We started by generating random instances of solvable puzzles by creating boards consistent with the solvability definition described in the Background section. In comparison, Russell and Norvig generated 1200 random boards, 100 for each even depth number. Our set-up however did not account for an equal number of problems at each depth. Since we used the random method in python to generate an ordering of numbers from 0 to 8, our boards were completely randomized in terms of the depth of the solution. We therefore decided to generate 10,000 boards spanning both even and odd depths, and supplied the same initial board to both of our heuristics. By generating a large number of boards, our hope was that our dataset will contain puzzles with solutions at every depth. Initially, we ran our algorithm on a dataset of 5,000 random boards, however, the number of boards was still insufficient in generating solutions at lower depths. By increasing our number of iterations to 10,000, we were successful in achieving solutions at all depths, as demonstrated later in our results.

Consistent with Russell and Norvig's implementation, we defined h_1 as the number of misplaced tiles, including the blank tile. In order to calculate this, we add 1 for a tile which is misplaced, and add 0 if it is in its correct place. For example, in the initial board shown in Figure 1, the following sum computes the h_1 value, starting with the tile in the top left corner, proceeding left to right and from the top row to the bottom row:

$$h_1 = 1 + 0 + 1 + 0 + 1 + 1 + 1 + 1 + 1 = 7$$

$$h_2 = 3 + 0 + 2 + 0 + 2 + 3 + 2 + 3 + 1 = 16$$

When implementing A*, the ordering of nodes in our priority queue was determined by a $f(n)$ value, where

$$f(n) = h(n) + g(n)$$

In the above equation, $h(n)$ is the estimate provided by the heuristic, while $g(n)$ is the path cost thus far, that is, the current depth. In the case of a tie where two board states in the priority queue have the same $f(n)$ value, we simply chose the one that had been pushed into the priority queue first. Our A* algorithm ran by making legal moves until we arrived at the goal state, at which point we recorded the number of nodes traversed and depth reached, as well as calculated the effective branching factor for that specific depth.

4 Results

Our results returned values that support our hypothesis of h_2 performing significantly better than h_1. As seen below in Figures 2 and 3, it is clear that the search cost for h_1 jumps significantly as the depth increases, compared to a much more steady incline for h_2. Our effective branching factor (EBF) is of a similar result, showing a consistent decrease between h_1 and h_2 until ultimately showing a distinction when reaching depths six and eight.

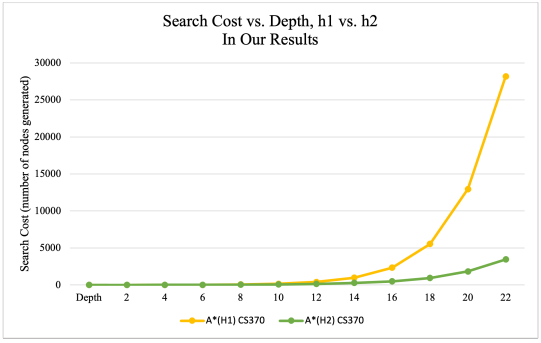

Figure 2: Search Cost vs. Depth in our Paper

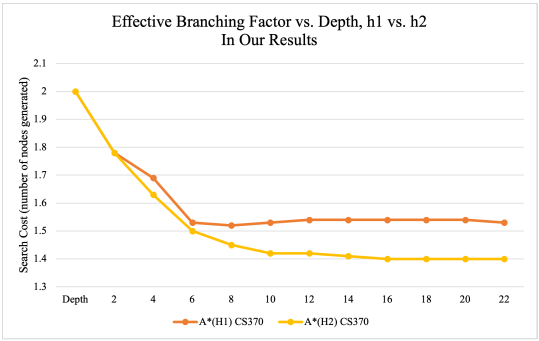

Figure 3: EBF vs. Depth in our Paper

Despite their similarities at lower depths, it's clear that these heuristics produce significantly different results. It is possible, as will be discussed further on, that our heuristics produced EBF results which are similar to one another because the equation we used to calculate EBF rounds values off which determined their final value.

In comparison to the results of Russell and Norvig, our results were similar, yet not entirely the same. For example, our search cost for h_1 was overall lower than Russell and Norvig, while our search cost for h_2 was overall higher than theirs. Additionally, our effective branching factors for h_1 and h_2 were overall higher than those of Russell and Norvig. These results can be seen below in Figures 4 and 5 respectively.

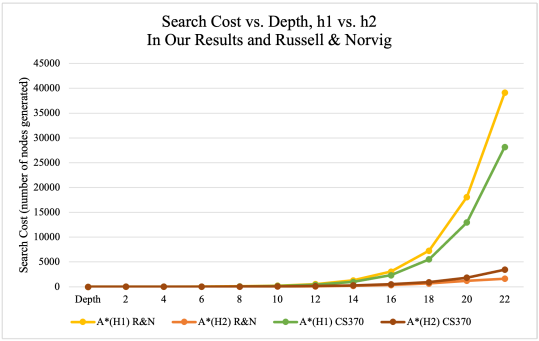

Figure 4: Search Cost vs. Depth in both Papers

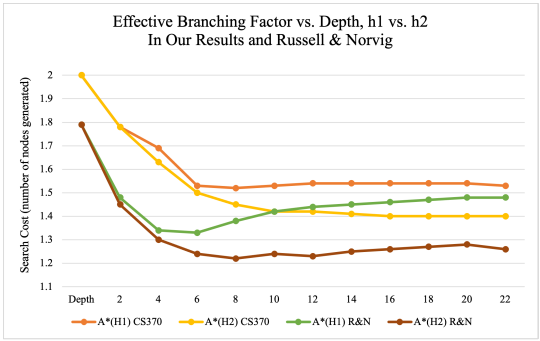

Figure 5: EBF vs. Depth in both Papers

We identified a few reasons as to why these discrepancies might exist. One possible reason is that Russell and Norvig do not define what "nodes generated" is in their case; in ours, it is the total number of nodes added to the priority queue throughout the process of the algorithm. Hence, their definition might differ from ours, and therefore our data for search cost would differ.

Depth	Search Cost h_1	Search Cost h_2	EBF h_1	EBF h_2
2	4	4	2	2
4	10	10	1.78	1.78
6	23	19	1.69	1.63
8	31	25	1.53	1.5
10	68	42	1.52	1.45
12	166	73	1.53	1.42
14	410	145	1.54	1.42
16	978	285	1.54	1.41
18	2346	500	1.54	1.4
20	5542	942	1.54	1.4
22	12960	1835	1.54	1.4
24	28184	3455	1.53	1.4

Figure 6: Our Results

Depth	Search Cost h_1	Search Cost h_2	EBF h_1	EBF h_2
2	6	6	1.79	1.79
4	13	12	1.48	1.45
6	20	18	1.34	1.3
8	39	25	1.33	1.24
10	93	39	1.38	1.22
12	227	73	1.42	1.24
14	539	113	1.44	1.23
16	1301	211	1.45	1.25
18	3056	363	1.46	1.26
20	7276	676	1.47	1.27
22	18094	1219	1.48	1.28
24	39135	1641	1.48	1.26

Figure 7: Russell and Norvig's Results

Another possible reason is that our effective branching factor formula, $N^{1/d}$, is an approximation of the equation provided by Russell and Norvig. Ergo, our effective branching values may differ slightly due to rounding errors or differences in equation approximation.

Finally, another important reason for varying differences is that Russell and Norvig's results are an average over 100 boards at every depth, whereas our results are highly varied since we ran the two heuristics on 10,000 instances, irrespective of depth. For smaller depths, we have much fewer boards than a 100 because due to the nature of the problem, it's far less likely to have a problem with depth two, three or four, whereas for larger depths we have much more data.

Regardless of these result disparities, h_2 performed better than h_1 in our dataset, for both search costs and effective branching factors. This is demonstrated in Figure 6. Likely, this is because the Manhattan distance takes into account the number of steps to the goal position for each tile and therefore calculates a heuristic closer to the actual cost, while the Hamming distance is less accurate to the solution on average. As this agrees with Russell and Norvig's findings in Figure 7 (Russell and Norvig 2003) as well as Lee and See's (Lee and See 2021), it's clear that this is a result which ultimately confirms their results: the Manhattan Distance Heuristic is superior to the Hamming Heuristic.

5 Conclusions

Initially, our goal was to investigate the impact of heuristic quality on search performance using the 8-puzzle. Specifically, we looked at two heuristics and compared the search costs and the effective branching factors for each heuristic with the A* algorithm. We found that in comparison to Russell and Norvig, our search cost for h_1 was generally lower while the effective branching factor was overall higher; and for h_2, both search cost and effective branching factor were generally higher. Finally, according to our results, h_2 performed better than h_1, for both search cost and effective branching factor.

6 Contributions

We were all able to offer input on every aspect of the assignment and complete the code as well as the paper during our in-person meetings. We each proof-read the entire document as well.

References

Lee, S. C., and See, T. H. 2021. Comparing the Hamming and Manhattan Heuristics in Solving the 8—Puzzle by A* Algorithm. *Proceedings of the 7th International Conference on the Applications of Science and Mathematics 2021* 273:189–195.

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