Analysis of Heuristic Functions for the 8-Puzzle Problem

Daniel Zhou,
Davidson College, Davidson, NC 28035
dazhou@davidson.edu,

Donald Lin
Davidson College, Davidson, NC 28035
dolin@davidson.edu

Abstract

In this paper, we delve into the complexities of the 8-puzzle, a classic problem in artificial intelligence, to scrutinize the efficiency of various heuristic functions within the context of the A^* search algorithm. Our study examines and recreates results uncovering discrepancies and introducing a novel heuristic approach, h_3, which exhibits slight superiority in terms of the effective branching factor. Through comprehensive experimentation on all 181440 valid configurations, we reveal the varying effectiveness of heuristic functions and the nuances of their impact on the search process. Our findings highlight the importance of heuristic selection. The insights gained contribute to a deeper understanding of heuristic search algorithms and open avenues for further exploration and enhancement of heuristic functions.

1 Introduction

The realm of artificial intelligence has long been captivated by the challenge of solving puzzles. These puzzles, seemingly simple in nature, often present intricate problems that require sophisticated algorithms to solve efficiently. Among these puzzles, the N-puzzle stands out as a classic problem that has been extensively studied and used as a benchmark in heuristic search algorithms. W. Johnson et al. (Johnson and Story 1879) charaterized the state space of the 15-puzzle as two disconnected components. O. Hansson et al. (Hansson, Mayer, and Yung 1985) generated efficient heuristic functions for the N-puzzle by criticizing solutions. S. Russell et al. (Russell and Norvig 2003) compared two classic heuristic functions of the 8-puzzle. In this paper, we extended the work in (Johnson and Story 1879) to give a full characterization of the state space of the 8-puzzle. We also compared the quality of three heuristic functions of 8-puzzle through careful implementation of the A^* algorithm.

2 Background

The 8-puzzle, a precursor to the more popular 15-puzzle, is a sliding puzzle that consists of a 3×3 board with eight numbered tiles and a blank space. The objective is to transition from an initial state to a goal state, where the tiles are in ascending order, by sliding the tiles using the blank space. Due to its combinatorial nature, the 8-puzzle serves as an ideal test-bed for heuristic search algorithms.

Our study was heavily influenced by Section 3.6 of (Russell and Norvig 2003). Their work implemented A^* algorithm with two heuristics,

- h_1: number of misplaced tiles
- h_2: Manhattan distance

and showcased results from 1200 simulations. They calculated the average effective branching factor for problems of various solution depths. On the high level, the effective branching factor represents the quality of an heuristic function. The smaller the effective branching factor, the better the heuristic function. In later sections, we will introduce our formula for calculating the effective branching factor. When implementing our formula, we used binary search to restrict numerical error within 0.001.

While the contributions of (Russell and Norvig 2003) have been foundational, our research embarked on a more exhaustive journey. With the observation in (Johnson and Story 1879), we examined the parity of permutations and generated all $181,440$ boards of the 8-puzzle that could reach the goal state, aiming for a comprehensive understanding of the problem space. Then, we calculated the effective branching factor of this whole space to compare the quality of h_1 and h_2 specified in (Russell and Norvig 2003). In addition, we introduced a new admissible heuristic, h_3, which builds upon the Manhattan distance (h_2) and, intriguingly, outperforms h_2 in our simulations.

3 Experiments

Our experiments are designed to investigate two primary aspects: the comparative efficiency of different heuristic functions in identifying solutions and a comprehensive overview of the 8-puzzle problem. For notational simplicity, the blank space is denoted as the 0-tile in subsequent discussions.

Generating Rroblems

As pointed by W. Johnson et al. (Johnson and Story 1879), not every tile configuration can be transformed into the goal configuration. This observation is grounded in the principle that an even permutation can be obtained as the composition of only an even number of exchanges. Consequently, precisely half of all permutations, amounting to $\frac{9!}{2} = 181440$ configurations, are deemed "valid," i.e., capable of reaching the goal configuration.

Generating the 181440 valid configurations presents a challenge. Initiating a random walk in reverse from the goal configuration is not a prudent approach. This is due to the fact that, on average, each step offers 3 potential directions, and as will be elucidated in the results section, some configurations necessitate a minimum of 31 steps to attain the goal configurations. Furthermore, this number of 31 is extrapolated from subsequent experiments, leaving us without prior knowledge of the requisite step number for the random walk to enumerate all valid configurations.

Nonetheless, armed with the key observation in (Johnson and Story 1879), we possess the capability to ascertain the validity of each configuration with a straightforward strategy — scrutinize each permutation of the tile, totaling $9! = 362880$ permutations. In order to verify a configuration, we initially relocate the 0-tile (blank space) to the right-down corner, employing interchanges to yield a new configuration. It is evident that any configuration accessible from the original can also be reached from this modified configuration. Given that the goal configuration similarly locates the 0-tile in the right-down corner, any potential solution path from this altered configuration to the goal configuration inherently forms a "loop" of the 0-tile's positions, thereby necessitating an even number of interchanges. Consequently, we can modify the new configuration into the goal configuration by exchanging adjacent tiles, and it is sufficient to determine whether the cumulative number of exchanges conducted is even. Through careful implementation, we force our code to find a sequence of at most 30 interchanges to reach the goal configuration from any configuration. Thus the number of exchanges can be calculated efficiently.

Heuristic Functions

Three heuristic functions, all admissible and consistent, are evaluated in our study:

- h_1 = the number of misplaced tiles.
- h_2 = the sum of Manhatton distances of the tiles from their goal positions.
- h_3 = the sum of Manhatton distances of the tiles from their goal positions plus 2 times the number of direct adjacent tile reversals.

The admissibility of h_1, h_2 is established in (Russell and Norvig 2003). They also exhibit consistency, given that each edge has unit length and h_1, h_2 alter by at most 1 when a single tile is moved to an adjacent position. Note that h_3 is a straightforward modification of h_2. The heuristic function h_3 is admissible, as for each pair of nodes in an adjacent reversal, relocating one tile to the alternate position necessitates the initial movement of the 0-tile to that position. This action displaces the other tile in the pair, requiring it to traverse a minimum of 3 times to reach the goal position. Such action marks an increase of 2 from its Manhattan distance of 1.

Similar to h_2, h_3 maintains consistency. If the movement of a tile neither forms nor disrupts an adjacent tile reversal, it is evident that h_3 changes by 1. Also, it is clear that moving a tile cannot simultaneously form and disrupt adjacent tile reversals. If the tile movement disrupts an adjacent tile reversal, the tile must be relocated from a position with a Manhattan distance of 1 to the goal position, to a position with a Manhattan distance of 2. This results in a change in h_3 by $-2 + 1 = -1$. Similarly, if the tile movement forms an adjacent tile reversal, it must be moved from a position with a Manhattan distance of 2 to the goal position to a position with a Manhattan distance of 1, leading to a change in h_3 by $+2 - 1 = 1$. Therefore, h_3 is consistent.

Now, we have established the admissibility and consistency of h_1, h_2, and h_3.

Effective Branching Factor b^*

Suppose A^* algorithm finds the solution to a problem. Denote d as the depth of this solution in the search tree, i.e. the number of steps to reach the goal state. Denote N as the number of nodes pushed into the frontier of the A^* algorithm throughout the loop procedure. It should be noted that the starting node is excluded from the N count as it is not appended during the loop. The frontier may contain multiple instances of the same configuration, each contributing to the N count. These repetitive configurations are perceived as leaves of the search tree, which, through careful implementation, are precluded from subsequent visits. Consequently, a minor modification in the A^* implementation suffices to enumerate N; specifically, we increment N by 1 each time a node is pushed into the frontier.

The variable N serves as a representation of the **search cost**. The rationale behind excluding the initial node from this calculation is that it is not discovered during the search process; rather, it is give to the algorithm as input. Nonetheless, the A^* search tree incorporates this initial node as the only node at depth 0, resulting in a total of $N+1$ nodes within the search tree.

The **effective branching factor** of A^* over a problem, denoted as b^*, is defined as the branching factor that a uniform tree of depth d must possess to accommodate an equivalent number of nodes as the search tree. This relationship is mathematically represented as:

$$N + 1 = \sum_{0}^{d} (b^*)^d.$$

Overview of Experiments

In our experiment, we implement A^* with h_1, h_2, or h_3 as its heuristic functions. Subsequently, for each of the 181440 valid configurations, we apply A^* to find the length of its optimal path, i.e. solution depth(d) to the goal configuration, and count the number of nodes(N) pushed into the frontier. To determine the solution depth, it is imperative that each node in our research tree encapsulates not only the configuration itself but also the corresponding depth of this configuration.

Following this, we will employ a binary search methodology to calculate the effective branching factor b^* for each problem solved with A^* algorithm. Our encoding of binary search forces the error to be within 0.001. The derived information will be documented in a text file (output file). In the concluding phase, a parser is used to extract data from the text file, enabling us to acquire the following: the number of problems at each solution depth, the average search cost(N) for problems at each solution depth, and the average effective branching factor(b^*) for problems at each solution depth.

4 Results

Distribution of Problems

The number of problems under each solution depths are the same in terms of h_1, h_2, and h_3, which reflected our implementation is correct and these heuristic functions are indeed admissible and consistent. The distribution of these problems is shown in Figure 1, where Num is the number of problems of a given depth. The y-axis is $log(Num)$ and x-axis is d(depth). We take log of Num because the number of problems varies a lot: depth 24 contains the most problems(24047) while depth 1 and 31 contain the least problems, each of only 2 problems. We ignore depth 0 problems(where the initial configuration is the goal configuration) because it's trivial. As shown in the figure, a valid configuration needs at most 31 steps to reach the goal configuration. The change of $log(Num)$ in Figure 1 is smooth, which makes sense as we would expect some exponential speed of increase and decrease. It should be noticed that the overall trend of increase then decrease doesn't hold everywhere locally, as we see a decrease from depth 22 to depth 23 but then an increase from depth 23 to depth 24.

Comparison of Heuristic Functions

In Table 1, we list the search cost(the number of nodes added to frontier) and effective branching factor averaged over all problems of the same depth. For example, when implementing A^* with h_1 as heuristic function, the table shows the average search cost(N) over all 24047 valid configurations of solution depth 24 is 25417.70. A visualization of these data can be found in Figure 2. The figure and table show that h_1, h_2, h_3 have close effective branching factor in depths 1 to 4. Starting from depth 5, we can see significant difference starts to occur between the effectiveness of h_1 and the effectiveness of h_2 or h_3. The effective branching factor of h_2 and h_3 are always very close, because h_3 is just a small modification of h_2. Nevertheless, we can still find h_3 slightly better than h_2 in terms of effective branching factor, as h_3 always give estimates at least the estimate of h_2. Since

h_2 and h_3 are both admissible, this means h_3 never guess farther from the true cost than h_2.

It should be noticed that h_2 and h_3 show a steady increase in effective branching factor from depth 10 to depth 30. This trend is only weakened a little at the end from depth 30 to 31. On the other hand, h_1 shows more fluctuation, probably because h_1 is not good extimate of the distance to goal configuration so it doesn't reflect properties of problems as precisely as h_2 or h_3. Nevertheless, h_1, h_2, and h_3 all show an overall trend of decrease, increase, then decrease, so they all reflect some properties of problems of different depths.

Comparision of Our Data with Data from W. Johnson et al.

In Figure 2, we also draw curves representing the average effective branching factor obtained by W. Johnson et al.. They share the same trend with curves from our data. However, there are some tiny differences. We don't know how (Russell and Norvig 2003) generated their data, but apparently they didn't use unique problems to do the average calculation, because they generated 100 problems with depth 2 but there are only 4 such unique problems. It would be difficult to guess the distribution of problems generated by them. In their generated problem space, some problems may have significantly larger multiplicity than others in the same depth, which would lead to their data not representative.

Observe that in most depths our data gives a smaller effective branching factor–with the only exception on depth 24 of h_2. This is probably because our implementation of A^* checks the frontier frequently to avoid adding or visiting meaningless nodes: each time before we try to push a node into the frontier, we will check whether the configuration has already been added; each time we pop a node, before exploring its children, we also check whether the configuration has already been added.

5 Conclusions

In this paper, we examined the behaviour of A^* algorithm implemented with three heuristic functions when solving all valid configurations of the 8-puzzle. We computed the average effective branching factor, a property representing the quality of heuristic functions, of problems with each solution depth. We found that for all three heuristic functions, the effective branching factor shows a decrease-increase-decrease pattern. Our results are in align with the data in (Russell and Norvig 2003) in terms of general pattern: heuristic function h_2 is often much better than h_1. This result reveals the importance of heuristic selection when implementing A^* algorithm. We also computed the effective branching factor of h_3, a modification of h_2 that is not presented in (Russell and Norvig 2003), whose behaviour is slightly better than h_2.

We also presented the distribution of all valid configurations of the 8-puzzle, in terms of solution depths. As expected, our experiments show an exponential increase-decrease pattern, with a maximum of 24047 at depth 24. Our experiments also show that a valid configuration needs at most 31 steps to transition into the goal configuration. A future research direction is to examine why 31 is the maximum solution depth. The authors believe the answer to this question could reveal some general properties of the N-puzzle problems.

6 Contributions

Both authors contributed to the study conception and design. Both authors read and approved the final manuscript. D. Zhou and D. Lin did pair programming during the implementation of A^*. Both authors generated figures and tables during the project, and those generated by D. Zhou are used for the final version. D. Zhou wrote the experiments and results sections. D. Lin wrote the abstract, introduction, background, conclusion, contribution, and acknowledge sections.

7 Acknowledgements

We would like to thank Raghu Ramanujan for suggestions about the struction of this paper. We would also like to acknowledge the assistance of chatGPT in refining the clarity and coherence of our writing.

References

[Hansson, Mayer, and Yung 1985] Hansson, O.; Mayer, A. E.; and Yung, M. 1985. Generating admissible heuristics by criticizing solutions to relaxed models.

[Johnson and Story 1879] Johnson, W. W., and Story, W. E. 1879. Notes on the "15" puzzle. *American Journal of Mathematics* 2(4):397–404.

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

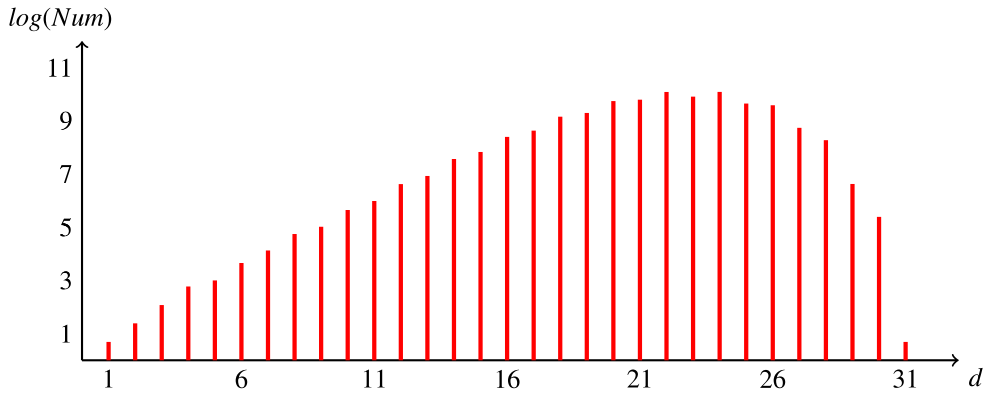

Figure 1: Discrete distribution of all valid problems over different optimal solution depth

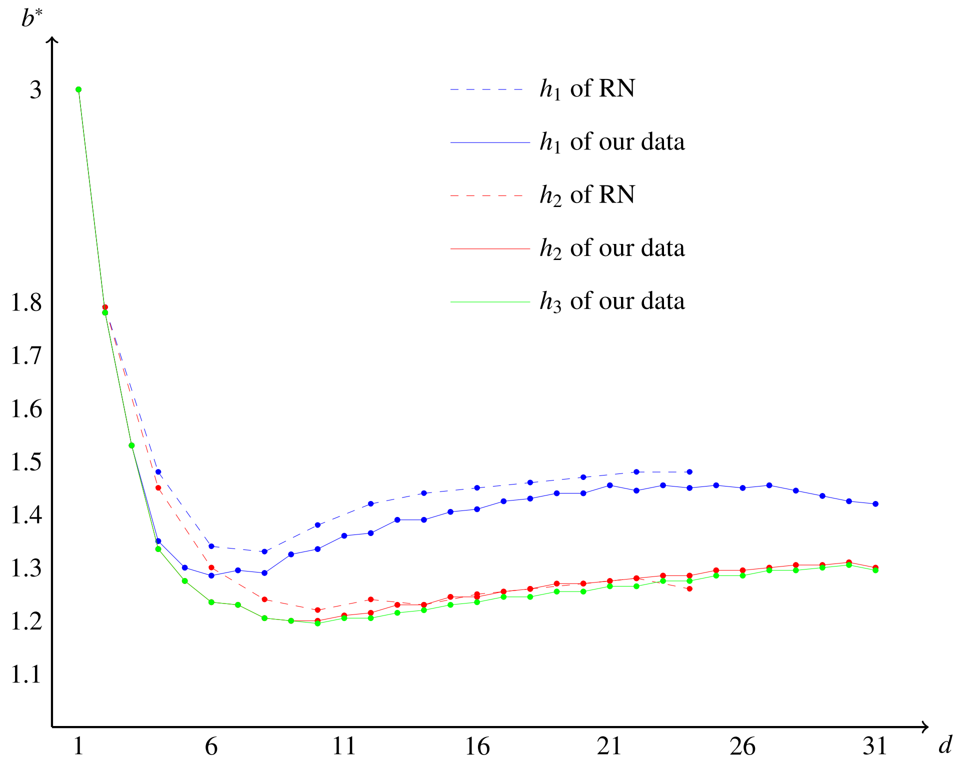

Figure 2: Comparison of the effective branching factors for different heuristic functions from W. Johnson et al.(RN) and our data

	Search Cost(N)			Effective Branching Factor(b^*)		
d	$A^*(h_1)$	$A^*(h_2)$	$A^*(h_3)$	$A^*(h_1)$	$A^*(h_2)$	$A^*(h_3)$
1	3.00	3.00	3.00	3.00	3.00	3.00
2	5.00	5.00	5.00	1.78	1.78	1.78
3	7.50	7.50	7.50	1.53	1.53	1.53
4	9.00	8.75	8.75	1.35	1.34	1.34
5	12.00	11.00	11.00	1.30	1.27	1.27
6	16.41	13.64	13.64	1.28	1.24	1.24
7	23.26	17.74	17.74	1.29	1.23	1.23
8	31.18	20.92	20.82	1.29	1.21	1.21
9	48.46	26.16	25.78	1.32	1.20	1.20
10	68.59	32.59	31.58	1.33	1.20	1.19
11	109.94	43.95	41.79	1.36	1.21	1.20
12	155.04	56.22	52.26	1.36	1.21	1.20
13	252.82	78.66	71.30	1.39	1.23	1.22
14	365.68	101.73	90.55	1.39	1.23	1.22
15	579.47	145.08	126.33	1.41	1.24	1.23
16	873.67	187.69	162.56	1.41	1.25	1.23
17	1373.78	262.60	225.70	1.42	1.26	1.24
18	2083.29	339.81	289.46	1.43	1.26	1.25
19	3440.55	476.00	407.50	1.44	1.27	1.26
20	4832.50	618.45	523.71	1.44	1.27	1.26
21	8321.82	873.79	736.93	1.45	1.28	1.27
22	11242.80	1140.24	958.49	1.45	1.28	1.27
23	18411.10	1626.61	1368.96	1.46	1.29	1.28
24	25417.70	2131.34	1799.96	1.45	1.29	1.28
25	39196.20	3102.94	2627.68	1.46	1.29	1.29
26	50880.90	4073.20	3472.52	1.45	1.30	1.29
27	78581.10	5940.17	5097.28	1.45	1.30	1.29
28	94319.50	7800.02	6744.01	1.44	1.30	1.30
29	122300.00	11553.90	9808.65	1.44	1.31	1.30
30	144704.00	15297.70	13079.50	1.43	1.31	1.30
31	179442.00	14877.50	13132.50	1.42	1.30	1.29

Table 1: Average of search cost and effective branching factor over all problems of given solution depth for A^* with h_1, h_2, or h_3 as heuristic funtion