Solving the 8-Puzzle

Pasha Sonkin and **Joshua Wallace**
{pasonkin,jowallace}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

In this paper we implemented the A^* algorithm on the 8-tile puzzle to compare the quality of different search heuristics, and to replicate and build on previous results in the field. Through analysis of the average branching factor, we determined that the Linear Conflict heuristic generated the fewest number of nodes before finding a solution, followed by the Manhattan Distance heuristic, which in turn was far superior to the Misplaced Tiles heuristic. These findings are in line with previous results.

1 Introduction

The 8-tile puzzle (Figures 1, 2) is a puzzle in which tiles numbered 1-8 are placed on a 3 by 3 board. These 8 tiles are permitted only to be moved if it is into the single empty space on the board, and it must be adjacent to this empty space. The goal of the game is to move the tiles back in to the goal state as shown in figure 2. In their experiment (Russell and Norvig 2003), Russell and Norvig generated 1200 different puzzles with solution depths ranging from 2 to 24. Their experiment applied Iterative Depth First Search and A* with two different heuristics on these puzzles in order to demonstrate the effectiveness of informed search and the relative efficacy of different search heuristics. They demonstrated that A^* using the Manhattan Distance heuristic performed better than A^* utilizing the Misplaced Tiles heuristic, which in turn was far superior to IDS, as measured by nodes generated and effective branching factor.

Russell and Norvig also described a method to generate more heuristics by examining heuristics as solutions to relaxed problems. Relaxed problems are versions of the original problem with rules removed or eased, and to be effective for use in A^*, should be easily solvable. According to Russell and Norvig, the closer the relaxed problem is to the original, the better it will act as a heuristic (Russell and Norvig 2003).

We recreated their experiment in order to confirm the results and also added a third heuristic, Linear Conflict. This heuristic had fewer rules relaxed than the two used by Russell and Norvig. Since it was closer to the original problem, we expected the Linear Conflict heuristic to perform better than the original two heuristics.

template/8puzzle1Random.png

Figure 1: Random Arrangement.

template/8puzzle1Goal.png

Figure 2: Goal State.

2 Background

Relaxed problems as described by Russell and Norvig have state-spaces that are supergraphs of the original problem's statespace, since one could always solve the new problem by playing by the harder problem's rules. This makes the heuristics admissible, since the solution length they will give will be at most equal to the actual solution length. We also have consistency because the heuristics give exact solutions for the relaxed problems, so they must obey the triangle inequality.

For the original tile problem, there are two rules:

1. Tiles can only be moved to vertically or horizontally adjacent tiles.
2. Tiles can only be moved to a space that is empty.

Below we describe the three heuristics we used, and which rules were relaxed to generate them.

1. Misplaced Tiles:
   For each of the 9 positions on the board, determine if the tile currently occupying the position is in its goal state. If the tile is in the correct location, nothing is added to the heuristic cost. If the tile is not in its goal state, 1 is added to the heuristic cost of the board.

   In this case, both rules are relaxed. We can move tiles to any space on the board, regardless of proximity or if the space is currently occupied.

2. Manhattan Distance:
   For each of the 8 numbered tiles on the board, determine the distance the tile must travel to reach its goal state. The constriction of moving into the empty space is removed for this estimate. Sum up the total distances of all tiles to get the heuristic cost.

   In this case, only the rule that states that the spaces we move tiles to must be empty are relaxed. Tiles can still only be moved to vertically or horizontally adjacent tiles.

3. Linear Conflict
   The sum of the Manhattan Distance cost of a board and an added cost of 1 for every instance of tiles being in linear conflict. Two tiles are in linear conflict if they occupy the same line, their goal positions are also on that same line, and the tiles must move past each other to reach their goal states.

   In this case, we partly include the restriction that the Manhattan Distance heuristic excluded, by slightly penalizing if there is a Linear Conflict. This is admissible since it takes more than one extra move for two tiles to switch places relative to passing through each other.

Russell and Norvig claimed that the closer the relaxed problem was to the original problem in terms of adjusted rules, the better the heuristic would perform. We then expect that Linear Conflict will perform better than Manhattan Distance, which will outperform Misplaced Tiles.

We also had to generate the 1200 different start states of different search depths. To accomplish this, we began an exhaustive breadth-first search from the goal state, which ensured that all tiles we would find would be solvable. We maintained a frontier of enqueued search tiles and we also had each tile keep track of its distance away from the goal state. Due to the properties of BFS, we knew that whenever the front of the frontier had a tile with a greater depth than the one we had just popped, the entire frontier only had tiles of that new depth. Therefore, every time we encountered a new depth we would permute the entire frontier, which would give us our random start states. We saved 100 problems for each depth, except for the depths 2,4,6, where there were only 4,16, and 40 total problems respectively.

3 Experiments

Our goals for this experiment were to compare the three different search heuristics, in order to replicate Russell and Norvig's results, and to explore one of their methods of new heuristic generation.

We did this by running A^* with each of the three heuristics on 100 (or fewer if there fewer total different boards) different tiles of search depths from 2 to 24, incrementing by 2.

We first tracked the total number of nodes generated. Every time that we popped a new node from our frontier, we generated a set of neighbors for that node. For each neighbor generated, we added 1 to the number of total nodes for that A^* search, and we also added 1 for the initial node. Then, we summed all of the A^* node sums for each depth, and divided them by the number of different problems for that depth to get our number for the average total number of nodes generated per problem per depth.

We then calculated the effective branching factor at each depth. According to Russell and Norvig, the effective branching factor (b^*) is one way to characterize the quality of a heuristic. If the total number of nodes generated by A^* for a particular problem is N and the solution depth is d, then b^* is the branching factor that a uniform tree of depth d would have to have in order to contain $N + 1$ nodes.(Russell and Norvig 2003) Since there is no closed-form solution for calculating b^*, we determined the effective branching factor using a binary search between 0 and $N^{1/d}$.

Finally, we tracked how the total runtime of A^* with each heuristic by timing the time it for each heuristic to solve all 1200 problems.

4 Results

As expected, our data (Figure 4) shows that h_1 - the Misplaced Tiles heuristic, was inferior to h_2 - Manhattan Distance, which in turn generated more nodes than Linear Conflict. These results are consistent with our expectations, and Russell and Norvig's results.

In terms of our data, the decrease in branching factor was substantially greater between h_1 and h_2, 9.25%, compared to the decrease between h_2 and h_3, only 1.36%. These findings may occur because the difference between the h_1 and h_2 heuristics is an entire rule, while between h_2 and h_3 we partially relax a rule. However, this is difficult to quantify. There also may be diminishing returns as a result of our heuristics getting closer to solving the actual problem. Another interesting aspect was the relative total time the heuristics took to run. While h_1 was clearly inferior with 154.73 total seconds, h_2 actually had a lower runtime than h_3 with a runtime of 20.33 to h_3's 23.38. This was caused by the more involved heuristic calculation of h_3, which was expensive enough that it took longer even with fewer total nodes generated. This is a 15% increase in time, which could imply that if time constraints were more important than memory constraints h_2 would be preferred. However, for bigger puzzles with higher default branching factors, the difference in memory performance between h_2 and h_3 could be higher, which could decrease the time gap.

We were able to replicate Russell and Norvig's data with reasonable accuracy. Treating their values for their branching factors as the targets and ours as the simulations, the root mean squared error was .047 and .054 for h_1 and

h_2 respectively. However, since these were calculated on random subsets of 100 problems per search depth we expected that different subsets would result in different branching factor values. While we had insufficient time to calculate the average variance in subsets, upon recalculating our random subsets, our branching factors differed from our previous results by up to .051, which leads us to believe that our data is near enough to theirs. Furthermore, Russell and Norvig's branching factor calculations were inconsistent with their generated node counts. For example, for h_2 at a depth of 24, they had 1641 generated nodes for a branching factor of 1.26. However, the effective branching factor for 1641 nodes at a depth of 24 is 1.277, making for a difference .017. With these two factors in mind, our results seem accurate.

Figure 3: Russell and Norvig's Data

	Search Cost (nodes generated)		Effective Branching Factor	
d	$A^*(h_1)$	$A^*(h_2)$	$A^*(h_1)$	$A^*(h_2)$
2	6	6	1.79	1.79
4	13	12	1.48	1.45
6	20	18	1.34	1.30
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 4: Our experimental data.

	Search Cost (nodes generated)			Effective Branching Factor		
d	$A^*(h_1)$	$A^*(h_2)$	$A^*(h_3)$	$A^*(h_1)$	$A^*(h_2)$	$A^*(h_3)$
2	6	6	6	1.791	1.791	1.791
4	10	9	9	1.352	1.298	1.298
6	18	16	16	1.308	1.269	1.269
8	33	25	25	1.307	1.243	1.243
10	69	40	39	1.336	1.239	1.234
12	158	70	63	1.369	1.253	1.238
14	381	130	109	1.398	1.270	1.249
16	908	256	210	1.418	1.289	1.269
18	2158	508	373	1.433	1.304	1.277
20	5124	959	722	1.445	1.312	1.290
22	12263	1898	1394	1.455	1.322	1.300
24	27694	3418	2577	1.459	1.324	1.306

5 Conclusions

We set out to implement the A^* algorithm on the 8-tile puzzle in order to compare the quality of different search heuristics and to replicate and build on previous results in the field. As Russell and Norvig claimed, heuristics that solved more restrictive relaxed subproblems of the original problem were more effective in cutting down the number of generated nodes.

However, a concern could be that with sufficiently complex subproblems, the time-cost of calculating the heurstic could be large enough that the space-savings would not be sufficient.

Our take-away message is that search heuristics that solve problems closer to the original problem are more effective. However, there may be a trade-off in the cost required to perform those heuristics.

Question for further exploration: How would the efficacy of the linear conflict heurisitic change with an increase of the penalty cost added for linear conflicts? If the penalty cost could be maximized while the heuristic is also still admissible, will it be more effective?

6 Contributions

Josh wrote/created: The heuristics, abstract, tables and figures, references, and acknowledgements

Pasha wrote/created: AStar, effective branching factor, experiments, generating random start tiles.

Both equally collaborated on: Introduction, Background, Results, Conclusions, proofreading entire document.

7 Acknowledgements

We would like to thank Dr. R for answering our questions and being generally informative during this project. Additionally, we would like to acknowledge we used Dr. Gabriel Ferrer's method[1] for calculating effective branching factor.

References

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

[1]http://ozark.hendrix.edu/ ferrer/courses/335/f11/lectures/effective-branching.html