A* and the 8-Puzzle

Jackson Spell and **Ben Wiley**
{jaspell,bewiley}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

An 8-puzzle is a tile puzzle which is solved by rearranging tiles via sliding to reach an ordered goal configuration. We use the A* local search algorithm to solve the 8-puzzle, comparing the behavior and efficiency of 3 heuristic functions in order to evaluate the difference in puzzle generation cost and effective branching factor. The heuristics used evaluate, respectively, the number of out-of-place tiles, the minimum number of moves needed for each tile to reach its goal position, and the sum of the minimum number of moves and total linear conflict, which is the number of out-of-order tile conflicts in each row and column.

1 Introduction

The 8-puzzle is one of a number of relatively simple problems, known as toy problems, which arise from logic tasks and are used to illustrate algorithm behavior, but do not describe real-world problems. The sliding-block family of puzzles are difficult to solve, with large state spaces, but have easily established representations and heuristics and have been historically used to evaluate search algorithms (Ratner and Warmuth 1986).

The 8-puzzle is a 3×3 board with 8 tiles, each of which can slide in the cardinal directions, and which cannot overlap. Moves between board states take the form of sliding a single tile. The goal configuration is an ordered board.

The 3×3 puzzle has 181,400 reachable states (Russell and Norvig 2003). Previous analyses exploring larger puzzles have shown that solving the N×N puzzle is NP-hard and thus computationally intractable (Ratner and Warmuth 1986). The problem is therefore an appropriate target for the lower memory costs and autonomous execution of a local search algorithm.

In this paper, we describe the heuristics used by the A* algorithm, detail our experimental approach, including our random board generation method, and then present the results of the comparison in heuristics.

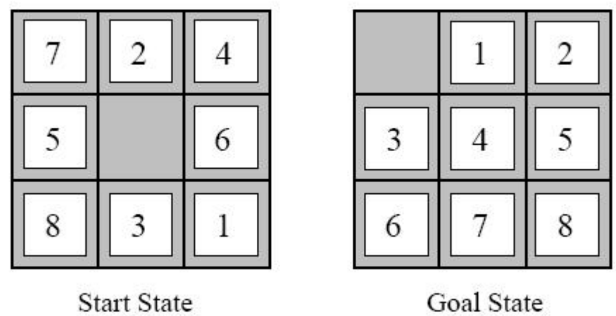

Figure 1: A randomized 8-puzzle and the goal configuration (Centurion 2015).

2 A* and Heuristics

We employ three heuristics based on board state and tile positions relative to the goal (Russell and Norvig 2003). The purpose of this experiment is to evaluate the difference in efficiency of A* search between the three heuristics. The first heuristic tallies the number of tiles out of place on the board, not including the empty space, and the second sums the "Manhattan distance" from each of the tiles to its target position in the goal configuration – the minimum number of moves needed to reach the target, ignoring the presence of other tiles. The third, the Linear Conflict heuristic, adds together the result of the second heuristic with the board's total linear conflict – the number of out-of-order tile conflicts in each of the board's rows and columns (Hansson, Mayer, and Yung 1985). Each heuristic is consistent, as each evaluates the minimum number of moves necessary to shift the tiles, which is the lower bound for the actual number of moves, considering other tiles may obstruct the shift.

Because each heuristic is consistent, our A* implementation makes use of a set of previously discovered nodes, termed the *closed set*. A consistent heuristic will always reach a node using the optimal path, so once generated, nodes need not be reopened. Due to the local nature of A*, though, the boards corresponding to nodes in the closed set may be recreated by the algo-

	h1		h2		h3	
Depth	Search Cost	Branching Factor	Search Cost	Branching Factor	Search Cost	Branching Factor
2	6 ± 0.19	2.95 ± 0.04	6 ± 0.19	2.95 ± 0.04	6 ± 0.19	2.95 ± 0.04
4	10 ± 0.23	2.01 ± 0.01	10 ± 0.13	1.99 ± 0.01	10 ± 0.13	1.99 ± 0.01
6	17 ± 0.87	1.73 ± 0.01	14 ± 0.48	1.69 ± 0.01	14 ± 0.35	1.68 ± 0.01
8	32 ± 1.76	1.63 ± 0.01	22 ± 1.06	1.56 ± 0.01	20 ± 0.99	1.55 ± 0.01
10	73 ± 3.22	1.61 ± 0.01	36 ± 2.34	1.50 ± 0.01	31 ± 2.11	1.48 ± 0.01
12	170 ± 6.17	1.6 ± 0.0	59 ± 4.9	1.46 ± 0.01	54 ± 5.66	1.44 ± 0.01
14	398 ± 15.48	1.59 ± 0.0	106 ± 8.94	1.44 ± 0.01	101 ± 11.99	1.43 ± 0.01
16	984 ± 40.17	1.59 ± 0.0	209 ± 18.95	1.44 ± 0.01	196 ± 28.03	1.42 ± 0.01
18	2365 ± 121.84	1.59 ± 0.0	400 ± 37.96	1.43 ± 0.01	308 ± 44.55	1.40 ± 0.01
20	5093 ± 387.1	1.59 ± 0.0	727 ± 78.31	1.44 ± 0.01	505 ± 52.59	1.39 ± 0.01
22	11412 ± 1099.62	1.59 ± 0.0	1344 ± 154.37	1.44 ± 0.01	876 ± 128.65	1.38 ± 0.01
24	24521 ± 2679.15	1.58 ± 0.0	2410 ± 327.77	1.43 ± 0.01	1152 ± 156.18	1.36 ± 0.01

Table 1: Experiment data.

rithm when exploring the frontier of possible moves from other board states. Such boards do not pollute the search operation, however, as their presence in the closed set prevents the algorithm from reentering the node into the priority queue.

3 Experiments

The objective in this experiment is to investigate the difference in search cost and effective branching factor between the three heuristic functions. We define the cost of the search to be the number of nodes generated – in this case, the number of boards created by moving tiles, then processed through the priority queue. The effective branching factor is calculated, for a solution of depth d and search cost of N, by

$$N + 1 = 1 + b^* + b^{*2} + ... + (b^*)^d$$

where b^* is the effective branching factor. This polynomial equation is solved using a numerical solver.

The heuristics are compared by averaging the search cost and effective branching factor for a given depth across 100 search operations, with the board randomly initialized each time. It should be noted that, though the maximum solution depth for an 8-puzzle is 31 moves, we evaluate boards of solution depths 2 - 24, which is sufficient to resolve the difference in efficiency between heuristic functions. Because half of the possible board configurations are unsolvable (Johnson and Story 1879), we create the randomized board by starting in the goal configuration and making a number of random legal moves, thus ensuring the board never enters an unsolvable state. The number of moves is itself a random number between 2 and 100 in order to fully populate the solution depth space.

The possibility of cycles in the moves during the board randomization means that the actual solving depth is bounded on the upper side by the number of randomization moves. The actual depth is found by solving the board with one heuristic (h3(n)). The other heuristics runs only if 100 search operations of the found depth have not already been completed; that is, h1(n) and h2(n) run only if more searches are needed at that depth in order to correctly average the algorithm's behavior.

4 Results

The results of the experiment are reported in Table 1 as the mean value with a 95% confidence interval.

The average search cost for the second heuristic, h2, is far lower than that of h1 for a given solution depth. This behavior is expected, as h2 more accurately evaluates the distance of the board state from the goal configuration. Unlike h1, h2 takes into account not only the fact that a tile is misplaced, but the minimum number of moves by which the tile has been separated from its goal location. h2 therefore prioritizes proximity to the goal configuration, which is far more valuable during the solving of the puzzle, as shifting the final tiles into the goal configuration requires displacing adjacent tiles. h3 performs more efficiently still, as it evaluates the number of tiles which are "in each other's way", an approximation of the number moves needed to reconcile the out-of-order tiles.

5 Conclusions

The 8-puzzle is a classic toy problem used to evaluate the performance of heuristic search algorithms. We evaluate the performance of the local search algorithm A* using three different heuristics based on the number of nodes generated during the search and the algorithm's effective branching factor. Compared to a heuristic evaluating simply the number of board tiles which are not in the final goal configuration, far better performance is seen from the heuristic which evaluates the distance of each component of the board from its location in the goal configuration, allowing the algorithm to evaluate marginal improvements to board state rather than the lower "resolution" of only detecting tiles in their final states. Further, the third heuristic's usage of linear conflict in addition to Manhattan distance ensures sharper accuracy than the first two heuristics. This experiment highlights the vast improvement in search efficiency using heuristics which more closely approximate the move cost between states.

6 Acknowledgements

The authors would like to thank Ross Kruse and Laura Hiatt for their assistance in optimizing data structures within the A* algorithm, as well as Dr. Ramanujan for his guidance in evaluating the algorithm's performance.

References

Centurion. 2015. Example state of an 8-puzzle. `http://centurion2.com/AIHomework/AI120/ai120.php`.

Hansson, O.; Mayer, A. E.; and Yung, M. 1985. Generating admissible heuristics by criticizing solutions to relaxed models. `http://academiccommons.columbia.edu/catalog/ac\%3A141289`.

Johnson, W. W., and Story, W. E. 1879. *Notes on the" 15" puzzle*.

Ratner, D., and Warmuth, M. K. 1986. Finding a shortest solution for the n× n extension of the 15-puzzle is intractable. In *AAAI*, 168–172.

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