CSCI 4511W Writing 5

Sid Lin (linx1052) and Noah Park (park1623)

April 12, 2020

One of the most common introductory topics to AI are search algorithms. Search algorithms are one of the most basic and familiar topics for computer scientists beginning to expand their learning into the realm of AI. These algorithms can be split into two categories, uninformed search and informed search. Many search algorithms have practical applications with familiar problems, such as the 8 queens, traveling salesperson problem, map coloring, N x N slide puzzle, and much more. For our project, we will be focusing on the 8-puzzle with informed search.

The 8-puzzle scenario is none other than the familiar slide puzzle; each state consists of a N x N grid with all numbered tiles, aside from a single missing one. The goal of the puzzle is to move each tile into a certain order with the fewest moves and time. It may appear as if the moves are made by moving the numbered tiles into the empty space, but a simpler explanation is that the empty tile moves around the board. A board that is 3 x 3 will have 9! possible permutations, but only half of them are solvable. The sheer amount of possible permutations that make the 8-puzzle is an ideal candidate for an informed search analysis [5].

Each configuration of the 181,440 possible instances of the 8-puzzle was tested with an optimal iterative deepening algorithm [7]. Richard Korf's IDA* algorithm was used in conjunction with the Manhattan distance heuristic to solve each configuration. Each solution had more than one optimal solution, but the average number of steps to solve the puzzle was consistent. The maximum number of steps to solve the 8-puzzle was 31, and the average was about 22 [7]. The ideal formulation of a search space for this problem would be a graph [6]. However, our instance of the 8-puzzle consisted of nodes that created a search tree. Graphs would be ideal due to the ability to undo a move, but for simplicity we chose to use nodes instead of vertices and edges.

One of the most popular informed search algorithms is A*. It is an extension of Dijkstra's algorithm. The A* algorithm uses a cost function that takes both path cost and heuristic cost into account. Normally, A* search spaces are implemented with nodes, which can have children and a predecessor node. Most importantly, A* uses a priority queue ordered with the lowest function cost to determine each node's order of expansion. The standard function is denoted f(x) = g(x) + h(x), where g and h are the path cost and heuristic cost respectively [4]. The optimality of this algorithm depends on the admissibility of the heuris-

tic function as well. Admissible heuristics are ones that will never overestimate the cost of reaching its goal. Furthermore, the other condition for optimality is known as consistency. Consistency, for every node n and every successor n' of n generated by any action a, is described as the estimated cost of reaching a goal from n being no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n'. This is also known as the triangle inequality [8].

With all informed search algorithms, there has to be some way of gathering more information from the search space; this is where the use of a heuristic makes these algorithms distinct. We chose to use a version of the Manhattan distance heuristic. The heuristic is traditionally the straight line distance from each tile to its goal location [2]. However, our implementation of the 8-puzzle was done in a one dimensional array, so we decided to use the index distance from a tile to its original place. The chosen heuristic is quite important, as it can change the way the A* algorithm will behave. Daniel R. Kunkle did extensive testing on heuristics with this problem. He noted that the most sophisticated heuristic is the Manhattan Distance, and that a heuristic of 0 will result in a BFS algorithmic behavior. It is important to choose the heuristic wisely when dealing with the A* algorithm [3].

Our implementation of the 8-puzzle was done in Swift and published on the iOS App Store. To measure the effectiveness of the A* search algorithm, we had a board randomization function with 4 levels of difficulty, correlated with the number of random moves made. Future advances on our solution could be led to genetic algorithms. An interesting take on this was found by Harsh Bhasin and Neha Singla. They found that by implementing the "natural selection" employment of the nature of genetic algorithms, many optimal solutions were found. They concluded that genetic algorithms were the best type of algorithm to solve these problems [1].

References

[1] Harsh Bhasin and Neha Singla. Genetic based algorithm for n-puzzle problem. *International Journal of Computer Applications*, 51(22), 2012.

[2] Ethan Andrew Burns, Matthew Hatem, Michael J Leighton, and Wheeler Ruml. Implementing fast heuristic search code. In *Fifth Annual Symposium on Combinatorial Search*, 2012.

[3] Daniel R Kunkle. Solving the 8 puzzle in a minimum number of moves: An application of the a* algorithm. *Introduction to Artificial Intelligence*, 2001.

[4] Masoud Nosrati, Ronak Karimi, and Hojat Allah Hasanvand. Investigation of the*(star) search algorithms: Characteristics, methods and approaches. *World Applied Programming*, 2(4):251–256, 2012.

[5] Rok Piltaver, Mitja Luštrek, and Matjaž Gams. The pathology of heuristic

search in the 8-puzzle. *Journal of Experimental & Theoretical Artificial Intelligence*, 24(1):65–94, 2012.

[6] Daniel Ratner and Manfred Warmuth. The (n2- 1)-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111–137, 1990.

[7] Alexander Reinefeld. Complete solution of the eight-puzzle and the bene t of node ordering in ida*. In *International Joint Conference on Artificial Intelligence*, pages 248–253, 1993.

[8] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.