Analysis of the Search Algorithms for Solving the 8-Puzzle Game

ANDREA NICOLE G. PRIVADO

2019-03716

X-1L

Introduction

The 8-puzzle is a puzzle that can be played on a 9-tiled 3-by-3 grid with 8 tiles labeled 1-8, plus a blank tile. The goal of this game is to rearrange the tiles through depending on the goal configuration defined by the game creator. The player is allowed to slide tiles either horizontally or vertically into the blank square.

This game can be solved in many ways. It can be played and finished manually where it is based on the player's capacity and skills or it could also be finished through systematic search solutions. In this paper, three search algorithms will be discussed, observed, and compared. These search strategies are all helpful to find a path to the goal state of the puzzle but here we can identify which one is the best for solving this 8-puzzle game.

Performance

This section will explore and discuss the definitions and differences of Breadth-First Search, Depth-First Search, and A-star Search when used as a path-finding solution for the 8-puzzle game. The definitions of these search algorithms are as follows:

Breadth-First Search (BFS) – an algorithm for traversing or searching tree that is implemented using a **queue** data structure which follows first in first out. Generally requires more memory than DFS and is optimal for finding the shortest distance.

Depth-First Search (DFS) – an algorithm for traversing or searching tree that is implemented using a **stack** data structure which follows first in last out. Generally requires less memory than BFS and is not optimal for finding the shortest distance.

A Star Search (A* Search) – A*search has been used widely in the field of pathfinding and graph traversal. Its main approach involves the use of a heuristic plus an open list (frontier) and a closed list (explored). A* algorithm is used to calculate the shortest distance between the source (initial state) and the destination or goal (final state).

The algorithms above have been tried and analyzed on the two test cases below. They are observed based on their performance considering the number of iterations (explored /closed list size), the space occupied (frontier/open list size + explored/closed list size), and the path cost. Note that the action sequence used in determining the next states is up-right-down-left. We can see the results of the tests below.

Test Case 1

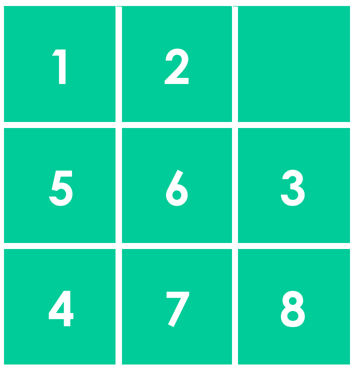

Fig 1. Test Case 1 Initial Puzzle Configuration

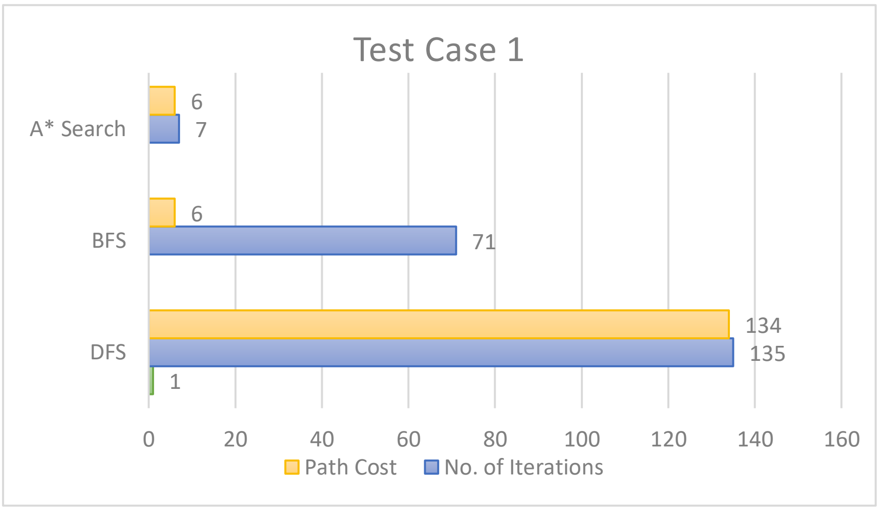

Fig 2. Test Case 1 Results

The results shown that both the BFS and the A* Search of this puzzle configuration have the lowest path cost (6). Meanwhile, using DFS, the resulted path cost is surprisingly high (134). Therefore, in terms of path cost, we can say that the path we found from DFS is not an optimal solution. Thus, both the A* Search and the BFS have found the shortest path to solve this puzzle.

Meanwhile, in terms of iterations or the explored states, A*Search has an incredibly low number of iterations (7) compared to the iterations of BFS (71) and DFS (135). Their differences are significantly great that it can also imply the overall efficiency of the algorithm. The number of iterations also reflects the time spent of the execution of the algorithm. Therefore, we can say that the DFS has the longest time spent and A* Search has the shortest time spent while finding a solution to this puzzle configuration.

Test Case 2

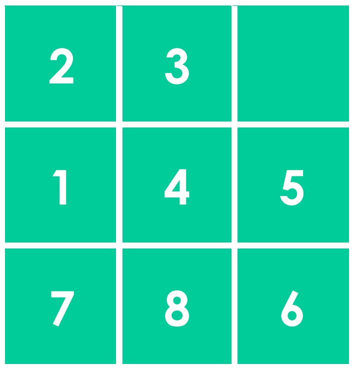

Fig 3. Test Case 2 Initial Puzzle Configuration

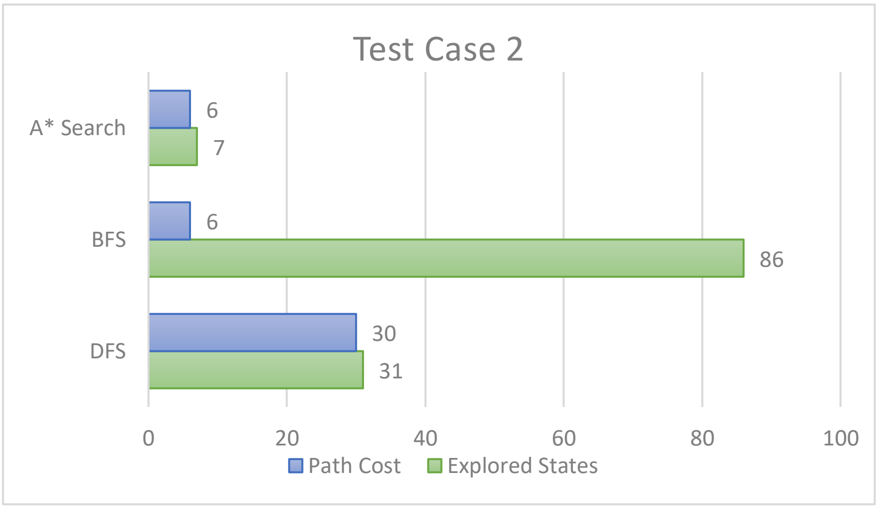

Fig 4. Test Case 2 Results

The results above have shown that both the BFS and the A* Search of this puzzle configuration have the lowest path cost (6) again. Meanwhile, using DFS, the resulted path cost is quite high (30). Therefore, in terms of path cost, we can say that the path we found from DFS is not an optimal solution. Thus, both the A* Search and the BFS have found the shortest path to solve this puzzle.

Furthermore, with regards to the iterations or the explored states, A*Search has an incredibly low number of iterations (7) compared to the iterations of DFS (31) and BFS (86). The number of iterations also reflects the time spent of the execution of the algorithm. Therefore, we can say that the BFS has the longest time spent and A* Search has the shortest time spent while finding a solution to this puzzle configuration.

Informed vs. Uninformed Search Strategies

Informed search strategy is a search strategy that has a way to know the information towards the goal state: it could be a formula to estimate the steps to the goal state or a function that approximates how close a state is to the goal state. One example of informed search strategy is the A* Search because it uses a heuristic value or the estimated cost of a node to the goal node. On the other hand, uninformed search strategy is a search strategy that has no additional information on the goal state except for the problem specification. Its way to the goal state just depends on the actions and length of actions. This search strategy is also called blind search. Examples of uninformed search strategy are BFS and DFS since they traverse blindly and just go on to the next state without considering the path to goal state.

To further compare the two strategies, we have the following basis: search direction, speed, cost, and space occupied.

Search Strategy	**Direction**	**Speed**	**Cost**	**Space**
Informed	has direction	faster solution search	low cost	lesser occupied space
Uninformed	no direction	slower solution search	higher cost	greater occupied space

Based on the comparison above, we can see that the performance and behavior of the informed search strategy is much better than the latter. Thus, making it more efficient to use for path-finding solutions for problems like the 8-puzzle game.

Conclusion and Recommendation

Based on the test cases and comparison of informed and uninformed search strategy, I believe the A*Search is the most appropriate and most efficient search algorithm to use to solve this 8-puzzle game. First, because it is an informed search strategy where we could have the information and direction to the goal state of the problem since we are computing for the estimated steps from a particular state to the goal state. Second, because of its amazingly low path cost. This search algorithm is indeed able to find the optimal solution or the shortest path to solve a puzzle configuration. Though its path cost are sometimes the same with the BF Search's, A* Search on the other hand is still better to use because of its low number of iterations or its shorter time spent. Therefore, I highly recommend the use of A* Search in finding solutions for the 8-puzzle game.

References

8-Puzzle game. (n.d.). 8-Puzzle Game. Retrieved October 5, 2021, from https://8-puzzle.readthedocs.io/en/latest/

Edpresso Team. (n.d.). *DFS vs. BFS*. Educative: Interactive Courses for Software Developers. Retrieved October 5, 2021, from https://www.educative.io/edpresso/dfs-vs-bfs

Edpresso Team. (n.d.). *What is the A* algorithm?* Educative: Interactive Courses for Software Developers. Retrieved October 5, 2021, from https://www.educative.io/edpresso/what-is-the-a-star-algorithm

GeeksforGeeks. (2021, February 26). *Difference between Informed and Uninformed Search in AI*. https://www.geeksforgeeks.org/difference-between-informed-and-uninformed-search-in-ai/

P, N. (2018, November 16). *Difference between Informed and Uninformed Search (with Comparison Chart)*. Tech Differences. https://techdifferences.com/difference-between-informed-and-uninformed-search.html