Prithviraj Yuvaraj

SID: 861258445

Pyuva001@ucr.edu

05/12/2021

Spring 2021

CS205 Project 1: 8 Puzzle

Overview

The project was created in Python 3 and tested using Python 3.8. As reference the Python v3.1.4 documentation: https://docs.python.org/3.1/index.html was used to help with specific data structures. In this case, the PriorityQueue was used to queue nodes and a custom Node object was created that could hold heuristic and cost parameters. Along with the Python documentation an 8-Puzzle solver: https://8puzzlesolver.com/index.php#step3 was used to check accuracy of the algorithms for shortest path. The Uniform Cost Search algorithm pseudo code was referenced from the textbook [1] and was modified to accept the different heuristics of A*. The code for the completed project can be found: https://github.com/pyuvaraj37/CS205_Project_1.

Introduction

To test three different search algorithms, an 8-Puzzle toy was used. "The 8-Puzzle consist of a 3x3 board with eight numbered tiles and a blank space. A tile adjacent to the blank space can slide into the space. The object is to reach a specified goal state." [1]. While there are many ways to represent operators to transform states, the simplest representation is to move the blank space. Fig 1 shows a scrambled 8-Puzzle, and Fig 2 shows the goal state that is trying to be achieved by the algorithms. The operators are moving the blank space up, down, left, and right without

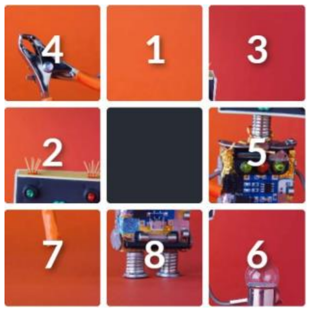

Figure 1: A scrambled 8-Puzzle

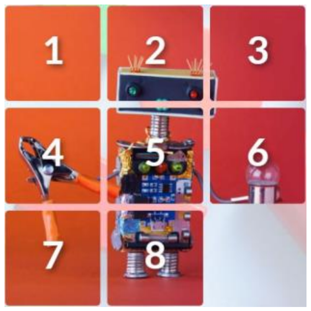

Figure 2: The goal state of the 8-Puzzle

making any illegal moves such as moving the blank out of bounds. The puzzle has about 180,000 different achievable states and has an expansion rate of 4 for each operator.

The three different algorithms that were implemented and tested are Uniform Cost Search (UCS), A* with the Missing Tile (AMT) heuristic, and A* with the Manhattan Distance (AMD) heuristic. In the python implementation the only a general algorithm was created but depending

[1] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach, Chapter 3, Page 70. Pearson. 3rd Edition. 2015.

on which method is specified a different h(n) is calculated. The cost of an action in this case is 1, so g(n) increases by 1 with each depth of the solution.

UCS is identical to A* but the only difference is that h(n) = 0, and only g(n) is used to establish state priorities. Another insight in the algorithm is that it is a glorified breadth-first search in that it tests all the nodes in a certain depth before moving onto the next one. AMT has a non-zero heuristic as well as the same g(n) as the other algorithms. The heuristic is calculated by counting the number of tiles which are misplaced, not including the blank space. For example, in Figure 1, the scrambled 8-Puzzle would have a heuristic value of 5. The 1, 2, 4, 5 and 6 tiles are in the incorrect spots. AMD also uses the miss placed tiles but sums their distance from their goal state as a heuristic, so Figure 1 would have a value of 6.

Example Execution

The following section is to show how the program runs, the blank text is output from the program and the red text is input of the user.

Below is an example of the test puzzle execution:

```
8-Puzzle

Are you going to be using a test puzzle? (1 for Yes, 0 for No): 1

Used for debugging

Choose the depth of the test puzzles: (1, 2, 4, 8, 16, 22): 4

0 1 2

4 5 3

7 8 6

Which method to solve the 8-Puzzle will you use? (1 for UCS, 2 for A* MT, 3 for A* MD): 1

Using UCS to solve the 8-Puzzle

Solution found!

Moves needed/Depth: 4

Nodes Expanded: 67
```

Below is an example of a custom inputted puzzle:

```
8-Puzzle

Are you going to be using a test puzzle? (1 for Yes, 0 for No): 0

Input custom 8-PUzzle

Enter each elements for a row, row by row with a space between the numbers. 0 for missing tile.

Input the first row: 0 1 2

Input the first row: 4 5 3
```

[1] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach, Chapter 3, Page 70. Pearson. 3rd Edition. 2015.

```
Input the first row: 7 8 6

0 1 2

4 5 3

7 8 6

Which method to solve the 8-Puzzle will you use? (1 for UCS, 2 for A* MT, 3 for A* MD): 1

Using UCS to solve the 8-Puzzle

Solution found!

Moves needed/Depth: 4

Nodes Expanded: 67
```

Analysis

For testing the different algorithms six different puzzles where used that have incrementally larger solutions depths. UCS can solve any depth of problem, but it takes a significant amount of time. For example, the execution time for the puzzle with a depth of 16 was about 10 minutes for UCS. With that in mind, a depth of 22 was not tried due to the algorithm needing to run for 30 minutes or longer. This is due to the h(n) heuristic being set to 0 so the algorithm learns no insight about the current state and is just blindly searching through the decision tree.

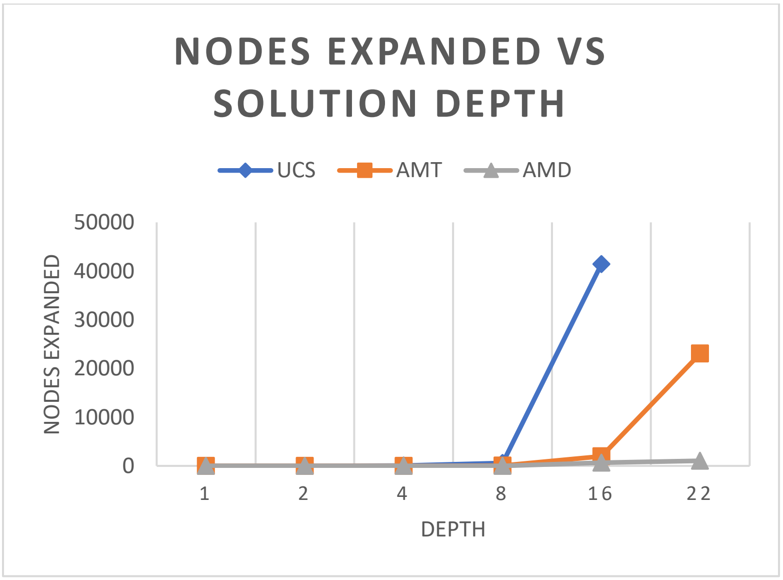

Figure 3: Nodes Expanded vs Depth chart. All the algorithms are similar while depth is less than 8.

Now comparing both the heuristic search which are both generally similar with a depth of 16 or less. The expansion of nodes greatly differs when a depth of 22 is used. The Miss Placed Tile heuristic has the local minimum problem where it may believe a goal state is being close to obtaining just because h(n) is getting smaller. Generally, when h(n) gets closer to zero a goal state is close, but its not guaranteed. While the Manhattan Distance can also suffer from this

[1] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach, Chapter 3, Page 70. Pearson. 3rd Edition. 2015.

issue, it is less likely than the Missing Tile since the heuristic evaluation function is more elaborate and considers more factors.

Conclusion

After comparing all three algorithms in terms of efficiency the Manhattan Distance heuristic gives the best results. With Uniform Cost Searches having the h(n) = 0, makes it like Bread-First Search a proper comparison to other algorithms would be other Blind Searches such as Depth-First or Iterative Deepening. The results are predicable with the Heuristic Searches running faster and using less space than the Blind search, but when comparing the Missing Tile and Manhattan Distance heuristic the results were surprising. Some further experimentation that can occur is with a 15-Puzzle to see if the gap between Missing Tile and Manhattan Distance widens.

[1] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach, Chapter 3, Page 70. Pearson. 3rd Edition. 2015.