Ashley McDaniel
5/5/2020

CS 170: Project 1 Eight-Puzzle Solver Report

Introduction

The program uses different variations of the A* algorithm to solve an eight-puzzle. The A* algorithm is the fastest search algorithm for any kind of given heuristic as it must expand a minimum of A* nodes. The different types of variations are uniform cost search, A* with the Misplaced Tile heuristic, and A* with Euclidean Distance heuristic. The program allows the eight-puzzle to be solves with any of the three options. The user can input their own puzzle if they choose to, otherwise there is a default puzzle that can be solved. The program will display the expanding state with its nodes and show the values of g(n) and h(n) for each state the algorithm finds. When the goal state is found then the program will tell the user “Goal !!!” and state how many nodes were expanded as well as the maximum number of nodes in the queue at a time before terminating.

Uniform Cost Search

In uniform cost search the h(n) will be set to zero which will allow for A* to choose the node that costs the least for g(n). So essentially the algorithm simply expands the node that costs the least and in this search each expanded node will cost one, making each node have a uniform cost.

A* with the Misplaced Tile heuristic

The A* with misplaced tile heuristic functions by storing the number of misplaced tiles in the problem and let h(n) be equal to the sum of misplaced tiles. Then the algorithm works by iterating through each node stored in the queue and when it comes across the node with the least h(n) plus g(n) value, then the algorithm will choose that node to expand. By storing the number of tiles that become misplaced the misplaced tile algorithm will have to save a large amount of data in memory making the memory cost for this algorithm quite great. Despite this, the misplaced tile algorithm is able to find a quicker path to solving the puzzle when compared with the uniform cost search algorithm.

A* with the Euclidean Distance heuristic

The A* with Euclidean distance heuristic uses the Euclidean distance formula to find the distance between the node and the goal state. The Euclidean formula specifically, is able to find the diagonal distance which can make the overall path cost cheap. The Euclidean distance heuristic works by counting the number of misplaced tiles and then using the Euclidean formula to see which node costs the least. The algorithm finds the lowest costing path by sorting through each of the nodes and expanding the cheapest nodes until the goal state is reached.

Challenges

Ashley McDaniel
5/5/2020

I had different challenges with each algorithm. With the A* Misplaced Tile heuristic I forgot to skip the blank space when counting the number of tiles misplaced which sometimes led to some errors. It is important to skip the blank space when iterating through the puzzle in this algorithm. Another problem I had was implementing the Euclidean formula $\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$ because the square root function will lower the runtime of the Euclidean distance heuristic algorithm which does not make sense. However, when leaving out the square root it performs much better. Also, I needed to include the math.h library in order to access functions such as pow and sqrt. Overall, these challenges in addition to the many bugs I found in my program made implementing these algorithms a challenge.

Data

Puzzles Used:

Trivial:
123
456
780

Very Easy:
123
405
786

Easy:
120
453
786

Hard:
412
503
786

Very Hard:
123
450
678

Impossible:
123
456
870

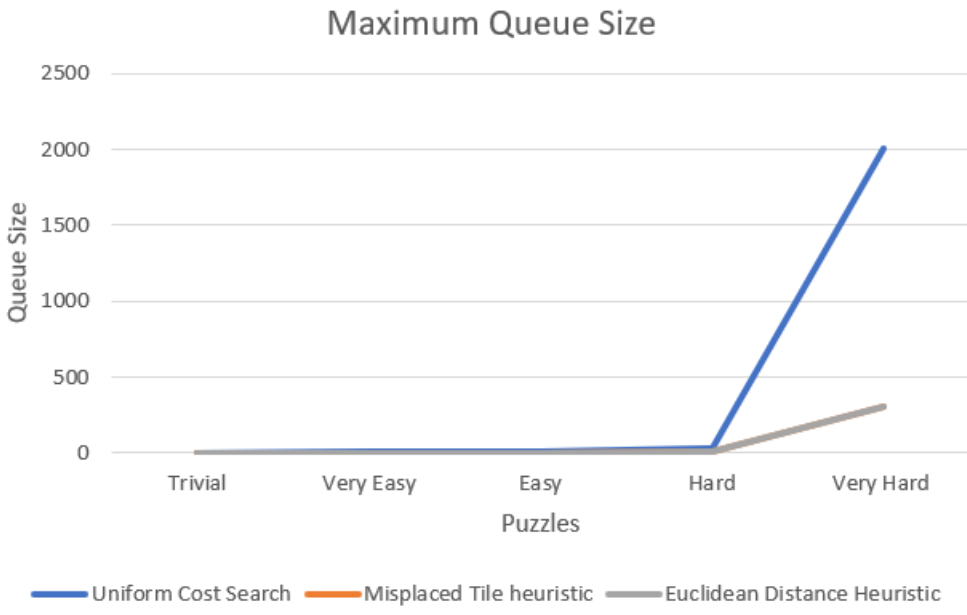
Ashley McDaniel
5/5/2020

Maximum Queue Size:

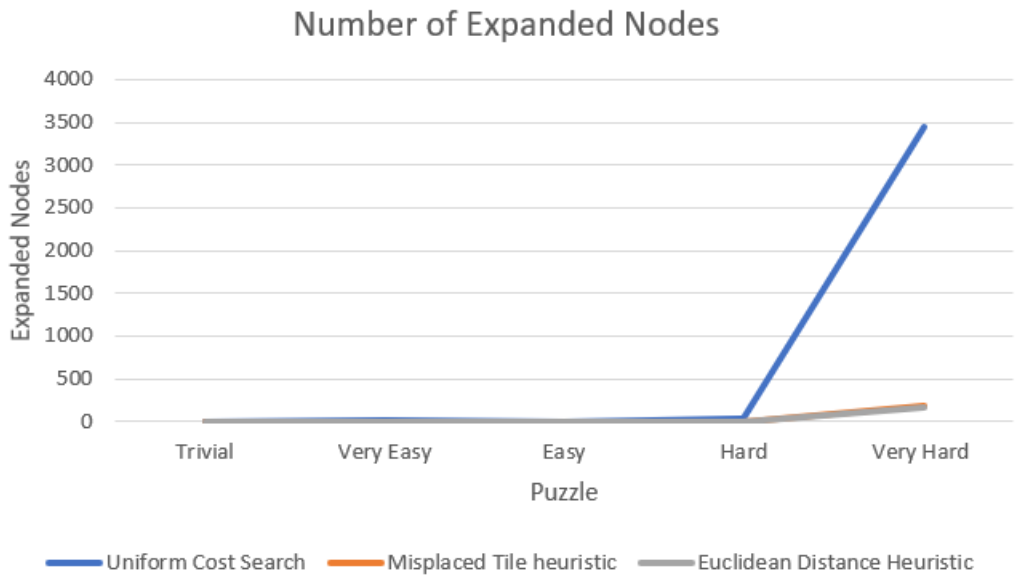
Puzzle	Uniform Cost Search	Misplaced Tile heuristic	Euclidean Distance Heuristic
Trivial	0	0	0
Very Easy	7	2	2
Easy	5	2	2
Hard	35	5	5
Very Hard	2008	302	303

Number of Expanded Nodes:

Puzzle	Uniform Cost Search	Misplaced Tile heuristic	Euclidean Distance Heuristic
Trivial	0	0	0
Very Easy	8	4	4
Easy	4	2	2
Hard	28	4	4
Very Hard	3436	190	158



Ashley McDaniel
5/5/2020



Conclusion

Overall, when looking at the graphs titled maximum queue size, and number of expanded nodes it is clear to see that the uniform cost search performs exponentially worse once the puzzle reaches the difficulty level of hard and very hard. The uniform cost search, misplaced tile heuristic, and Euclidean distance heuristic perform the same or almost the same for the puzzles with the level of difficulty: trivial, very easy, and easy. This is because the number of expanded nodes and the maximum queue size is about the same for all three algorithms until the puzzle level of difficulty reaches hard.

Once the level of hard is reached then the uniform cost search uses much more memory to store the max queue size and the number of expanded nodes. Thus, that means the misplaced tile heuristic and Euclidean distance heuristic performed much better than the uniform cost search. They perform better because they consider h(n) and g(n) when finding the best path.

In the puzzle, each node is representative of a number tile that can be moved around the puzzle using operators. Moreover, these two algorithms have an advantage because they are taking into consideration the cost from the current node to get to the

Ashley McDaniel
5/5/2020

goal state which is h(n), and the total sum from the starting node to the current node which is g(n).

In conclusion, it is of great importance to use an efficient heuristic such as the misplaced tile heuristic or the Euclidean distance heuristic because they are faster at finding the solution to difficult puzzles. There did not seem to be a noticeable difference in the performance between the misplaced tile heuristic and the Euclidean distance heuristic so either are a good choice for hard or very hard puzzles in this program.

Ashley McDaniel
5/5/2020

Traceback:

Uniform Cost:

```
Program Beginning

Hello! Welcome to 861232971 eight-puzzle solver.
Type 1 to use a default puzzle, or 2 to enter your own puzzle.
1

103
426
758
Enter your choice of algorithm
[1]: Uniform Cost Search
[2]: A* with the Misplaced Tile Heuristic
[3]: A* with the Euclidean Distance Heuristic

1
expanding state
103
426
758

The best state to expand with g(n) = 1 and h(n) = 0 is...
123
406
758
expanding this node...

The best state to expand with g(n) = 2 and h(n) = 0 is...
123
456
708
expanding this node...


 Goal !!!

The maximum number of nodes in the queue at any one time: 15.
To solve this problem the search algorithm expanded a total of 10 nodes

Program Ending
```

Ashley McDaniel
5/5/2020

A* with Misplaced Tile Heuristic:

```
Hello! Welcome to 861232971 eight-puzzle solver.
Type 1 to use a default puzzle, or 2 to enter your own puzzle.
1

103
426
758
Enter your choice of algorithm
[1]: Uniform Cost Search
[2]: A* with the Misplaced Tile Heuristic
[3]: A* with the Euclidean Distance Heuristic

2
Using Misplaced Tile Heuristic

expanding state
103
426
758

The best state to expand with g(n) = 1 and h(n) = 2 is...
123
406
758
expanding this node...

The best state to expand with g(n) = 2 and h(n) = 1 is...
123
456
708
expanding this node...


 Goal !!!

The maximum number of nodes in the queue at any one time: 3.
To solve this problem the search algorithm expanded a total of 5 nodes

Program Ending
```

Ashley McDaniel
5/5/2020

Euclidean Distance Heuristic:

```
Hello! Welcome to 861232971 eight-puzzle solver.
Type 1 to use a default puzzle, or 2 to enter your own puzzle.
1

103
426
758
Enter your choice of algorithm
[1]: Uniform Cost Search
[2]: A* with the Misplaced Tile Heuristic
[3]: A* with the Euclidean Distance Heuristic

3
Using Euclidean Distance Heuristic

expanding state
103
426
758

The best state to expand with g(n) = 1 and h(n) = 2 is...
123
406
758
expanding this node...

The best state to expand with g(n) = 2 and h(n) = 1 is...
123
456
708
expanding this node...


 Goal !!!

The maximum number of nodes in the queue at any one time: 3.
To solve this problem the search algorithm expanded a total of 5 nodes

Program Ending
```

Ashley McDaniel
5/5/2020

Sources Cited

1. Lecture 2 Slides-Blind Search
2. Lecture 3 Slides-Heuristic Search
3. https://www.geeksforgeeks.org/uniform-cost-search-dijkstra-for-large-graphs/
4. https://www.geeksforgeeks.org/a-search-algorithm/
5. https://www.geeksforgeeks.org/check-instance-8-puzzle-solvable/
6. https://faramira.com/solving-the-8-puzzle-problem-using-a-star-algorithm/
7. https://www.geeksforgeeks.org/euclidean-algorithms-basic-and-extended/
8. https://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html