Assignment 2
CSC520: Artificial Intelligence

Harshit Patel
(200314248)

September 25, 2019

1 Heuristic

Define a novel heuristic function for this puzzle. Include an explanation of this heuristic in your report along with a justification for why it will guarantee an optimal solution with A* Search. Use this heuristic in your implementation below.
Ans.

For this puzzle, the heuristic function that I have used is as follows:
$h = min(dist(tile_i, globejunctions) + dist(globejunctions, tile_j))$
where, $tile_i$ is tiles current position, and $tile_j$ is tiles goal position.

This is very accurate to how we would calculate to real distance between a tile and its goal position. If a tile is at position x and it has to go to a position y, then in most cases it would have to pass through one of the six junction points of the globe (Globe Junction: (0, 0), (180, 180), (90, 90), (90, 0), (90, 270), (90, 180). In certain cases it does not pass through them, for example, if a tile have to move from (90, 30) to (90, 60). At such time it may overestimate the cost. In order to compensate for that, I am taking an average of h for all tiles. Finally I use floor function to make it an integer less than what it would be if ceiling function was used.

Thus, this function is a very good heuristic for A* Search. It is very conservative but in certain cases it overestimates, which is compensated by average. Therefore, it is an admissible heuristic. Hence, it would guarantee an optimal solution.

Puzzles	Breadth First Search	A* Search
Puzzle2-0	Explored states: 15218, Max queue size: 58137	Explored states: 3592, Max queue size: 16049
Puzzle2-1	Explored states: 22945, Max queue size: 87684	Explored states: 5346, Max queue size: 23465
Puzzle2-2	Explored states: 23259, Max queue size: 88126	Explored states: 6476, Max queue size: 25693
Puzzle2-3	Explored state: 27653 Max queue size: 125697	Explored states: 10068, Max queue size: 32496

Tabell 1: Number of Explored States for each of the three algorithm (The algorithm are slow and hence it was taking very long to output. Hence, I have put number of states expanded and maximum queue size)

2 Implementation

The code files are provided in the folder.

3 Analysis

Using the code above calculate solution results for each of your algorithms on the Puzzle files. Report the minimum, average, and maximum values for the number of states expanded and the queue size for each algorithm and identify the hardest puzzle for each. Consider the relative performance of the algorithms and state which algorithm is best for this problem. Justify your answer.

Ans. Due to certain inefficient data structures that I used, my algorithms were not able to solve any of the puzzle in time. After hours, operation on puzzle0 both Breadth First Search and A* Search exceeded memory. For Recursive Best First Search, I am getting an error that childnodeconstructor is not an iterable object. Which, I think is due to the recursion. When the algorithm finally computes the result, I get it right if I see it in debug mode. However, when I run it, it is throwing the error state above. As, I did not had time to rewrite the whole code again, I just submitted what I had for it.

The results of Breadth First Search and A* Search algorithms when ran on puzzle's 2-0 to 2-3 for about 45 minutes are given in Tabell 1.

From the information provided in data we can easily calculate minimum , maximum and average states of expanded, and size of queue. All of this information is as given below:

1. Breadth First Search:
   $average(explorednodes) = 22268.75$
   $average(Maxqueuesize) = 89911$
   $max(explorednodes) = 27653$
   $max(Maxqueuesize) = 125697$
   $min(explorednodes) = 15218$
   $min(Maxqueuesize) = 58137$

2. A* Search:
   $average(explorednodes) = 6370.5$
   $average(Maxqueuesize) = 24425.75$
   $max(explorednodes) = 10068$
   $max(Maxqueuesize) = 32496$
   $min(explorednodes) = 3592$
   $min(Maxqueuesize) = 16049$

From the above given data, it is easy to conclude that A* is multifolds better over BFS if given a very good admissible heuristic. The hardest and easiest puzzle for both the algorithms were the same, that is puzzle2-3 and puzzle2-0 respectively. Though, we can see that A* got to the goal state in using a very optimal path. My heuristic, even though hypothetically is admissible, due to inefficient or slightly incorrect implementation, it overestimates at times. Still, for most puzzle it is optimal (I have not solved all of them, but I am concluding from this from the data for given four puzzles.