Comparisons of Heuristic Search Algorithms

Christopher Mendez, Rupika Dikkala

ABSTRACT

There are many different ways to leverage a search algorithm to solve a 4x4 puzzle board. In this assignment, we implemented two memory-bounded heuristic search algorithms, namely IDA* and RBFS, to find the Manhattan distance and Misplaced Tile heuristics across the board. We compare and contrast the algorithm performances with each other, as well as to themselves across the functions. To further study how the search performance varies with problem difficulty, we also generated random problems by perturbing the goal state with a random sequence of moves. To conclude the paper, we explore methods to improve the search capabilities and heuristic evaluation function.

INTRODUCTION

Artificial Intelligence systems commonly leverage search algorithms to navigate from the initial state to the goal state in order to achieve a solution [1]. When comparing the search capacities of different kinds of AI, it makes sense to vary the algorithms and heuristics. In this paper, we compare the performance of two memory-bounded search algorithms across two different heuristics to better understand the implications of different algorithms on heuristics. The IDA* algorithm was effective at solving the puzzle for both heuristics, but overall took longer to solve. With this particular algorithm, the Misplaced Tile heuristics took longer to reach a solution - it expanded more nodes and resulted in a longer solution length than the Manhattan distance heuristic. The RBFS algorithm was an effective puzzle solver when there was a lower *m* amount of shuffling, but misplaced more tiles as *m* was increased. The recursive nature of this algorithm resulted in less time taken than the A* approach; there were no noticeable performance differences between the two heuristics.

DESCRIPTION OF ALGORITHMS

The RBFS Algorithm for each point in time selected the node with the best heuristic value. It f-limit was defined locally, as a function of the heuristic value + distance from the start. The idea is that as distance from start increases the heuristic value should be decreasing and it is not worth pursuing high cost heuristics early on or diving too far into the tree. These f-limits were local and once reached the function recursed out.

By comparison the IDA* algorithm used both heuristic and distance to make its decision. Its f-limit is global and when reached the function restarts if it fails. This is what makes it different than a normal A* function. Of note,while this approach is a way to save some space it doesn't always save time so we do also cut of either our RBFS and IDA* if they run for too long, meaning they won't always find solutions. In these cases we record how many misplaced tiles are left (any amount greater than 0 means there wasn't a total solution).

EXPERIMENTAL SETUP

Our experiment consisted of testing the different algorithms and heuristic performance across randomized puzzle instances with various degrees of shuffling. To randomize the puzzle instances, we wrote a scrambler function that performed a set of *m* random moves on the "solved" puzzle. By randomizing the baseline "solved" puzzle, we are guaranteed that all our randomized puzzles are solvable and by varying *m* we can control how scrambled the puzzles can get. We ran 50 trials on each algorithm and each heuristic combination (200 trials total). For each of the 50 trials, 10 trials were run on the 5 different values of *m* move sequences (10, 20, 30, 40, 50).

RESULTS

In the graphs below, note the heuristics are coded as follows - **H1: Misplaced Tiles, H2: Manhattan Distance.**

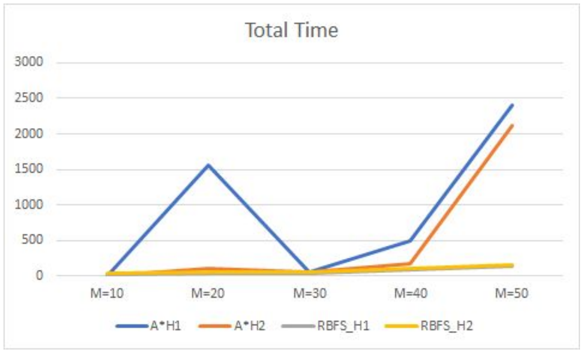

Figure 1. Total time (ms) spent on each algorithm and heuristic combination.

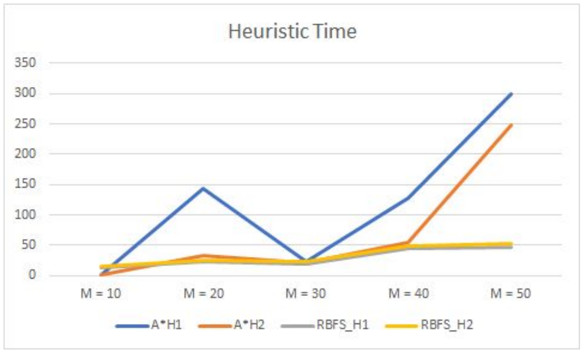

Figure 2. Time (ms) spent evaluating the heuristics for the A* and RBFS algorithms.

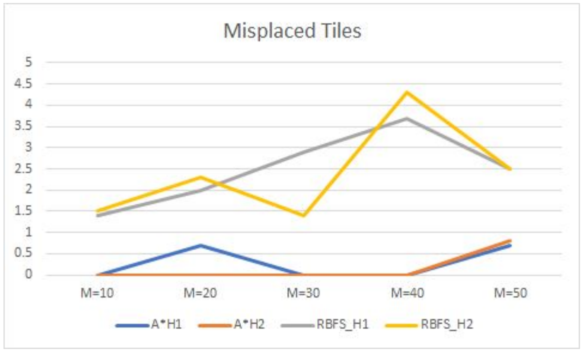

Figure 3. Number of Misplaced Tiles (meaning that the tiles weren't in the current position) across the board.

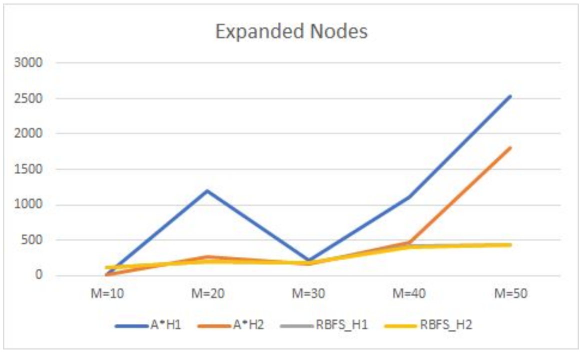

Figure 4. Number of expanded nodes while performing the search.

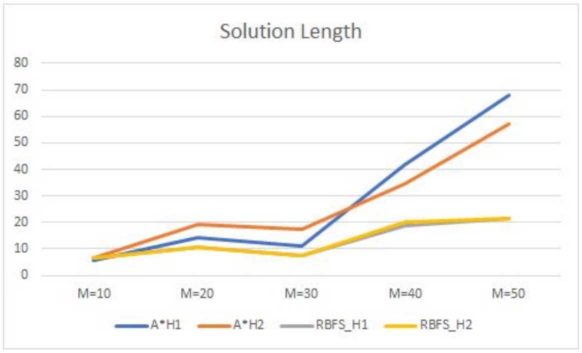

Figure 5. The number of moves it took the algorithm to reach the solution (solved puzzle).

Performance of IDA* Algorithm

The IDA* algorithm both were effective at solving the puzzle, averaging less than 1 misplaced tile per trial across all 50 trials (Figure 3). This means most of the time, they were able to solve the puzzle before we timed them out. As a tradeoff to this, the amount of time these algorithms took was longer than the RBFS algorithms.

Comparing the Misplaced Tiles Heuristic to the Manhattan Distance it is clear the Misplaced Tiles Heuristic took longer to calculate (Figure 2) and it led to more expanded nodes (Figure 4) and a longer solution length (Figure 5). This all leads to a longer overall time without better performance with regards to the final solution.

Performance of RBFS Algorithm

The RBFS algorithm was very effective when M was lower but suffered as M increased. For M of 10, 20 and 30 the RBFS algorithm performed as well or better as the A* algorithm by quite a bit on total time, time on heuristic, expanded nodes and solution length. Its main drawback was failure to completely solve many of the problems, averaging 2-3 misplaced tiles per trial for the lower M's and climbing up to around 4 misplaced tiles per trial in the upper M's.

This is likely due to how the algorithm was implemented. By having a local f limit, there will be a point where each function stops recursion and solving the problem. Originally the performance of the RBFS was very bad so we tripled the f limit and while that improved it, it still failed to solve many of the problems but did complete rather quickly. This seems to be the main feature of the algorithm, much less memory and CPU time used but it comes at a cost of not running as long so it might not get all of the way to a solution.

The difference in Manhattan Distance and Misplaced Tiles Heuristic was not that pronounced, and in many cases on in the figures they are overlapping. It seems as though while there were some small differences in the heuristics it didn't matter as much as the overall algorithm.

DISCUSSION

While there wasn't much of a conspicuous difference between the Manhattan distance and the Misplaced Tile heuristics for the RBFS algorithm, in the A* algorithm we observed that the Misplaced Tile heuristic had more performance bottlenecks compared to the MD. It took more time to reach a solution, and along the way expanded more nodes thus resulting in a longer solution length. Therefore, the Manhattan Distance is the more preferred heuristic.

A small sacrifice to optimality can surely lead to a reduction in the number of nodes expanded alongside optimizing CPU time as showcased in the RBFS algorithm. There just is definitely a fine line as with the wrong shuffle or with a slightly lower f-limit we would expect to see a very sharp drop in optimality.

Time spent on heuristic evaluation was between 10% and 15% of the total time spent (Figure 1, Figure 2) evaluating the function across the A* and RBFS algorithms and Manhattan Distance/Misplaced Tile heuristic combinations - therefore not a significant time was spent on the evaluation itself.

For the heuristic evaluation function we considered part of the lecture slides on search which mention misplaced tiles as something which could be quantified. From there, it made sense to see if an algorithm that followed a heuristic to move towards states with fewer misplaced tiles would be a good algorithm. It wasn't really, as in most, if not all cases manhattan distance performed better.

There was a linear relationship between the amount of search and CPU time of an algorithm. Meaning when more nodes were searched, more CPU time was used proportionally. From the results, it is clear that searching fewer nodes would save CPU time but also be less likely to find an optimal solution. If you favor CPU time, then the RBFS was a superior algorithm but if you favor finding the optimal solution, the IDA* was the superior algorithm.

REFERENCES

[1] Russell, S. J., & Norvig, P. (2009). *Artificial Intelligence: a modern approach*.