Marissa Ginel

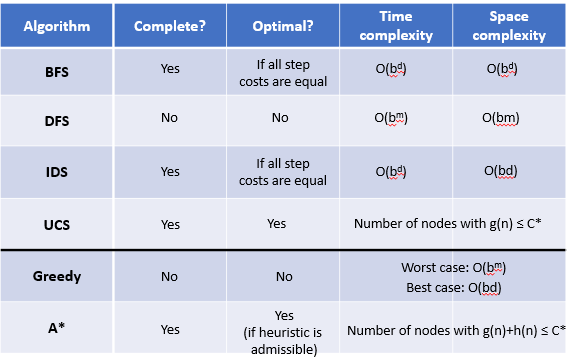
A\* Puzzle

Dr. Burns

Disclaimer: I could not get my full code to execute, so I decided to do a little bit of reading and share what I learned.

A\* is an informed search, which means that the search always knows how far away it is from the goal. Every time A\* makes a move it will have take the move that costs the least. We predict this by using the evaluation function which is: f(n) = g(n) + h(n), where g(n) is the path cost (cost so far), and h(n) is the estimated cost from n to the goal (heuristic). Since your path cost is based off your moves so far you want to make sure that your heuristic is quick and admissible, you are always getting the cheapest move to add to your path cost. Admissible heuristics never overestimate the cost if anything they will underestimate. If h(n) is admissible, then A\* is optimal in the tree search. A heuristic also needs to be consistent where if for every node n, every successor n' of n generated by any action a, h(n) ≤ c(n,a,n') + h(n').Consistency implies admissibility, which naturally comes from admissible heuristics.

For the 8-puzzle I used misplaced tiles (*h*1(*n*)*)* and total Manhattan distance (*h*2(*n*)*:* number of squares away from goal location. If 8-puzzzle if a relaxed problem where a tile can move anywhere *h*1(*n*) gives the shortest solution. If the rules are relaxed, so a tile can move to any adjacent square then *h*2(*n*) gives the shortest solution. Comparatively, *h*1(*n*) will always explore more nodes because A\* search expands every node with f(n) < C\*, were C\* is the actual path cost of the goal node selected.



As you can see the A\* search is complete and optimal as long as the heuristic is admissible. My prediction would be that the Manhattan Distance heuristic will be better than the misplaced tiles. The lowest the Manhattan distance could possibly be is equal to the misplaced tiles. While they both underestimate the true distance, meaning they are admissible, the Manhattan distance will take less time, since the tile can only move to an adjacent tile there will be less nodes to explore and there fore will take less time and space, while still being admissible and consistent.

Unlike what we previously discussed a local search does care for the path or sequence of actions. So, rather than having a search tree that exponentially grows, we have the initial state which we will slightly modify each time and examine and repeat. In this case we just need to find the solution, not hot to get it. This has search algorithms that are more applicable than search agents such as A\* and BFS. This is because local search agents make the path irrelevant. The good part to the is that it has a constant time and space complexity. My prediction would be that the local search, random walk in the 8-puzzle would have taken a very long time to complete and probably would not even find the goal. This is because a random walk is moving a from the current state to a random neighbor. This is extremely inefficient, because rather than checking and knowing where to move you could face the possibility of exhausting every possible move before finding the correct move last. I do not believe this would have worked for the 8 puzzles consistently, or at all.

The hill climbing search probably would have found a solution here and there, but it does not seem like it would be a likely occurrence. Since hill climbing search never make “downhill” moves towards state with a lower value, it is guaranteed to be incomplete because of the local maxima. This could possibly get stuck in a loop or reach a destination and have no where else to go. This seems just as, if not more inefficient than the random walk. I predict this would have been very difficult to find a solution for the 8 -puzzle but could have passed very rarely.