HW1 Wet in 046203 Planning and Reinforcement Learning  
Submitter 1: David Valensi 342439643  
Submitter 2: Yaniv Galron 206765646

Solving 8-Puzzle with Dijkstra’s Algorithm

1. We have 9 numbers that can be placed in each of the 9 possible locations on the grid. Therefore, the number of possible states is . In fact, the number of reachable states is half of that number and equals . This number is too big to run minimization of all the states in each iteration. Instead, we will not save all the states but only the nodes we reach to and its neighbors (that are not already in the graph)
2. CODE

Solving 8-Puzzle with A\* Algorithm

1. The admissible heuristic we will use for the 8-Puzzle will be

when we sum over all tiles x,y current position vs x,y end position. This heuristic is admissible ( because at each action only one of the digits is moved by a distance of ‘1’ to the x or y direction. The shortest path possible is If we could move each digit directly to the end position and we would get

1. CODE
2. This heuristic is admissible also and we can know that from the fact that if a tile is not in the correct place than the value of this function will be 1 and the value of the admissible heuristic we used before will be greater of equal to 1 so

With this “num – incorrect” heuristic the algorithm visited 1816 states, compared to 218 with the MD heuristic. The MD visits less states and takes 9 times less time. We think this is because that the MD heuristic is more bounded and closer to the real cost of the states.

1. The following eight-puzzle instance take 27 moves to solve:

Dijkstra algorithm took: 25.78 seconds to complete and it visited 176184 states  
 A\* algorithm took 0.37 seconds and it visited 2194 states

1. Let’s compare the result on the given puzzle.

For

For

Same number of visited states but longer time (3.59 seconds)

For

We can see that as we increase alpha the number of visited states and the running time is decreasing but if we make alpha too large the heuristic becomes non-admissible, and we got a non-optimal solution