Samanyu Satheesh

Dr. Tang

CS 4200

24 July 2023

Project 1 Report

In this project, we were tasked with creating a program that could use the A* algorithm to solve the 8-Puzzle game, in which a player has to arrange 8 tiles in order in a 3x3 grid. Since A* contains a heuristic function, we were also assigned to test two different heuristic functions, and study their performance in relation to each other. The first heuristic function is known as Hamming, and is calculated by the number of tiles in incorrect positions. The second heuristic is known as the Manhattan heuristic, and is calculated by adding up the distances from each tile to its correct position, horizontally and vertically.

After programming and testing these two heuristic functions, I found that the Manhattan heuristic function outperformed the Hamming heuristic function, especially as the complexity (depth) of the problem grew. I believe this is because it is simply more knowledgeable about the state of the board. While the Hamming approach only knows how many wrong tiles there are, the Manhattan approach is actually aware of the distances that need to be covered in order to progress the board into the goal state.

When it comes to the experience gained by doing this project, it was a good refresher to programming in Java, and it was my first time actually coding an AI. Being able to program code that solves a problem feels like an accomplishment, and I learned a lot about how these programming approaches work in the context of AI and algorithms.

Below is a table detailing my findings.

I ran 100+ randomized test cases and recorded the findings here. As you can see, the two heuristic functions are basically the same at VERY small depths, but there is a enormous difference in nodes generated and runtime as the problem gets more in-depth and complex

		H1		H2	
Depth	# of Cases	Avg Runtime (ms)	Avg Nodes Generated	Avg Runtime (ms)	Avg Nodes Generated
2	1	9	5	6	5
4	1	13	10	10	10
8	1	20	24	13	22
14	1	68	539	20	114
15	1	16	927	17	233
16	3	20	1691	21	391
17	2	40	2797	50	367
18	5	37	3648	55	596
19	5	33	5979	44	787
20	11	53	9454	36	1013
21	6	44	14353	38	1347
22	11	64	22992	49	2129
23	13	76	37659	42	2413
24	18	86	50142	43	4289
25	13	137	90073	53	6068
26	6	150	98771	57	8987
27	1	310	176110	56	21563
28	3	358	235069	83	11771

29	1	311	298449	59	12735