Akash Kumar

Dr. Burns

CSC 575-01

2/23/2021

A* 8-Puzzle Runtime Analysis

Setup:
To output statistics on the time required to solve the 8 tile sliding-block puzzle, I used Python's *time* module. Specifically, I used the *time*() function in that module to measure the start and end times. The start time is recorded immediately before the function call, while the end time is recorded immediately after the call returns. Subtracting the start time from the end time will give us the time elapsed for the function call. This time is measured in seconds. To convert to milliseconds (ms), I multiplied the time by 10^3.

Here are the results from running the A*search 10,000 times on puzzle 1 and averaging the results:
A* search:

puzzle1_start = [[1, 2, 5], [4, 8, 7], [3, 6, ' ']]
puzzle_goal = [[' ', 1, 2], [3, 4, 5], [6, 7, 8]]

Time taken to find the solution using the misplaced tiles heuristic: 4.098 ms.
Number of nodes explored: 28

Time taken to find the solution using the Manhattan distance heuristic: 4.418 ms.
Number of nodes explored: 14

Here are the results from running the A* search 100 times on puzzle 3 and averaging the results:
A* search:

puzzle3_start = [[1, 3, 2], [4, 6, 5], [' ', 7, 8]]
puzzle_goal = [[' ', 1, 2], [3, 4, 5], [6, 7, 8]]

Time taken to find the solution using the misplaced tiles heuristic: 98.725 ms.
Number of nodes explored: 691

Time taken to find the solution using the Manhattan distance heuristic: 74.844 ms.
Number of nodes explored: 409

Let h_1 be the misplaced tiles heuristic.
Let h_2 be the Manhattan distance heuristic.
From the results shown above, we see that the A* search based on h_1 expanded more nodes than h_2. Since $h_2(n) >= h_1(n)$ for all n, h_2 **dominates** h_1. Domination translates directly into efficiency. Hence, using the h_2 heuristic is faster than the h_1 heuristic.

Here are the results from running the random walk search 10,000 times on puzzle 1 and averaging the results:
Random walk:
Number of times the agent finds a solution: 52
Percent of times the random agent finds a solution: 0.01%.
Average time taken to find a solution: 0.625 ms.

Here are the results from running the random walk search 10,000 times on puzzle 3 and averaging the results:
Random walk:
Number of times the agent finds a solution: 3
Percent of times the random agent finds a solution: 0.00%.
Average time taken to find a solution: 0.664 ms.

Based on above results for the random walk agent, we can conclude that the random agent finds a solution 0 – 0.01% of the time. On average, it took less than 0.7 ms to find a solution.

Here are the results from running hill climbing search on puzzle 1 using the misplaced tiles heuristic:
Hill climbing: (gradient descent)
No solution found!
Solution path: []

Here are the results from running hill climbing search on puzzle 1 using the Manhattan distance heuristic:
Hill climbing: (gradient descent)
Solution found!
Time taken to find the solution using the Manhattan distance heuristic: 5.000 ms.
Solution path: ['N', 'W', 'W', 'S', 'E', 'E', 'N', 'N', 'W', 'W']

Here are the results from running hill climbing search on puzzle 3 using both heuristics:
Hill climbing: (gradient descent)
No solution found!
Solution path: []

The hill climbing search found a solution for puzzle 1 using the Manhattan distance heuristic, but the misplaced tiles heuristic did not work. While the search found a solution for puzzle 1, for puzzle 3, no solution was found using either heuristic. This may be because the agent became stuck at a local maximum, and so was unable to find the path to the solution.

Comparing the runtime for A*, random walk, and hill climbing searches, random walk is the fastest, hill climbing is the slowest, and A* is somewhere in between. In terms of reliability, A* is more reliable (since it is complete) than random walk and hill climbing.