Andrew Davison

CSC 372: Survey of Artificial Intelligence

Dr. Thomas Allen

3/7/2023

**A2: IDA\* on the 2x2x2 Rubik’s Cube**

Approaching the Search Problem

Overview of IDA\*

To approach the problem of solving a 2x2x2 Rubik’s Cube, I used a version of IDA\*, or Iterative-Deeping-A\*, is a method of informed search that looks for a solution by generating possible moves to take from a scrambled cube and then selecting which action to take based off of a heuristic and path cost. This is done until all possible actions are examined up to a certain depth; if a solution is not found to that depth, the depth is increased or the program is terminated. The program is usually only terminated if the time elapsed in the search becomes increasingly long.

In my implementation, I implemented IDA\* by starting with a scrambled cube. The first depth is set to the heuristic evaluation of that cube, and then iteratively deepened until a solution is found. The program takes the scrambled cube, applies all 12 possible moves to the cube excepting ones that would immediately reverse the action taken prior when the search gets deeper, and then evaluates all of the states according to their heuristic again. This is done recursively, according to Korf’s recommendations, but also includes a small priority queue for each set of children generated when a node is explored. This way, of the 11 children to evaluate, the program first considers those with a heuristic best indicating the path to a solution.

Heuristic

In this discussion, it’s important to address the heuristic used for this search. A heuristic is a way to evaluate a state to try and give a decent guess towards the number of steps that a given state is away from a solved state. This is incredibly useful in our search, as it would be better to select states that are closer to a solved state.

My heuristic used in this search is based distinctly around my model for the cube. I model the 2x2x2 cube with a vector of eight orientations, where a solved state is an instance of that vector where all of the orientations are the same. Since each position can have 24 different orientations, there are 24 different solution states. To account for this, my heuristic is similar to Hamming distance, but instead of comparing to a specific solution state, the current cube is considered by the number of different orientations currently in the cube. The heuristic counts the number of different orientations in the eight different cubelets, and then divides that by 2, since each move can affect 4 cubelets, or half of the cube.

In my experimental data, there was never a case where the heuristic overestimated the number of steps to a solved state. It was often not exactly correct, but because it never overestimated, it is an admissible heuristic.

Implementing the Code

I implemented this code in C++ using packages from the STL library. Most of the data structures in the code were C++ vectors or sets, in a few cases. I used the chrono library to record real-time computation data for search performance. In addition to the utilities of the language, I wrote a few structs to be used for the search in a Node struct and SolveResult struct to hold the solution with some data surrounding that solution (number of explored nodes and time elapsed).

Although my code is functional, I think that it is far from optimized. Since I explore all 11-12 possible children, the branching factor of the search is very large, but there was not a noticeable speedup when the branching factor was reduced. This leads me to believe that the slowdown most likely comes from the intermediate priority queues, the function calls towards modifying the state of a cube, or simply the fact that my model of the cube makes it possible for there to be a large number of identical states that are visited many times, greatly increasing the search space.

For simplicity, my code randomized cubes to a certain depth and solved them 10 times for each depth, and then iterated the depth up to 14, which is the maximum number of quarter turns needed to solve a 2x2x2 cube. This meant that running the program overall took longer, but that the results were always ran on the same version of the code.

When the program is running, all of the output goes to a files names idastar\_output.txt. This file has a block of text for each cube test run. First is the initial heuristic, then the sequence of randomized moves to scramble the cube, the solution found, the start state of the cube in orientation notation, and then the number of nodes explored and time elapsed in microseconds.

Results

Time Complexity in Microseconds (μs) According to Randomization Depth

Depth Trials

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 | Avg |
| 1 | 60 | 102 | 68 | 143 | 77 | 56 | 66 | 62 | 49 | 66 | 76 |
| 2 | 79 | 77 | 78 | 79 | 84 | 150 | 85 | 148 | 76 | 76 | 93 |
| 3 | 254 | 52 | 308 | 264 | 201 | 234 | 239 | 311 | 183 | 302 | 235 |
| 4 | 179 | 90 | 1139 | 351 | 491 | 151 | 1713 | 443 | 208 | 67 | 483 |
| 5 | 3696 | 167 | 3210 | 3101 | 2912 | 183 | 2266 | 5469 | 6816 | 55 | 2788 |
| 6 | 49502 | 1850 | 14184 | 41118 | 22746 | 568 | 22379 | 28376 | 114343 | 102 | 29516 |
| 7 | 222345 | 312 | 654245 | 572949 | 290701 | 3648 | 462661 | 120182 | 192062 | 79 | 251918 |
| 8 | 3161601 | 983 | 3158157 | 17796 | 3576465 | 8898 | 8990115 | 32530 | 2328227 | 114 | 2127489 |
| 9 | 387677 | 3187 | 376768 | 4435 | 58541432 | 511125 | 35758725 | 327385 | 50452770 | 94 | 14636360 |
| 10 | 1000979 | 882 | 4067062 | 15106 | 62128778 | 3950588 | 258543971 | 5870872 | 11895892 | 4424513 | 35189864 |
| 11 | 462582 | 3267 | 45617777 | 74813 | 108587970 | 671917 | 514096483 | 12115351 | N/A | N/A | 85203770 |

Space Complexity in Nodes Explored According to Randomization Depth

Depth Trials

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 | Avg |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 2 | 2 | 2 | 14 | 2 | 14 | 2 | 2 | 4.4 |
| 3 | 61 | 1 | 85 | 49 | 51 | 37 | 49 | 37 | 15 | 85 | 47 |
| 4 | 28 | 2 | 429 | 94 | 144 | 14 | 707 | 138 | 38 | 2 | 160 |
| 5 | 1547 | 37 | 1442 | 1221 | 1214 | 37 | 959 | 2424 | 2981 | 1 | 1186 |
| 6 | 21830 | 822 | 6283 | 18626 | 10374 | 194 | 9643 | 12306 | 52812 | 2 | 13289 |
| 7 | 99427 | 49 | 296551 | 254871 | 130667 | 1530 | 206609 | 55285 | 87704 | 1 | 113269 |
| 8 | 1428526 | 375 | 1425217 | 8466 | 1633164 | 4006 | 400078 | 14771 | 1070860 | 2 | 598546 |
| 9 | 176589 | 1325 | 167914 | 1689 | 26684497 | 233990 | 16115612 | 149768 | 23132347 | 1 | 6666373 |
| 10 | 455882 | 274 | 1825831 | 6954 | 28323993 | 1815423 | 116844343 | 2673488 | 5484167 | 2016321 | 15944668 |
| 11 | 211324 | 1156 | 20398744 | 33707 | 49143178 | 310563 | 230056789 | 5472658 | N/A | N/A | 38203515 |

For consideration, the system that this test was run on consists of an AMD Ryzen 7 5700G processor, 32 GB of RAM, and a NVIDIA GeForce GTX 1060 3GB GPU.

The trials, as expected, began to scale increasingly worse with depth. The trials at depth 11 stopped because of a one-hour time constraint. It is clear that the heuristic and search implementation behaved as expected, though, since even deep into the randomization process the solver was able to find simple solutions to barely scrambled cubes down even to 9 random turns.

Lessons and Takeaways

In this assignment, I learned a lot about how intelligent design can make a significant impact on the ability to solve problems effectively and efficiently. However, even when intelligent design is implemented, there is always room for improvement. It is clear to me that even though I have an intelligent cube solver, there is much more optimization that could be done, or other approaches to be taken.

Acknowledgements and References

I found a significant part of research on this blog, which strongly inspired my data structure for this problem:

<https://k-l-lambda.github.io/2020/12/14/rubik-cube-notation/>

I read and used a modified version of Korf’s method for IDA\*:

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.288&rep=rep1&type=pdf

I used cplusplus.com for language reference:

https://cplusplus.com/

Additionally, I discussed my ideas for this structure with classmates Michael Liao and Christian Fronk.