Cube AI

Team 4 • Ben Duggan & Connor Altic

11/7/18

Background information

Our team is attempting to find an efficient AI algorithm to solve Rubik's cubes of various sizes. The goal is to observe how efficient AI techniques are at solving Rubik's cubes, of different sizes n. This can be done by measuring the time our algorithm takes to solve the cube and by counting the number of moves our algorithm takes to solve the cube. For n <= 4 the maximum number of moves away from solved, called god's number, is known but not known for n > 5. For this reason, using the number of moves taken to solve the cube may not be useful for n > 5. One reason this problem is challenging is because sometimes the cube needs to become less solved to make it more solved.

People have made many solvers for 3x3 Rubik's cubes, but we can't find any algorithm where they have generalized this. One algorithm we could use is Thisthethwaite's algorithm which restrict the cube positions into 5 different groups of cube positions which can then be solved using a certain set of moves.

We are wanting to make several heuristics such as a heuristic that counts the number of cubes in the correct position and counts the number of pockets, region of correctly oriented cubes. We additionally want to look at a MiniMax/Alpha-Beta style algorithm, but one that only maximizes as this isn't a two-player game. We plan on making a bi-directional search algorithm, one that searches from the unsolved and solved states and meets in the middle. We want to experiment with adding unnecessary moves which allow for faster solve but at the cost of a larger branching factor. Finally, we want to try solving using a graph instead of a tree as there are many states which repeat each other, thus we will save on memory usage and processing time, allowing us to search deeper.

Third-party libraries

We are using pygames as a third-party library. We are only using this to make a GUI so we can easily see the cube and interact with it. This should make it easier to track how the cube is being solved, instead of just looking at print statements.

Timeline

Week 1 (by 11/12/18): Finish python class that can represent n size cubes

Week 2 (by 11/19/18): Work on heuristic that counts the correct number of correctly positioned pieces, maximizing tree algorithm and pruning algorithm for the maximizing tree algorithm.

Week 3 (by 11/26/18): Experiment with different maximizing algorithms and bidirectional search

Week 4 (by 12/3/18): Experiment with using unnecessary moves in the solution (increasing the branching factor, k) and using a graph and finding the longest path or solved state.

Week 5 (by 12/10/18): Refine algorithms and collect data points.