Justin Svegliato

Professor Benjamin Carle

Artificial Intelligence

26 October 2012

Rubik’s Cube Solver Analysis

**Generation and Storage of Tables**

Three separate tables, which are stored in text files, containing a heuristic value for every possible combination of all possible corner and edge configurations were generated by starting at the goal state of the cube and subsequently applying all unique valid moves, namely a 90, 180, or 270-degree turn of a face. After the expansion of the goal state, which is the root node, the same process was subsequently applied to the resulting configurations, i.e., the child nodes. Every child; that is, every resulting configuration of a particular expanded parent node, was then itself expanded as well until a depth of eleven was reached (a depth of eleven is the maximum value that may be calculated by the heuristic specified in Korf’s paper). Further, the depth of a node is representative of the total number of movements away from the goal state.

An index for each possible combination of the edges and corner configurations were not stored in the pattern databases with the heuristic value. Instead, following the suggestion in Korf’s publication, an encoding function that took in the line number of a particular heuristic value yielded a unique configuration. For instance, in the corner configuration table, there were 88, 179, 840 (the number of total possible combinations) heuristic values ranging from zero to eleven with no corresponding index listed in the file. While reading in all the values, a counter in the application kept track of the current line number, which was then used in the encoding function to compute the associated configuration. By doing this, the size of the three tables, namely the corner and two edge files, were dramatically reduced in size.

**Execution Instructions**

In order to compute the solution of a particular configuration, the three heuristic tables must be generated beforehand. However, it is important to note that two distinct classes generated the corner and two edge heuristic tables in order to reduce memory requirements. After computing the tables, the solver can then be run in order to compute the solution of a particular combination. Furthermore, the application always reads from a file named “cube.txt” and it is important to have the heuristic tables, namely “heuristic1.txt”, “heuristic2.txt”, and “heuristic3.txt” stored in the appropriate directory in order to be used by the application.

**Solutions to Sample Configurations**

The solving of a particular configuration either resulted in a solution or was stopped manually. That is, none of the sample configurations were terminated due to memory issues. With sufficient time, the optimal solution of the sample configurations is likely to be solved by the application.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sample Cube** | **Solution** | **Initial Heuristic Value** | **Time** |
| Cube0 | Y3 R3 B2 W3 R3 W2 G3 O2 W1 | 7 | 8 seconds |
| Cube1 | No solution | 9 | 47 minutes |
| Cube2 | No solution | 10 | 10 minutes |
| Cube3 | No solution | 9 | 12 minutes |
| Cube 4 | No solution | 8 | 6 minutes |
| Cube 5 | No solution | 8 | 6 minutes |
| Cube 6 | No solution | 8 | 5 minutes |
| Cube 7 | No solution | 8 | 6 minutes |
| Cube 8 | No solution | 9 | 5 minutes |