Moving-Window Rubik's Cube Solver Gecco Competition 2009

Wolfgang Steitz, Uli Golle, Franz Rothlauf

Working Paper 02/2009

June 2009

Working Papers in Information Systems and Business Administration

Johannes Gutenberg-University Mainz
Department of Information Systems and Business Administration
D-55128 Mainz/Germany
Phone +49 6131 39 22734, Fax +49 6131 39 22185
E-Mail: sekretariat[at]wi.bwl.uni-mainz.de
Internet: `http://wi.bwl.uni-mainz.de`

Abstract

Rubik's Cube is a popular three-dimensional mechanical puzzle developed in the 1970's. The aim of the puzzle is to restore a scrambled cube such that each of the six faces of the cube is a solid color. This paper proposes an evolutionary optimization solver that finds a sequence of turns which restores an arbitrarily scrambled cube with a low number of turns. The solver, called Moving Window Solver (MWS), splits search in three phases. Each phase uses different objective functions and evolutionary search operators.

1 Introduction

The well-known Rubik's cube was introduced in the late 1970's by Erno Rubik. The original cube is of dimension 3×3×3 and contains six faces, each covered by nine stickers. In the initial state of the cube, all stickers located at one face have the same color (see Fig. 1). Thus, there are six different colors overall. Since the cube can be turned independently by a pivot mechanism, the cube can be arbitrarily scrambled such that the faces are not a solid color anymore. The aim is to restore such a scrambled cube by applying a sequence of turns such that the cube is in its initial state again and all faces have the same color. Finding a sequence with lowest number of turns is a challenging optimization problem. The current upper bound on the number of necessary turns to solve any scrambled cube is 22 [4].

For determining a sequence of turns that solves a cube, some solution procedures have been developed, e.g. Iterative-Deepening A* in combination with look-up tables [2, 3]. Exact approaches find a sequence with minimum turns. However, to limit computational effort, exact approaches are usually combined with look-up tables that store information about different states of a cube. Due to look-up tables, memory demands are high. To reduce time and space effort, other solution approaches use turn patterns instead of single turns to form a complete sequence of turns that solves a cube [1]. In contrast to single turns of a cube, which affect positions of stickers on four different faces, turn patterns only affect positions of a few stickers of the cube leaving the remaining cube unchanged. Turn patterns are constructed from single turns, resulting in very long turn sequences. Therefore, existing evolutionary algorithms based on evolution strategy and usage of turn patterns produce turn sequences that are quite large and typically need several hundreds of turns to solve a cube.

In contrast, the approach proposed in this paper produces solutions of moderate size in reasonable time without the usage of look-up tables or other stored informations. The Moving Window Solver (MWS) directly encodes the turn sequence as a string where each allele encodes a turn. Alleles are solved stepwise and the search process is split in three different solution phases. In the first phase, MWS solves a 2×2×3 part of the cube. In the second phase, a turn sequence is found that transfers the cube into a Two-Generator state. The final, third phases completes the cube. For each phase, appropriate objective functions and search operators are developed. No look-up tables are used. The resulting evolutionary MWS easily solves randomly scrambled cubes in a few seconds.

MWS participates in the Parabon Sponsored GECCO Competition 2009.

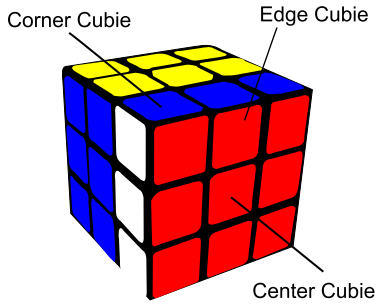

Figure 1: A Rubik's cube after a quarter-turn clockwise on the front face.

The goal is to evolve a program that can solve an arbitrarily-scrambled cube in the fewest number of turns. The contestants will be judged on the basis of 1,000 randomly scrambled cubes. Every move included in the solution produced by the solver will result in one penalty point. A penalty of 200 points will be added for each cube not solved in less than 200 moves or in less than 10 seconds. The contestants will not solely be judged on lowest penalties but also on originality of the approach.

The paper is structured as follows. Sect. 2 introduces notations. In Sect. 3, we describe MWS. Then, the performance of the proposed method is analyzed in Sect. 4. The paper ends with concluding remarks.

2 Rubik's Cube notations

The original Rubik's Cube with dimension 3×3×3 consists of 26 visible pieces, which are called cubies (the center piece of the cube is not visible). The cubies are of three different types: corner cubies, edge cubies and center cubies. The corner cubies are located at the corners of the cube, each having three visible stickers. The edge cubies are placed at the edges of the cube and have two visible stickers. In the middle of each face is one center cubie each having only one visible sticker. Since all center cubies are connected to the invisible center piece of the cube, their positions can not be changed in relation to one another. Thus the color of the sticker on the center cubie will define the color of the face. Overall, a cube has eight corner cubies, twelve edge cubies, and six center cubies.

Every single face of the cube can be independently rotated by quarter-turns. The turns are named by initials according to the face they are applied on. U for turning the Up-face, D for Down-face, R for Right-face, L for Left-face, F for Front-face and B for Back-face. Since the faces can only be moved by quarter-turns, there are three possible turns per face:

- a quarter-turn clockwise,
- a quarter-turn counterclockwise,
- and a half turn consisting of two quarter-turns clockwise, resp. counter-clockwise.

A quarter-turn clockwise of a face is denoted by using the single initial of the face. An additional "′" indicates a quarter-turn counterclockwise. A half turn is denoted by "2". Thus, for example F^2 denotes a half turn of the front face, while R is a quarter-turn clockwise of the right face. A sequence of turns is denoted as a composition of single turns applied from left to right. Therefore, the sequence $U'B^2L$ first conducts a quarter-turn counterclockwise on the Up-face, then a half turn on the Back-face, and finally a quarter-turn clockwise on the Left-face.

3 Solving Rubik's Cube

In the initial state of the cube, the stickers on each face match the same color. By applying a series of turns the cube can be arbitrarily scrambled, which usually leads to a state where the colors of the stickers on each face do not have the same color anymore. The aim of MWS is to find a sequence of turns as short as possible to restore a randomly scrambled cube. The overall solution process is divided into three phases which gradually restore the cube.

3.1 Moving Window Solver

The Moving Window Solver works on a turn sequence of fixed length l. The turns are initialized randomly. The solver gradually slides through the sequence from left to right and determines in each step the turn on the leftmost position of the window. Since every single turn in the sequence transfers the cube into a different state, it is possible that a subsequence of turns leads to a better result than the whole sequence. Thus, we determine the fitness of a sequence not only after applying the whole turn sequence but also after applying subsequences. The MWS considers only a window of the whole sequence at once starting from the first position in the sequence. In each step, the fitness of the turns in the window is determined by applying the turns within the window to the cube and measuring the improvement in restoring the cube. After that, some moves within the window are randomly mutated by replacing them with other moves and the fitness of the so-changed window is again determined. This process is repeated several times and the best window is kept. Thereupon the window is slided one position further. After the first slide it will starts with the second turn in the sequence. Thus, the first turn is fixed as it cannot be changed anymore. This procedure is continued until the end of the sliding window reaches the end of the turn sequence.

The overall fitness of the move sequence is finally determined by gradually applying the turns of the sequence from left to right to the scrambled cube and keeping the fitness of the cube after each move. The best fitness is the fitness of the whole sequence. There are different fitness measures conceivable to evaluate the improvement in restoring the cube. One could be to count the number of stickers on each face which match the color of the center cubie on this face. Or to count the number of cubies which are located in the right position.

3.2 Solution Process

The solution process is divided into three phases. Throughout each phase the aforementioned Moving Window Solver is applied. The main difference between the phases is the used fitness function and the parameter setting for the Moving Window Solver. The phases also differ in the moves that are allowed in the sequence.

In the first phase, a 2×2×3 part of the cube is solved, called a subcubie. This is the entire cube except two adjacent faces. A cube has twelve possible subcubies. In our approach, we start twelve independent optimization runs in parallel, each trying to solve one of the twelve different subcubies. If an optimization run finishes phase 1 and correctly finds its subcubie, it continues with the second phase. Fitness measure for phase 1 is the number of adjacent sticker pairs having the same color on each face of the respective subcubie. This leads to a total fitness of 22 when a subcubie is solved.

In the second phase the cube is transfered into a state called Two-Generator. The Two-Generator is a subgroup of all possible cube states. A cube is in the Two-Generator if it is scrambled by rotating only two adjacent faces. There are $73,483,200$ possible states in the Two-Generator in comparison with the 4.3×10^{19} states accessible by rotating all faces. Transferring the cube into the Two-Generator allows to restore it by only using the respective two faces and therefore leads to a reduction of the solution space. Since every cube in the Two-Generator contains one 2×2×3 solved subcubie, there are six corner cubies and seven edge cubies left to be positioned and oriented correctly. However, not every cube with a solved subcubie is also in the Two-Generator. The remaining cubies need to be positioned and oriented in a certain configuration so that they can be solved by moving the two faces left. In this phase the remaining cubies of the cube are positioned and oriented to meet such a configuration without destroying the already solved subcubie obtained in the first phase. This is achieved by using a fitness function consisting of the fitness function of phase one and functions which evaluate the positions and orientations of the corner and edge cubies.

In the third phase the whole cube is restored. After the second phase it is assured that the cube is in the Two-Generator. Therefore, it is sufficient to turn only the two remaining faces to restore the whole cube. Hence, in contrast to the other phases only moves turning those two faces are allowed. Given that there are two faces left and three possible moves per face, in phase 3, we only allow six possible turns. Since two or more consecutive moves of the same face will always be replaceable by just one move of that face, the move sequence used in phase three is initialized by alternating moves of the two faces. This is also considered when mutating moves of the sequence. As fitness function, the same function as in phase one is utilized, just applied on the unsolved parts of the cube.

WMS terminates if one of the twelve independent optimization run returns a correct solution.

scrambles		10	20	50
cubes		100	50	50
success rate		1.0	1.0	1.0
length	mean	69.45	80.96	90.7
	stdev	35.7	24.78	25.13
CPU time	mean	182.29	275.27	340.49
	stdev	189.43	142.78	239.03

Table 1: Experimental results of MWS for instances with 10, 20 and 50 random scrambles.

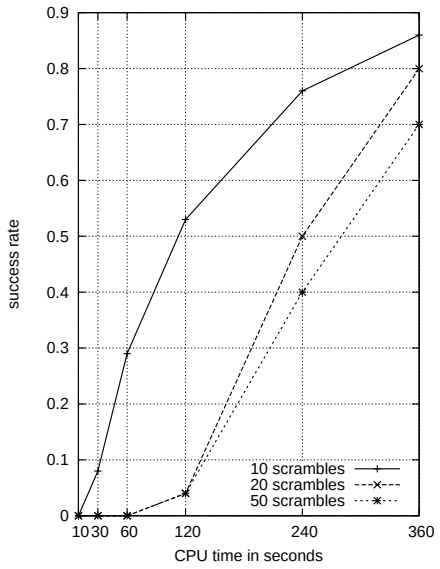

Figure 2: Success rate of MWS after different time intervals.

4 Experimental Results

In this section, we analyze the performance of our approach, by solving various randomly scrambled cubes. We study cubes with 10 (easy), 20 (medium) and 50 (hard) random scrambles. Table 1 gives an overview of our results. The table lists the success rate, the average solution length and the overall CPU time. Our solver was able to solve every instance, with an average solution length of 90.7 moves for hard instances. The solution length as well as the CPU time increases with the difficulty of the cube instance. CPU time has a large standard deviation.

Figure 2 plots the success rate after some time intervals. After 240 seconds we solved 75% of the easy, 50% of the medium instances and 40% of the hard instances.

5 Conclusions

This work describes a new approach to solve the Rubik's cube puzzle called Moving Window Solver (MWS). In contrast to existing approaches, MWS does not rely on lookup tables or other stored informations. The basic idea is to transfer the randomly scrambled cube in a Two-Generator state, which leads to a reduction of the solution space. The solution process is divided into three phases. All phases share the same optimization concept, the Moving Window Solver, but use different objective functions and search operators. An experimental analysis of the approach showed that every scrambled cube can be solved, but the solution length and the CPU increase with the number of random moves the instance is scrambled. Over 50% of the instances are solved after 240 seconds, with an average solution length of < 91 moves.

References

[1] M. Herdy. Application of the evolutionsstrategie to discrete optimization problems. In *Parallel Problem Solving from Nature*. Springer, 1991.

[2] H. Kociemba. Cube explorer. http://kociemba.org/cube.htm, 1992.

[3] R.E. Korf. Finding optimal solutions to rubik's cube using pattern databases. In *AAAI-97 Proceedings*, pages 700–705, 1997.

[4] T. Rokicki. Twenty-two moves suffice. http://cubezzz.homelinux.org/drupal/?q=node/view/121, 2008.