Design and Comparison of two Evolutionary Approaches for Solving the Rubik's Cube

Nail El-Sourani and Markus Borschbach

University of Applied Sciences,
Faculty of Computer Science, Chair of Optimized Systems,
Hauptstr. 2, D-51465 Bergisch Gladbach, Germany
nail@elsourani.com,markus.borschbach@fhdw.de

Abstract. Solutions calculated by Evolutionary Algorithms have come to surpass exact methods for solving various problems. The Rubik's Cube multiobjective optimization problem is one such area. In this paper we design, benchmark and compare two different evolutionary approaches to solve the Rubik's Cube. One is based on the work of Michael Herdy using predefined swapping and flipping algorithms, the other adapting the Thistlethwaite Algorithm. The latter is based on group theory, transforming the problem of solving the Cube into four subproblems. We give detailed information about realizing those Evolutionary Algorithms regarding selection method, fitness function and mutation operators. Finally, both methods are benchmarked and compared to enable an interesting view of solution space size and exploration/exploitation in regard to the Rubik's Cube.

1 Introduction

Since the Rubik's Cube's invention by Erno Rubik in 1974 and its commercialization in 1980 it has been the interest of not only hobbyists but also scientific research. Primarily mathematicians found themselves working on the Rubik's Cube as a discrete optimization problem trying to find efficient ways to solve it. With its simple structure the classic Rubik's Cube can reach a total number of $4.3 \cdot 10^{19}$ different configurations which induces an underlying complex optimization problem. To this day it is impossible to calculate all of those configurations. Also, the shortest length of sequences to reach any of those configurations (*God's Number*) is still unknown and subject to ongoing research [8].

In this paper we use two different evolutionary approaches to solve the Rubik's Cube as a discrete optimization problem. First, we briefly describe some characteristics and notation for the classic Rubik's Cube puzzle. We then introduce our two evolutionary approaches, one extending a method by Micheal Herdy [5], the other building upon Thistlethwaite's group-theoretic approach [2], [3], [10]. In contrast to [3] where we introduced our Thistlethwaite Evolution Strategy in detail, this work concentrates on our improvement of the Herdy Evolution Strategy and a thorough examination of both ES' mechanics. The careful benchmark and comparison of both algorithms provide an interesting

analysis of solution space size and relation between exploration and exploitation in regard to the Rubik's Cube.

2 The Rubik's Cube

2.1 Structure

The classic 3^3 Rubik's Cube is widely known and the one subject to this paper. It consists of 26 pieces: 8 corner pieces, 12 edge pieces and 6 center pieces, distributed equally on the six sides of the Cube. Each side of the Cube will be called *face*, each 2-dimensional square on a face will be referred to as *facelet*.

Corners, edges and centers are all *cubies* - representing the physical object. A corner shows 3 facelets, an edge 2 and a center 1. Each side of the Rubik's Cube can be rotated clockwise (CW) and counterclockwise (CCW). Every such single move changes the position of 4 edges and 4 corners - note that the center facelet on every of the Cube's faces always stays in the same position. Thus, the color of a solved face is always determined by its center color.

For each edge and corner it is of great importance to distinguish between *position* and *orientation*: i.e. an edge can be in its right position (defined by the two adjacent center colors) but in the wrong orientation (flipped).

2.2 Notation

There are several known notations [6] for applying single moves on the Rubik's Cube. We will use F, R, U, B, L, D to denote a clockwise quarter-turn of the front, right, up, back, left, down face and Fi, Ri, Ui, Bi, Li, Di for a counterclockwise quarter-turn. Every such turn is a *single move*. In Cube related research half-turns ($F2, R2, U2, B2, L2, D2$) are also counted as single move, we will do so as well. This notation is dependent on the users viewpoint to the cube rather than the center facelets' colors.

However, as a convention used for this research work we assume the classic Rubik's Cube color configuration which is $white : yellow, red : orange, blue : green$ where : denotes *opposite of*. The starting orientation for the scrambles will be $F = white, R = red, U = blue, B = yellow, L = orange, D = green$.

2.3 Characteristics

Obviously the Rubik's Cube fulfills the characteristics of a mathematical group [4], [9]. The number of all attainable states $4.3 \cdot 10^{19}$ depicts the order of the Cube group $G_C =< F, R, U, B, L, D >$. All configuration of the Rubik's Cube can be reached by using combinations of single moves in this group, thus the single moves *generate* G_C. Further,

- there is always a neutral element, i.e. $F \cdot FFFF = FFFFF = F$ and $F^4 = 1$ (also showing the order of each generator in G_C is 4)
- there always exists an inverse: $Fi \cdot F = 1$ and $Fi = FFF$

The inverse of any operation is quickly calculated by reversing the order of single moves and their direction. For example the inverse of $FRiDLBUi$ would be $UBiLiDiRFi$.

Further we can define subgroups of the group G_C. Let $H =< R, U >$ be a such a subgroup. If we only use moves from H there are just $2^6 \cdot 3^8 \cdot 5^2 \cdot 7 = 73483200$ different configurations we can attain [7]. This significantly reduces the number of possible states a Cube can reach, but induces certain constraints like unchangeable edge orientations.

3 Related Work

Only a few evolutionary approaches dedicated to solve the Rubik's Cube exist. In 1994 Herdy devised a method which successfully solves the Cube [5] using pre-defined sequences as mutation operators that only alter few cubies, resulting in very long solutions. Another approach by Castella could not be verified due to a lack of documentation. Recently Borschbach and Grelle [1] devised a 3-stage Genetic Algorithm based on a common human "SpeedCubing" [6] method, first transforming the Cube into a 2x2x3 solved state, then into a subgroup where it can be completed using only two adjacent faces (two-generator group).

4 Groundwork

When designing an ES to solve the Rubik's Cube a few things come to mind. While ES turn out to be very useful problem-solving algorithms for complex optimization problems, typically those problems are described in a system of continuous variables. Small variations of these variables usually lead to small changes of the value of the fitness function (called *strong causality*). This is a fundamental behavior needed in ES to ensure a steady search for better solutions in the solution space. It is of high importance to transfer this behavior to discrete optimization problems. This can be done by adapting suitable mutation operators that work well with the fitness function used.

Obviously, the individuals that will be evolved are actual representations of a Rubik's Cube. Starting from the solved state, a certain configuration of a Cube is defined through a sequence of moves applied. A distinct state most certainly has multiple sequences leading to it, the state itself however is unique and one of the $4.3 \cdot 10^{19}$ possible states. A scrambled Cube can be evolved by applying moves that hopefully near it to the solved state. This will be the ground principle of the two forthcoming ES.

4.1 Individual Representation

To enable us a better opportunity for comparison all ES designed for this work use the same representation of a Rubik's Cube. Each face is implemented as a 3×3 2-dimensional matrix containing values from 1 to 6 where each value

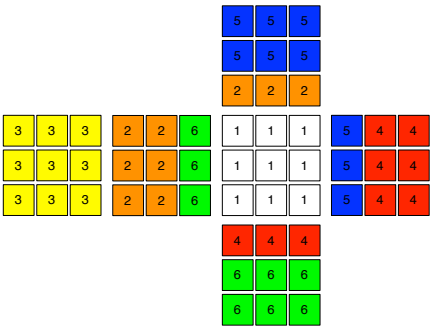

Fig. 1. Internal Representation of a Rubik's Cube Individual after Applying F to Solved State.

depicts one color. Thus, one individual is described by 6 matrices, describing each facelets color configuration.

Every quarter- and half-turn can be applied to this representation, yielding a total of 18 different single moves while leaving the Rubik's Cube integrity intact. Additionally the whole Cube can be rotated clockwise and counterclockwise. This guarantees easy human verification with a real physical Cube.

4.2 Mutation

Mutation being the primary search operator of ES is easily realized by not modifying a single facelet's color but applying a sequence of moves to the Cube. This guarantees that the Cube's integrity stays intact at all times and makes a separate integrity test superfluous. Allowing mutations which operate on single facelets and change their color could yield non-existent Cube states, due to its structure.

Every Cube saves the mutations it has undergone, i.e. a list of moves that have been applied. To keep this list as small as possible, redundant moves are removed automatically. To clarify: we assume the above Cube where only F has been applied. Let the next mutation be $FRRiB$. This will automatically yield in $F \cdot FRRiB = F2B$. We will go into further detail when describing each of the Evolution Strategies.

5 Two Evolutionary Approaches to the Rubik's Cube Puzzle

5.1 Herdy Evolution Strategy

In 1994 Michael Herdy presented an (1,50) Evolution Strategy solving the Rubik's Cube in a mean of 38.7 generations, calculating only a mean of 1935 of all possible configurations [5]. This algorithm was enhanced and implemented as follows.

Fitness Calculation To calculate the fitness of an individual the standard fitness function proposed by Michael Herdy was used. Three qualities q_1, q_2, q_3 are defined:

- q_1 is increased by 1 for each facelet whose color differs from the center facelet on the same face
- q_2 is increased by 4 for each wrong positioned edge, orientation is not considered
- q_3 is increased by 6 for each wrong positioned corner, orientation is not considered

As we see, each of those qualities can reach a maximum of 48, leading us to $max\{q_1 + q_2 + q_3\} = 144$. Obviously the Cube is in a solved state when the sum's value reaches 0.

Mutation Operators Herdy proposed the following mutation operators which change the fitness of each individual just slightly: two-edge-flip, two-corner-flip, three-edge-swap, two-edge/two-corner-swap, three-corner-swap, two-corner-swap and two-edge-swap - each in both directions (meaning the inverse sequence). Those were slightly extended using mirrors and adding three-inslice-edge-swap [2]. Note that each such mutation operator is depicted by a sequence of $x, 6 \leq x \leq 14$ single moves.

Table 1. Herdy ES Mutation Operators (omitting Mirrors).

mutation	sequence	length
two edge flip cw	FRBLULiUBiRiFiLiUiLUi	14
two edge flip ccw	FiLiBiRiUiRUiBLFRURiU	14
two corner flip cw	LDiLiFiDiFUFiDFLDLiUi	14
two corner flip ccw	RiDRFDFiUiFDiFiRiDiRU	14
three edge swap cw	UF2UiRiDiLiF2LDR	10
three edge swap ccw	UiF2ULDRF2RiDiLi	10
two edge/corner swap cw	RiURUiRiUFRBiRBRFiR2	14
two edge/corner swap ccw	LUiLiULUiFiLiBLiBiLiFL2	14
three corner swap cw	FiUBUiFUBiUi	8
three corner swap ccw	FUiBiUFiUiBU	8
three inslice edge swap cw	RLiU2RiLF2	6
three inslice edge swap ccw	LiRU2LRiF2	6

The above operators change the state of a Cube just slightly by only affecting 2-4 positions while leaving the rest of the Cube intact. This guarantees a slow and steady exploration of the solution space, slightly improving the population per generation. To fully utilize the above operators potential, the face being the physical front of the Cube has to be randomly chosen before each mutation as well as the orientation of the Cube. Thus an actual mutation step looks like this:

1. choose a random face to become the new front, this involves the entire Cube to be rotated, there are 6 faces to choose from

2. choose a random orientation of the front by rotating the whole Cube cw/ccw 1 or 2 times

Looking at the mutation we see how only very specialized operators are used. Tests with random sequences or single moves turned out to always get stuck in local optima which corresponds to the discovery made by Borschbach and Grelle. Particularly with this fitness function just a single quarter turn of one face will drastically change the fitness of an individual.

Selection Method After having applied random mutations to every individual in the population, they are sorted by their fitness in ascending order (remember 0=solved). Now, the μ best individuals are selected for duplication. There are several methods to select an individual for duplication from the selection pool, ranging from very simple ones (gaussian random) to quite complex e.g. non-linear fitness-scaling. It turns out that simply choosing random individuals (with a random function that generates values that are equally distributed within the specified range) is the most effective.

5.2 Thistlethwaite Evolution Strategy

We can say that the grade of specialization of mutation operators corresponds to the grade of restriction of an Evolutionary Strategy. With the Herdy ES turning out to be a very restricted ES we wanted to design an ES with the slightest possible grade of restriction i.e. truly random mutation operators. To achieve this, a common method is to break very complex optimization problems down into smaller parts which are solved independently, at best returning a solution for each subproblem that can finally be joined to get a solution for the original problem. This is exactly how the Algorithm devised by Thistlethwaite in 1981 works. In the original algorithm each subproblem is solved by finding a solution in precalculated lookup-tables. Our ES does not use those lookup-tables to solve those subproblems, but rather evolves Cubes solving each problem by using a dedicated fitness function.

Fitness Calculation The Thistlethwaite Algorithm (TWA) starts with the Cube in the G_o group, a group where all moves are allowed to be applied to the Cube. This starts a chain of nested groups, i.e. $G_o > G_1 > G_2 > G_3$, with the order of magnitude of each subsequent group being smaller than the one before. The groups are defined as follows [10]:

- $G_0 =< F, R, U, B, L, D >, |G_0| = 4.33 \cdot 10^{19}$
- $G_1 =< F, R2, U, B, L2, D >, |G_1| = 2.11 \cdot 10^{16}$
- $G_2 =< F2, R2, U, B2, L2, D >, |G_2| = 1.95 \cdot 10^{10}$
- $G_3 =< F2, R2, U2, B2, L2, D2 >, |G_3| = 6.63 \cdot 10^{5}$

As we can see, with every subsequent group the total number of possible states the Cube can achieve (and therefore the number of states the Cube has to be in,

to be solved) by only using moves from this distinct group, is greatly reduced (detailed group order calculation in [3]). Naturally the higher the group index, the more constraints we have to fulfill. Once a Cube is maneuvered into a subgroup, we can freely apply all moves of this subgroup and it will stay in this group. Furthermore we must only use moves of this subgroup to put it into the next group and so forth. We can not use any moves from previous groups which are not part of the current subgroup without destroying what we have reached so far.

Assuming we have some random scrambled Cube. Naturally this Cube will be in G_0 which allows us to apply any move we want. Now, by using moves from G_0 we put the Cube into G_1. Applying R would put it back into G_0 again, as this move is not part of G_1 and so forth. Thus, we choose to calculate fitness separately for each group transition. This is done via

$$phase_i = weight \cdot v + c, i = 0, 1, 2, 3 \tag{1}$$

where v is a count for wrong positioned/oriented cubies in regard to the constraints induced by the particular subgroup and c the length of moves already applied to the current individual. Thus, not only the necessary group transition but also a short solution sequence length is promoted as the desired goal.

To clarify the constraints induced by different subgroups let us take a closer look at the group G_1. Changing an edge's orientation on the Rubik's Cube implicates the ability to use quarter-turns of adjacent faces. Evidently, this is not possible once in G_1. Thus, G_1 induces the constraint that each edge cubie must be oriented (not necessarily positioned) right. Further subgroups induce further constraints, detailed in [3].

Mutation Operators Dividing the problem of solving the Rubik's Cube into 5 phases by splitting it into 4 subproblems gives us the ability to use truly random sequences of single moves as mutation operators. The only restriction is given by the group transition the Cube is undergoing. Conveniently the maximum sequence length needed to transform the Cube from one subgroup to another is given by Thistlethwaite: $G_0 \rightarrow G_1 : 7$, $G_1 \rightarrow G_2 : 13$, $G_2 \rightarrow G_3 : 15$ and $G_3 \rightarrow solved : 17$.

In each phase, a sequence of random length between 0 and *max. sequence length (=i)* is generated by filling each position with a random single move of the according group. By allowing the sequence length to be 0, we are artificially inducing characteristics of an $(\mu+\lambda)$-ES while actually being an (μ, λ)-ES. Unlike the Herdy ES where with each mutation the number of applied moves increases, here a decrease is possible induced by the automatic cleanup of move sequences.

Selection Method Tests have shown no need for a dedicated Selection method. Standard EA selection methods except for random choice tend to decrease the population's diversity after phase transitions. In early versions this would sometimes lead to a local optimum, thus not solving the Cube. This results from the TW ES being very sensitive about selection pressure. To counter this

property a gaussian random function is used to choose from the selection pool for duplication.

6 Benchmark and Comparison

For benchmark and comparison a total of 100 random scrambles between 10 and 50 single moves were generated and solved by both ES in 5 repetitions. These tests were conducted on the same hardware and conditions using the shared Rubik's Cube framework proposed earlier (individual representation and mutation mechanic). The numbers used for μ and λ were chosen after preleminary benchmarks found them to be the most effective. The high parent population is necessary to provide the possibility for a broad spectrum of different sequence combinations in later generations. In short, large parent populations enable a high individual diversity - which is crucial in a solution space of this size. This is even more critical to the Thistlethwaite ES due to the random mutation operators used, contrary to the hard-coded sequences in the Herdy ES.

6.1 Herdy Evolution Strategy

The Herdy Evolution Strategy with $(\mu, \lambda) = (100, 5000)$ is used. Calculations were repeated 5 times to enable a statistically more precise evaluation.

Table 2. Solutions of 100 random scrambles, 5 repetitions, Herdy ES.

	run 1	**run 2**	**run 3**	**run 4**	**run 5**
avg. generations	18.13	18.28	18.06	18.42	18.11
avg. moves	232.27	234.50	233.43	236.62	236.21
avg. time(s)	5.66	5.12	5.00	4.66	4.75

Fast performance, a small number of generations needed and long solution sequences seem to be typical properties of the Herdy ES. Results for this particular Evolution Strategy are very predictable and perfectly fit the mechanics described above.

96% of the scrambles could be solved in x moves with $180 < x < 280$, roughly resembling a normal distribution around the x value with $y = max(x)$. This is when providing a sufficient number and/or versatility of scrambles.

Comments The Herdy ES is really a theoretically simple ES for solving the Rubik's Cube. It performs amazingly fast, only calculating a tiny fraction of the total number of possible Cube states, solving the Cube in only few generations. On the downside the mean length of the solution sequences calculated by this algorithm can be estimated by

$$11 \cdot g \,|\, g := number\, of\, generations\, needed \tag{2}$$

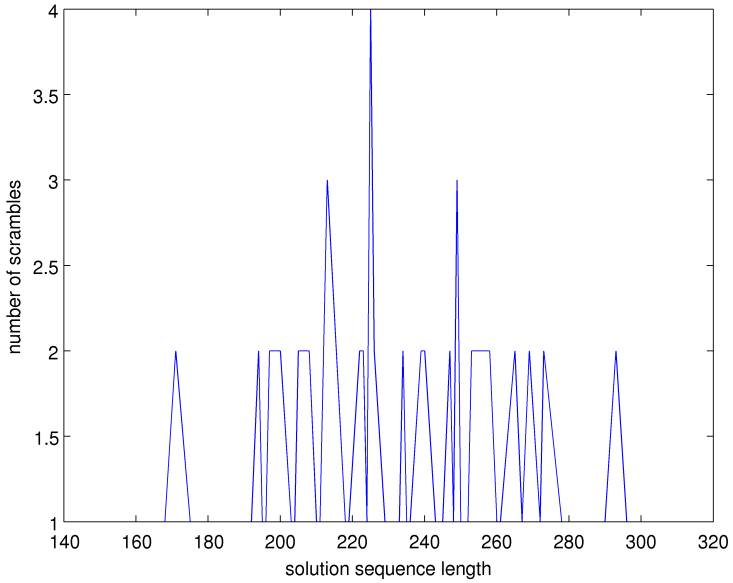

Fig. 2. 100 Random Scramble Test: Distribution of scrambles on solution length, Herdy ES.

which is quite huge. Solution sequences with a length of well above 100 are quite normal. In addition the "random element" which makes up the "evolutionaryness" of an ES is quite small - only affecting which cubies will be swapped or flipped as the mutation sequences are all hard-coded. Tests were conducted that added the length of the applied moves in each generation to the fitness function. No improvements could be observed which is not surprising regarding the above arguments.

6.2 Thistlethwaite Evolution Strategy

The following settings were used: $(\mu, \lambda) = (1000, 50000)$, weighing factors are $(5, 5, 5, 5, 5)$, mutation lengths $(5, 5, 13, 15, 17)$ and maximum generations before reset (250).

Table 3. Solutions of 100 random scrambles, 5 repetitions, Thistlethwaite ES.

	run 1	run 2	run 3	run 4	run 5
avg. generations	95.72	100.63	92.71	99.66	92.22
avg. moves	50.67	50.32	50.87	50.23	49.46
avg. time(s)	321.78	381.68	393.99	312.98	287.93

Compared to the Herdy ES calculation time is greater by a factor of about 60. This results not only from a higher population size but also, for some scrambles, the TW ES maneuvers into a local optimum which triggers a self-reset. A further consequence of this are the relatively high numbers of generations needed. The solution sequence hits an average of about 50 single moves which is slightly lower than the upper bound of 52 for the classic Thistlethwaite Algorithm [10].

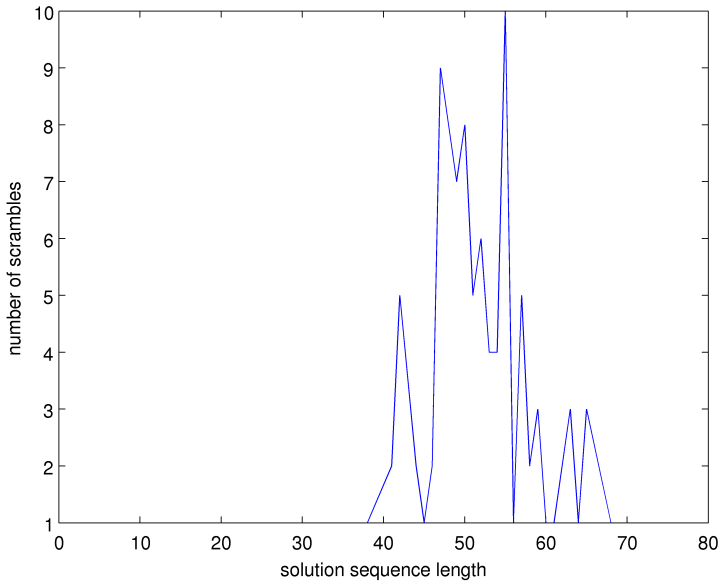

Fig. 3. 100 Random Scramble Test: Distribution of scrambles on solution length, Thistlethwaite ES.

Comments The Thistlethwaite ES yields results that resemble the theoretical solution lengths of the classic Thristlethwaite's Algorithm without any use of precalculated lookup-tables. Unlike the Herdy ES which is efficient with a small population size, the TW ES needs much larger populations to guarantee a successful solve. This directly affects calculation time. Even though the Cube solving is broken down into 4 smaller subproblems, each subproblem is still of great order.

7 Conclusion

When comparing the results of both Evolution Strategies we notice a tradeoff between computing time and quality of solution. Although very different in

nature both ES solve the same optimization problem. Obviously there are some important differences. The Herdy ES

- is a $(100, 5000)$ ES
- uses predefined mutation operators
- incorporates a very simple fitness function

Each of those are reasons for the faster performance of the Herdy ES. We can look at it the opposite way. The Thistlethwaite ES

- is a $(1000, 50000)$ ES
- uses randomly generated mutation operators
- incorporates a complex fitness function dividing the solving process into four subproblems

The differences seem obvious. While the Herdy ES provides fast but long solutions to any scramble, the Thistlethwaite ES delivers better results in terms of solution length. This is easily explained with the mutation operators used. While the TW ES could solve the very simple scramble F by simply returning Fi as the solution, the Herdy ES would need to perform 2 edge and 2 corner swaps, resulting in a very long solution for such a simple scramble. This disadvantage becomes less important when solving more complex scrambles, however the TW ES performs a shorter solution at all times. Running the TW ES with smaller μ, λ yields faster calculation but can not guarantee a successful solve. More tests will have to be conducted for further examination of the TW ES's behavior with different parameters and sequence optimization which could result in faster computation and/or shorter solution lengths and occurrence reduction of self resets.

References

1. Borschbach, M., Grelle, C.: Empirical Benchmarks of a Genetic Algorithm Incorporating Human Strategies. Technical Report, University of Applied Sciences, Bergisch Gladbach (2009)
2. El-Sourani, N.: Design and Benchmark of different Evolutionary Approaches to Solve the Rubik's Cube as a Discrete Optimization Problem. Diploma Thesis, WWU Muenster, Germany (2009)
3. El-Sourani, N., Hauke, S., Borschbach, M.: An Evolutionary Approach for Solving the Rubik's Cube Incorporating Exact Methods. Applications of Evolutionary Computations, pp. 80-90. LNCS 6024, Springer (2010)
4. Frey, A., Singmaster, D.: Handbook of Cubic Math. Enslow, Hillside (1982)
5. Herdy, M., Patone, G.: Evolution Strategy in Action, 10 ES-Demonstrations. Technical Report, International Conference on Evolutionary Computation (1994)
6. Petrus, L.: Solving Rubik's Cube for speed. `http://lar5.com/Cube/`
7. Reid, M.: Cube Lovers Mailing List. `http://www.math.rwth-aachen.de/~Martin.Schoenert/Cube-Lovers/Index_by_Author.html`
8. Rokicki, T.: Twenty-Three Moves Suffice. `http://Cubezzz.homelinux.org/drupal`
9. Singmaster, D.: Notes on Rubik's Magic Cube. Enslow, Hillside (1981)
10. Thistlethwaite, M.B.: The 45-52 Move Strategy. London CL VIII (1981)