An Evolutionary Approach for Solving the Rubik's Cube Incorporating Exact Methods

Nail El-Sourani, Sascha Hauke, 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, sascha.hauke@fhdw.de, 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 work we present an evolutionary approach to solve the Rubik's Cube with a low number of moves by building upon the classic Thistlethwaite's approach. We provide a group theoretic analysis of the subproblem complexity induced by Thistlethwaite's group transitions and design an Evolutionary Algorithm from the ground up including detailed derivation of our custom fitness functions. The implementation resulting from these observations is thoroughly tested for integrity and random scrambles, revealing performance that is competitive with exact methods without the need for pre-calculated lookup-tables.

1 Introduction

Solving the Rubik's Cube is a challenging task. Both the size of the solution space induced by the number of attainable states and multiple desirable side-objectives next to restoring the Cube (favorably in the smallest possible number of moves and lowest calculation complexity) make this an interesting optimization problem. Although invented in 1974, the number of moves required to solve any state of Rubik's Cube (the so-called *God's Number*) is yet to be determined after 30 years.

Various algorithms were devised to decrease the upper bound. However, all those approaches are strictly exact methods and the most recent ones rely on terabytes of pre-calculated lookup-tables. This is reflected by the current lowest upper bound of 22 moves achieved by Rokicki [11]. This number was attained by applying the same method he had used earlier for pushing the upper bound to 26, 25 and then 23 moves - using the very same algorithm only on more powerful hardware and a longer calculation time [10], [11].

Evolutionary Algorithms have been successfully applied in a variety of fields, especially highly complex optimization problems [2], [8], [14]. Oftentimes, superior solutions - as compared to classical algorithms have been achieved - notably in multiobjective cases (for example multiconstraint knapsack problems [4]). This gives rise to the idea of applying Evolutionary Algorithms to the Rubik's Cube problem. All relevant approaches are based on dividing the solution space of the Rubik's Cube into mathematical groups, starting with Thistlethwaite using 4 [13], then Reid combining two of Thistlethwaite's groups resulting in total of 3 [9] and finally Kociemba's [7] and Rokicki's approach using 2 subgroups. This makes the group theoretic approach a reasonable starting point for designing Evolutionary Algorithms. It is of particular interest to us to determine how such an EA can solve the Cube without relying on extensive lookup-tables.

2 Notation and Basic Concepts

2.1 Rubik's Cube

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

A corner cubie shows 3 facelets, an edge 2 and a center 1. Each side of the Cube can be rotated clockwise (CW) and counterclockwise (CCW). Each single move changes the position of 4 edges and 4 corners. The center facelets remain fixed in position. They determined their face's color.

For each edge and corner we 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).

There are several known notations 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 moves. This notation is independent of colors but depends on the viewpoint. A sequence of moves, an operation, is created by concatenating single moves and is called *operation* (i.e. $FRBiL2$).

2.2 Applied Group Theory

A *group* G is a set together with multiplication and identity e ($eg = g$), inverse ($gg^{-1} = g^{-1}g = e$) and an associative law. A *subgroup* $H < G$ is a subset H that is closed under group operations. $S \subseteq G$, written $G =< S >$ is a *generator* of G if any element of G can be written as a product of elements of S and their inverses. The *order* of the group is the number of elements in it, $|G|$. Given a group G and a subgroup $H < G$, a *coset* of H is the set $Hg = hg : h \in H$; thus, $H < G$ partitions G into cosets. The set of all cosets is written $H \diagdown G$.

Obviously, all possible states of a Rubik's Cube are described by the group generated by its applicable moves $G_C =< F, R, U, B, L, D >$, also called the *Cube Group* ($|G_C| = 4.3 \cdot 10^{19}$). Let $H =< L, R, F, B, U2, D2 >$ be a subgroup of G_C, representing a Cube where only the edge positions matter, as no edge orientations can be altered. Thus, $H \diagdown G_C$ depicts the left coset space which contains all possibly attainable states when only flipping edge cubies (changing an edges orientation). For extended explanation refer to [5], [12].

3 Related Work

3.1 Non-evolutionary Approaches

There are several computational approaches for solving the Rubik's Cube, the three most important being the work of Thistlethwaite, Kociemba and Rokicki. Their advanced algorithms are based on group theory concepts and apply advanced concepts such as symmetry cancelation and dedicated traversal methods (e.g. Iterative Deep Searching, IDA*).

Thistlethwaite's Algorithm (TWA) works by dividing the problem into 4 subproblems - specifically subgroups and subsequently solving those. By using precalculated lookup-tables, sequences are put together that move a Cube from one group into another until it is solved [13].

Kociemba's Algorithm takes the idea of dividing the problem into subgroups from Thistlethwaite, but reduces the number of needed subgroups to only 2. This method uses an advanced implementation of IDA*, generating small maps, calculating and removing symmetries from the search tree and tends to solve the Cube close to the shortest number of moves possible. Kociemba made his approach available in form of a program called *Cube Explorer* which can be found at [7].

Rokicki realised that the initial parts of the pathways computed by Kociemba's Algorithm are solutions to a large set of related configurations. He exploits this property by dividing the problem into 2 billion cosets, each containing around 20 billion related configurations. With this method he was able to push the upper bound to 22 moves sufficing to solve the Cube from any initial scrambled configuration [10], [11].

3.2 Evolutionary Approaches

Only a few evolutionary approaches dedicated to solving the Rubik's Cube exist. In 1994 Herdy devised a method which successfully solves the Cube [6] 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" 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 Thistlethwaite's Algorithm

The basic idea of the TWA is to divide the problem of solving the Cube into four independent subproblems by using the following four nested groups: $G_0 =<$

$F, R, U, B, L, D >, G_1 =< F, U, B, D, R2, L2 >, G_2 =< U, D, R2, L2, F2, B2 >$ $, G_3 =< F2, R2, U2, B2, L2, D2 >, G_4 = I$. Obviously, $G_0 = G_C$. The functional principle of Thistlethwaite's Algorithm is to put the Cube into a state where it can be solved by only using moves from G_i which again has to be achieved by only using moves from G_{i-1} for $i = 1, \ldots 4$, thus named *nested groups.*

Specifically, every stage of the algorithm is simply a lookup table showing a transition sequence for each element in the current coset space $G_{i+1} \diagdown G_i$ to the next one ($i = i+1$). These coset spaces, each describing a reduced form of the 3^3 Rubik's Cube puzzle, induce different kinds of constraints. This directly results in the total number of attainable states being reduced by using only moves from some subgroup G_{i+1}. The exact orders for each group are calculated as follows:

G_0 $|G_0| = 4.33 \cdot 10^{19}$ represents the order of the Cube Group.

G_1 The first coset space $G_1 \diagdown G_0$ contains all Cube states, where the edge orientation does not matter. This is due to the impossibility of flipping edge cubies when only using moves from G_1. As there are 2^{11} possible edge orientations,

$$|G_1 \diagdown G_0| = 2^{11} = 2048 \tag{1}$$

the order of $|G_1|$ is

$$|G_1| \equiv \frac{|G_0|}{|G_1 \diagdown G_0|} = 2.11 \cdot 10^{16} \ . \tag{2}$$

G_2 Using only moves from G_2, no corner orientations can be altered (eliminating 3^7 states). Additionally, no edge cubies can be transported to or from the middle layer (eliminating $\frac{12!}{(8! \cdot 4!)}$ states). The coset space $G_2 \diagdown G_1$ thus depicts a reduced puzzle of the order

$$|G_2 \diagdown G_1| = 3^7 \cdot \frac{12!}{(8! \cdot 4!)} = 1082565 \tag{3}$$

and

$$|G_2| \equiv \frac{|G_1|}{|G_2 \diagdown G_1|} = 1.95 \cdot 10^{10} \ . \tag{4}$$

G_3 Once in the coset space $G_3 \diagdown G_2$, the Cube can be solved by only using moves from G_3, here the edge cubies in the L, R layers can not transfer to another layer (eliminating $\frac{8!}{(4! \cdot 4!)} \cdot 2$ states) and corners are put into their correct orbits, eliminating $\frac{8!}{(4! \cdot 4!)} \cdot 3$ states). Thus,

$$|G_3 \diagdown G_2| = (\frac{8!}{(4! \cdot 4!)})^2 \cdot 2 \cdot 3 = 29400 \tag{5}$$

and

$$|G_3| \equiv \frac{|G_2|}{|G_3 \diagdown G_2|} = 6.63 \cdot 10^5 \ . \tag{6}$$

G_4 As G_4 represents the solved state - obviously $|G_4| = 1$ which means there exist a mere $|G_3|$ possible states for which a solution needs to be calculated to transfer from $G_4 \diagdown G_3$ to solved state.

Most essential to TWA are the groups G_1, G_2, G_3 as G_0 simply describing the Cube Group G_C and G_4 the solved state. To further exemplify how the coset spaces simplify the Rubik's Cube puzzle the following may prove helpful. When looking at the constraints induced by $G_2 \diagdown G_1 \diagdown G_0$ carefully (combining the constraints induced by $G_2 \diagdown G_1$ and $G_1 \diagdown G_2$) it is essentially a Rubik's Cube with only 3 colors - counting two opposing colors as one. This representation can be reached for each distinct coset space by examining and applying its effect to the complete Rubik's Cube puzzle.

While solving the Rubik's Cube in a divide and conquer manner, breaking it down into smaller problems (by generating groups and coset spaces) is effective, there exists one major flaw. Final results obtained by concatenating shortest subgroup solution do not necessarily lead to the shortest solution, globally.

5 The Thistlethwaite ES - An Evolution Strategy Based on the Thistlethwaite's Algorithm

As seen above, in the classic TWA the order of each subproblem is significantly smaller than $|G_C|$ and is reduced from stage to stage. The four resulting problem spaces are much more suitable to be solved via ES, as calculation complexity and the probability of ending up in local minima is decreased. Further, this enables the use of truly random mutation operators (otherwise highly ineffective in all of $|G_C|$ [1]) opposed to the hard-coded sequence approach used by Herdy [6].

Thus, we present a 4-phase ES with each phase calculating one group transition (will be referred to as Thistlethwaite Evolution Strategy, TWES). These phases share the same basic selection method but differ in mutation operators and fitness functions. Effectively, the presented ES can best be described as four consecutive ES, each using the solution of the previous one as starting individual to be duplicated (the first using the scrambled input Cube).

5.1 Basic Workflow

A scrambled Cube is duplicated λ times and the main loop is started using a fitness function $phase_0$ and only mutation sequences from G_0. As soon as μ Cubes which solve $phase_0$ have been evolved, the phase transition begins.

During phase transition, from those μ $phase_0$-solving Cubes, a random Cube is chosen and duplicated. This is repeated λ times and yields in the first population after the phase transition. Now the phase-counter is increased by one, and the main ES loop is entered again. This process is repeated until $phase_4$ is solved (i.e. $phase_5$ is reached), presenting a solution sequence to the originally scrambled Cube. This workflow is demonstrated in Fig. 1 (for in-depth implementation details see [3]).

In order to avoid the TWES getting stuck in local optima an upper bound for calculated generations is introduced. As soon as this upper bound is reached,

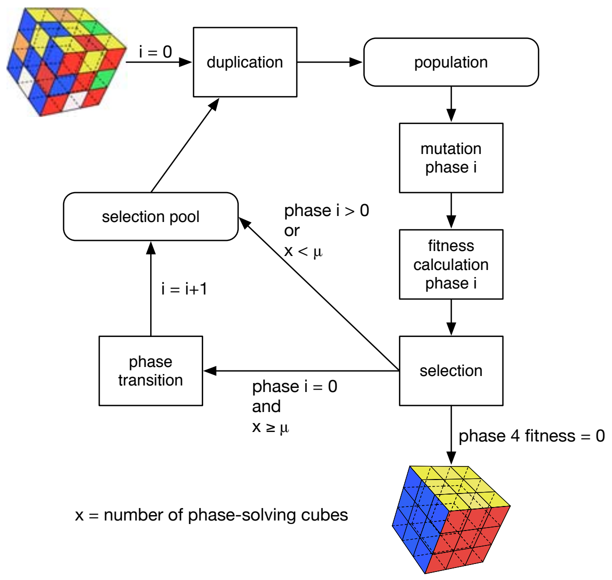

Fig. 1. Basic workflow of Thistlethwaite ES, $i = 0, \ldots, 5$

the TWES resets itself and starts over again. Based on testing several scrambles, the default upper bound is set to 150 generations.

5.2 Rubik's Cube as an Individual

The Rubik's Cube is represented using 6 2D matrices containing values from 1 to 6, each representing one color. Every quarter- and half-turn can be applied to this representation, yielding a total of 18 different single moves while still leaving the Cube's integrity intact.

Thus, mutation 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.

Every individual remembers 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 automatically removed. For example an individual that has been mutated with F and is then mutated with $FRRiB$ will only remember the optimized sequence $F \cdot FRRiB = F2B$, preventing redundancy. Essentially, this is realized via a while-loop, eliminating redundant moves in each pass until no further optimizations can be made: e.g. $F2BBiR2R2F$ is optimized to Fi by first removing BBi, then removing $R2R2$ and finally transforming $F2F$ into Fi.

5.3 Fitness Function

Translating the TWA into an appropriate Fitness Function for an Evolutionary Algorithm essentially forces the design of four distinct subfunctions. As each subgroup of G_0 has different constraints, custom methods to satisfy these constraints are proposed.

$\mathbf{G_0 \rightarrow G_1}$ To reach G_1 from any scrambled Cube, we have to orient all edge pieces right while ignoring their position. The fitness function for this phase simply increases the variable $phase_0$ by 2 for each wrong oriented edge. Furthermore, we add the number of moves that have already been applied to the particular individual in order to promote shorter solutions. Finally, we adjust the weight between w (number of wrong *oriented* edges) and c (number of moves applied to current Cube individual). This will be done similarly in all subsequent phases.

$$phase_0 = 5 \cdot (2w) + c \tag{7}$$

With a total of 12 edges which can all have the wrong orientation this gives $max\{2w\} = 24$. The Cube has been successfully put into G_1 when $phase_0 = c$. Reaching G_1 is fairly easy to accomplish, thus making the weight-factor 5 a good choice.

$\mathbf{G_1 \rightarrow G_2}$ In order to fulfill G_2 the 8 corners have to be oriented correctly. Edges that belong in the middle layer get transferred there. Tests with the Thistlethwaite ES showed it somewhat problematic to do this in one step. Oftentimes, the algorithm would get stuck in local optima. To solve this, the process of transferring a Cube from G_1 to G_2 has been divided into two parts. First, edges that belong into the middle layer are transferred there. Second, the corners are oriented the right way. The first part is fairly easy and the fitness function is similar to that from $phase_0$ except for w (number of wrong *positioned* edges), i.e. edges that should be in the middle layer but are not.

$$phase_1 = 5 \cdot (2w) + c \tag{8}$$

In the second part, for each wrong positioned corner, 4 penalty points are assigned as they are more complex to correct than edges. Obviously, in order to put the Cube from G_1 to G_2 both phases described here have to be fulfilled, which yields:

$$phase_2 = 10 \cdot (4v) + phase_1 \tag{9}$$

where v represents the number of wrong *oriented* corners. The weighing factor is increased from 5 to 10 to promote a successful transformation into G_2 over a short sequence of moves.

$\mathbf{G_2 \rightarrow G_3}$ We now have to put the remaining 8 edges in their correct orbit. The same is done for the 8 corners which also need to be aligned the right way. Thus, the colors of two adjacent corners in one circuit have to match on two faces. In G_3 the Cube will only have opposite colors on each face. Let x (number of wrong *colored* facelets) and y (number of wrong *aligned* corners), then

$$phase_3 = 5 \cdot (x + 2 \cdot y) + c \, . \tag{10}$$

$\mathbf{G_3} \rightarrow \mathbf{G_4}$(solved) The Cube can now be solved by only using half-turns. For the fitness function we simply count wrong colored facelets. Let z be the number of wrong colored facelets, then

$$phase_4 = 5 \cdot z + c\ . \tag{11}$$

To summarize, 5 different fitness functions are needed for the Thistlethwaite ES. $phase_i$ is solved if $phase_i = c$, $i = 0, ..., 4$ and with the properties of nested groups we can conclude, given the above, a solved Cube implies:

$$\sum_{0}^{4} phase_i = c\ . \tag{12}$$

Fulfilling the above equation satisfies the constraints induced by the groups $G_0, \ldots, G_4$, with the final fitness value c describing the final solution sequence length. The weight factors chosen are based on consecutive testing throughout development. The ratio is dictated by the size of the nested groups. Finding optimal weights presents a seperate optimization problem and may be subject to future work.

5.4 Mutation Operators

The mutation operators are dictated by the subgroups used. Conveniently, the maximum sequence length (s) needed to transform the Cube from one subgroup to another is given by Thistlethwaite [13]. Those lengths are 7,13,15,17 (the sum of which is 52, hence "52 Move Strategy") for each group transition respectively. An individual in phase i is mutated by:

1. generating a random length (l) with $0 \leq l \leq s$, according to i ($i = 0 \rightarrow s = 7, i = 1 \rightarrow s = 13, i = 2, 3 \rightarrow s = 15, i = 4 \rightarrow s = 17$)
2. concatinating l random single moves from the according group G_i
3. applying this sequence to the current Cube individual

For example: Let $i = 2$ (transitioning from $G_2 \rightarrow G_3$). The maximum sequence length for this step is $s = 15$. Let random $l = 4, (0 \leq 4 \leq 15)$. Next, we chose a random single move from G_2, repeat this a total of 4 times and concatinate these to form a sequence. Let those 4 single moves be $D, F2, R2, U$. This results in the sequence $DF2R2U$ representing the present mutation which is applied to the current Cube individual.

In case of $l = 0$ the mutation is an empty sequence, leaving the current individual untouched. Given an appropriate fitness, this allows distinct Cubes to survive multiple generations.

5.5 Selection Method

The selection method used is a modified truncation selection. The selection pool is generated by chosing the μ best individuals from the current population. Therefrom, each individual is duplicated with the same probability $\frac{1}{\mu}$, repeated λ times to form the new population. This approach implicates a low selection pressure and hence favors a higher diversity, as it is a key importance to enable the survival of alleged suboptimal individuals. The combination of random mutation operators, phase transitions and redundant move removal can result in an abrupt fitness improvement of such individuals.

Furthermore, similar to Kociemba's key idea of continuing calculation after some solution to one phase has been found [7], our ES continues until μ *different* such individuals have been evolved. These are then duplicated as described above to form the inital population for the subsequent phase. Put simply, $phase_{i+1}$ starts with a population pool of λ $phase_i$-solving Cubes of high diversity. This can have a positive effect on overall solution length and calculation time, as remarked by Kociemba [7] and counters the major flaw of the classic TWA mentioned in section 4.

6 Benchmarks

To provide a brief performance overview 100 random scrambles of minimum length 10 and maximum length 50 were generated and and solved in 5 repetitions. Solution lengths and calculation time are of particular interest to us. The test was conducted with the TWES using $(\mu, \lambda) = (1000, 50000)$, weighing factors $(5, 5, 5, 5, 5)$, mutation lengths $(5, 5, 13, 15, 17)$ and maximum generations before reset (250).

Table 1. 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

As seen in Table 1, the solution sequences hit an average of about 50 single moves, further demonstrating a consistent performance throughout the repetitions. Most scrambles are solved in 35-45 moves, outliers are responsible for the higher average count. Extensive additional benchmarks can be found in [3].

7 Conclusion

The benchmarks are promising, yielding comparable results to the classic TWA. Outliers calculated by TWES provide both significantly shorter and longer solutions. This is most probably due to inter-group dependencies and future focus lies on increasing our TWES' tendency to such shorter results. Instead of obtaining static solutions dictated by the lookup-table used in the classic TWA, the dynamic evolution process enables those shorter solution sequences not previously possible.

Regarding the Rubik's Cube optimization problem, our evolutionary approach is evidently competitive with the exact method it adepts. As this was the first such attempt - based on the first group theoretic exact approach using lookup-tables (Thistlethwaite) - future work promises further improvement. This algorithm only solves the classic 3^3 Rubik's Cube, just as the exact method it is based on does. However, our modular EA can also be used to solve higher dimensional Rubik's Cubes by appropriately substituting the current fitness functions.

The next developmental step will adept approaches that reduce the number of subgroups to 3 and then 2, potentially yielding further improvement in solution sequence length. Conveniently, our implementation already provides such possibilities for extensions, enabling quick testing of different subgroup combinations.

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. Boyzejko, W., Wodecki, M.: A Hybrid Evolutionary Algorithm for some Discrete Optimization Problems. In: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications, pp. 326–331. IEEE Computer Society, Washington (2005)
3. El-Sourani, N.: Design and Benchmark of different Evolutionary Approaches to Solve the Rubiks Cube as a Discrete Optimization Problem. Diploma Thesis, WWU Muenster, Germany (2009)
4. Florios, K., Mavrotas, G., Diakoulaki, D.: Solving Multiobjective, Multiconstraint Knapsack Problems Using Mathematical Programming and Evolutionary Algorithms. European Journal of Operational Research 203, 14–21 (2009)
5. Frey, A., Singmaster, D.: Handbook of Cubic Math. Enslow, Hillside (1982)
6. Herdy, M., Patone, G.: Evolution Strategy in Action, 10 ES-Demonstrations. Technical Report, International Conference on Evolutionary Computation (1994)
7. Kociemba, H.: Cube Explorer, `http://kociemba.org/Cube.htm`
8. Muehlenbein, H., Mahnig, T.: FDA - A Scalable Evolutionary Algorithm for the Optimization of Additively Decomposed Functions. Evol. Comput. 7, 353–376 (1999)
9. Reid, M.: Cube Lovers Mailing List, `http://www.math.rwth-aachen.de/~Martin.Schoenert/Cube-Lovers/Index_by_Author.html`
10. Rokicki, T.: Twenty-Five Moves Suffice for Rubik's Cube, `http://Cubezzz.homelinux.org/drupal/?q=node/view/121`
11. Rokicki, T.: `http://www.newscientist.com/article/mg19926681.800-cracking-the-last-mystery-of-the-rubiks-Cube.html`
12. Singmaster, D.: Notes on Rubik's Magic Cube. Enslow, Hillside (1981)
13. Thistlethwaite, M.B.: The 45-52 Move Strategy. London CL VIII (1981)
14. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Penn State (1999)