Solving Full NxNxN Rubik's Supercube Using Genetic Algorithm

Robert Świta (robert.swita@wp.pl)
Koszalin University of Technology https://orcid.org/0000-0002-3551-295X
Zbigniew Suszyński
Koszalin University of Technology: Politechnika Koszalinska

Research Article

Keywords: Rubik Cube, Genetic Algorithm, 3D orientation, Euler angles, cubie clusters, macro moves

Posted Date: March 22nd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-885980/v1

Solving Full NxNxN Rubik's Supercube using Genetic Algorithm

Ph.D. Robert Świta [1*], Prof. Zbigniew Suszyński [1]

[1] Faculty of Electronics and Informatics, Koszalin University of Technology, Koszalin, POLAND
[*]robert.swita@wp.pl

Abstract: This article presents an algorithm using an evolutionary approach to the problem of solving a full Rubik's NxNxN supercube i.e. orienting all cubies, including internal ones, not only by their face colors, but to the same orientation in 3D space. The problem is formally defined by a matrix representation using affine transformations of cubies. The solving strategy consists of series of genetic algorithms which try to find better cube configuration from the current one. After finding one, moves are executed, changing current configuration. This strategy is repeated until solving the cube. Genetic algorithm restricts movements to the current cluster, solving the cube in stages from the center of the cube to the outside. Movements which solve clusters are saved in the form of macros and used for training and acceleration of the algorithm. The goal of the algorithm is to minimize solving time, not the count of moves.

Keywords: Rubik Cube, Genetic Algorithm, 3D orientation, Euler angles, cubie clusters, macro moves.

1. Problem

The problem under consideration concerns solving the full N-segment Rubik's N×N×N supercube, including the internal cubies, using the genetic algorithm. The problem posed is much more general than solving of traditional cubes due not only the need to correct orientation of the off-border cubies (and not only to the appropriate face color), but also to orient the inner cubies. To our knowledge such structures, though being natural generalizations of Rubik's Cube, were not yet investigated.

The traditional cube has three-layers, it does not contain inner cubies and the orientation of the middle cubies is usually not important. The first algorithms of its solving were created in the early 1980s, and the most popular of them is Morwen B. Thistlethwaite's [1], based on the division of the task into 4 stages, in which the possibilities of movement are gradually restricted. These restrictions apply to the possible angles of rotation for certain segments to only 180°. In the last step, all segments are subject to this limitation. For cubes larger than traditional ones, it is more important to limit the movements of the increasing number of segments, but dividing the task into stages with constraints on the movements is also applicable in this scenario.

The problem of arranging a traditional cube with the use of the shortest possible sequence of movements is often solved by algorithms for iterative A* searching of the state tree down to a specific depth (eg IDA *). A typical example is Kociemba's algorithm [2], which also breaks the problem into substages, but uses state tree search and heuristic functions. The most common approach, however, is to use special macro-movement sequences (macros) that change the orientation of a small number of cubies, smaller than that resulting from a single movement [3]. Recent research has allowed to determine the minimum number of moves necessary to arrange N×N×N cubes without internal cubies from any configuration to $O(N^2/\log N)$, although the optimal algorithm is not yet known [4] [5]. Already in 1994 M. Herdy solved the cube using a genetic algorithm that used macro movements as genetic operators [6]. Later studies tried to develop strategies without using expert knowledge [7] [8] or to use already known heuristic algorithms [9] [10].

The aim of the presented work was to develop the most effective and stable algorithm to solve the problem in the shortest possible time, regardless of the number of movements made. With the problem posed in this way, the presented solution is based on the execution of a sequence of genetic algorithms improving the configuration of the cube in accordance with the adopted evaluation function and defined genetic operators. The solution found by the genetic algorithm is used by executing movements written in the chromosome of the best individual only if it improves the cube configuration, otherwise the attempt is repeated. Algorithm solving the cube is divided into stages, and the sequences that complete the stages are memorized and used for teaching the algorithm.

2. Cubies orientation

Rubik's N×N×N Cube has been defined in a coordinate system whose center coincides with geometric center of the cube. Each cubie of a cube is related to its local coordinate system defined in relation to the parent system of the entire cube. Movements of cube slices, in such defined system, results in additional rotations R of the cubies around their selected *axis*. Each cubie from the slice changes its origin position from p to p' accordind to the linear formula:

$$p' = Rp \tag{1}$$

After transformation, cubies swap their position and change orientation, therefore configuration of the cube can be represented as a 3D array of cubies composite transformations. All cubies that can occupy the same position as a result of rotation belong to the same set, called a cluster.

Positions p of the cubies don't have to be stored in the matrix, because they are dependent on their indices v in the 3D matrix. Since the indices are always non-negative in the range $[0: N-1]$, the cubie coordinates p can be defined e.g. by shifting their indices v by the indices of the cube center C, so that a shifted center will move to the origin of the cube coordinate system:

$$p = v - C$$
$$p' = v' - C$$

where: $C = \left[\frac{N-1}{2}, \frac{N-1}{2}, \frac{N-1}{2}\right]$

Positions p would be then origins of the cubies in the range $\left[-\frac{N-1}{2}, \frac{N-1}{2}\right]$ in each dimension, with an offset step of 1 from one another. The matrix of their transformation is then a combination of the translation to the center of the cube's system, the orthogonal rotation matrix and invert of this translation:

$$v' = R(v - C) + C \tag{2}$$

The orientation and position of the cubies are thus clearly defined by the hitherto position of the cubie p and the rotation matrix R, which can be expressed as the composition of basic transformation matrices of the cube slices rotation in the planes of its system, around the X, Y or Z axis by 90°. In column notation, these matrices are defined by:

$$R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}, R_Y = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}, R_Z = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Performing successive rotations in the same axis results in the combination of the same transformations, i.e. the exponentiation of these matrices. Obviously their fourth powers are identity matrices (full angle rotations), hence:

$$R_{axis}^{\varphi} = R_{axis}^{\varphi \pm 4}.$$

Since four orientations with the face in front are possible for each face of the cube (four multiples of the angle of 90° around the vector perpendicular to the face), the total number of possible cubie orientations (as well as the maximum number of different positions of the cubie p' or the maximum number of cubies in a cluster) is 24. Therefore, these orientations can be written shortly as a 5-bit number, but the most convenient way is to define it by using Euler angles, presenting rotation R as a combination of elementary rotations around the cube axis in the order X, Y and Z respectively, by the angles $[\alpha', \beta', \gamma'] = \frac{\pi}{2}[\alpha, \beta, \gamma]$, where $\alpha, \beta, \gamma = 0{:}3$.

$$R = R_Z^{\gamma} R_Y^{\beta} R_X^{\alpha} = \tag{3}$$

$$\begin{bmatrix} \cos\gamma' & -\sin\gamma' & 0 \\ \sin\gamma' & \cos\gamma' & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta' & 0 & \sin\beta' \\ 0 & 1 & 0 \\ -\sin\beta' & 0 & \cos\beta' \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha' & -\sin\alpha' \\ 0 & \sin\alpha' & \cos\alpha' \end{bmatrix}$$

The determination of orientation by means of Euler angles is not unequivocal. Different combinations of angles may specify the same orientations. For any rotation Q, a conjugation, like:

$$Q^{-1} R_{axis} Q = R_{axis'} \tag{4}$$

changes only rotation axis from $axis$ to $axis' = Q^{-1}axis$. An real eigenvector $axis'$ corresponds to the eigenvalue λ = 1 and doesn't change its direction during rotation $Q^{-1}R_{axis}Q$ and hence it's an axis of this rotation:

$$Q^{-1} R_{axis} Q \cdot Q^{-1}axis = Q^{-1}axis$$

Using as a Q, e.g. 90° rotation about the X axis, changes rotation axis from the Z axis to the Y axis, because:

$$R_X^{-1}[0 \quad 0 \quad 1]^T = [0 \quad 1 \quad 0]^T$$

Therefore, the following equations are true:

$$\begin{gathered} R_Y^{-1} R_X^{\gamma} R_Y = R_Z^{\gamma} \\ R_X^{-1} R_Z^{\beta} R_X = R_Y^{\beta} \\ R_Y^{-2} R_X^{\gamma} R_Y^{2} = R_{-X}^{\gamma} = R_X^{-\gamma} \\ R_Z^{-2} R_X^{\gamma} R_Z^{2} = R_{-X}^{\gamma} = R_X^{-\gamma} \end{gathered} \tag{5}$$

For α, β, γ taking values in the range [0: 3], all possible cubie orientations can be obtained with just two movements, i.e. one of the angles can be considered to be equal zero:

$$\begin{gathered} R_Z^{0} R_Y^{\beta} R_X^{\alpha} = \begin{bmatrix} \boldsymbol{\cos\beta'} & \sin\alpha'\sin\beta' & \cos\alpha'\sin\beta' \\ \boldsymbol{0} & \boldsymbol{\cos\alpha'} & \boldsymbol{-\sin\alpha'} \\ \boldsymbol{-\sin\beta'} & \sin\alpha'\cos\beta' & \cos\alpha'\cos\beta' \end{bmatrix} \\ R_Z^{\gamma} R_Y^{0} R_X^{\alpha} = \begin{bmatrix} \boldsymbol{\cos\gamma'} & -\sin\gamma'\cos\alpha' & \sin\gamma'\sin\alpha' \\ \boldsymbol{\sin\gamma'} & \cos\gamma'\cos\alpha' & -\cos\gamma'\sin\alpha' \\ \boldsymbol{0} & \boldsymbol{\sin\alpha'} & \boldsymbol{\cos\alpha'} \end{bmatrix} \\ R_Z^{\gamma} R_Y^{\beta} R_X^{0} = \begin{bmatrix} \cos\beta'\cos\gamma' & \boldsymbol{-\sin\gamma'} & \sin\beta'\cos\gamma' \\ \cos\beta'\sin\gamma' & \boldsymbol{\cos\gamma'} & \sin\beta'\sin\gamma' \\ \boldsymbol{-\sin\beta'} & \boldsymbol{0} & \boldsymbol{\cos\beta'} \end{bmatrix} \end{gathered} \tag{6}$$

Which orientation definition can be used, may be determined e.g. by the position of the zero element in the rotation matrix (in bold in eq.6). Due to the adopted order of transformations, rotations in the form $R_Z^{\gamma} R_X^{\alpha}$ and $R_Y^{\beta} R_X^{\alpha}$, or $R_Z^{\gamma} R_X^{\alpha}$ and $R_Z^{\gamma} R_Y^{\beta}$ are enough to determine any orientation. We chose the first option. Using eq.5, it can be proved, that for every possible value of β, the matrix R can be represented as a combination of just such rotations:

$$\begin{aligned} & R(\beta = 0) = \boldsymbol{R_Z^{\gamma} R_X^{\alpha}} \\ & R(\beta = 1) = R_Z^{\gamma} R_Y R_X^{\alpha} = R_Y^{-1} R_X^{\gamma} R_Y^{2} R_X^{\alpha} = \\ & = R_Y^{-1}\left(R_X^{\gamma} R_Y^{2}\right) R_X^{\alpha} = R_Y^{-1}\left(R_Y^{2} R_X^{-\gamma}\right) R_X^{\alpha} = \boldsymbol{R_Y R_X^{\alpha-\gamma}} \\ & R(\beta = 2) = R_Z^{\gamma} R_Y^{2} R_X^{\alpha} = R_Z^{\gamma}(R_X^{-1} R_Z^{2} R_X) R_X^{\alpha} = \\ & = R_Z^{\gamma}(R_Z^{-2} R_X) R_X^{\alpha+1} = \boldsymbol{R_Z^{\gamma\pm2} R_X^{\alpha\pm2}} \\ & R(\beta = 3) = R_Z^{\gamma} R_Y^{3} R_X^{\alpha} = \left(R_Y^{-1} R_X^{\gamma} R_Y\right) R_Y^{3} R_X^{\alpha} = \boldsymbol{R_Y^{3} R_X^{\alpha+\gamma}} \end{aligned} \tag{7}$$

Since the angles are multiples of 90°, it is easy to determine them from the signs of the sine and cosine functions, which are elements of the rotation matrix. For $\varphi = \alpha, \beta, \gamma$, if:

$$\begin{gathered} \cos\varphi' > 0 \Rightarrow \varphi = 0 \\ \sin\varphi' > 0 \Rightarrow \varphi = 1 \\ \cos\varphi' < 0 \Rightarrow \varphi = 2 \\ \sin\varphi' < 0 \Rightarrow \varphi = 3 \end{gathered} \tag{8}$$

From the point of view of the movements performed, the more favorable orientations are those that are composed of fewer of them. An unarranged cubie has an orientation of only one or two movements. The orientations obtained as a result of a single movement must have one positive value in the matrix (equal to one in the absence of cubie scaling) on its diagonal. This number can also be determined directly from the values of γ, β, α. Accurate orientation information is also important as it will allow evaluation function to recognize and remember cube configurations. The cubie's orientation vector, briefly referred to as its state, was encoded on 8 bits. The first two bits define the number of moves, and the remaining ones, the numbers of turns γ, β, α respectively (also 2 bits each, see Table 1):

$$\begin{gathered} moveCount = sign(\gamma) + sign(\beta) + sign(\alpha) \\ state = moveCount \ll 6 \mid \gamma \ll 4 \mid \beta \ll 2 \mid \alpha \end{gathered} \tag{9}$$

Orientation $R_Z^{2} R_X^{2}$ can be more preferably achieved with one move. In this case it's worth to change it's definition to R_Y^{2}.

Table 1 presents possible cubie orientations in the form of its rotation matrix. For any cube configuration, this matrix can be recreated on the basis of a 3x3 sub-matrix of cubie's affine transformations - without taking into account its position (translation). Because each cubie is also scaled by positive valued cubieScale (Figure 1), only meaningful signs of matrix elements are presented.

The cube config can be saved as the orientation table of all cubies. Due to the small number of orientations of a single cubie (i.e. 24), they can be encoded as alphanumeric characters and the cube configuration as a string. This makes it easy to compare different cube configurations.

Table 1. Possible orientations of the cubie in the form of Euler angles $(\alpha', \beta', \gamma')$, matrix R, coded *state* and cubie positions p' after transformation by rotation R

(γ', β')	$(0, \mathbf{0})$	$(90°, \mathbf{0})$	$(180°, \mathbf{0})$	$(270°, \mathbf{0})$	$(\mathbf{0}, 90°)$	$(\mathbf{0}, 270°)$
$\alpha' = 0$ R $state$ p'	$(0, 0, 0)$ I $\begin{bmatrix} + & 0 & 0 \\ 0 & + & 0 \\ \mathbf{0} & 0 & + \end{bmatrix}$ 00000000 $[x, y, z]$	$(90°, 0, 0)$ R_z^1 $\begin{bmatrix} 0 & - & 0 \\ + & 0 & 0 \\ \mathbf{0} & 0 & + \end{bmatrix}$ 01010000 $[-y, x, z]$	$(180°, 0, 0)$ R_z^2 $\begin{bmatrix} - & 0 & 0 \\ 0 & - & 0 \\ \mathbf{0} & 0 & + \end{bmatrix}$ 01100000 $[-x, -y, z]$	$(270°, 0, 0)$ R_z^3 $\begin{bmatrix} 0 & + & 0 \\ - & 0 & 0 \\ \mathbf{0} & 0 & + \end{bmatrix}$ 01110000 $[y, -x, z]$	$(0, 90°, 0)$ R_Y^1 $\begin{bmatrix} 0 & 0 & + \\ \mathbf{0} & + & 0 \\ - & 0 & 0 \end{bmatrix}$ 01000100 $[z, y, -x]$	$(0, 270°, 0)$ R_Y^3 $\begin{bmatrix} 0 & 0 & - \\ \mathbf{0} & + & 0 \\ + & 0 & 0 \end{bmatrix}$ 01001100 $[-z, y, x]$
$\alpha' = 90°$ R $state$ p'	$(0, 0, 90°)$ R_X^1 $\begin{bmatrix} + & 0 & 0 \\ 0 & 0 & - \\ \mathbf{0} & + & 0 \end{bmatrix}$ 01000001 $[x, -z, y]$	$(90°, 0, 90°)$ $R_z^1 R_X^1$ $\begin{bmatrix} 0 & 0 & + \\ + & 0 & 0 \\ \mathbf{0} & + & 0 \end{bmatrix}$ 10010001 $[z, x, y]$	$(180°, 0, 90°)$ $R_z^2 R_X^1$ $\begin{bmatrix} - & 0 & 0 \\ 0 & 0 & + \\ \mathbf{0} & + & 0 \end{bmatrix}$ 10100001 $[-x, z, y]$	$(270°, 0, 90°)$ $R_z^3 R_X^1$ $\begin{bmatrix} 0 & 0 & - \\ - & 0 & 0 \\ \mathbf{0} & + & 0 \end{bmatrix}$ 10110001 $[-z, -x, y]$	$(0, 90°, 90°)$ $R_Y^1 R_X^1$ $\begin{bmatrix} 0 & + & 0 \\ \mathbf{0} & 0 & - \\ - & 0 & 0 \end{bmatrix}$ 10000101 $[y, -z, -x]$	$(0, 270°, 90°)$ $R_Y^3 R_X^1$ $\begin{bmatrix} 0 & - & 0 \\ \mathbf{0} & 0 & - \\ + & 0 & 0 \end{bmatrix}$ 10001101 $[-y, -z, x]$
$\alpha' = 180°$ R $state$ p'	$(0, 0, 180°)$ R_X^2 $\begin{bmatrix} + & 0 & 0 \\ 0 & - & 0 \\ \mathbf{0} & 0 & - \end{bmatrix}$ 01000010 $[x, -y, -z]$	$(90°, 0, 180°)$ $R_z^1 R_X^2$ $\begin{bmatrix} 0 & + & 0 \\ + & 0 & 0 \\ \mathbf{0} & 0 & - \end{bmatrix}$ 10010010 $[y, x, -z]$	$(180°, 0, 180°)$ $R_z^2 R_X^2 = R_Y^2$ $\begin{bmatrix} - & 0 & 0 \\ 0 & + & 0 \\ \mathbf{0} & 0 & - \end{bmatrix}$ 01001000 $[-x, y, -z]$	$(270°, 0, 180°)$ $R_z^3 R_X^2$ $\begin{bmatrix} 0 & - & 0 \\ - & 0 & 0 \\ \mathbf{0} & 0 & - \end{bmatrix}$ 10110010 $[-y, -x, -z]$	$(0, 90°, 180°)$ $R_Y^1 R_X^2$ $\begin{bmatrix} 0 & 0 & - \\ \mathbf{0} & - & 0 \\ - & 0 & 0 \end{bmatrix}$ 10000110 $[-z, -y, -x]$	$(0, 270°, 180°)$ $R_Y^3 R_X^2$ $\begin{bmatrix} 0 & 0 & + \\ \mathbf{0} & - & 0 \\ + & 0 & 0 \end{bmatrix}$ 10001110 $[z, -y, x]$
$\alpha' = 270°$ R $state$ p'	$(0, 0, 270°)$ R_X^3 $\begin{bmatrix} + & 0 & 0 \\ 0 & 0 & + \\ \mathbf{0} & - & 0 \end{bmatrix}$ 01000011 $[x, z, -y]$	$(90°, 0, 270°)$ $R_z^1 R_X^3$ $\begin{bmatrix} 0 & 0 & - \\ + & 0 & 0 \\ \mathbf{0} & - & 0 \end{bmatrix}$ 10010011 $[-z, x, -y]$	$(180°, 0, 270°)$ $R_z^2 R_X^3$ $\begin{bmatrix} - & 0 & 0 \\ 0 & 0 & - \\ \mathbf{0} & - & 0 \end{bmatrix}$ 10100011 $[-x, -z, -y]$	$(270°, 0, 270°)$ $R_z^3 R_X^3$ $\begin{bmatrix} 0 & 0 & + \\ - & 0 & 0 \\ \mathbf{0} & - & 0 \end{bmatrix}$ 10110011 $[z, -x, -y]$	$(0, 90°, 270°)$ $R_Y^1 R_X^3$ $\begin{bmatrix} 0 & - & 0 \\ \mathbf{0} & 0 & + \\ - & 0 & 0 \end{bmatrix}$ 10000111 $[-y, z, -x]$	$(0, 270°, 270°)$ $R_Y^3 R_X^3$ $\begin{bmatrix} 0 & + & 0 \\ \mathbf{0} & 0 & + \\ + & 0 & 0 \end{bmatrix}$ 10001111 $[y, z, x]$

For a cubie $p = [x, y, z]$, the remaining cubies in the cluster are p' cubies with coordinates Rp given in Table 1. Solving the cube corresponds to the situation when for all cubies their rotation matrices are identity matrices - if the cubie is correctly oriented, it must be in the correct position.

Since the orientation of the inner cubies is important and the arrangement of the cubies begins from its center, all cubies are separated from each other by a certain constant spacing and scrambled cubies are presented as partially transparent (Figure 1).

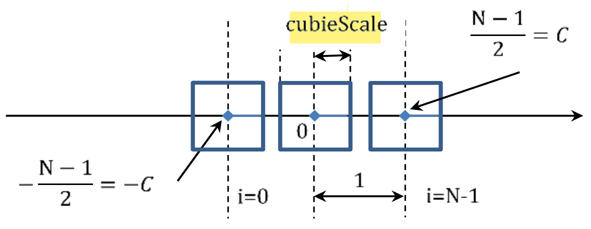

Figure 1. Arrangement of cubies in the cube coordinate system

3. Algorithm

Cube movements were defined by means of three parameters: rotation *axis* X, Y and Z [0:2], index of the *slice* [0:N-1] and the number of 90° slice rotations *angle* [0:2]. The number of different possible moves is therefore equal to 9N. Movements are coded to the values of integer, linear indices in the range [0:9N-1] and thus represented as genes in the chromosome containing the solution:

$$move(slice, axis, angle) = 9slice + 3axis + angle \quad (10)$$

The gene values can then also be easily decoded into the these 3 parameters, determining movement.

The cube configuration can be represented as a block vector $\mathbb{p}$, elements of which are the cubies rotation (orientation) matrices. For a single movement, permutation of the cubie p in the block vector can then be described as:

$$\mathbb{p}'(Mp) = M\mathbb{p}(p) \quad (11)$$

where the matrix M is the transformation of the rotation by the *angle* around the *axis* for cubies in the *slice*, and the identity matrix for the others:

$$M = \begin{cases} R_{axis}^{angle+1}, & for\ p(axis) = slice \\ I, & otherwise \end{cases} \quad (12)$$

Since all cubes in the *slice* segment are rotated (permutated) in the cube vector, the cubie's transformation can be determined based on the M transformation combined with the transformation stored in the cube vector at the position from inversed transformation performed on that position:

$$\mathbb{p}'(p) = M\mathbb{p}(M^{-1}p) \tag{13}$$

By saving the cube configuration as a vector, it is also possible to define a transformation that performs any permutation of the elements of this vector (cubies in a cube), composite with the M transformation and present the movement of the cubie slice as a matrix operation:

$$\mathbb{p}' = \mathbb{R}_{axis,slice}^{angle+1}\mathbb{p} \tag{14}$$

Matrix $\mathbb{R}$ has the size $N^3 \times N^3$ and its elements are the M matrices of the cubies. There is one nonzero M transformation at the position specified by undoing the M transformation at the original cubie position and converting it to a linear index in each row and column. Knowing the current cubies orientations, it is possible to construct such a matrix $\mathbb{R}$, which is a composite of all previous cubie's transformations and represents cube configuration.

For example, for a cube with size N = 2 and unfolding to the vector $\mathbb{p} = [\mathbb{p}_{XYZ}]$, the single rotations of the $slice = 0$ around the Z and Y axes are represented by matrices (dots represent zero matrices):

$$\mathbb{R}_{Z,0} = \left[\begin{array}{cccc|cccc} \cdot & \cdot & R_Z & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & I & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & R_Z & \cdot \\ \cdot & \cdot & \cdot & I & \cdot & \cdot & \cdot & \cdot \\ \hdashline R_Z & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & I & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & R_Z & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & I \end{array}\right]$$

$$\mathbb{R}_{Y,0} = \left[\begin{array}{cccc|cccc} \cdot & \cdot & \cdot & \cdot & R_Y & \cdot & \cdot & \cdot \\ R_Y & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & I & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & I & \cdot & \cdot & \cdot & \cdot \\ \hdashline \cdot & \cdot & \cdot & \cdot & \cdot & R_Y & \cdot & \cdot \\ \cdot & R_Y & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & I & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & I \end{array}\right],$$

and commutator $\mathbb{K} = \mathbb{R}_{Y,0}^T\mathbb{R}_{Z,0}^T\mathbb{R}_{Y,0}\mathbb{R}_{Z,0}$:

$$\mathbb{K} = \left[\begin{array}{cccc|cccc} \cdot & \cdot & R_Z & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & I & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ R_X & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & I & \cdot & \cdot & \cdot & \cdot \\ \hdashline \cdot & \cdot & \cdot & \cdot & \cdot & R_X & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & R_Y^T & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & I & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & I \end{array}\right]$$

It follows that only 4 cubes changed their transformations, but they swapped their places $\mathbb{p}_{000} - \mathbb{p}_{010}$ and $\mathbb{p}_{100} - \mathbb{p}_{101}$ and are transformed by a single move. Executing a double commutator will result in the absence of a permutation of the positions of these cubies with composite transformation from 2 movements:

$$\mathbb{K}^2 = diag(R_ZR_X, I, R_XR_Z, I, R_XR_Y^T, R_Y^TR_X, I, I) \tag{15}$$

Thanks to this, the movements of commutators can be used to position the corner cubies of this cube.

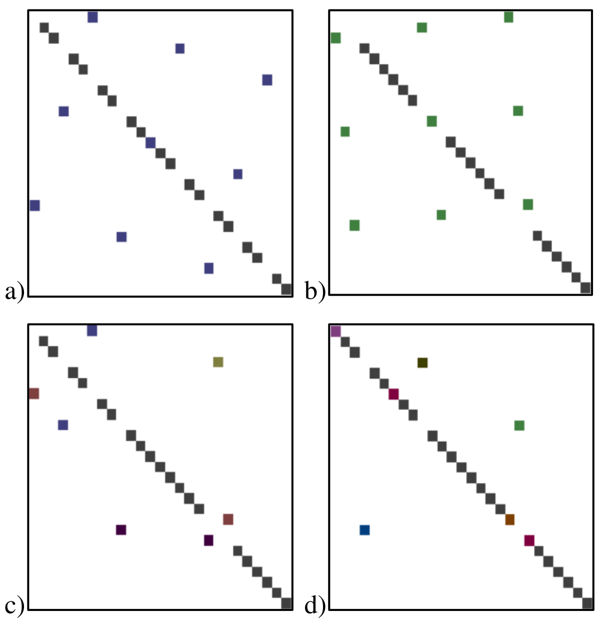

Figure 2. Examples of rotation matrices for a 3x3x3 cube: a) $\mathbb{R}_{Y,0}$ b) $\mathbb{R}_{Y,0}$ c) $\mathbb{K}$, d) $\mathbb{K}^2$

The cube configuration changes through consecutive movements:

$$\mathbb{p}' = \mathbb{R}_{move_k} \dots \mathbb{R}_{move_2}\mathbb{R}_{move_1}\mathbb{p} = \mathbb{R}\mathbb{p} \tag{16}$$

Finding the movements that reverse these transformations, that is, solving the cube, is related to the decomposition of the matrix $\mathbb{R}^T$ into the sequence of the cube movements:

$$\mathbb{p} = \mathbb{R}^T\mathbb{p}' = \mathbb{R}_{move_1}{}^T\mathbb{R}_{move_2}{}^T \dots \mathbb{R}_{move_k}{}^T\mathbb{p}' \tag{17}$$

Each of these transformations is orthogonal to a given segment. As in the one-side Jacobi transform used to derive the SVD of a matrix by means of a Q^TRQ transformation to a similar matrix (maintaining the same eigenvalues), such a sequence in the case of a cube transformation, called rotation conjugate of R, also changes only a limited number of elements. Presented algorithm will also use a strategy similar to Gaussian elimination or “chasing zero” in QR iterations, by solving the cube, cubie by cubie in a specific order, so as not to cause deterioration of the existing configuration.

The number of possible configurations of a cube with N segments can be estimated, by assuming that each cubie can be in any orientation, on:

$$c = 24^{N^3} \tag{18}$$

This number therefore increases exponentially very quickly and the chromosome evaluation function must take into account not only the orientation of the N^3 cubies themselves, but also their position, changed as a result of rotation.

Due to the possibility of repeating the same cubie position coordinates during rotation, there are 6 types of clusters. They differ in the number of segments that can affect the position of cubies of a given cluster type. For each axis and selected angle, the index of the segment influencing the cubie in the cluster can take the value of the cubie coordinate p' after transformation, i.e. the cubie coordinate $p = [x, y, z]$ or the opposite number (Table 1), which gives a maximum of 6 options: $slice = (x, -x, y, -y, z, -z)$.

Apart from the possibility of changing the *axis* and the *angle* of rotation for each segment, the type of the C_k cluster can be determined precisely by the number of k = 1..6 segments affecting it. Figure 3 shows all possible cubies positions from given C_k cluster as red squares in a transparent cube. Vertical lines mark segments for chosen *axis*. If we try to arrange a cubie or cubies only from a given cluster, then the values of genes in the chromosome can be restricted to movements that affect these cubies. The number of such movements for any cube, taking into account the possibility of choosing one of the three axes and one of the three angles, is 9k (maximum 54).

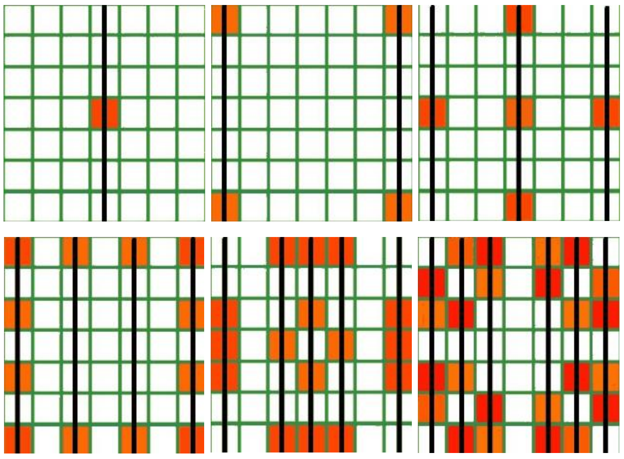
Figure 3. **Six types of clusters C_k k = 1..6, defined by the number of segments that affect those clusters**

Clusters C_2 consist of 8 corner cubies for cube or for inner cubes. For odd-sized cubes, the C_1 cluster constitutes one, middle cube, and the C_3 clusters consist of 6 cubies located in the middle of the cube segments (except for the middle segments) or 12 corner cubies for the middle segments for the cube or inner cubes. The remaining types of clusters contain maximal number of cubies i.e 24 (Figure 4). Even N size cubes do not contain odd cluster types.

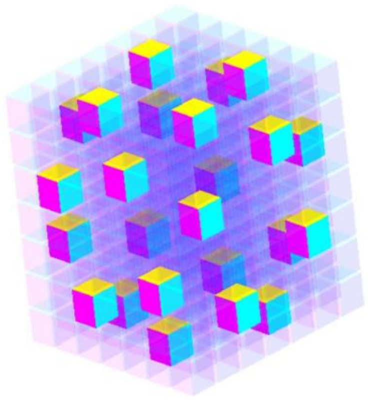
Figure 4. Typical cluster, containing 24 cubies

A set of cubies in a cluster can be created on the basis of a selected cubie *p* by applying to it any possible rotation *R* and obtaining positions (or orientations) of cubies *p'*, which belong to the same cluster (Table 1).

Based on restricted *slice* indexes and any possible value of *axis* and *angle*, list of restricted moves is created. It serves as a genes pool for creatng initial population of chromosomes in genetic algorithm. The chosen genetic operators will not violate this restrictions and whole algorithm will run on restricted list of acceptable moves.

Because in the definition of movement, as the size of a cube N increases, only the number of segments increases, the number of possible moves for a cube should be limited by blocking the movements of specific slices. Additionally, when the cube is rotated, cubies do not change their distance to the cube center. Movements of the slices in planes distant from the center by a distance greater than *r* do not affect the position of cubies in planes less distant. Thus, these cubies can be arranged only by means of slice movements that are distant from the center by less than *r*. The adopted strategy of solving the cube is to orient the cubies one by one, preserving orientation of the cubies previously arranged, whereby selected (active) unoriented cubie is always that, which is closest to the middle of the inner cube of the smallest possible size. This way the algorithm will try to arrange the inner cubies in turn, from the inside to the outside of the cube. The algorithm tries to correctly orientate active cubie by limiting the movements to the movements influencing its cluster. By arranging the cube from the inside, one can gradually limit the movements of the interior slices. The last clusters of the cube (and also inner cubes') are always the clusters containing the corner cubies - most distant from the center. The movements of these clusters are restricted to only movements of the 6 outermost walls of a given cube (Figure 5).

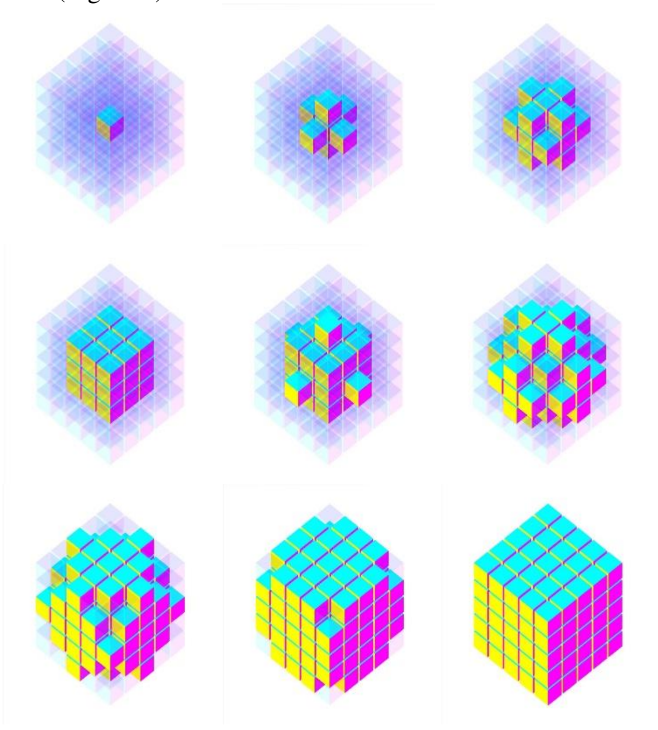
Figure 5. Stages of solving 5x5x5 Cube

4. Fitness function

A chromosome containing a sequence of cube movements may contain movements that can be combined into a single movement or that are completely self-canceling. Prior to the evaluation of the chromosome, a correction procedure can be performed that combines the movements of the same slice of the cube into one movement if they are not separated by movements in different planes, or to remove both genes when movements are canceled.

The fitness function evaluates the cube's alignment status for each sub-sequence of movements in the chromosome starting with the first move. The first movement on a chromosome should change the orientation of the active cubie, changing the number of the movement's *slice* to the segment containing the active cubie, taking into account the axis of rotation specified in the gene. This will increase the probability of finding the active cubie orienting sequence.

First, a copy of the cube is created, which will be subject to movements written in the chromosome. The chromosome genes are processed sequentially. The gene is decoded to the movement and performed on a copy of the cube, which is then evaluated. The sequence of movements is successively supplemented with successive movements defined in the genes of the chromosome, creating an ever longer substring all the way to the entire chromosome. The length of the substring with the best score is remembered and its score becomes the overall chromosome score. In this way, the possibility of finding solutions with a different number of movements using chromosomes of a fixed length has been effectively implemented.

The cube is assessed on the basis of the sum of the assessments of individual cubies belonging to the selected set of cubies. This collection consists of two groups. Group A consists of all cubies belonging to clusters, which were already active cubies of the cube. Position of cubies belonging to this group cannot be changed by the algorithm, therefore they gain a weight greater than the sum of the assessment of all cubies from group B, which are cubies belonging to the current active cubie cluster. The value of 0 is the best fitness of the cube arrangement and individual cubies. The fitness of the X = A, B group is the sum of the scores of non-oriented cubies, each in the range [0, 2) and defined as:

$$f(X) = \sum_{X_i.state \neq 0} 1 + X_i.state/255 \qquad (19)$$

Where X_i denotes orientation of the i-th cubie, encoded as its state (see eq.9 and Table 1.) in A or B group

Cube fitness $error$ is a combination of both group fitnesses:

$$error = f(A)\max\big(f(B)\big) + f(B) \qquad (20)$$

where the maximum value of group B's fitness is the double number of cubies in this group (according to eq.16).

$$\max\big(f(B)\big) = 2 \cdot count(B)$$

By taking into account the exact orientation of the cube, the fitness function causes the genetic algorithm to prefer placing cubies in a certain order, always bringing the cube to similar configurations and reducing the number of cube layouts to a limited number with individual cubies not in their final position. The number of such configurations for k non-oriented cubies in the entire cube is less than:

$$c_k = 24^k \binom{N^3}{k} \qquad (21)$$

After arranging cubies from clusters in groups A and B, the value of the cube fitness function drops to zero. A new active cubie is designated. If the cube is not yet solved, then such a cubie exists and is located at a distance from the center of the cube greater than the cubies in groups A and B. The group B is then joined with A, and the cubies from the active cubie cluster form a new group B. This ends the cube building stage, and the solution that led to it, if not a sequence derived from a saved solution, is stored in the solution file and can be used to speed up the algorithm. The cube's fitness is set to its maximum value and another genetic algorithm is run.

5. Selection, crossover and mutation

An important step in the genetic algorithm is selection. In the case of a Rubik's cube, the chromosomes will often differ in genes at distant positions in the chromosome, which will not affect the value of its fitness function. When selecting winners, a kind of ranking selection can be used, which omits individuals with the value of the fitness function the same as the already selected winner. The advantage of this approach is also low computational complexity.

The most important thing in the algorithm, however, is the use of efficient genetic operators. While the crossover may be a simple one-point recombination of chromosomes, the mutation should allow finding a solution for a cube in a configuration of only a few cubies with the wrong orientation, i.e. macros. Such R' sequences are in the form of a commutator or a conjugation in which the matrix Q can be composed of several movements around different axes:

$$R' = QR_{axis}Q^{-1} \qquad (22)$$

By limiting the movements to the movements that change the active cubie's cluster, it is much easier to find the sequence improving the cube's configuration in accordance with the fitness function, because if the R_{axis} performs rotation on a slice more distant from the center of the cube than the inner slices of the already oriented cube, it is an identity transformation for the cubies of the inner cube. This means, that also the entire R' transformation is an identity transformation for these cubies and does not change their arrangement.

The proposed mutation procedure performs the following actions:

1. Random selection of an idx less than half-length of the chromosome.
2. Genes are processed from the start of the chromosome to $idx - 1$ and inverse transformations to them are coded in genes from index $2idx$ downto $idx + 1$.

Due to the high efficiency of finding macro sequences, the mutation rate in the algorithm can be extremely high, within 100% of the population size.

6. Learning

The sequences found by the genetic algorithm that end with the arrangement of all cubies from the assessed set cause the change of position of only a few, $k \ll N^3$ cubies, not changing orientations of the rest. These are key sequences that can be used not only in the last stage of solving the cube. The algorithm can remember such sequences, implementing the ability to learn and arrange the cube faster and faster.

The memorized sequences are used each time a better cube configuration is found to further improve it. The movements of these sequences are restricted to a certain cluster (maximum 6 segments) and can be mapped to active cubie cluster movements by replacing the segments with active cluster segments. This enables, among others, using the saved cube movements with fewer segments not only to orient the inner cubes, but for all clusters of the same type in the cube.

Movements that, after composing, can be represented in the form of an orthogonal matrix R, can be additionally complemented by a conjugation to the form $Q^{-1}RQ$, in which the matrix Q is a single rotation, selected optimally among all possible movements of the active cluster. This will enable a much wider use of memorized movements, taking into account similar patterns resulting from the cube symmetry.

7. Results

The way the algorithm works can be summarized in four main points:

1. The algorithm arranges one cubie at a time, selecting as an active cubie closest to its middle and contained in the inner cube of the smallest possible size.
2. Gene values in chromosomes (movements) are limited to segments that influence the current cubie cluster, i.e. the current cluster.
3. Only cubies belonging to the current cluster are assessed. Cubies from oriented clusters cannot change orientation.
4. The algorithm has the ability to learn macros, which are the sequence of movements that complete orientation of the current cluster. Macros are mapped to the movements of the current cluster by changing segment indexes to segment indexes of the current cluster.

The results of the algorithm's operation on full cubes with sizes from 2 to 6 are presented. Each cube was scrambled with a 100 random moves before the algorithm was run.

The parameters of the algorithm chose by us assumed the length of the chromosome equal to 27, the number of selection winners equal to 10% of the population, the population itself at the level of 100 individuals and the number of generations at the level of 50 iterations, due to the attempt to find only the cube configuration better than the one found so far. For the duration of the experiment, the ability of the algorithm to learn was suspended. The algorithm has had already known and used approximately 100 macro sequences. Table 2 and Table 3 show the number of genetic algorithms performed and the time it took to complete the cube in 10 consecutive experiments.

Table 2. Number of used genetic algoritms

size	2x2x2	3x3x3	4x4x4	5x5x5	6x6x6
GA1	2	31	99	130	364
GA2	2	21	88	190	459
GA3	4	17	85	175	405
GA4	3	17	44	280	727
GA5	4	16	46	144	493
GA6	3	18	95	134	545
GA7	2	16	84	256	836
GA8	3	13	69	225	776
GA9	2	14	66	184	507
GA10	3	14	112	166	493

Table 3. Calculation time [s]

size	2x2x2	3x3x3	4x4x4	5x5x5	6x6x6
GA1	1	29	204	464	2000
GA2	1	18	181	655	2559
GA3	1	14	169	616	2279
GA4	1	12	72	1025	4181
GA5	1	10	80	481	2789
GA6	1	13	198	425	2933
GA7	1	11	177	897	4755
GA8	1	9	135	776	4400
GA9	1	9	132	657	3089
GA10	1	8	227	580	3205

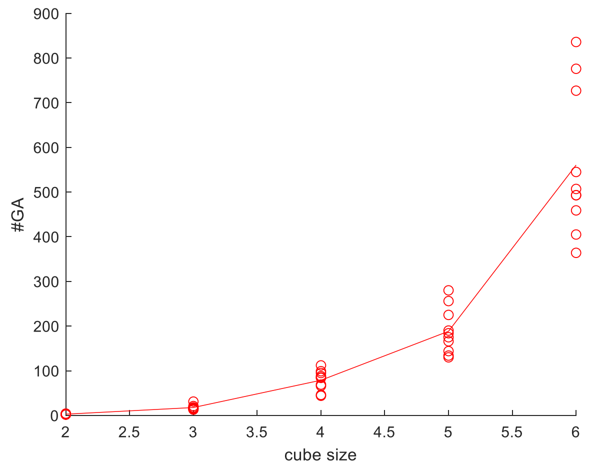

Figure 6. Function of GA count vs cube's size

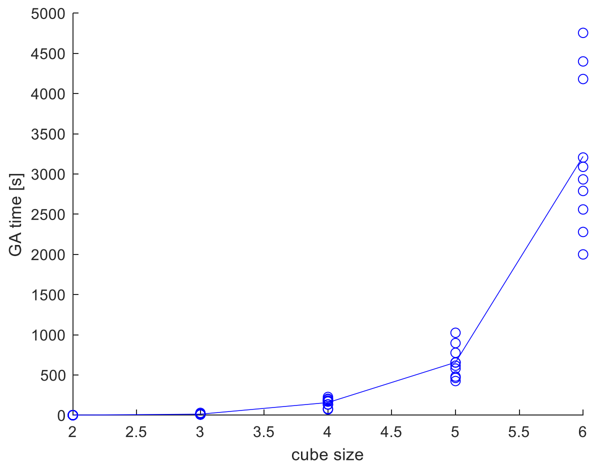

Figure 7. Function of calculation time vs cube's size

The total number of analyzed moves equals approximately to the product of number of genetic algorithms, population count, length of the chromosome and the number of generations (in our case $135E3 \cdot \#GA$), and that is why we tried to minimize number of genetic algorithms (and not number of solution moves).

Table 4 and Fig.8 show number of moves, which led to a solution. Some of them could stand from learned macros.

Table 4. Solutions lengths

size	2x2x2	3x3x3	4x4x4	5x5x5	6x6x6
GA1	75	396	1078	1970	4588
GA2	70	350	913	2239	4089
GA3	100	329	1018	2171	4134
GA4	87	333	935	2299	4116
GA5	87	325	926	2477	4261
GA6	82	363	1056	2239	4088
GA7	43	301	1038	2322	4655
GA8	76	253	976	2054	4734
GA9	68	282	967	1940	4412
GA10	82	265	893	2120	3896

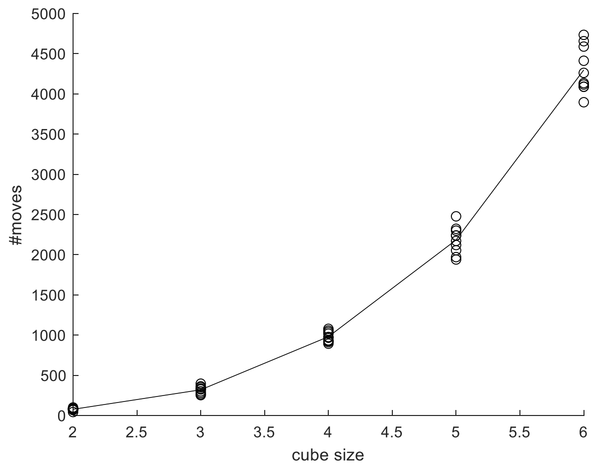

Figure 8. Solution length vs cube's size

8. Conclusions

Presented method deals with a generic problem of solving full Rubik's supercube of any size, taking into account orientation of the off-border and internal cubies. The cube is represented as a model of the composite transformations of the block rotation matrices around X, Y and Z axes. The resulting cube configurations are therefore always correct and it is not necessary to color-map the faces of objects (cubies) and select correct permutations to determine their orientation.

Problem of solving the cube can be formulated as a decomposition of configuration block rotation matrix into orthogonal move block matrices (see eq.17). Such analysis can be treated as a generalization of the SVD decomposition for higher dimensions and similarities of the presented algorithm to this algorithms are therefore highlighted. Solving strategy arranging cubies one by one from the current cluster allows for strong restriction of the possible moves (gene values in GA), which depends on the type of cluster, but is indifferent to the size of the cube (number of segments). The method also doesn't use prior expert knowledge of the existing macro-moves until it finds them on its own.

Rubik's Cube can fascinate, but it is not only a toy, it's also used in data encryption, search algorithms in 3D structures or solving mechanical problems. It is an inspiration to build scientific analogies in the theory of groups, permutations, cycles, but also in 3D graphics. Like fractals, multi-dimensional regular polyhedra or block tensors, it is a gateway to new dimensions and discovering rules and beauty in mathematics.

Declarations:

Funding:
No funds, grants, or other support was received.

Conflicts of interest/Competing interests:
The authors have no relevant financial or non-financial interests to disclose.

Availability of code, data and material:
The code and datasets are available in the GitHub repository, https://github.com/robertswita/RubikCube

Authorship contributions:
RŚ prepared concept, wrote the manuscript, code and conducted experiments. ZS adapted genetic algorithms for solving supercube.

9. References

[1] Thistlethwaite M., *The 52 Move Strategy* (1981), http://www.jaapsch.net/puzzles/thistle.htm. Accessed August 3, 2020.

[2] Herbert Kociemba, *The Two-Phase-Algorithm*, http://kociemba.org/twophase.htm. Accessed August 3, 2020.

[3] Richard E. Korf, *Finding Optimal Solutions to Rubik's Cube Using Pattern Databases*, AAAI-97 Proceedings, pp.700-705 (1997)

[4] Rokicki T., Kociemba H., Davidson M. and Dethridge J., *The Diameter of the Rubik's Cube Group is Twenty*, SIAM J. Discrete Math, Vol. 27, No. 2, pp. 1082–1105 (2013)

[5] Demaine Erik, Demaine Martin, Eisenstat Sarah, Lubiw Anna, Winslow Andrew, *Algorithms for Solving Rubik's Cubes,* Algorithms – ESA 2011. 19th annual European symposium, Saarbrücken, Germany, September 5–9, 2011. Proceedings, pp. 689-700, doi:10.1007/978-3-642-23719-5_58. (2011)

[6] Herdy M., Patone G., *Evolution Strategy in Action, 10 ES-Demonstrations. Technical Report*, International Conference on Evolutionary Computation (1994)

[7] Baum, E. B. and Durdanovic, I. (2000). *Evolution of cooperative problem-solving in an artificial economy*. Neural Computation, 12:2743–2775.

[8] Agostinelli, F., McAleer, S., Shmakov, A. et al. *Solving the Rubik's cube with deep reinforcement learning and search*, Nat Mach Intell 1, 356–363 (2019). https://doi.org/10.1038/s42256-019-0070-z

[9] El-Sourani, N., Hauke, S., and Borschbach, M. (2010). *An evolutionary approach for solving the Rubik's cube incorporating exact methods*. In EvoApplications Part – 1: EvoGames, vol. 6024 of LNCS, pp. 80–89

[10] Borschbach, M., Grelle, C.: *Empirical Benchmarks of a Genetic Algorithm Incorporating Human Strategies. Technical Report*, University of Applied Sciences, Bergisch Gladbach (2009)

Prof. Zbigniew SUSZYŃSKI, Sc. D, Eng.

Head of the Department of Multimedia Systems and Artificial Intelligence in Koszalin University of Technology. Studies and PhD defense - Department of Electron Technology at the Technical University of Kiev. Habilitation colloquium at the Institute of Electron Technology Academy of Sciences in Warsaw. Interests: active thermography, thermal tomography, signal and image processing and the use of genetic algorithms and artificial neural networks.

e-mail: zbigniew.suszynski@tu.koszalin.pl

Ph. D. Robert ŚWITA, Eng.

Graduated from the Faculty of Electronics, Telecommunications and Computer Technologies in Gdańsk University of Technology. PhD degree in Informatics, since 2010 associated with the Faculty of Electronics and Computer Science in Koszalin University of Technology. Adjunct in the Department of Multimedia Systems and Artificial Intelligence. Research interests: linear algebra, algorithms for creating 3D graphics, signal and data processing, multimedia technologies, AI algorithms.

e-mail: robert.swita@wp.pl