Solving the Rubik's Cube Optimally is NP-complete

Erik D. Demaine
MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar St., Cambridge, MA 02139, USA
edemaine@mit.edu

Sarah Eisenstat
MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar St., Cambridge, MA 02139, USA

Mikhail Rudoy[1]
MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar St., Cambridge, MA 02139, USA
mrudoy@gmail.com

Abstract

In this paper, we prove that optimally solving an $n \times n \times n$ Rubik's Cube is NP-complete by reducing from the Hamiltonian Cycle problem in square grid graphs. This improves the previous result that optimally solving an $n \times n \times n$ Rubik's Cube with missing stickers is NP-complete. We prove this result first for the simpler case of the Rubik's Square – an $n \times n \times 1$ generalization of the Rubik's Cube – and then proceed with a similar but more complicated proof for the Rubik's Cube case. Our results hold both when the goal is make the sides monochromatic and when the goal is to put each sticker into a specific location.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Combinatorial Puzzles, NP-hardness, Group Theory, Hamiltonicity

Digital Object Identifier 10.4230/LIPIcs.STACS.2018.24

Related Version A full version of the paper is available [4], `https://arXiv.org/abs/1706.06708`.

1 Introduction

The Rubik's Cube is an iconic puzzle in which the goal is to rearrange the stickers on the outside of a $3 \times 3 \times 3$ cube so as to make each face monochromatic by rotating $1 \times 3 \times 3$ (or $3 \times 1 \times 3$ or $3 \times 3 \times 1$) slices. In some versions where the faces show pictures instead of colors, the goal is to put each sticker into a specific location. The $3 \times 3 \times 3$ Rubik's Cube can be generalized to an $n \times n \times n$ cube in which a single move is a rotation of a $1 \times n \times n$ slice. We can also consider the generalization to an $n \times n \times 1$ figure. In this simpler puzzle, called the $n \times n$ Rubik's Square, the allowed moves are flips of $n \times 1 \times 1$ rows or $1 \times n \times 1$ columns. These two generalizations were introduced in [3].

The overall purpose of this paper is to address the computational difficulty of optimally solving these puzzles. In particular, consider the decision problem which asks for a given

[1] Now at Google.

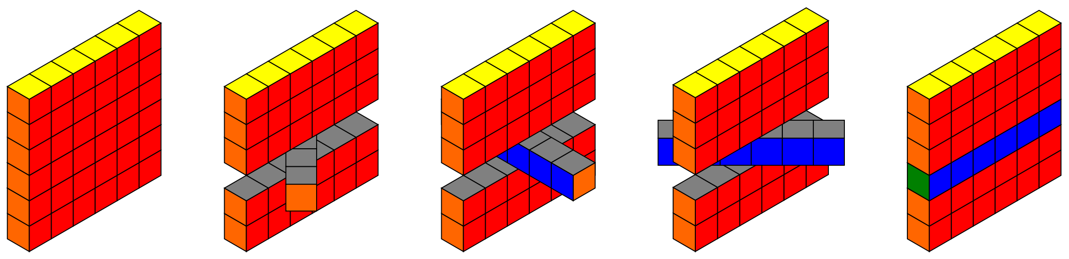


Figure 1 A single move in an example 6 × 6 Rubik's Square.

puzzle configuration whether that puzzle can be solved in a given number of moves. We show that this problem is NP-complete for the $n \times n$ Rubik's Square and for the $n \times n \times n$ Rubik's Cube under two different move models. These results close a problem that has been repeatedly posed as far back as 1984 [1, 8, 5] and has until now remained open [7].

In Section 2, we formally introduce the decision problems regarding Rubik's Squares and Rubik's Cubes whose complexity we will analyze. Then in Section 3, we introduce the variant of the Hamiltonicity problem that we will reduce from – Promise Cubical Hamiltonian Path – and prove this problem to be NP-hard. Next, we prove that the problems regarding the Rubik's Square are NP-complete in Section 4 by reducing from Promise Cubical Hamiltonian Path. After that, we apply the same ideas in Section 5 to a more complicated proof of NP-hardness for the problems regarding the Rubik's Cube. Finally, we discuss possible next steps in Section 6. Membership in NP, as well as other omitted proofs, can be found in the full version of this paper [4].

2 Rubik's Cube and Rubik's Square problems

2.1 Rubik's Square

We begin with a simpler model based on the Rubik's Cube which we will refer to as the Rubik's Square. In this model, a puzzle consists of an $n \times n$ array of unit cubes, called *cubies* to avoid ambiguity. Every cubie face on the outside of the puzzle has a colored (red, blue, green, white, yellow, or orange) sticker. The goal of the puzzle is to use a sequence of moves to rearrange the cubies such that each face of the puzzle is monochromatic in a different color. A *move* consists of flipping a single row or column in the array through space via a rotation in the long direction as demonstrated in Figure 1.

We are concerned with the following decision problem:

▶ **Problem 1.** *The **Rubik's Square** problem has as input an $n \times n$ Rubik's Square configuration and a value k. The goal is to decide whether a Rubik's Square in configuration C can be solved in k moves or fewer.*

Note that this type of puzzle was previously introduced in [3] as the $n \times n \times 1$ Rubik's Cube. In that paper, the authors showed that deciding whether it is possible to solve the $n \times n \times 1$ Rubik's Cube in a given number of moves is NP-complete when the puzzle is allowed to have missing stickers (and the puzzle is considered solved if each face contains stickers of only one color).

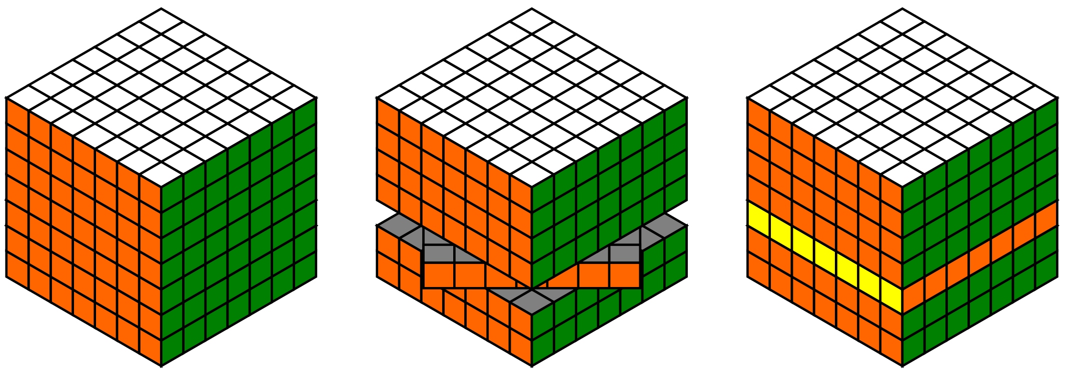


Figure 2 A single slice rotation in an example $7 \times 7 \times 7$ Rubik's Cube.

2.2 Rubik's Cube

Next consider the Rubik's Cube puzzle. An $n \times n \times n$ Rubik's Cube is a cube consisting of n^3 unit cubes called *cubies*. Every face of a cubie that is on the exterior of the cube has a colored (red, blue, green, white, yellow, or orange) sticker. The goal of the puzzle is to use a sequence of moves to reconfigure the cubies in such a way that each face of the cube ends up monochromatic in a different color. A *move count metric* is a convention for counting moves in a Rubik's Cube. Several common move count metrics for Rubik's Cubes are listed in [9]. As discussed in [2], however, many common move count metrics do not easily generalize to $n > 3$ or are not of any theoretical interest. In this paper, we will restrict our attention to two move count metrics called the Slice Turn Metric and the Slice Quarter Turn Metric. Both of these metrics use the same type of motion to define a move. Consider the subdivision of the Rubik's Cube's volume into n *slices* of dimension $1 \times n \times n$ (or $n \times 1 \times n$ or $n \times n \times 1$). In the Slice Turn Metric (STM), a *move* is a rotation of a single slice by any multiple of 90°. Similarly, in the Slice Quarter Turn Metric (SQTM), a *move* is a rotation of a single slice by an angle of 90° in either direction. An example SQTM move is shown in Figure 2.

We are concerned with the following decision problems:

▶ **Problem 2.** *The **STM/SQTM Rubik's Cube problem** takes as input a configuration of a Rubik's Cube together with a number k. The goal is to decide whether a Rubik's Cube in configuration C can be solved in at most k STM/SQTM moves.*

2.3 Notation

Next we define some notation for dealing with the Rubik's Cube and Rubik's Square problems.

To begin, we need a way to refer to cubies and stickers. For this purpose, we orient the puzzle to be axis-aligned. In the case of the Rubik's Square we arrange the $n \times n$ array of cubies in the x and y directions and we refer to a cubie by stating its x and y coordinates. In the case of the Rubik's Cube, we refer to a cubie by stating its x, y, and z coordinates. To refer to a sticker in either puzzle, we need only specify the face on which that sticker resides (e.g. "top" or "$+z$") and also the two coordinates of the sticker along the surface of the face (e.g. the x and y coordinates for a sticker on the $+z$ face).

If $n = 2a + 1$ is odd, then we will let the coordinates of the cubies in each direction range over the set $\{-a, -(a-1), \ldots, -1, 0, 1, \ldots, a-1, a\}$. This is equivalent to centering the puzzle at the origin. If, however, $n = 2a$ is even, then we let the coordinates of the cubies in each direction range over the set $\{-a, -(a-1), \ldots, -1\} \cup \{1, \ldots, a-1, a\}$. In this case, the coordinate scheme does not correspond with a standard coordinate sheme no matter how we translate the cube. This coordinate scheme is a good idea for the following reason: under this scheme, if a move relocates a sticker, the coordinates of that sticker remain the same up to permutation and negation.

Next, we need a way to distinguish the sets of cubies affected by a move from each other.

In the Rubik's Square, there are two types of moves. The first type of move, which we will call a *row move* or a y *move*, affects all the cubies with some particular y coordinate. The second type of move, which we will call a *column move* or an x *move* affects all the cubies with some particular x coordinate. We will refer to the set of cubies affected by a row move as a *row* and refer to the set of cubies affected by a column move as a *column*. In order to identify a move, we must identify which row or column is being flipped, by specifying whether the move is a row or column move as well as the index of the coordinate shared by all the moved cubies (e.g. the index -5 row move is the move that affects the cubies with $y = -5$).

In the Rubik's Cube, each STM/SQTM move affects a single slice of n^2 cubies sharing some coordinate. If the cubies share an x (or y or z) coordinate, then we call the slice an x *(or y or z) slice*. As with the Rubik's Square, we identify the slice by its normal direction together with its cubies' index in that direction (e.g. the $x = 3$ slice). We will also refer to the six slices at the boundaries of the Cube as *face slices* (e.g. the $+x$ face slice).

A move in a Rubik's Cube can be named by identifying the slice being rotated and the amount of rotation. We split this up into the following five pieces of information: the normal direction to the slice, the sign of the index of the slice, the absolute value of the index of the slice, the amount of rotation, and the direction of rotation. Splitting the information up in this way allows us not only to refer to individual moves (by specifying all five pieces of information) but also to refer to interesting sets of moves (by omitting one or more of the pieces of information).

To identify the normal direction to a slice, we simply specify x, y, or z; for example, we could refer to a move as an x move whenever the rotating slice is normal to the x direction. We will use two methods to identify the sign of the index of a moved slice. Sometimes we will refer to positive moves or negative moves, and sometimes we will combine this information with the normal direction and specify that the move is a $+x$, $-x$, $+y$, $-y$, $+z$, or $-z$ move. We use the term *index-v move* to refer to a move rotating a slice whose index has absolute value v. In the particular case that the slice rotated is a face slice, we instead use the term *face move*. We refer to a move as a *turn* if the angle of rotation is $90°$ and as a *flip* if the angle of rotation is $180°$. In the case that the angle of rotation is $90°$, we can specify further by using the terms *clockwise turn* and *counterclockwise turn*. We make the notational convention that clockwise and counterclockwise rotations around the x, y, or z axes are labeled according to the direction of rotation when looking from the direction of positive x, y, or z.

We also extend the same naming conventions to the Rubik's Square moves. For example, a positive row move is any row move with positive index and an index-v move is any move with index $\pm v$.

2.4 Group-theoretic approach

An alternative way to look at the Rubik's Square and Rubik's Cube problems is through the lens of group theory. The transformations that can be applied to a Rubik's Square or Rubik's Cube by a sequence of moves form a group with composition as the group operation. Define RS_n to be the group of possible sticker permutations in an $n \times n$ Rubik's Square and define RC_n to be the group of possible sticker permutations in an $n \times n \times n$ Rubik's Cube.

Consider the moves possible in an $n \times n$ Rubik's Square or an $n \times n \times n$ Rubik's Cube. Each such move has a corresponding element in group RS_n or RC_n.

For the Rubik's Square, let $x_i \in RS_n$ be the transformation of flipping the column with index i in an $n \times n$ Rubik's Square and let y_i be the transformation of flipping the row with index i in the Square. Then if I is the set of row/column indices in an $n \times n$ Rubik's Square we have that RS_n is generated by the set of group elements $\bigcup_{i \in I}\{x_i, y_i\}$.

Similarly, for the Rubik's Cube, let x_i, y_i, and z_i in RC_n be the transformations corresponding to clockwise turns of x, y, or z slices with index i. Then if I is the set of slice indices in an $n \times n \times n$ Rubik's Cube we have that RC_n is generated by the set of group elements $\bigcup_{i \in I}\{x_i, y_i, z_i\}$.

Using these groups we obtain a new way of identifying puzzle configurations. Let C_0 be a canonical solved configuration of a Rubik's Square or Rubik's Cube puzzle. For the $n \times n$ Rubik's Square, define C_0 to have top face red, bottom face blue, and the other four faces green, orange, yellow, and white in some fixed order. For the $n \times n \times n$ Rubik's Cube, let C_0 have the following face colors: the $+x$ face is orange, the $-x$ face is red, the $+y$ face is green, the $-y$ face is yellow, the $+z$ face is white, and the $-z$ face is blue. Then from any element of RS_n or RC_n, we can construct a configuration of the corresponding puzzle by applying that element to C_0. In other words, every transformation $t \in RS_n$ or $t \in RC_n$ corresponds with the configuration $C_t = t(C_0)$ of the $n \times n$ Rubik's Square or $n \times n \times n$ Rubik's Cube that is obtained by applying t to C_0.

Using this idea, we define a new series of problems:

▶ **Problem 3.** *The **Group Rubik's Square** problem has as input a transformation $t \in RS_n$ and a value k. The goal is to decide whether the transformation t can be reversed by a sequence of at most k transformations corresponding to Rubik's Square moves. In other words, the answer is "yes" if and only if the transformation t can be reversed by a sequence of at most k transformations of the form x_i or y_i.*

▶ **Problem 4.** *The **Group STM/SQTM Rubik's Cube** problem has as input a transformation $t \in RC_n$ and a value k. The goal is to decide whether the transformation t can be reversed by a sequence of at most k transformations corresponding with legal Rubik's Cube moves under move count metric STM/SQTM.*

We can interpret these problems as variants of the Rubik's Square or Rubik's Cube problems. For example, the Rubik's Square problem asks whether it is possible (in a given number of moves) to unscramble a Rubik's Square configuration so that each face ends up monochromatic, while the Group Rubik's Square problem asks whether it is possible (in a given number of moves) to unscramble a Rubik's Square configuration so that each sticker goes back to its exact position in the originally solved configuration C_0. As you see, the Group Rubik's Square problem, as a puzzle, is just a more difficult variant of the puzzle: instead of asking the player to move all the stickers of the same color to the same face, this variant asks the player to move each stickers to the exact correct position. Similarly, the Group STM/SQTM Rubik's Cube problem as a puzzle asks the player to move each sticker to an exact position. These problems can have practical applications with physical puzzles. For example, some Rubik's Cubes have pictures split up over the stickers of each face instead of just monochromatic colors on the stickers. For these puzzles, as long as no two stickers are the same, the Group STM/SQTM Rubik's Cube problem is more applicable than the STM/SQTM Rubik's Cube problem (which can leave a face "monochromatic" but scrambled in image).

We formalize the idea that the Group version of the puzzle is a strictly more difficult puzzle in the following lemmas:

▶ **Lemma 2.1.** *If (t, k) is a "yes" instance to the Group Rubik's Square problem, then $(t(C_0), k)$ is a "yes" instance to the Rubik's Square problem.*

▶ **Lemma 2.2.** *If (t, k) is a "yes" instance to the Group STM/SQTM Rubik's Cube problem, then $(t(C_0), k)$ is a "yes" instance to the STM/SQTM Rubik's Cube problem.*

At this point it is also worth mentioning that the Rubik's Square with SQTM move model is a strictly more difficult puzzle than the Rubik's Square with STM move model:

▶ **Lemma 2.3.** *If (C, k) is a "yes" instance to the SQTM Rubik's Cube problem, then it is also a "yes" instance to the STM Rubik's Cube problem. Similarly, if (t, k) is a "yes" instance to the Group SQTM Rubik's Cube problem, then it is also a "yes" instance to the Group STM Rubik's Cube problem.*

3 Hamiltonicity variants

To prove the problems introduced above hard, we need to introduce several variants of the Hamiltonian cycle and path problems.

It is shown in [6] that the following problem is NP-complete.

▶ **Problem 5.** *A* square grid graph *is a finite induced subgraph of the infinite square lattice. The* Grid Graph Hamiltonian Cycle *problem asks whether a given square grid graph with no degree-*1 *vertices has a Hamiltonian cycle.*

Starting with this problem, we prove that the following promise version of the grid graph Hamiltonian path problem is also NP-hard.

▶ **Problem 6.** *The* Promise Grid Graph Hamiltonian Path *problem takes as input a square grid graph G and two specified vertices s and t with the promise that any Hamiltonian path in G has s and t as its start and end respectively. The problem asks whether there exists a Hamiltonian path in G.*

The above problem is more useful, but it is still inconvenient in some ways. In particular, there is no conceptually simple way to connect a grid graph to a Rubik's Square or Rubik's Cube puzzle. It is the case, however, that every grid graph is actually a type of graph called a "cubical graph". Cubical graphs, unlike grid graphs, can be conceptually related to Rubik's Cubes and Rubik's Squares with little trouble.

So what is a cubical graph? Let H_m be the m dimensional hypercube graph; in particular, the vertices of H_m are the bitstrings of length m and the edges connect pairs of bitstrings whose Hamming distance is exactly one. Then a *cubical graph* is any induced subgraph of any hypercube graph H_m.

Notably, when embedding a grid graph into a hypercube, it is always possible to assign the bitstring label $00\ldots0$ to any vertex. Suppose we start with Promise Grid Graph Hamiltonian Path problem instance (G, s, t); then by embedding G into a hypercube graph, we can reinterpret this instance as an instance of the promise version of cubical Hamiltonian path:

▶ **Problem 7.** *The* Promise Cubical Hamiltonian Path *problem takes as input a cubical graph whose vertices are length-m bitstrings $l_1, l_2, \ldots, l_n$ with the promise that (1) $l_n = 00\ldots0$ and (2) any Hamiltonian path in the graph has l_1 and l_n as its start and end respectively. The* *problem asks whether there exists a Hamiltonian path in the cubical graph. In other words, the problem asks whether it is possible to rearrange bitstrings $l_1, \ldots, l_n$ into a new order such that each bitstring has Hamming distance one from the next.*

First, we reduce from the Grid Graph Hamiltonian Cycle problem to the Promise Grid Graph Hamiltonian Path problem.

▶ **Lemma 3.1.** *The Promise Grid Graph Hamiltonian Path problem (Problem 6) is NP-hard.*

Second, we reduce from the Promise Grid Graph Hamiltonian Path problem to the Promise Cubical Hamiltonian Path problem.

▶ **Theorem 3.2.** *The Promise Cubical Hamiltonian Path problem (Problem 7) is NP-hard.*

4 (Group) Rubik's Square is NP-complete

4.1 Reductions

To prove that the Rubik's Square and Group Rubik's Square problems are NP-complete, we reduce from the Promise Cubical Hamiltonian Path problem of Section 3.

Suppose we are given an instance of the Promise Cubical Hamiltonian Path problem consisting of n bitstrings $l_1, \ldots, l_n$ of length m (with $l_n = 00\ldots0$). To construct a Group Rubik's Square instance we need to compute the value k indicating the allowed number of moves and construct the transformation $t \in RS_s$.

The value k can be computed directly as $k = 2n - 1$.

The transformation t will be an element of group RS_s where $s = 2(\max(m,n) + 2n)$. Define a_i for $1 \le i \le n$ to be $(x_1)^{(l_i)_1} \circ (x_2)^{(l_i)_2} \circ \cdots \circ (x_m)^{(l_i)_m}$ where $(l_i)_1, (l_i)_2, \ldots, (l_i)_m$ are the bits of l_i. Also define $b_i = (a_i)^{-1} \circ y_i \circ a_i$ for $1 \le i \le n$. Then we define t to be $a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$.

Outputting (t,k) completes the reduction from the Promise Cubical Hamiltonian Path problem to the Group Rubik's Square problem. To reduce from the Promise Cubical Hamiltonian Path problem to the Rubik's Square problem we simply output $(C_t, k) = (t(C_0), k)$. These reductions clearly run in polynomial time.

4.2 Intuition

The key idea that makes this reduction work is that the transformations b_i for $i \in \{1, \ldots, n\}$ all commute. This allows us to rewrite $t = a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$ with the b_is in a different order. If the order we choose happens to correspond to a Hamiltonian path in the cubical graph specified by $l_1, \ldots, l_n$, then when we explicitly write the b_is and a_1 in terms of x_js and y_is, most of the terms cancel. In particular, the number of remaining terms will be exactly k. Since we can write t as a combination of exactly k x_js and y_is, we can invert t using at most k x_js and y_is. In other words, if there is a Hamiltonian path in the cubical graph specified by $l_1, \ldots, l_n$, then (t,k) is a "yes" instance to the Group Rubik's Square problem.

In order to more precisely describe the cancellation of terms in t, we can consider just one local part: $b_i \circ b_{i'}$. We can rewrite this as $(a_i)^{-1} \circ y_i \circ a_i \circ (a_{i'})^{-1} \circ y_{i'} \circ a_{i'}$. The interesting part is that $a_i \circ (a_{i'})^{-1}$ will cancel to become just one x_j. Note that

$$a_i \circ (a_{i'})^{-1} = (x_1)^{(l_i)_1} \circ (x_2)^{(l_i)_2} \circ \cdots \circ (x_m)^{(l_i)_m} \circ (x_1)^{-(l_{i'})_1} \circ (x_2)^{-(l_{i'})_2} \circ \cdots \circ (x_m)^{-(l_{i'})_m},$$

which we can rearrange as

$$(x_1)^{(l_i)_1 - (l_{i'})_1} \circ (x_2)^{(l_i)_2 - (l_{i'})_2} \circ \cdots \circ (x_m)^{(l_i)_m - (l_{i'})_m}.$$

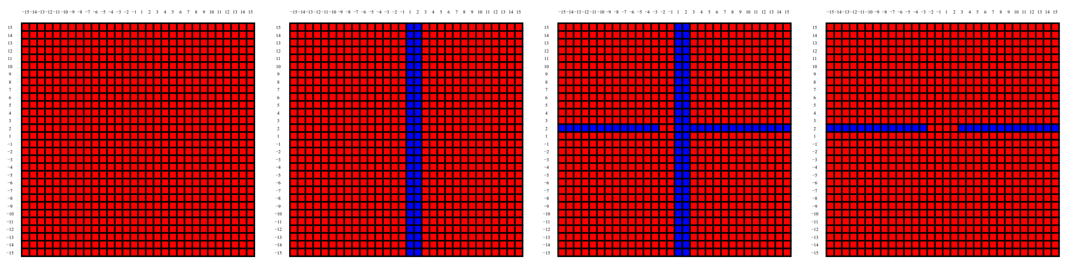


Figure 3 Applying b_2 to C_0 step by step (only top face shown).

Next, if b_i and $b_{i'}$ correspond to adjacent vertices l_i and $l_{i'}$, then $(l_i)_j - (l_{i'})_j$ is zero for all j except one for which $(l_i)_j - (l_{i'})_j = \pm 1$. Thus the above can be rewritten as $(x_j)^1$ or $(x_j)^{-1}$ for some specific j. Since $x_j = (x_j)^{-1}$ this shows that $(a_{i_1})^{-1} \circ a_{i_2}$ simplifies to x_j for some j.

This intuition is formalized in the following sequence of results.

▶ **Lemma 4.1.** *The transformations b_i all commute.*

▶ **Theorem 4.2.** *If $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem, then (t, k) is a "yes" instance to the Group Rubik's Square problem.*

▶ **Corollary 4.3.** *If $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem, then (C_t, k) is a "yes" instance to the Rubik's Square problem.*

4.3 Coloring of C_t

In order to show the other direction of the proof, it will be helpful to consider the coloring of the stickers on the top and bottom faces of the Rubik's Square. In particular, if we define $b = b_1 \circ \cdots \circ b_n$ (so that $t = a_1 \circ b$), then it will be very helpful for us to know the colors of the top and bottom stickers in configuration $C_b = b(C_0)$.

Consider for example the instance of Promise Cubical Hamiltonian Path with $n = 5$ and $m = 3$ defined by $l_1 = 011$, $l_2 = 110$, $l_3 = 111$, $l_4 = 100$, $l_5 = 000$. For this example, C_0 is an $s \times s$ Rubik's Square with $s = 2(\max(m, n) + 2n) = 30$.

To describe configuration C_b, we need to know the effect of transformation b_i. For example, Figure 3 shows the top face of a Rubik's Square in configurations C_0, $a_2(C_0)$, $(y_2 \circ a_2)(C_0)$, and $b_2(C_0) = ((a_2)^{-1} \circ y_2 \circ a_2)(C_0)$ where a_2 and y_2 are defined in terms of $l_2 = 110$ as in the reduction.

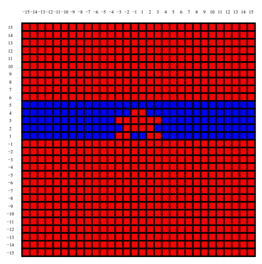
The exact behavior of a Rubik's Square due to b_i is described by the following lemma:

▶ **Lemma 4.4.** *Suppose $i \in \{1, \ldots, n\}$, and $c, r \in \{1, \ldots, s/2\}$. Then*

1. *if $r = i$ and $c \leq m$ such that bit c of l_i is 1, then b_i swaps the cubies in positions $(c, -r)$ and $(-c, r)$ without flipping either;*
2. *if $r = i$ and either $c > m$ or $c \leq m$ and bit c of l_i is 0, then b_i swaps the cubies in positions (c, r) and $(-c, r)$ and flips them both;*
3. *all other cubies are not moved by b_i.*

We can apply the above to figure out the effect of transformation $b_1 \circ b_2 \circ \cdots \circ b_n$ on configuration C_0. In particular, that allows us to learn the coloring of configuration C_b.

▶ **Theorem 4.5.** *In C_b, a cubie has top face blue if and only if it is in position (c, r) such that $1 \leq r \leq n$ and either $|c| > m$ or $|c| \leq m$ and bit $|c|$ of l_r is 0.*


(a) The top face of C_b for the example input $l_1, \dots, l_n$.

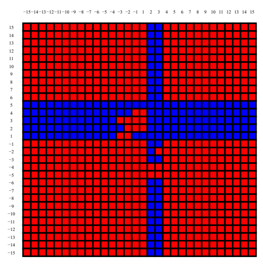

(b) The top face of C_t for the example input $l_1, \dots, l_n$.

Figure 4 The coloring of the Rubik's Square for the example input $l_1, \dots, l_n$.

This concludes the description of C_b in terms of colors. The coloring of configuration C_t – the configuration that is actually obtained by applying the reduction to $l_1, \dots, l_n$ – can be obtained from the coloring of configuration C_b by applying transformation a_1.

Applying Theorem 4.5 to the previously given example, we obtain the coloring of the Rubik's Square in configuration C_b as shown in Figure 4a. Note that the $n \times m$ grid of bits comprising $l_1, \dots, l_n$ is actually directly encoded in the coloring of a section of the Rubik's Square. In addition, the coloring of the Rubik's Square in configuration C_t is shown for the same example in Figure 4b.

4.4 (Group) Rubik's Square solution $\rightarrow$ Promise Cubical Hamiltonian Path solution

In the full paper, [4], we prove the following theorem:

▶ **Theorem 4.6.** *If (C_t, k) is a "yes" instance to the Rubik's Square problem, then $l_1, \dots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.*

By Lemma 2.1, this immediately implies the following corollary:

▶ **Corollary 4.7.** *If (t, k) is a "yes" instance to the Group Rubik's Square problem, then $l_1, \dots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.*

4.5 Conclusion

Theorems 4.2 and 4.6 and Corollaries 4.3 and 4.7 show that the polynomial-time reductions given are answer preserving. As a result, we conclude that

▶ **Theorem 4.8.** *The Rubik's Square and Group Rubik's Square problems are NP-complete.*

5 (Group) STM/SQTM Rubik's Cube is NP-complete

5.1 Reductions

Below, we introduce the reductions used for the Rubik's Cube case. These reductions very closely mirror the Rubik's Square case, and the intuition remains exactly the same: the b_i terms commute, and so if the input Promise Cubical Hamiltonian Path instance is a "yes" instance then the b_is can be reordered so that all but k moves in the definition of t will cancel; therefore in that case t can be both enacted and reversed in k moves.

There are, however, several notable differences from the Rubik's Square case. The first difference is that in a Rubik's Cube, the moves x_i, y_i, and z_i are all quarter turn rotations rather than self-inverting row or column flips. One consequence is that unlike in the Rubik's Square case, the term a_i does not have the property that $(a_i)^{-1} = a_i$. A second difference is that in a Rubik's Square, the rows never become columns or visa versa. In a Rubik's Cube on the other hand, rotation of the faces can put rows of stickers that were once aligned parallel to one axis into alignment with another axis. To avoid allowing a solution of the puzzle due to this fact in the absence of a solution to the input Promise Cubical Hamiltonian Path instance, the slices in this construction which take the role of rows 1 through n in the Rubik's Square case and the slices which take the role of columns 1 through m in the Rubik's Square case will be assigned entirely distinct indices.

To prove that the STM/SQTM Rubik's Cube and Group STM/SQTM Rubik's Cube problems are NP-complete, we reduce from the Promise Cubical Hamiltonian Path problem of Section 3 as described below.

Suppose we are given an instance of the Promise Cubical Hamiltonian Path problem consisting of n biststrings $l_1, \ldots, l_n$ of length m (with $l_n = 00\ldots0$). To construct a Group STM/SQTM Rubik's Square instance we need to compute the value k indicating the allowed number of moves and construct the transformation t in RC_s.

The value k can be computed directly as $k = 2n - 1$.

The transformation t will be an element of group RC_s where $s = 6n + 2m$. Define a_i for $1 \leq i \leq n$ to be $(x_1)^{(l_i)_1} \circ (x_2)^{(l_i)_2} \circ \cdots \circ (x_m)^{(l_i)_m}$ where $(l_i)_1, (l_i)_2, \ldots, (l_i)_m$ are the bits of l_i. Also define $b_i = (a_i)^{-1} \circ z_{m+i} \circ a_i$ for $1 \leq i \leq n$. Then we define t to be $a_1 \circ b_1 \circ b_2 \circ \cdots \circ b_n$.

Outputting (t, k) completes the reduction from the Promise Cubical Hamiltonian Path problem to the Group STM/SQTM Rubik's Cube problem. To reduce from the Promise Cubical Hamiltonian Path problem to the STM/SQTM Rubik's Cube problem we simply output $(C_t, k) = (t(C_0), k)$. As with the Rubik's Square case, these reductions are clearly polynomial-time reductions.

5.2 Promise Cubical Hamiltonian Path solution → (Group) STM/SQTM Rubik's Cube solution

The proof of this direction is not substantively different from the proof of the first direction for the Rubik's Square problems. The differences in these proofs are all minor details that are only present to account for the differences (listed above) between the Rubik's Square and Rubik's Cube reductions. See [4], the full paper, for details.

5.3 Coloring of C_t

As in the Rubik's Square case, it is helpful for the second direction of the proof to know the coloring of the Cube's configuration. As before, we define $b = b_1 \circ \cdots \circ b_n$ (so that $t = a_1 \circ b$) and determine the colors of the stickers in configuration $C_b = b(C_0)$.

Consider the example instance of Promise Cubical Hamiltonian Path with $n = 5$ and $m = 3$ introduced in Section 4.3. For this example instance, the Rubik's Cube configuration produced by the reduction is an $s \times s \times s$ Rubik's Cube with $s = 2m + 6n = 36$. Furthermore, the coloring of the stickers in C_b for this example is shown in Figure 5. Note that the $n \times m$ grid of bits comprising $l_1, \ldots, l_n$ is actually directly encoded in the coloring of each face.

In the full paper, we formalize the general pattern of colors intuited from this example.

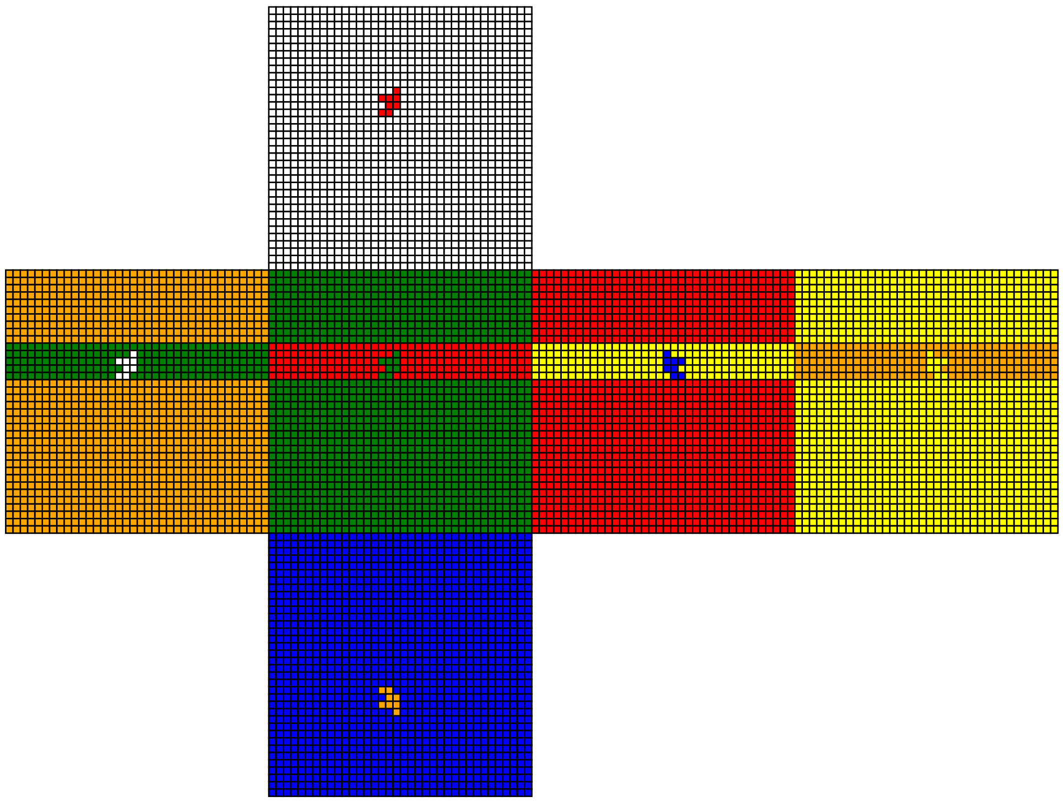


Figure 5 The faces of C_b for the example input $l_1, \ldots, l_n$. In this figure, the top and bottom faces are the $+z$ and $-z$ faces, while the faces in the vertical center of the figure are the $+x$, $+y$, $-x$, and $-y$ faces from left to right.

5.4 (Group) STM/SQTM Rubik's Cube solution $\rightarrow$ Promise Cubical Hamiltonian Path solution: proof outline

In [4], the full paper, we prove the following:

▶ **Theorem 5.1.** *If (C_t, k) is a "yes" instance to the STM Rubik's Cube problem, then $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.*

By Lemmas 2.2 and 2.3, this immediately implies the following corollary:

▶ **Corollary 5.2.** *If (t, k) is a "yes" instance to the Group STM/SQTM Rubik's Cube problem or (C_t, k) is a "yes" instance to the STM/SQTM Rubik's Cube problem, then $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.*

The intuition behind the proof of Theorem 5.1 is similar to that used in the Rubik's Square case, but there is added complexity due to the extra options available in a Rubik's Cube. Most of the added complexity is due to the possibility of face moves (allowing rows of stickers to align in several directions over the course of a solution). Below, we describe an outline of the proof using several high-level steps.

Consider a hypothetical solution to the (C_t, k) instance of the STM Rubik's Cube problem consisting of a sequence of STM Rubik's Cube moves $m_1, \ldots, m_{k'}$ with $k' \leq k$ such that $C' = (m_{k'} \circ \cdots \circ m_1)(C_t)$ is a solved configuration of the Rubik's Cube.

Using the fact that the side-length of the cube is large compared to the number of allowed moves, we prove the following preliminary facts:

- There are no indices $i \in \{1, \ldots, n\}$ such that $m_1, \ldots, m_{k'}$ contains exactly zero index-$(m+i)$ moves.
- If $m_1, \ldots, m_{k'}$ contains exactly one index-$(m+i)$ move, then the sole index-$(m+i)$ move must be a counterclockwise z turn. In this case, call the move in question an O-move (where O stands for "one").
- If $m_1, \ldots, m_{k'}$ contains exactly two index-$(m+i)$ moves, then the two index-$(m+i)$ moves must be a clockwise z turn and a z flip in some order. In this case, call the moves in question T-moves (where T stands for "two").
- All O- and T-moves must occur at a time when faces $+x$, $+y$, $-x$, and $-y$ all have zero rotation and any move of z slice $-(m+i)$ must occur at a time when these faces all have rotation $180°$.

Next, we introduce a new concept, paired stickers, and use it to prove the following. Suppose that $j \in \{1, 2, \ldots, m\}$ is a value such that l_{i_1} and l_{i_2} differ in bit j, and $i_1, i_2 \in \{1, \ldots, n\}$ are indices for which $m_1, \ldots, m_{k'}$ contains an index-$(m+i_1)$ O-move and an index-$(m+i_2)$ O-move. Then it must be the case that between these two moves there is either at least one index-j move or at least one face move of faces $+x$, $+y$, $-x$, and $-y$ (which by the previous results actually requires at least two such face moves).

After that, we use a counting argument to significantly restrict the possible moves in $m_1, \ldots, m_{k'}$. In particular, we classify the moves into several types and use the previous results to bound the number of moves of each type. Adding these bounds together and simplifying, we find that $k' \geq k$. Since we already know that $k' \leq k$, we learn that equality must hold in each of our computed bounds. This immediately constrains the quantity of each type of move even further. In particular, we learn that $m_1, \ldots, m_{k'}$ consists of three types of moves: O-moves, T-moves, and index-j moves for $j \in \{1, 2, \ldots, m\}$ (call these J-moves). Furthermore, there is exactly one J-move between every consecutive pair of O-moves.

After the types of moves in $m_1, \ldots, m_{k'}$ are restricted to this extent, several possibilites that we previously had to consider are no longer relevent (i.e. there are no face moves). As a consequence, the earlier results are actually strengthened and can be reapplied to learn even more about the types of O-, T-, and J-moves in $m_1, \ldots, m_{k'}$.

Finally, by applying the idea of paired stickers to our now-highly-constrained sequence of moves $m_1, \ldots, m_{k'}$, we are able to show that there are no T-moves in $m_1, \ldots, m_{k'}$. At this point, we can conclude that $m_1, \ldots, m_{k'}$ consists entirely of alternating O- and J-moves with one O-move of an index-$(m+i)$ slice for every $i \in \{1, \ldots, n\}$. If three consecutive O-, J-, and O-moves rotate index-$(m+i_1)$, index-j, and index-$(m+i_2)$ slices, then it must be the case that l_{i_1} and l_{i_2} differ in bit j and in no other bit. Thus, if we consider all of the O-moves in the order in which they occur, the corresponding elements $i \in \{1, \ldots, n\}$ in the same order have the property that each bitstring l_i is at Hamming distance one from the next. In other words, we have our desired result: that $l_1, \ldots, l_n$ is a "yes" instance to the Promise Cubical Hamiltonian Path problem.

5.5 Conclusion

Sections 5.2 and 5.4 show that the polynomial-time reductions given are answer preserving. As a result, we conclude that

▶ **Theorem 5.3.** *The STM/SQTM Rubik's Cube and Group STM/SQTM Rubik's Cube problems are NP-complete.*

6 Future work

In this paper, we resolve the complexity of optimally solving Rubik's Cubes under move count metrics for which a single move rotates a single slice. It could be interesting to consider the complexity of this problem under other move count metrics.

Of particular interest are the Wide Turn Metric (WTM) and Wide Quarter Turn Metric (WQTM), in which the puzzle solver can rotate any contiguous group of layers including a face. These metrics correspond most directly to how one would physically solve a real-world $n \times n \times n$ Rubik's Cube: by grabbing some number of layers (including a face) from the side of the cube and rotating thm together. We can also consider the $1 \times n \times n$ analogue of the Rubik's Cube with WTM move count metric: this would be a Rubik's Square in which a single move flips a contiguous sequence of rows or columns including a row or column at the edge of the Square. Solving this toy model could help point us in the right direction for the WTM and WQTM Rubik's Cube problems. If the toy model resists analysis, it could be interesting to consider this toy model with missing stickers.

References

1 Stephen A. Cook. Can computers routinely discover mathematical proofs? *Proceedings of the American Philosophical Society*, 128(1):40–43, 1984. URL: `http://www.jstor.org/stable/986492`.

2 Cride5. Move count metrics for big cubes - standards and preferences. Speed Solving Forum, August 2010. URL: `https://www.speedsolving.com/forum/showthread.php?23546-Move-count-metrics-for-big-cubes-standards-and-preferences`.

3 Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Anna Lubiw, and Andrew Winslow. Algorithms for solving Rubik's Cubes. In *Proceedings of the 19th European Conference on Algorithms*, ESA'11, pages 689–700, Berlin, Heidelberg, 2011. Springer-Verlag. URL: `http://dl.acm.org/citation.cfm?id=2040572.2040647`.

4 Erik D. Demaine, Sarah Eisenstat, and Mikhail Rudoy. Solving the Rubik's Cube optimally is NP-complete. arXiv:1706.06708, 2017. URL: `https://arXiv.org/abs/1706.06708`.

5 Jeff Erickson. Is optimally solving the n×n×n Rubik's Cube NP-hard? Theoretical Computer Science Stack Exchange. URL: `https://cstheory.stackexchange.com/q/783` (version: 2010-10-23).

6 Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, November 1982.

7 Graham Kendall, Andrew J. Parkes, and Kristian Spoerer. A survey of NP-complete puzzles. *ICGA Journal*, 31:13–34, 2008.

8 Daniel Ratner and Manfred K. Warmuth. Nxn puzzle and related relocation problem. *J. Symb. Comput.*, 10(2):111–138, 1990. `doi:10.1016/S0747-7171(08)80001-6`.

9 Wiki. Metric. Speed Solving Wiki, May 2010. URL: `https://www.speedsolving.com/wiki/index.php/Metric`.