Algorithms for Solving Rubik's Cubes

Erik D. Demaine[1], Martin L. Demaine[1], Sarah Eisenstat[1],
Anna Lubiw[2], and Andrew Winslow[3]

[1] MIT Computer Science and Artificial Intelligence Laboratory,
Cambridge, MA 02139, USA, {edemaine,mdemaine,seisenst}@mit.edu
[2] David R. Cheriton School of Computer Science,
University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, alubiw@uwaterloo.ca
[3] Department of Computer Science, Tufts University,
Medford, MA 02155, USA, awinslow@cs.tufts.edu

Abstract. The Rubik's Cube is perhaps the world's most famous and iconic puzzle, well-known to have a rich underlying mathematical structure (group theory). In this paper, we show that the Rubik's Cube also has a rich underlying algorithmic structure. Specifically, we show that the $n \times n \times n$ Rubik's Cube, as well as the $n \times n \times 1$ variant, has a "God's Number" (diameter of the configuration space) of $\Theta(n^2/\log n)$. The upper bound comes from effectively parallelizing standard $\Theta(n^2)$ solution algorithms, while the lower bound follows from a counting argument. The upper bound gives an asymptotically optimal algorithm for solving a general Rubik's Cube in the worst case. Given a specific starting state, we show how to find the shortest solution in an $n \times O(1) \times O(1)$ Rubik's Cube. Finally, we show that finding this optimal solution becomes NP-hard in an $n \times n \times 1$ Rubik's Cube when the positions and colors of some cubies are ignored (not used in determining whether the cube is solved).

Keywords: combinatorial puzzles, diameter, God's number, combinatorial optimization

1 Introduction

A little over thirty years ago, Hungarian architecture professor Ernő Rubik released his "Magic Cube" to the world.[4] What we now all know as the *Rubik's Cube* quickly became a sensation [26]. It is the best-selling puzzle ever, at over 350 million units [15]. It is a tribute to elegant design, being part of the permanent collection of the Museum of Modern Art in New York [18]. It is an icon for difficult puzzles—an intellectual Mount Everest. It is the heart of World Cube Association's speed-cubing competitions, whose current record holders can solve a cube in under 7 seconds (or 31 seconds blindfold) [1]. It is the basis for cube art, a form of pop art made from many carefully unsolved Rubik's Cubes. (For example, the recent movie *Exit Through the Gift Shop* features the street cube

[4] Similar puzzles were invented around the same time in the United States [9][17], the United Kingdom [6], and Japan [10] but did not reach the same level of success.

artist known as Space Invader.) It is the bane of many computers, which spent about 35 CPU years determining in 2010 that the best algorithm to solve the worst configuration requires exactly 20 moves—referred to as *God's Number* [22].

To a mathematician, or a student taking abstract algebra, the Rubik's Cube is a shining example of group theory. The configurations of the Rubik's Cube, or equivalently the transformations from one configuration to another, form a subgroup of a permutation group, generated by the basic twist moves. This perspective makes it easier to prove (and compute) that the configuration space falls into two connected components, according to the parity of the permutation on the cubies (the individual subcubes that make up the puzzle). See [7] for how to compute the number of elements in the group generated by the basic Rubik's Cube moves (or any set of permutations) in polynomial time.

To a theoretical computer scientist, the Rubik's Cube and its many generalizations suggest several natural open problems. What are good algorithms for solving a given Rubik's Cube puzzle? What is an optimal worst-case bound on the number of moves? What is the complexity of optimizing the number of moves required for a given starting configuration? Although God's Number is known to be 20 for the $3 \times 3 \times 3$, the optimal solution of each configuration in this constant-size puzzle still has not been computed [22]. While computing the exact behavior for larger cubes is out of the question, how does the worst-case number of moves and complexity scale with the side lengths of the cube? In parallel with our work, these questions were recently posed by Andy Drucker and Jeff Erickson [4]. Scalability is important given the commercially available $4 \times 4 \times 4$ Rubik's Revenge [25]; $5 \times 5 \times 5$ Professor's Cube [13]; the $6 \times 6 \times 6$ and $7 \times 7 \times 7$ V-CUBEs [27]; Leslie Le's 3D-printed $12 \times 12 \times 12$ [14]; and Oskar van Deventer's $17 \times 17 \times 17$ Over the Top and his $2 \times 2 \times 20$ Overlap Cube, both available from 3D printer *shapeways* [28].

Diameter / God's Number. The diameter of the configuration space of a Rubik's Cube seems difficult to capture using just group theory. In general, a set of permutations (moves) can generate a group with superpolynomial diameter [3]. If we restrict each generator (move) to manipulate only k elements, then the diameter is $O(n^k)$ [16], but this gives very weak (superexponential) upper bounds for $n \times n \times n$ and $n \times n \times 1$ Rubik's Cubes.

Fortunately, we confirm that the general approach taken by folk algorithms for solving Rubik's Cubes of various fixed sizes can be generalized to perform a constant number of moves per cubie, for an upper bound of $O(n^2)$. This result is essentially standard, but we take care to ensure that all cases can be handled.

Surprisingly, this bound is not optimal. Each twist move in the $n \times n \times n$ and $n \times n \times 1$ Rubik's Cubes simultaneously transforms $n^{\Theta(1)}$ cubies (with the exponent depending on the dimensions and whether a move transforms a plane or a half-space). This property offers a form of parallelism for solving multiple cubies at once, to the extent that multiple cubies want the same move to be applied at a particular time. We show that this parallelism can be exploited to reduce the number of moves by a logarithmic factor, to $O(n^2/\log n)$. Furthermore, an easy counting argument shows an average-case lower bound of $\Omega(n^2/\log n)$.

Thus we settle the diameter of the $n \times n \times n$ and $n \times n \times 1$ Rubik's Cubes, up to constant factors. These results are described in Sections 4 and 3, respectively.

n^2-1 *puzzle.* Another puzzle that can be described as a permutation group given by generators corresponding to valid moves is the $n \times n$ generalization of the classic Fifteen Puzzle. This $n^2 - 1$ *puzzle* also has polynomial diameter, though without any form of parallelism, the diameter is simply $\Theta(n^3)$ [20]. Interestingly, computing the shortest solution from a given configuration of the puzzle is NP-hard [21]. More generally, given a set of generator permutations, it is PSPACE-complete to find the shortest sequence of generators whose product is a given target permutation [5,11]. These papers mention the Rubik's Cube as motivation, but neither addresses the natural question: is it NP-hard to solve a given $n \times n \times n$ or $n \times n \times 1$ Rubik's Cube using the fewest possible moves? Although the $n \times n \times n$ problem was posed as early as 1984 [2,21], both questions remain open [12]. We give partial progress toward hardness, as well as a polynomial-time exact algorithm for a particular generalization of the Rubik's Cube.

Optimization algorithms. We give one positive and one negative result about finding the shortest solution from a given configuration of a generalized Rubik's Cube puzzle. On the positive side, we show in Section 6 how to compute the exact optimum for $n \times O(1) \times O(1)$ Rubik's Cubes. Essentially, we prove structural results about how an optimal solution decomposes into moves in the long dimension and the two short dimensions, and use this structure to obtain a dynamic program. This result may prove useful for optimally solving configurations of Oskar van Deventer's $2 \times 2 \times 20$ Overlap Cube [28], but it does not apply to the $3 \times 3 \times 3$ Rubik's Cube because we need n to be distinct from the other two side lengths. On the negative side, we prove in Section 5 that it is NP-hard to find an optimal solution to a subset of cubies in an $n \times n \times 1$ Rubik's Cube. Phrased differently, optimally solving a given $n \times n \times 1$ Rubik's Cube configuration is NP-hard when the colors and positions of some cubies are ignored (i.e., they are not considered in determining whether the cube is solved).

2 Common Definitions

We begin with some terminology. An $\ell \times m \times n$ Rubik's Cube is composed of ℓmn *cubies*, each of which has some position (x, y, z), where $x \in \{0, 1, \ldots, \ell-1\}$, $y \in \{0, 1, \ldots, m-1\}$, and $z \in \{0, 1, \ldots, n-1\}$. Each cubie also has an orientation. Each cubie in a Rubik's Cube has a color on each visible face. There are six colors in total. We say that a Rubik's Cube is *solved* when each face of the cube is the same color, unique for each face.

An *edge cubie* is any cubie which has at least two visible faces which point in perpendicular directions. A *corner cubie* is any cubie which has at least three visible faces which all point in perpendicular directions.

A *slice* of a Rubik's Cube is a set of cubies that match in one coordinate (e.g. all of the cubies such that $y = 1$). A legal move on a Rubik's Cube involves

rotating one slice around its perpendicular[5]. To preserve the shape of the cube, there are restrictions on how much the slice can be rotated. If the slice to be rotated is a square, then the slice can be rotated 90° in either direction. Otherwise, the slice can only be rotated by 180°. Finally, note that if one dimension of the cube has length 1, we disallow rotations of the only slice in that dimension. For example, we cannot rotate the slice $z = 0$ in the $n \times n \times 1$ cube.

A *configuration* of a Rubik's Cube is a mapping from each visible face of each cubie to a color. A *reachable configuration* of a Rubik's Cube is a configuration which can be reached from a solved Rubik's Cube via a sequence of legal moves.

For each of the Rubik's Cube variants we consider, we will define the contents of a *cubie cluster*. The cubies which belong in this cubie cluster depend on the problem we are working on; however, they do share some key properties:

1. Each cubie cluster consists of a constant number of cubies.
2. No sequence of legal moves can cause any cubie to move from one cubie cluster into another.

Each cubie cluster has a *cluster configuration* mapping from each visible face of the cubie cluster to its color. Because the number of cubies in a cubie cluster is constant, the number of possible cluster configurations is also constant.

We say that a move *affects* a cubie cluster if the move causes at least one cubie in the cubie cluster to change places. Similarly, we say that a sequence of moves affects a cubie cluster if at least one cubie in the cubie cluster changes position or orientation after the sequence of moves has been performed.

3 Diameter of $n \times n \times 1$ Rubik's Cube

When considering an $n \times n \times 1$ Rubik's Cube we omit the third coordinate of a cubie, which by necessity must be 0. For simplicity, we restrict the set of solutions to those configurations where the top of the cube is orange. We also assume that n is even, and ignore the edge and corner cubies. A more rigorous proof, which handles these details, is available in the full version of this paper.[6]

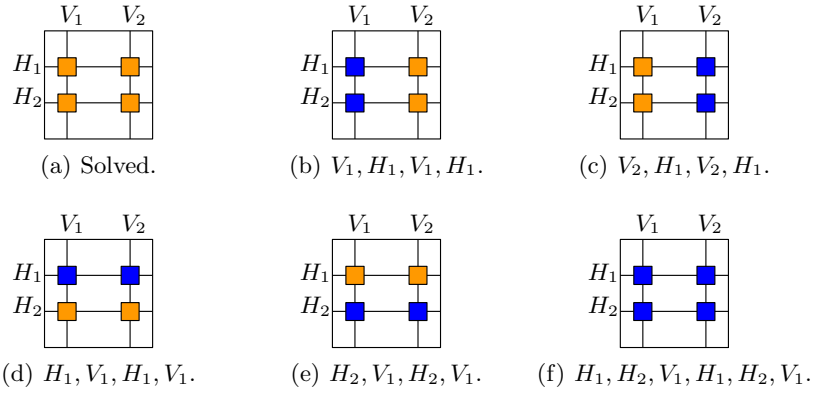
Consider the set of locations reachable by a cubie at position (x, y). If we flip column x, the cubie will move to position $(x, n-y-1)$. If we instead flip row y, it will move to position $(n-x-1, y)$. Hence, there are at most four reachable locations for a cubie that starts at (x, y): (x, y), $(x, n-y-1)$, $(n-x-1, y)$, and $(n-x-1, n-y-1)$. We call this set of locations the *cubie cluster* (x, y).

We begin by showing that for any reachable cluster configuration, there exists a sequence of moves of constant length which can be used to solve that cluster without affecting any other clusters. Figure 1 gives just such a sequence for each potential cluster configuration.

In the remainder of Section 3, we use the notation H_1, H_2 and V_1, V_2 to denote the two rows and columns containing cubies from a single cubie cluster.

[5] While other definitions of a legal move exist (e.g. rotating a set of contiguous parallel slices), this definition most closely matches the one used in popular move notations.

[6] http://arxiv.org/abs/1106.5736



(a) Solved. (b) V_1, H_1, V_1, H_1. (c) V_2, H_1, V_2, H_1.

(d) H_1, V_1, H_1, V_1. (e) H_2, V_1, H_2, V_1. (f) $H_1, H_2, V_1, H_1, H_2, V_1$.

Fig. 1. The reachable cluster configurations and the move sequences to solve them.

We also use the same symbols to denote single moves affecting these rows and columns. In the special cases of cross and center cubie clusters, we denote the single row or column containing the cluster by H or V, respectively.

3.1 $n \times n \times 1$ Upper Bound

There are n^2 clusters in the $n \times n \times 1$ Rubik's Cube. If we use the move sequences given in Fig. 1 to solve each cluster individually, we have a sequence of $O(n^2)$ moves for solving the entire cube. In this section, we take this sequence of moves and take advantage of parallelism to get a solution with $O(n^2/\log n)$ moves.

Say that we are given columns X and rows Y such that all of the clusters $(x, y) \in X \times Y$ are in the cluster configuration depicted in Fig. 1(b). If we solve each of these clusters individually, the number of moves required is $\Theta(|X| \cdot |Y|)$.

Consider instead what would happen if we first flipped all of the columns $x \in X$, then flipped all of the rows $y \in Y$, then flipped all of the columns $x \in X$ again, and finally flipped all of the rows $y \in Y$ again. What would be the effect of this move sequence on a particular $(x^*, y^*) \in X \times Y$? The only moves affecting that cluster are the column moves x^* and $(n-1-x^*)$ and the row moves y^* and $(n-1-y^*)$. So the subsequence of moves affecting (x^*, y^*) would consist of the column move x^*, followed by the row move y^*, followed by the column move x^* again, and finally the row move y^* again. Those four moves are exactly the moves needed to solve that cluster.

A generalization of this idea gives us a technique for solving all cubie clusters $(x, y) \in X \times Y$ using only $O(|X| + |Y|)$ moves, if each one of those clusters is in the same configuration. Our goal is to use this technique for a related problem: solving all of the cubie clusters $(x, y) \in X \times Y$ that are in a particular cluster configuration c, leaving the rest of the clusters alone.

For each $y \in Y$, we define $S_y = \{x \in X \mid \text{cluster } (x, y) \text{ is in configuration } c\}$. For each $S \subseteq X$, we define $Y_S = \{y \in Y \mid S_y = S\}$. For each of the $2^{|X|}$ values of

S, we use a single sequence of moves to solve all $(x, y) \in S \times Y_S$. This sequence of moves has length $O(|S| + |Y_S|) = O(|X| + |Y_S|)$. When we sum the lengths up for all Y_S, we find that the number of moves is bounded by

$$O\left(\sum_S (|X| + |Y_S|)\right) = O\left(|X| \cdot 2^{|X|} + \sum_S |Y_S|\right) = O\Big(|X| \cdot 2^{|X|} + |Y|\Big).$$

To make this technique cost-effective, we partition all $\lfloor n/2 \rfloor$ columns into sets of size $\frac{1}{2} \log n$, and solve each such group individually. This means that we can solve all clusters in a particular configuration c using

$$O\left(\frac{\frac{n}{2}}{\frac{1}{2}\log n} \cdot \left(\frac{1}{2}\log n \cdot 2^{\frac{1}{2}\log n} + \frac{n}{2}\right)\right) = O\left(\frac{n^2}{\log n}\right).$$

moves. When we construct that move sequence for all 6 cluster configurations, we have the following result:

Theorem 1. *Given an $n \times n \times 1$ Rubik's Cube configuration, all cubie clusters can be solved in $O(n^2/\log n)$ moves.*

3.2 $n \times n \times 1$ Lower Bound

Using calculations involving the maximum degree of the graph of the configuration space and the total number of reachable configurations, we have the matching lower bound:

Theorem 2. *Some configurations of an $n \times n \times 1$ Rubik's Cube are $\Omega(n^2/\log n)$ moves away from being solved.*

Omitted proofs may be found in the full version of this paper.

4 Diameter of $n \times n \times n$ Rubik's Cube

For simplicity, we again assume that n is even and ignore all edge and corner cubies. A more rigorous proof, which handles these details, is available in the full version of this paper.

Because the only visible cubies on the $n \times n \times n$ Rubik's Cube are on the surface, we use an alternative coordinate system. Each cubie has a face coordinate $(x, y) \in \{0, 1, \ldots, n-1\} \times \{0, 1, \ldots, n-1\}$. Consider the set of reachable locations for a cubie on the front face with coordinates (x, y). A face rotation of the front face will let it reach the coordinates $(n-y-1, x)$, $(n-x-1, n-y-1)$, and $(y, n-x-1)$ on the front face. Row or column moves will allow the cubie to move to another face, where it still has to have one of those four coordinates. Hence, it can reach 24 locations in total. We define the *cubie cluster* (x, y) to be those 24 positions that are reachable by the cubie (x, y).

Just as in the case of the $n \times n \times 1$ cube, our goal is to prove that for each cluster configuration, there is a sequence of $O(1)$ moves that can be used to solve

the cluster, while not affecting any other clusters. For the $n \times n \times 1$ cube, we wrote these solution sequences using the symbols H_1, H_2, V_1, V_2 to represent a general class of moves, each of which could be mapped to a specific move once the cubie cluster coordinates were known. Here we introduce more formal notation.

Because of the coordinate system we are using, we distinguish two types of legal moves. *Face moves* involve taking a single face and rotating it 90° in either direction. *Row or column moves* involve taking a slice of the cube (not one of its faces) and rotating the cubies in that slice by 90° in either direction. Face moves come in twelve types, two for each face. For our purposes, we will add a thirteenth type which applies the identity function. If a is the type of face move, we write F_a to denote the move itself. Given a particular index $i \in \{1, 2, \ldots, \lfloor n/2 \rfloor - 1\}$, there are twelve types of row and column moves that can be performed — three different axes for the slice, two different indices (i and $n - i - 1$) to pick from, and two directions of rotation. Again, we add a thirteenth type which applies the identity function. If a is the type of row or column move, and i is the index, then we write $RC_{a,i}$ to denote the move itself.

A *cluster move sequence* consists of three type sequences: face types $a_1, \ldots, a_\ell$, row and column types $b_1, \ldots, b_\ell$, and row and column types $c_1, \ldots, c_\ell$. For a cluster (x, y), the sequence of actual moves produced by the cluster move sequence is $F_{a_1}, RC_{b_1,x}, RC_{c_1,y}, \ldots, F_{a_\ell}, RC_{b_\ell,x}, RC_{c_\ell,y}$. A *cluster move solution* for a cluster configuration d is a cluster move sequence with the following properties:

1. For any $(x, y) \in \{1, 2, \ldots, \lfloor n/2 \rfloor - 1\} \times \{1, 2, \ldots, \lfloor n/2 \rfloor - 1\}$, if cluster (x, y) is in configuration d, then it can be solved using the sequence of moves $F_{a_1}, RC_{b_1,x}, RC_{c_1,y}, \ldots, F_{a_\ell}, RC_{b_\ell,x}, RC_{c_\ell,y}$.
2. The move sequence $F_{a_1}, RC_{b_1,x}, RC_{c_1,y}, \ldots, F_{a_\ell}, RC_{b_\ell,x}, RC_{c_\ell,y}$ does not affect cubie cluster (y, x).
3. All three of the following sequences of moves do not affect the configuration of any cubie clusters:

$$F_{a_1}, RC_{b_1,x}, F_{a_2}, RC_{b_1,x}, \ldots, F_{a_\ell}, RC_{b_\ell,x};$$
$$F_{a_1}, RC_{c_1,y}, F_{a_2}, RC_{c_1,y}, \ldots, F_{a_\ell}, RC_{c_\ell,y};$$
$$F_{a_1}, F_{a_2}, \ldots, F_{a_\ell}.$$

Our goal is to construct a cluster move solution for each possible cluster configuration, and then use those solutions to solve multiple cubie clusters in parallel.

In the speed cubing community, there is a well-known technique for solving $n \times n \times n$ Rubik's Cubes in $O(n^2)$ moves, involving a family of constant-length cluster move sequences. These sequences are attributed to Ingo Schütze [24], but due to their popularity in the speed cubing community, their exact origins are unclear. These cluster move sequences can be combined to construct constant-length cluster move solutions for all possible cluster configurations, which is precisely what we wanted. A detailed explanation and proof of correctness for this method can be found in the full version of this paper.

4.1 $n \times n \times n$ Upper Bound

As in the $n \times n \times 1$ case, we wish to solve several clusters in parallel, so that the length of the solution is reduced from $O(n^2)$ to $O(n^2/\log n)$. Say we have a set of columns $X = \{x_1, \ldots, x_\ell\}$ and rows $Y = \{y_1, \ldots, y_k\}$ such that $X \cap Y = \emptyset$ and all cubie clusters $(x, y) \in X \times Y$ have the same cluster configuration d. Solving each cluster individually requires a total of $\Theta(|X| \cdot |Y|)$ moves.

Instead, we will attempt to parallelize. The cluster configuration d must have a constant-length cluster move solution with type sequences $a_1, \ldots, a_m$, $b_1, \ldots, b_m$, and $c_1, \ldots, c_m$. To construct a parallel move sequence, we use the following sequence of moves as a building block:

$$\textsc{Bulk}_i = F_{a_i}, RC_{b_i,x_1}, RC_{b_i,x_2}, \ldots, RC_{b_i,x_\ell}, RC_{c_i,y_1}, RC_{c_i,y_2}, \ldots, RC_{c_i,y_k}.$$

The full sequence we use is $\textsc{Bulk}_1, \textsc{Bulk}_2, \ldots, \textsc{Bulk}_m$. A careful case-by-case analysis using the properties of cluster move solutions reveals that this sequence of $O(|X| + |Y|)$ moves will solve all clusters $X \times Y$, and that the only other clusters it may affect are the clusters $X \times X$ and $Y \times Y$.

Now say that we are given a cluster configuration d and a set of columns X and rows Y such that $X \cap Y = \emptyset$. Using the same row-grouping technique that we used for the $n \times n \times 1$ case, it is possible to show that there exists a sequence of moves of length $O(|X| \cdot 2^{|X|} + |Y|)$ solving all of the clusters in $X \times Y$ which are in configuration d and limiting the set of other clusters affected to $(X \times X) \cup (Y \times Y)$. By dividing up X into groups of roughly size $\frac{1}{2} \log |Y|$, just as we did for the $n \times n \times 1$ cube, we may show that there exists a sequence of moves with the same properties, but with length $O(|X| \cdot |Y| / \log |Y|)$.

To finish constructing the move sequence for the entire Rubik's Cube, we must account for two differences between this case and the $n \times n \times 1$ case: the requirement that $X \cap Y = \emptyset$ and the potential to affect clusters in $(X \times X) \cup (Y \times Y)$. We handle both cases by taking the initial set of columns $\{1, 2, \ldots, \lfloor n/2 \rfloor - 1\}$ and dividing it into groups $X_1, \ldots, X_j$ of size $\sqrt{n/2}$. We partition the initial set of rows into sets $Y_1, \ldots, Y_j$ in a similar fashion. We then loop through pairs (X_i, Y_j), where $i \neq j$, to solve all clusters in configuration d for all but the clusters $(X_1 \times Y_1) \cup \ldots \cup (X_j \times Y_j)$. Because $j = |X_i| = |Y_i| = \sqrt{n/2}$, the total number of moves required for this step is $O(n^2/\log n)$. To solve the clusters $(X_1 \times Y_1) \cup \ldots \cup (X_j \times Y_j)$, we simply solve each cluster individually, which requires a total of $O(n^{3/2}) < O(n^2/\log n)$ moves. If we add up that cost for each of the $O(1)$ different configurations, the total number of moves is $O(n^2/\log n)$.

Theorem 3. *Given an $n \times n \times n$ Rubik's Cube configuration, all cubie clusters can be solved in $O(n^2/\log n)$ moves.*

4.2 $n \times n \times n$ Lower Bound

Just as we did for the $n \times n \times 1$ lower bound, we can calculate a matching lower bound using the maximum degree of the graph of the configuration space and the total number of reachable configurations:

Theorem 4. *Some configurations of an $n \times n \times n$ Rubik's Cube are $\Omega(n^2/\log n)$ moves away from being solved.*

5 Optimally Solving a Subset of the $n \times n \times 1$ Rubik's Cube is NP-Hard

In this section, we consider a generalization of the problem of computing the optimal sequence of moves to solve a Rubik's Cube. Say that we are given a configuration of an $n \times n \times 1$ Rubik's Cube and a list of *important* cubies. We wish to find the shortest sequence of moves that solves the important cubies. Note that the solution for the important cubies may cause other cubies to leave the solved state, so this problem is only equivalent to solving an $n \times n \times 1$ Rubik's Cube when all cubies are marked important.

In this section, we prove the NP-hardness of computing the length of this shortest sequence. More precisely, we prove that the following decision problem is NP-hard: is there a sequence of k moves that solves the important cubies of the $n \times n \times 1$ Rubik's Cube? Our reduction ensures that the cubies within a single cluster are either all important or all unimportant, and thus it does not matter whether we aim to solve cubies (which move) or specific cubie positions (which do not move). Therefore the problem remains NP-hard if we aim to solve the puzzle in the sense of unifying the side colors, when we ignore the colors of all unimportant cubies.

Certain properties of the Rubik's Cube configuration can affect the set of potential solutions. For the rest of this section, we will consider only Rubik's Cubes where n is odd and where all edge cubies and cross cubies are both solved and marked important. This restriction ensures that for any cluster, the number of horizontal moves and vertical moves affecting it must both be even. In addition, we will only consider Rubik's Cubes in which all cubie clusters are in the cluster configurations depicted in Figures 1(a), 1(b), and 1(d). This restriction means that the puzzle can always be solved using moves only of types H_1 and V_1. This combination of restrictions ensures that each unsolved cluster must be affected by both vertical and horizontal moves.

Suppose that we are given a configuration and a list of important cubies. Let u_r be the number of rows of index $\leq \lfloor n/2 \rfloor$ that contain at least one important unsolved cubie. Let u_c be the number of columns of index $\leq \lfloor n/2 \rfloor$ that contain at least one important unsolved cubie. Then we say that the *ideal number of moves* for solving the given configuration is $2(u_r + u_c)$. In other words, the ideal number of moves is equal to the smallest possible number of moves that could solve all the important cubies. An *ideal solution* for a subset of the cubies in a particular $n \times n \times 1$ puzzle is a solution for that set of cubies which uses the ideal number of moves. For the types of configurations that we are considering, the ideal solution will contain exactly two of each move, and the only moves that occur will be moves of type H_1 or V_1.

Definition 1. *Let $I_k(m)$ denote the index in the solution of the kth occurrence of move m.*

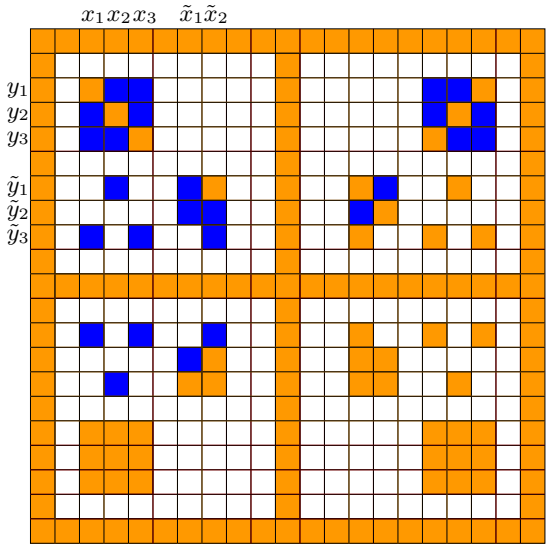


Fig. 2. A sample of the betweenness gadget from Lemma 1. Important cubies are orange (solved) and blue (unsolved). Unimportant cubies are white. Any ideal solution must either have $I_1(x_1) < I_1(x_2) < I_1(x_3)$ or $I_1(x_3) < I_1(x_2) < I_1(x_1)$.

For our hardness reduction, we develop a gadget (depicted in Fig. 2) which forces a betweenness constraint on the ordering of three different row moves:

Lemma 1. *Given three columns $x_1, x_2, x_3 \leq \lfloor n/2 \rfloor$, there is a gadget using six extra rows and two extra columns ensuring that $I_1(x_2)$ lies between $I_1(x_1)$ and $I_1(x_3)$. This gadget also forces $I_2(x_2) < I_2(x_1)$, $I_2(x_2) < I_2(x_3)$, and*

$$\max_{x \in \{x_1, x_2, x_3\}} I_1(x) < \min_{x \in \{x_1, x_2, x_3\}} I_2(x).$$

The *betweenness problem* is a known NP-hard problem [8,19]. In this problem, we are given a set of triples (a, b, c), and wish to find an ordering on all items such that, for each triple, either $a < b < c$ or $c < b < a$. In other words, for each triple, b should lie between a and c in the overall ordering. Lemma 1 gives us a gadget which would at first seem to be perfectly suited to a reduction from the betweenness problem. However, because the lemma places additional restrictions on the order of all moves, we cannot reduce directly from betweenness.

Instead, we provide a reduction from another known NP-hard problem, Not-All-Equal 3-SAT [8,23]. In this problem, sometimes known as $\neq$-SAT, the input is a 3-CNF formula ϕ and the goal is to determine whether there exists an assignment to the variables of ϕ such that there is at least one true literal and one false literal in every clause. Our reduction from $\neq$-SAT to ideal Rubik solutions closely follows the reduction from hypergraph 2-coloring to betweenness [19].

Theorem 5. *Given a $\neq$-SAT instance ϕ, it is possible to compute in time polynomial in the size of ϕ an $n \times n \times 1$ configuration and a subset of the cubies that has an ideal solution if and only if ϕ has a solution, i.e., belongs to $\neq$-SAT.*

6 Optimally Solving an $O(1) \times O(1) \times n$ Rubik's Cube

For the $c_1 \times c_2 \times n$ Rubik's Cube with $c_1 \neq n \neq c_2$, the asymmetry of the puzzle leads to a few additional definitions. We call a slice *short* if the matching coordinate is z; otherwise, a slice is *long*. A *short move* involves rotating a short slice; a *long move* involves rotating a long slice. We define cubie cluster i to be the pair of slices $z = i$ and $z = (n-1) - i$. This definition means that any short move affects the position and orientation of cubies in exactly one cubie cluster.

Each short move affects exactly one cluster. Hence, in the optimal solution, the number of short moves affecting a particular cluster is at most the number of configurations of that cluster. Each cluster has $O(1)$ configurations, so in any optimal solution, any particular short move will be performed $O(1)$ times.

Any sequence of long moves corresponds to an arrangement of $c_1 c_2$ blocks of cubies with dimensions $1 \times 1 \times n$. We call each such arrangement a *long configuration*. There are a constant number of long configurations, so there must exist a *long move tour*: a constant-length sequence of long moves which passes through every long configuration before returning to the initial long configuration.

The effect of a short move depends only on the current long configuration. Hence, to solve a particular $c_1 \times c_2 \times n$ puzzle, it is sufficient to know a sequence of short moves for each cluster, annotated with the long configuration that each such move should be performed in. If we have such a sequence for each cluster, we may construct a full solution to the puzzle by repeatedly performing a long move tour, and inserting short moves into the appropriate places. Then we are guaranteed to be able to perform the kth short move for every cluster during the kth long move tour. Hence, the number of long move tours necessary is bounded by the maximum length of any short move sequence, which is $O(1)$ in any optimal solution. Therefore, any optimal solution contains $O(1)$ long moves.

This bound on the number of long moves allows us to construct an algorithm that does the following:

Theorem 6. *Given any $c_1 \times c_2 \times n$ Rubik's Cube configuration, it is possible to find the optimal solution in time polynomial in n.*

References

1. World Cube Association. Official results. http://www.worldcubeassociation.org/results/, 2010.
2. Stephen A. Cook. Can computers routinely discover mathematical proofs? *Proceedings of the American Philosophical Society*, 128(1):40–43, 1984.
3. James R. Driscoll and Merrick L. Furst. On the diameter of permutation groups. In *Proceedings of the 15th Annual ACM Symposium on Theory of computing*, pages 152–160, 1983.

4. Andy Drucker and Jeff Erickson. Is optimally solving the $n \times n \times n$ Rubik's Cube NP-hard? Theoretical Computer Science — Stack Exchange post, August–September 2010. http://cstheory.stackexchange.com/questions/783/is-optimally-solving-the-nnn-rubiks-cube-np-hard.
5. Shimon Even and Oded Goldreich. The minimum-length generator sequence problem is NP-hard. *Journal of Algorithms*, 2(3):311–313, 1981.
6. Frank Fox. Spherical 3x3x3. U.K. Patent 1,344,259, January 1974.
7. Merrick Furst, John Hopcroft, and Eugene Luks. Polynomial-time algorithms for permutation groups. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, pages 36–41, 1980.
8. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman & Co Ltd, first edition edition, January 1979.
9. Wiliam O. Gustafson. Manipulatable toy. U.S. Patent 3,081,089, March 1963.
10. Terutoshi Ishige. Japan Patent 55-8192, 1976.
11. Mark R. Jerrum. The complexity of finding minimum-length generator sequences. *Theoretical Computer Science*, 36(2–3):265–289, June 1985.
12. Graham Kendall, Andrew Parkes, and Kristian Spoerer. A survey of np-complete puzzles. *International Computer Games Association Journal*, 31(1):13–34, 2008.
13. Udo Krell. Three dimensional puzzle. U.S. Patent 4,600,199, July 1986.
14. Leslie Le. The world's first 12x12x12 cube. twistypuzzles.com forum post, November 2009. http://www.twistypuzzles.com/forum/viewtopic.php?f=15&t=15424.
15. Seven Towns Ltd. 30 years on... and the Rubik's Cube is as popular as ever. Press brief, May 2010. http://www.rubiks.com/i/company/media_library/pdf/Rubiks%20Cube%20to%20celebrate%2030th%20Anniversary%20in%20May%202010.pdf.
16. Pierre McKenzie. Permutations of bounded degree generate groups of polynomial diameter. *Information Processing Letters*, 19(5):253–254, November 1984.
17. Larry D. Nichols. Pattern forming puzzle and method with pieces rotatable in groups. U.S. Patent 3,655,201, April 1972.
18. Museum of Modern Art. Rubik's cube. http://www.moma.org/collection/browse_results.php?object_id=2908.
19. Jaroslav Opatrny. Total ordering problem. *SIAM Journal on Computing*, 8(1):111–114, 1979.
20. Ian Parberry. A real-time algorithm for the (n^2-1)-puzzle. *Information Processing Letters*, 56(1):23–28, 1995.
21. Daniel Ratner and Manfred Warmuth. The (n^2-1)-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10:111–137, 1990.
22. Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge. God's number is 20, 2010. http://cube20.org.
23. Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 216–226, San Diego, CA, 1978.
24. Ingo Schütze. V-cubes Solutions. http://solutions.v-cubes.com/solutions2/.
25. Peter Sebesteny. Puzzle-cube. U.S. Patent 4,421,311, December 1983.
26. Jerry Slocum. *The Cube: The Ultimate Guide to the World's Bestselling Puzzle — Secrets, Stories, Solutions*. Black Dog & Leventhal Publishers, March 2009.
27. V-CUBE. V-cube: the 21st century cube. http://www.v-cubes.com/.
28. Oskar van Deventer. Overlap cube 2x2x23. shapeways design. http://www.shapeways.com/model/96696/overlap_cube_2x2x23.html.