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 of the 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

[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.

example, the recent movie *Exit Through the Gift Shop* features the street cube 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]; even writing down the first move in each solution would take about 8 exabytes (after factoring out symmetries). 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 [28] whose design enables cubes as large as $11 \times 11 \times 11$ according to Verdes's design patent [30]; 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* [29].

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 show 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 indeed 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 $\Omega(n^2/\log n)$, because the number of configurations is $2^{\Theta(n^2)}$ and there are $O(n)$ possible moves from each configuration.

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 lacking any form of parallelism, the diameter is simply $\Theta(n^3)$ [20]. Interestingly, however, computing the shortest solution from a given configuration of the puzzle is NP-complete [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 address the natural question: is it NP-complete 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 [29], 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.

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]. In order 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 will change 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 red and the bottom of the cube is blue.

Consider the set of locations that a cubie at position (x, y) can reach. If we flip column x, the cubie will move to position $(x, n-y-1)$. If we instead flip row y, the cubie will move to position $(n-x-1, y)$. Consequently, 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).

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

If a cubie is in the first or last row or column, we call it an *edge cubie*. If a cluster contains an edge cubie, then we call it an *edge cluster*, because all of its cubies are edge cubies. The special edge cluster which contains the cubie in the first row and first column is called the *corner cluster*, and its cubies are *corner cubies*. If n is odd, then there is a cluster with one cubie which is in both the median row and the median column. We will call this cubie the *center cubie* or *center cluster*. In addition, if n is odd then there are also clusters with two cubies found in the median row or column. We call the clusters *cross clusters* and the cubies in them *cross cubies*.

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.

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. We also use the same symbols to denote single moves affecting these rows and columns. Recall that for a $n \times n \times 1$ cube there is only one valid non-identity operation on a row or column: rotation by 180°. 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.

We begin by proving the following lemma about cubie cluster configurations.

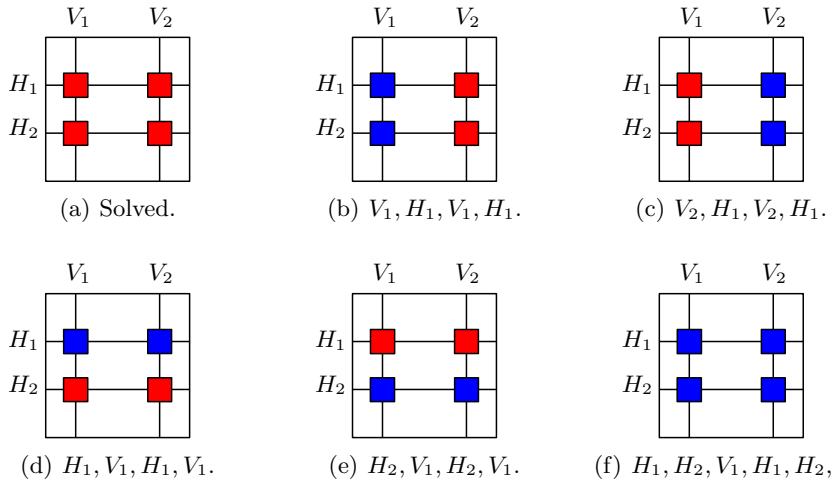
Lemma 1. *In a solvable $n \times n \times 1$ Rubik's Cube, the colors on the top faces of the cubies in a cubie cluster can only be in the six configurations in Fig. 1.*

Proof. Consider what happens to a cubie cluster when a move is performed. If the move does not affect the cubie cluster, then its cubie configuration will not change. Otherwise, the move will swap two cubies in the same row or column while reversing the color of each cubie. If both cubies are the same color, then both cubies will become the other color. In other words, if one cubie is red and the other is blue, then the color configuration will not change.

Figure 1(a) shows the solved configuration, which is quite obviously reachable. The four moves which affect this cubie cluster result in configurations 1(b), 1(c), 1(d), and 1(e). Consider how the four possible moves affect each of configurations 1(b), 1(c), 1(d), and 1(e). For each configuration, two of the four possible moves involve one red cubie and one blue cubie, and therefore do not affect the colors. In addition, one move for each configuration is the inverse of the move used to reach that configuration, and therefore leads back to configuration 1(a). The final move for each configuration results in configuration 1(f), thus completing the set of reachable configurations. □

Because of these limitations on cubie cluster configurations, we can prove the following lemma.

Lemma 2. *All six cluster configurations from Lemma 1 can be solved using a sequence of at most six moves that does not affect the position of any cubies not in the cubie cluster. (See Fig. 1.)*



(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.

Proof. The correct move sequence for each configuration from Lemma 1 is given in Fig. 1. The fact that we always use each move twice ensures that all other clusters will not be affected by the move sequence. The correctness of these sequences can be verified by the reader. □

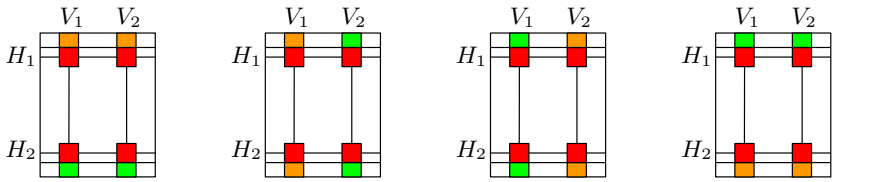
In order to handle the edge, corner, and cross clusters, we need a more complicated sequence of moves. These clusters cannot always be solved without affecting any other cubes. So rather than show that we can solve each cluster individually, we show that we can solve all such clusters together.

Lemma 3. *Given a solvable configuration of an $n \times n \times 1$ Rubik's Cube, there exists a sequence of moves of length $O(n)$ which can be used to solve the edge clusters and cross clusters.*

Proof. We begin by solving the corner cluster, and, if n is odd, the center cluster and the two edge cross clusters. These four clusters combined have only $O(1)$ reachable configurations, and so all four can be obviously be fixed in $O(1)$ moves, disregarding the effect that these moves might have on any other clusters. Next we solve the other edge clusters. Our goal is to solve each cluster without affecting any of the clusters we have previously solved.

Without loss of generality, say that we are trying to solve an edge cluster with coordinates $(x, 0)$. We begin by using the move sequences from Lemma 2 to make sure that the cluster has all four red stickers visible. Then it will be in one of the states depicted in Fig. 2. To solve the cluster, we can use the move sequences given in Fig. 2. Although the given move sequences are not guaranteed to apply the identity permutation to all other clusters, they do have the property that any horizontal move will be performed an even number of times. Hence, this

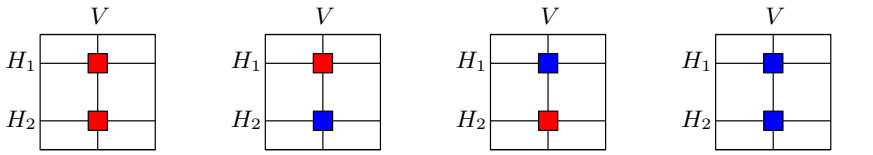
move sequence will apply the identity permutation to all other edge clusters. In addition, none of the move sequences affect the center cubie.



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

Fig. 2. When all four red stickers are facing forwards, these are the possible configurations for an edge cluster, and the move sequences used to solve them.

Once all the edge clusters have been solved, we want to solve the cross clusters. We know that the center cluster has already been solved. We also know that there are $O(n)$ cross clusters, so if we can solve each cross cluster in $O(1)$ moves without affecting the rest of the clusters, then we will have solved the edge and cross clusters in a total of $O(n)$ moves. Without loss of generality, say that we are trying to solve the cross cluster $((n-1)/2, y)$. The four possible states for a cross cluster are depicted in Fig. 3.



(a) Solved. (b) n/a. (c) n/a. (d) H_1, V, H_1, V.

Fig. 3. The four reachable configurations for a cross cluster. Two of them can be solved without affecting the rest of the clusters.

However, because we have already solved all of the edge clusters, we know that the set of possible configurations for our cross cluster is more restricted. Both horizontal moves affecting our cross cluster will cause one of the two cubies to change color. No matter what state the cross cluster is in, the vertical move cannot change the color of only one cubie. Therefore, if the cross cluster is in the configuration depicted in Fig. 3(b) or the configuration depicted in Fig. 3(c), then the rest of the solution must perform either the move H_1 or the move H_2 an odd number of times.

Consider the effect of this on the edge cluster affected by H_1 and H_2. Each move in the edge cluster causes a swap of two cubies. If an order is placed on

the cubies in the cluster, each swap is a permutation of this order, and the set of permutations constructible using swaps is equivalent to the permutation group on 4 elements, S_4. By permutation group theory, if a particular cluster configuration can be solved using an even number of swaps, then *any* solution for that cluster configuration has an even number of swaps. We know that the edge cluster is already solved, so it can be solved using an even number of swaps. So if the rest of the solution contains an odd number of moves H_1 and H_2, then it must also contain an odd number of edge moves V_1 and V_2.

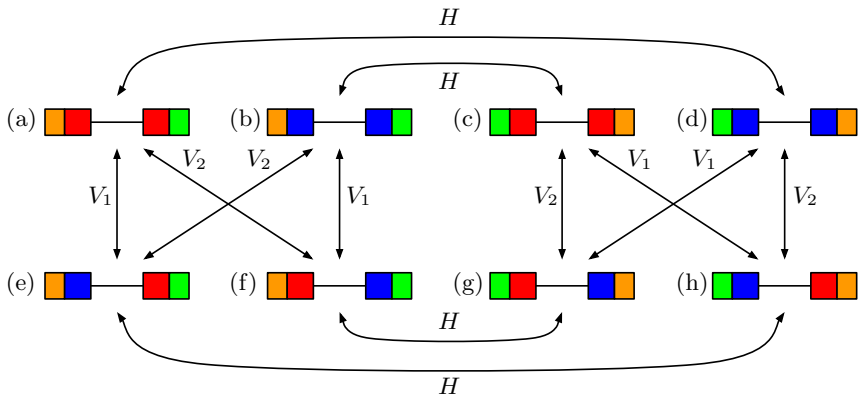


Fig. 4. The possible configurations of an edge cross cluster.

Now consider the effect of an odd number of edge moves on the affected edge cross cluster. Figure 4 gives the configuration space for that cluster. We know that it is currently in the solved state labelled (a). Any sequence of moves which contains an odd number of edge moves V_1 and V_2 will cause the edge cross cluster to leave the solved state. So the rest of the solution cannot contain an odd number of edge moves V_1 and V_2. This means that the rest of the solution cannot contain an odd number of horizontal moves H_1 and H_2 affecting a single cross cluster. So every cross cluster must be in one of the states depicted in Figs. 3(a) and 3(d), each of which can be solved without affecting any other clusters using the sequence of moves listed for each state. □

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 cubie clusters $(x, y) \in X \times Y$ are in the configuration depicted in Fig. 1(b). If we attempted

to solve each of these clusters individually, the number of moves required would be $O(|X| \cdot |Y|)$.

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.

This idea allows us to parallelize the solutions for multiple clusters, resulting in the following lemma.

Lemma 4. *Given an $n \times n \times 1$ Rubik's Cube configuration and sets $X, Y \subseteq \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$, if all cubie clusters $(x, y) \in X \times Y$ are in the same cluster configurations, then all cubie clusters $(x, y) \in X \times Y$ can be solved in $O(|X|+|Y|)$ moves without affecting the rest of the cubies.*

Proof. By Lemma 2, we know that there exists a sequence of moves of length at most six which will solve the configuration of a single cubie cluster $(x, y) \in X \times Y$. We write this sequence of moves in a general form as in Lemma 2. We then replace each move V_1 with a sequence of moves that flips each column $x \in X$. Similarly, we replace each move V_2 with a sequence of moves that flips each column $n - x - 1$, where $x \in X$. We perform similar substitutions for H_1 and H_2, using the rows y and $n - y - 1$ instead. Because the original sequence of moves had length at most six, it is easy to see that the length of the new move sequence is $O(|X| + |Y|)$.

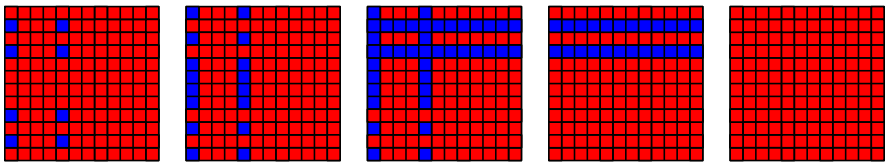


Fig. 5. Solving four cubie clusters at the same time.

We claim that this new sequence of moves will solve all cubie clusters $(x, y) \in X \times Y$, and that no other cubie clusters will be affected. To see that this is true, consider how some cubie cluster $(x^*, y^*) \in \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\} \times \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$ will be affected by the move sequence. If $x^* \notin X$ and $y^* \notin Y$, then none of the moves in the sequence will affect the position of any cubies in the cubie cluster, and therefore the cubie cluster will be unaffected. If $x^* \in X$ and $y^* \in Y$, then

the subsequence of moves which affect this cubie cluster will be exactly the sequence of moves necessary to solve this cubie cluster. Otherwise, either $x^* \in X$ or $y^* \in Y$, but not both. Therefore, the subsequence of moves which affect this cubie cluster will be either all vertical, or all horizontal, and each move will occur exactly twice. This means that the subsequence of moves which affect this cubie cluster will apply the identity permutation to this cubie cluster. □

This technique allows us to solve 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. To that end, we divide up the columns X according to the pattern of rows that are in configuration c, and solve each subset of the rows using the technique of Lemma 4. More formally:

Lemma 5. *Suppose we are given an $n \times n \times 1$ Rubik's Cube configuration, a cluster configuration c, and sets $X, Y \subseteq \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$ such that $|X| = \ell$. Then all cubie clusters $(x, y) \in X \times Y$ that are in configuration c can be solved in $O(\ell 2^\ell + |Y|)$ moves without affecting the rest of the cubies.*

Proof. For each $y \in Y$, we define $S_y = \{x \in X \mid \text{the cubie cluster } (x, y) \text{ is in configuration } c\}$. For each $S \subseteq X$, we let $Y_S = \{y \in Y \mid S_y = S\}$. There are 2^ℓ possible values for S. For each Y_S, we use the sequence of moves which is guaranteed to exist by Lemma 4 to solve all $(x, y) \in S \times Y_S$. This sequence of moves has length $O(|S| + |Y_S|) = O(\ell + |Y_S|)$. When we sum the lengths up for all Y_S, we find that the number of moves is bounded by

$$O\left(\ell \cdot 2^\ell + \sum_S |Y_S|\right) = O\left(\ell \cdot 2^\ell + |Y|\right).$$

□

Unfortunately, this result is not cost-effective. We need to make sure that ℓ is small enough to prevent an exponential blowup in the cost of solving the Rubik's Cube. To that end, we divide up the columns into small groups to get the following result:

Lemma 6. *Suppose we are given an $n \times n \times 1$ Rubik's Cube configuration, a cluster configuration c, and sets $X, Y \subseteq \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$. Then all cubie clusters $(x, y) \in X \times Y$ that are in configuration c can be solved in $O(|X| \cdot |Y| / \log |Y|)$ moves without affecting the rest of the cubies.*

Proof. Let $\ell = \frac{1}{2} \log_2 |Y|$, so that $2^\ell = \sqrt{|Y|}$. Let $k = \lceil |X|/\ell \rceil$. Partition the set X into a series of sets $X_1, \ldots, X_k$ each of which has size $\leq \ell$. For each X_i, we solve the cubie clusters in $X_i \times Y$ using the sequence of moves that is guaranteed to exist by Lemma 5. The number of moves required to solve a single X_i is

$$O\left(\ell 2^\ell + |Y|\right) = O\left(\left(\frac{1}{2} \log_2 |Y|\right) \sqrt{|Y|} + |Y|\right) = O(|Y|).$$

Therefore, if we wish to perform this for k sets, the total number of moves becomes:

$$O\left(k \cdot |Y|\right) = O\left(\frac{|X|}{\frac{1}{2}\log_2 |Y|} \cdot |Y|\right) = O\left(\frac{|X| \cdot |Y|}{\log |Y|}\right)$$

□

As a result of this parallelization, we get the following upper bound on the diameter of the configuration space:

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.*

Proof. In order to solve the Rubik's Cube, we must solve all cubie clusters $(x, y) \in \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\} \times \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$. To do so, we loop through the six possible cluster configurations, then use the sequence of moves guaranteed to exist by Lemma 6 to solve all of the cubie clusters which are in a particular configuration. For a single configuration, the number of moves that this generates is

$$O\left(\frac{\lfloor n/2 \rfloor \cdot \lfloor n/2 \rfloor}{\log(\lfloor n/2 \rfloor)}\right) = O\left(\frac{n^2}{\log n}\right).$$

When we add this cost up for the six different configurations, the total number of moves is $6 \cdot O(n^2/\log n) = O(n^2/\log n)$. □

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

In this section, we establish 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.*

Proof. Lemma 2 shows that for every possible configuration of a cubie cluster, there exists a sequence of moves to solve the cubie cluster while leaving the rest of the cubies in the same location. Hence, the inverse of such a sequence will transform a solved cubie cluster to an arbitrary cluster configuration without affecting any other cubies. Not counting the edge cubies and the cross cubies, there are $(\lfloor n/2 \rfloor - 1)^2$ cubie clusters, each of which can be independently placed into one of six different configurations. This means that there are at least $6^{(\lfloor n/2 \rfloor - 1)^2}$ reachable configurations.

There are $2n$ possible moves. Therefore, the total number of states reachable using at most k moves is at most

$$\frac{(2n)^{k+1} - 1}{2n - 1} \leq (2n)^{k+1}.$$

Therefore, if k is the number of moves necessary to reach all states, it must have the property that:

$$\begin{aligned} 6^{(\lfloor n/2 \rfloor - 1)^2} &\leq (2n)^{k+1}, \\ (\lfloor n/2 \rfloor - 1)^2 &\leq \log_6 \left((2n)^{k+1}\right) = \frac{(k+1)\log 2n}{\log 6}, \\ \frac{(\lfloor n/2 \rfloor - 1)^2 \log 6}{\log 2n} - 1 &\leq k. \end{aligned}$$

Hence, there must exist some configurations which are $\Omega(n^2/\log n)$ moves away from solved. □

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

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 with coordinates $(x, y) \in \{0, 1, \ldots, n-1\} \times \{0, 1, \ldots, n-1\}$ on the front face. 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. For this problem, we define the cubie cluster (x, y) to be those 24 positions that are reachable by the cubie (x, y).

We define *edge cubies* to be cubies with more than one face visible. If a cluster has an edge cubie, then all of its cubies are edge cubies. We call such clusters *edge clusters*. We define *corner cubies* to be cubies with more than two faces visible. All corner cubies are in a single cluster known as the *corner cluster*. If n is odd, we must also define several other types of cubies. We first define *cross cubies* to be cubies with face coordinates of the form $(x, (n-1)/2)$ or $((n-1)/2, y)$. If a cluster contains a cross cubie, then all of its cubies are cross cubies, and the cluster is called a *cross cluster*. We define *center cubies* to be the six cubies with face coordinates $((n-1)/2, (n-1)/2)$. They form a special cluster which we will call the *center cluster*. Our goal in solving the Rubik's Cube will be to make each side match the color of its center cluster. Hence, there is no need to solve the center cluster.

Given a particular cluster configuration, this configuration can be converted to the solved color configuration by performing a sequence of pairwise cubie swaps. If an order is placed on the cubies in the cluster, as in Figure 6, each pairwise cubie swap is a permutation of this order, and the set of all cubies swaps generates the permutation group on 24 elements, S_{24}. By permutation group theory, if an even (odd) number of swaps can be applied to a color configuration to transform it to the solved color configuration, then *any* sequence of swaps transforming a configuration to the solved configuration has an even (odd) number of swaps. We call this the *parity* of a color configuration.

Just as it was for 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 which 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 cubie 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 cubie cluster configuration, and then use the cluster move solution 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 cluster move sequences can be combined to construct constant-length cluster move solutions for all possible cluster configurations, which is precisely what we wanted.

Before dealing with the general solution, we address fixing the parity of the cubie clusters. This allows us to assume that the parity of all clusters is even for the remainder of the paper.

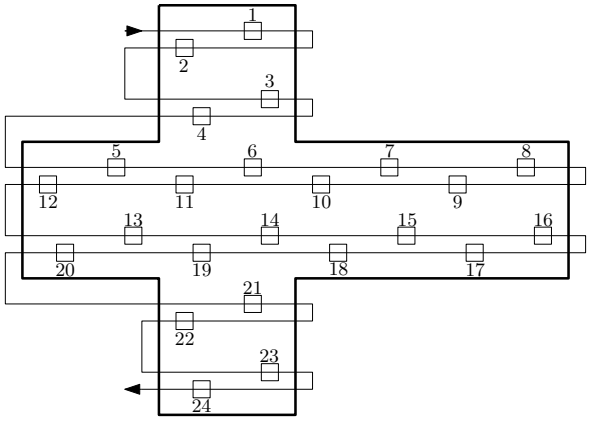


Fig. 6. An ordering of a 24-cubie cluster.

Lemma 7. *Given a solvable $n \times n \times n$ Rubik's Cube configuration, the parity of all cubie clusters can be made even in $O(n)$ moves.*

Proof. By definition, the center cluster is already solved, and therefore we may assume that its parity is already even. In addition, any cluster containing at least two indistinguishable cubies can be considered to have even parity or odd parity depending on the chosen label for the indistinguishable cubies. Therefore, we may assume that all such clusters have even parity. This means that all non-edge clusters, including the non-edge cross clusters, can be assumed to have the correct parity no matter how many moves are performed. So we need only fix the parity of the edge clusters.

We begin by fixing the parity of the corner cluster and the edge cross cluster (if it exists). Because the cube is solvable, we know that the corner cluster and the edge cross cluster can be solved. Because the corner cluster has $O(1)$ reachable states and the edge cross cluster has $O(1)$ reachable states, we know that we can solve both in $O(1)$ moves. Once those two clusters are solved, we know that their parities must be correct. Therefore, there is a sequence of $O(1)$ moves which can be used to fix the parity of those clusters.

Consider the effect of a face move on the parity of a non-cross edge cluster. For a particular edge cluster, a face move affects the location of eight cubies, due to the fact that a face move also acts like a row or column move for edge cubie groups. The color of each cubie is rotated 90° in the direction of the face's rotation. This means that the permutation applied consists of two permutation cycles each containing four elements. Therefore, if the elements whose colors

are changed are $1, 2, 3, \ldots, 8$, then we can write the applied permutation as $(1\ 3)(1\ 5)(1\ 7)(2\ 4)(2\ 6)(2\ 8)$, or six swaps. Hence face moves cannot be used to fix the parity of the edge clusters.

Now consider the effect of a row or column move on the parity of a non-cross edge cluster. A row or column move affects the colors of four cubies, one for each corner of the rotated slice. The color of each cubie is transferred to the adjacent cubie in the direction of the move rotation. So if the elements whose colors are changed are $1, 2, 3, 4$, then the applied permutation is $(1\ 2\ 3\ 4) = (1\ 2)(1\ 3)(1\ 4)$. Because the permutation can be written as an odd number of swaps, the parity of the cluster has changed. Note, however, that there is exactly one edge cluster whose parity is affected by this movement. Therefore, we can correct the parity of each odd edge cluster by performing a single row or column move that affects the cluster in question. The total number of moves required is therefore proportional to the number of edge clusters, or $O(n)$. □

Lemma 8. *For each permutation in Table 1, there exists a cluster move sequence of length $O(1)$ which can be used to apply the given permutation to the cluster while applying the identity permutation to every other cluster.*

Proof. First, we introduce some more notation for Rubik's Cube moves. It is more specific than the formalism of the cluster move sequence introduced above, making it easier to express a particular set of moves, but is more difficult to analyze when we parallelize this move sequence.

Consider facing the cube from in front (the front face is the face in the xz-plane with the more negative y-value). From this view, there are horizontal moves that rotate a slice of the cube parallel to the xy-plane. Rotating the ith slice from the top in the clockwise direction 90° as viewed from above the cube is denoted by H_i^{CW}. Rotating this same slice in the opposite direction is denoted by H_i^{CCW}. Similarly, rotating a slice parallel to the yz-plane is a vertical move, and rotating the jth slice from the left side of the cube in the clockwise direction as viewed from left of the cube is denoted V_j^{CW}, while the counter-clockwise version is denote V_j^{CCW}. Finally, we define a third type of move which rotates a slice parallel to the front face. Counting inward from the front face, we denote rotating the kth slab 90° clockwise as D_k^{CW}, while rotating it 90° in the opposite direction is D_k^{CCW}. See Figure 7.

We claim that the move sequence $S = V_n^{CCW} \circ D_m^{CW} \circ V_n^{CW} \circ D_m^{CCW} \circ H_0^{CW} \circ D_m^{CW} \circ V_n^{CCW} \circ D_m^{CCW} \circ V_n^{CW} \circ H_0^{CCW}$ swaps the colors of three cubies in a single cluster, while leaving the color configurations of all other clusters the same. This move sequence is attributed to Ingo Schütze [24], but is fairly well-known within the speed cubing community, so its origins are unclear. The effect of applying S is shown by case analysis of individual cubies. When applying the sequence S, only cubies lying on the union of the slices rotated by the V_n, D_n, and H_0 moves are affected.

Blocks lying on the bottom face are unaffected, as they never reach the top face and thus only have the subsequence $V_n^{CCW} \circ D_m^{CW} \circ V_n^{CW} \circ D_m^{CCW} \circ D_m^{CW} \circ V_n^{CCW} \circ D_m^{CCW} \circ V_n^{CW}$ applied, which does not affect the final location of a

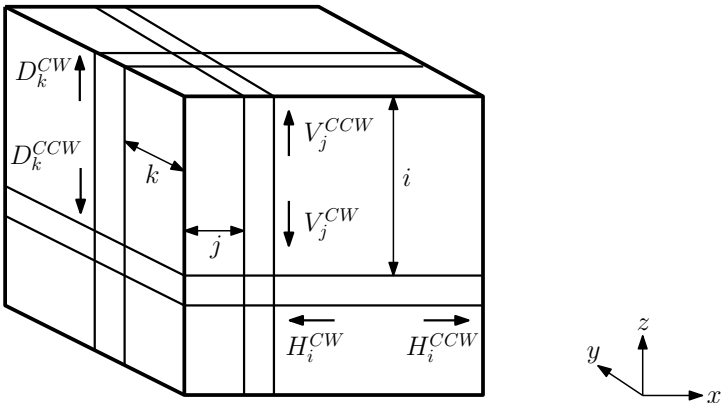


Fig. 7. The definitions of the various moves defined on an $n \times n \times n$ Rubik's Cube.

cube. Blocks starting on the back and right faces never move to the upper face, as each move that could place these on the upper face (moves V_n^{CW} and D_m^{CCW}) is preceded by a move rotating these cubies onto the bottom face (moves V_n^{CCW} and D_m^{CW}).

Now consider the cubies on left face in the slice rotated by D_m^{CW} and D_m^{CCW}. Exactly one of these cubies is in the same cluster as the cubie on the upper face that lies in the slices rotated by *both* V_n^{CW} and D_m^{CW}. All other such cubies cannot be affected by D_m^{CW} and D_m^{CCW} moves, so applying S has the same effect as applying the sequence $D_m^{CW} \circ D_m^{CCW} \circ H_0^{CW} \circ D_m^{CW} \circ D_m^{CCW} \circ H_0^{CCW}$. Canceling the D_m^{CW} and D_m^{CCW} moves yields H_0^{CW}, H_0^{CCW} and thus these cubies are unaffected. Now consider the single cubie on the left face in the same cluster as the cubie on the upper face that lies in the slices rotated by *both* V_n^{CW} and D_m^{CW}. Tracing the locations visited by this cubie when S is applied to it shows that the cubie travels to the upper face (via D_m^{CW}), the front face (via L_n^{CW}), the upper face (via V_n^{CCW}), and then the left face (via D_m^{CCW}). So the cubie's location is unaffected by S.

Next consider the cubies on the front face. Exactly one of these cubies is in the same cluster as the cubie on the upper face that lies in the slices rotated by both V_n^{CW} and D_m^{CW}. All other such cubies cannot be affected by V_n^{CW} and V_n^{CCW} moves, so applying S has the same effect as applying the sequence $D_m^{CW} \circ D_m^{CCW} \circ H_0^{CW} \circ D_m^{CW} \circ D_m^{CCW} \circ H_0^{CCW}$. For the cubie in the same cluster as the cubie on the upper face that lies on both the V_n^{CW} and D_m^{CW} slices, applying S to it results in moving it, in sequence, to the left side of the upper face (V_n^{CCW}), right face (D_m^{CW}), upper face (D_m^{CCW}), the back side of the upper face (H_0^{CW}), and the left side of the upper face (H_0^{CCW}). So applying S to this cubie moves it to the location of the cubie in its cluster in the left side of the upper face.

Finally, consider the cubies on the upper face. Divide the cubies into three sets based upon the cubies in their clusters. Each cubie on the upper face either

is in the cluster containing the cubie lying in slices rotated by both V_n^{CW} and D_m^{CW}, or is in a cluster containing a cubie lying in exactly one of the slices rotated by V_n^{CW} and D_m^{CW}, or is in a cluster that does not contain any elements in either of the slices rotated by V_n^{CW} and D_m^{CW}. If a cubie lies in a cluster that does not contain any elements in either of the slices rotated by V_n^{CW} and D_m^{CW}, then it cannot be affected by V_n^{CW}, V_n^{CCW}, D_m^{CW}, or D_m^{CCW} moves. So applying S to it is equivalent to applying H_0^{CW}, H_0^{CCW} to it, and thus does not affect its position. If a cubie lies in the V_n^{CW} slice but is not in the cluster containing the cubie in both V_n^{CW} and D_m^{CW} slices, then applying S to it is equivalent to applying $V_n^{CCW} \circ V_n^{CW} \circ H_0^{CW} \circ V_n^{CCW} \circ V_n^{CW} \circ H_0^{CCW}$ (the identity) as it can never lie in the D_m^{CW} slice. Similarly, if a cubie lies in the D_m^{CW} slice but is not in the cluster containing the cubie in both V_n^{CW} and D_m^{CW} slices, then it can never lie in the V_n^{CW} slice, and so applying S to it is equivalent to applying $D_m^{CW} \circ D_m^{CCW} \circ H_0^{CW} \circ D_m^{CW} \circ D_m^{CCW} \circ H_0^{CCW}$, the identity.

Now consider the four cubies on the upper face in the same cluster as the cubie lying in both V_n^{CW} and D_m^{CW} slices. The cubie lying on the left side of the upper face *is* the cubie lying in both the V_n^{CW} and D_m^{CW} slices, and applying S to it results in moving it, in sequence, to the upper side of the back face (V_n^{CCW}), the left side of the upper face (V_n^{CW}), the bottom side of the left face (D_m^{CCW}), the left side of the upper face (D_m^{CW}), the upper side of the back face (V_n^{CCW}), the left side of the upper face (V_n^{CW}), and the front side of the upper face (H_0^{CCW}). So the result is moving the cubie from the left side to the front side of the upper face. The cubie lying on the front side of the upper face initially lies in neither the V_n^{CW} nor the D_m^{CW} slice. So the moves in S before H_0^{CW} do no affect it. Applying the subsequence of moves starting at H_0^{CW} move it, in sequence, to the left side of the upper face (H_0^{CW}), the upper side of the right face (D_m^{CW}), the left side of the upper face (D_m^{CCW}), and the left side of the front face (V_n^{CW}). So the result is moving the cubie from the front side of the upper face to the left side of the front face. The cubie lying on the back side of the upper face moves visits the back face (V_n^{CCW}, V_n^{CW}) and the right side of the upper face (H_0^{CW}, H_0^{CCW}), but is not affected by S, and the cubie lying on the right side of the upper face only visits the front side of the upper face (H_0^{CW}, H_0^{CCW}) and thus is not affected by S.

In summary, applying S to the cube results in changing the locations of exactly three cubies of a single cluster, those lying on the left and front sides of the upper face, and the cubie lying on the left side of the front face. All other cubies of the cube are left unchanged. The three affected cubies each move into the location of another, with the cubie on the left side of the upper face moving to the location of the cubie on the front side of the upper face, the cubie on the front side of the upper face moving to the location of the cubie on the left side of the front face, and the cubie on the left side of the front face moving to the location of the cubie on the left side of the upper face. As seen in Figure 8. the result of applying S to a cluster is to "rotate" the locations of three cubies, and in effect rotate the colors of the cubies at these three locations. Note that

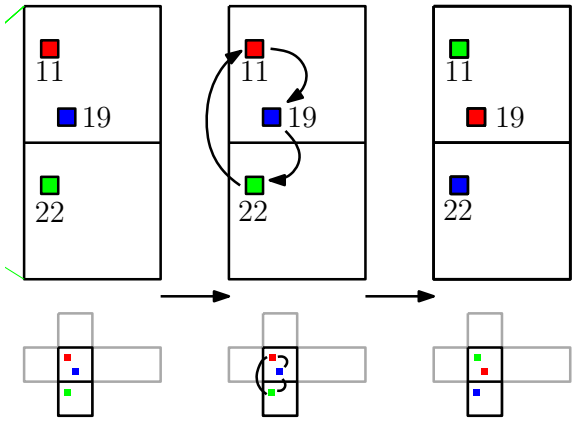


Fig. 8. The resulting movement of the three cubies of a cluster resulting from applying the move sequence $S = V_n^{CCW} \circ D_m^{CW} \circ V_n^{CW} \circ D_m^{CCW} \circ H_0^{CW} \circ D_m^{CW} \circ V_n^{CCW} \circ D_m^{CCW} \circ V_n^{CW} \circ H_0^{CCW}$.

(1 2 12)	(4 3 10)	(2 4 11)	(3 1 9)	(5 12 8)	(20 13 19)
(12 20 24)	(13 5 4)	(6 11 5)	(19 14 18)	(11 19 22)	(14 6 3)
(7 10 6)	(18 15 17)	(10 18 21)	(15 7 1)	(8 9 7)	(17 16 20)
(9 17 23)	(16 8 2)	(21 22 13)	(24 23 15)	(22 24 16)	(23 21 14)

Table 1. A set of 24 permutations that can be applied to a cluster while leaving all other clusters unchanged.

rotating three elements is equivalent to performing a pair of swaps, just as the permutation (11 19 22) = (11 19)(11 22)

The choices for which faces are front, left and upper are arbitrary, and there are 24 choices for such a set (six choices for the front face, and four choices for the upper face for each choice of front face). For a specific cluster, each choice of front, left and upper faces implies a permutation resulting from applying S. Using the ordering of the cubies in a cluster defined in Figure 6, a resulting set of 24 permutations is generated (as seen in Table 1). □

Lemma 9. *Any cluster configuration with even parity can be solved using a cluster move solution of length* $O(1)$.

Proof. By Lemma 8, there exist a set of permutations that can be applied to any single cluster while applying the identity permutation to every other cluster. It can be shown using the GAP software package [27] that this set of permutations generates A_{24}, the set of even permutations on 24 elements. Thus any even permutation can be written as a composition of these permutations and has an inverse that can also be written as the composition of these permutations. Because each cluster has finite size, the inverse composition must have $O(1)$ length. So there exists a $O(1)$-length sequence of moves that can be applied to

the cube that results in one cluster having the solved color configuration, and all other cubie clusters having unchanged color configurations. □

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

As in the $n \times n \times 1$ case, our goal here is to find a way to solve several different clusters in parallel, so that the number of moves for the solution is reduced from $O(n^2)$ to $O(n^2/\log n)$. We use the same parallelization technique as we did for the $n \times n \times 1$ Rubik's Cube, although some modification is necessary because of differences in the types of moves allowed.

Lemma 10. *Suppose we are given an $n \times n \times n$ Rubik's Cube configuration and sets $X, Y \subseteq \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$ such that $X \cap Y = \emptyset$. If all cubie clusters in $X \times Y$ have the same cluster configuration, then they can all be solved in a sequence of $O(|X|+|Y|)$ moves that only affects cubie clusters in $(X \times Y) \cup (X \times X) \cup (Y \times Y)$.*

Proof. Let d be the cluster configuration of all of the clusters in $X \times Y$. By Lemma 9, we know that there is a constant-length cluster move solution for d. Let $a_1, \ldots, a_m$, $b_1, \ldots, b_m$, and $c_1, \ldots, c_m$ be the type sequences of that cluster move solution. Let $x_1, \ldots, x_\ell$ be the elements of X, and let $y_1, \ldots, y_k$ be the elements of Y. To build a sequence of moves to solve all the clusters in $X \times Y$, we begin by defining:

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

Note that this sequence consists of $|X| + |Y| + 1$ moves. We then construct the full sequence of moves to be the following:

$$\textsc{Bulk}_1, \textsc{Bulk}_2, \ldots, \textsc{Bulk}_m.$$

Because the original sequence of moves had length $O(1)$, we know that $\ell = O(1)$, and so the total length of this sequence will be $O(|X| + |Y| + 1)$.

Consider the effect of this constructed move sequence on a cubie cluster $(x, y) \in \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\} \times \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$. First, consider the effect on $(x, y) \in X \times Y$. The subsequence of moves which affect this cubie cluster will be $F_{a_1}, RC_{b_1,x}, RC_{c_1,y}, \ldots, F_{a_\ell}, RC_{b_\ell,x}, RC_{c_\ell,y}$. This is precisely the set of moves generated by the cluster move solution for solving the cluster (x, y) so this cluster will be solved. The subsequence of moves affecting the cluster $(y, x) \in Y \times X$. will be the same as the subsequence for the cluster (x, y). By Property 2 of cluster move solutions, this cluster will not be affected by the sequence of moves.

We need not consider the effect on clusters $X \times X$ or $Y \times Y$, because our lemma places no restrictions on what happens to those clusters. So all of the remaining clusters we must consider have at most one coordinate in $X \cup Y$. Suppose we have some $x \in X$ and some $z \notin X \cup Y$. Then the sequence of moves affecting the clusters (x, z) and (z, x) will be $F_{a_1}, RC_{b_1,x}, F_{a_2}, RC_{b_2,x} \ldots, F_{a_\ell}, RC_{b_\ell,x}$. By Property 3 of cluster move solutions, this sequence of moves does not affect

any clusters, and so (x, z) and (z, x) will both remain unaffected. Similarly, suppose we have some $y \in Y$ and some $z \notin X \cup Y$. Then the sequence of moves affecting the clusters (y, z) and (z, y) is $F_{a_1}, RC_{c_1,y}, F_{a_2}, RC_{c_2,y} \ldots, F_{a_\ell}, RC_{c_\ell,y}$. According to Property 3, this move sequence does not affect the configuration of clusters (y, z) or (z, y). Finally, consider the effect on some cluster $(w, z) \in \overline{X \cup Y} \times \overline{X \cup Y}$. Then the sequence of moves affecting (w, z) is $F_{a_1}, F_{a_2}, \ldots, F_{a_\ell}$. Once again, by Property 3 of cluster move solutions, this move sequence will not affect the configuration of cluster (w, z). □

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, we show the following lemma.

Lemma 11. *Suppose we are given an $n \times n \times n$ Rubik's Cube configuration, a cluster configuration c, and sets $X, Y \subseteq \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$, such that $\ell = |X|$ and $X \cap Y = \emptyset$. Then there exists a sequence of moves of length $O(\ell 2^\ell + |Y|)$ such that:*

- *All cubie clusters $(x, y) \in X \times Y$ in configuration c will be solved.*
- *All cubie clusters $(x, y) \in (X \times X) \cup (Y \times Y)$ may or may not be affected.*
- *All other cubie clusters will not be affected.*

Proof. For each row $y \in Y$, let $S_y = \{x \in X \mid \text{cubie cluster } (x, y) \text{ is in configuration } c\}$. For each set $S \subseteq X$, let $Y_S = \{y \in Y \mid S_y = S\}$. Because $S \subseteq X$, there are at most 2^ℓ different values for S. For each S, we will use the results of Lemma 10 to construct a sequence of moves to solve each cubie cluster $(x, y) \in S \times Y_S$. This move sequence will have length $O(|S| + |Y_S|) = O(\ell + |Y_S|)]$. When we sum up this cost over all sets $S \subseteq X$, we get the following number of moves:

$$O\left(\ell \cdot 2^\ell + \sum_S |Y_S|\right) = O\left(\ell \cdot 2^\ell + |Y|\right).$$

□

Just as we did for the $n \times n \times 1$ Rubik's Cube, we avoid exponential blow-up by dividing the set of columns X into smaller groups, solving each such group individually. More formally:

Lemma 12. *Suppose we are given an $n \times n \times n$ Rubik's Cube configuration, a cluster configuration c, and sets $X, Y \subseteq \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$, such that $X \cap Y = \emptyset$. Then there exists a sequence of moves of length $O(|X| \cdot |Y| / \log |Y|)$ such that:*

- *All cubie clusters $(x, y) \in X \times Y$ in configuration c will be solved.*
- *All cubie clusters $(x, y) \in (X \times X) \cup (Y \times Y)$ may or may not be affected.*
- *All other cubie clusters will not be affected.*

Proof. Let $\ell = \frac{1}{2} \log_2 |Y|$, so that $2^\ell = \sqrt{|Y|}$. Let $k = \lceil |X|/\ell \rceil$. Partition the set X into a series of sets $X_1, \ldots, X_k$ each of which has size $\leq \ell$. For each X_i, we

solve the cubie clusters in $X_i \times Y$ using the sequence of moves that is guaranteed to exist by Lemma 11. The number of moves required to solve a single X_i is

$$O\left(\ell 2^{\ell} + |Y|\right) = O\left(\left(\frac{1}{2}\log_2 |Y|\right)\sqrt{|Y|} + |Y|\right) = O(|Y|).$$

Therefore, if we wish to perform this for k sets, the total number of moves becomes

$$O\left(k \cdot |Y|\right) = O\left(\frac{|X|}{\frac{1}{2}\log_2 |Y|} \cdot |Y|\right) = O\left(\frac{|X| \cdot |Y|}{\log |Y|}\right).$$

□

To finish constructing the move sequence for the entire Rubik's Cube, we need to 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 $(X \times X) \cup (Y \times Y)$.

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.*

Proof. In order to solve the Rubik's Cube, we must first fix the parity. Using the techniques of Lemma 7, we can perform this step in $O(n)$ moves. Then we solve each edge cluster individually. Each edge cluster requires $O(1)$ moves to solve, and there are $O(n)$ edge clusters, so this preliminary step takes time $O(n)$.

Once the edges have been solved, we want to solve the non-edge clusters. Let $k = \sqrt{n/2}$. Partition $\{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$ into a series of sets $G_1, \ldots, G_k$ each of which has size $\leq k$. For each pair i, j such that $i \neq j$ and each cluster configuration c, we use the sequence of moves guaranteed to exist by Lemma 12 to solve all $(x, y) \in G_i \times G_j$ with the configuration c. This ensures that all cubie clusters $(x, y) \in G_i \times G_j$ will be solved. For each i, we must also solve all cubie clusters $(x, y) \in G_i \times G_i$. There are $k^2(k-1)/2$ such cubie clusters, so we can afford to solve each such cubie cluster individually.

What is the total number of moves required? For a single pair $i \neq j$ and a single configuration c, the number of moves required will be $O(k^2/\log k) = O(n/\log n)$. There are a constant number of possible configurations, so solving a single pair i, j for all configurations will also require $O(n/\log n)$ moves. There are $k^2 - k$ such pairs, for a total of $O(n^2/\log n)$. If we then add in the extra $O(k^3)$ from the diagonals, then the total number of moves will be $O(n^2/\log n + n^{3/2}) = O(n^2/\log n)$. □

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

We derive a matching lower bound using a technique identical to the one used for the $n \times n \times 1$ lower bound:

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.*

Proof. Lemma 9 shows that for every possible configuration of a cubie cluster, there exists a sequence of moves to solve the cubie cluster while leaving the rest of the cubies in the same location. Hence, the inverse of such a sequence will transform a solved cubie cluster to an arbitrary configuration without affecting any other cubies. Not counting the edges, there are $(\lfloor n/2 \rfloor - 1)^2$ cubie clusters, each of which can be independently placed into one of $(24!)/(4!)^6$ different color configurations. This means that there are at least

$$\left(\frac{24!}{(4!)^6}\right)^{(\lfloor n/2 \rfloor - 1)^2}$$

reachable configurations.

There are $6n$ possible moves, so the total number of states reachable using at most k moves is at most

$$\frac{(6n)^{k+1} - 1}{6n - 1} \leq (6n)^{k+1}.$$

Therefore, if k is the number of moves necessary to reach all states, it must have the property that

$$\begin{aligned}
\left(\frac{24!}{(4!)^6}\right)^{(\lfloor n/2 \rfloor - 1)^2} &\leq (6n)^{k+1}, \\
(\lfloor n/2 \rfloor - 1)^2 \cdot \log\left(\frac{24!}{(4!)^6}\right) &\leq \log\left((6n)^{k+1}\right) = (k+1)\log(6n), \\
\frac{(\lfloor n/2 \rfloor - 1)^2 \log\left(\frac{24!}{(4!)^6}\right)}{\log(6n)} - 1 &\leq k.
\end{aligned}$$

Hence, there must exist some configurations which are $\Omega(n^2/\log n)$ moves away from 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 problem which generalizes 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

cubie 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.*

For our hardness reduction, we develop two main gadgets. The first gadget forces an ordering on the second occurrences of row moves, and is used in the construction of the second gadget. The second gadget forces a betweenness constraint on the ordering of the first occurrences of row moves.

Lemma 13. *Given two sets of columns $X_1, X_2 \subseteq \{0, 1, \ldots, \lfloor n/2 \rfloor - 1\}$, there is a gadget using three extra rows and two extra columns ensuring that, for all $x_1 \in X_1$ and for all $x_2 \in X_2$, $I_2(x_1) < I_2(x_2)$. As a side effect, this gadget also forces*

$$\max_{x_1 \in X_1} I_1(x_1) < \min_{x_1 \in X_1} I_2(x_1) \qquad \text{and} \qquad \max_{x_2 \in X_2} I_1(x_2) < \min_{x_2 \in X_1} I_2(x_2).$$

Proof. Let $\tilde{y}_1, \tilde{y}_2, \tilde{y}_3 \leq \lfloor n/2 \rfloor$ be three rows not used elsewhere in the construction. Let $\tilde{x}_1, \tilde{x}_2 \leq \lfloor n/2 \rfloor$ be two columns not used elsewhere in the construction. Make cubie clusters $(\tilde{x}_1, \tilde{y}_2)$ and $(\tilde{x}_2, \tilde{y}_3)$ have the configuration depicted in Fig. 1(b); make cubie clusters $(\tilde{x}_1, \tilde{y}_1)$ and $(\tilde{x}_2, \tilde{y}_2)$ have the configuration

depicted in Fig. 1(d); and make cubie cluster $(\tilde{x}_2, \tilde{y}_1)$ remain in the solved configuration. Mark all of these cubie clusters as important.

These cluster configurations enforce the following constraints:

$$I_1(\tilde{x}_1) < I_1(\tilde{y}_2) < I_2(\tilde{x}_1) < I_2(\tilde{y}_2), \qquad I_1(\tilde{x}_2) < I_1(\tilde{y}_3) < I_2(\tilde{x}_2) < I_2(\tilde{y}_3),$$
$$I_1(\tilde{y}_1) < I_1(\tilde{x}_1) < I_2(\tilde{y}_1) < I_2(\tilde{x}_1), \qquad I_1(\tilde{y}_2) < I_1(\tilde{x}_2) < I_2(\tilde{y}_2) < I_2(\tilde{x}_2).$$

We can use these inequalities to construct the following chains:

$$I_1(\tilde{y}_1) < I_1(\tilde{x}_1) < I_1(\tilde{y}_2) < I_1(\tilde{x}_2) \quad \text{and} \quad I_2(\tilde{y}_1) < I_2(\tilde{x}_1) < I_2(\tilde{y}_2) < I_2(\tilde{x}_2).$$

Because $(\tilde{x}_2, \tilde{y}_1)$ must remain solved, and because of the above constraints, there is only one possible ordering for the pairs of moves involving $\tilde{y}_1$ and $\tilde{x}_2$: $I_1(\tilde{y}_1) < I_2(\tilde{y}_1) < I_1(\tilde{x}_2) < I_2(\tilde{x}_2)$. If we combine this with the constraint $I_1(\tilde{x}_2) < I_1(\tilde{y}_3) < I_2(\tilde{x}_2) < I_2(\tilde{y}_3)$, we know that $I_1(\tilde{y}_1) < I_2(\tilde{y}_1) < I_1(\tilde{y}_3) < I_2(\tilde{y}_3)$.

Now, for each $x_1 \in X_1$ and $x_2 \in X_2$, make cubie clusters $(x_1, \tilde{y}_1)$ and $(x_2, \tilde{y}_3)$ have the configuration depicted in Fig. 1(b), and mark them important. No other cubie clusters involving $\tilde{y}_1, \tilde{y}_2, \tilde{y}_3$ or $\tilde{x}_1, \tilde{x}_2$ should be marked important. This constraint ensures that for all $x_1 \in X_1$, $I_2(x_1)$ must lie between $I_1(\tilde{y}_1)$ and $I_2(\tilde{y}_1)$. In addition, for all choices of $x_2 \in X_2$, $I_2(x_2)$ must lie between $I_1(\tilde{y}_3)$ and $I_2(\tilde{y}_3)$. Therefore, $I_2(x_1) < I_2(x_2)$.

As a side effect, these constraints ensure that for all $x_1 \in X_1$, $I_1(x_1)$ must lie before $I_1(\tilde{y}_1)$, while $I_2(x_1)$ lies after $I_1(\tilde{y}_1)$. Therefore,

$$\max_{x_1 \in X_1} I_1(x_1) < \min_{x_1 \in X_1} I_2(x_1).$$

In addition, the constraints ensure that for all choices of $x_2 \in X_2$, $I_1(x_2)$ must lie before $I_1(\tilde{y}_3)$, while $I_2(x_2)$ lies after $I_1(\tilde{y}_3)$. This ensures that

$$\max_{x_2 \in X_2} I_1(x_2) < \min_{x_2 \in X_1} I_2(x_2).$$

We have shown that these gadgets can enforce a constraint. We must also show that these gadgets do not enforce any constraints other than the ones expressed in the lemma. In other words, given any solution which satisfies the requirements given in the lemma, we must be able to insert the moves for our new rows and columns in such a way that all important clusters will be solved. In order to make sure that clusters $(\tilde{x}_1, \tilde{y}_2)$, $(\tilde{x}_2, \tilde{y}_3)$, $(\tilde{x}_1, \tilde{y}_1)$, $(\tilde{x}_2, \tilde{y}_2)$, and $(\tilde{x}_2, \tilde{y}_1)$ are all solved, it is sufficient to ensure that the moves $\tilde{x}_1, \tilde{x}_2, \tilde{y}_1, \tilde{y}_2, \tilde{y}_3$ occur in the following order:

$$I_1(\tilde{y}_1), I_1(\tilde{x}_1), I_2(\tilde{y}_1), I_1(\tilde{y}_2), I_2(\tilde{x}_1), I_1(\tilde{x}_2), I_2(\tilde{y}_2), I_1(\tilde{y}_3), I_2(\tilde{x}_2), I_2(\tilde{y}_3).$$

So if we can find the correct way to interleave this sequence with the existing move sequence, we will have a sequence that solves all clusters.

First, we observe that the only important clusters in row $\tilde{y}_2$ and in columns $\tilde{x}_1$ and $\tilde{x}_2$ are the ones which the above sequence will solve. So we need only determine how to correctly interleave the moves for rows $\tilde{y}_1$ and $\tilde{y}_3$ with the

existing move sequence. We know from the statement of the lemma that the existing move sequence satisfies the following constraints:

$$\max_{x_1 \in X_1} I_1(x_1) < \min_{x_1 \in X_1} I_2(x_1) \leq \max_{x_1 \in X_1} I_2(x_1) < \min_{x_2 \in X_2} I_2(x_2).$$

So to ensure that each cluster $(x_1, \tilde{y}_1)$ is solved, we insert the two copies of the move $\tilde{y}_1$ to satisfy:

$$\max_{x_1 \in X_1} I_1(x_1) < I_1(\tilde{y}_1) < \min_{x_1 \in X_1} I_2(x_1) \leq \max_{x_1 \in X_1} I_2(x_1) < I_2(\tilde{y}_1) < \min_{x_2 \in X_2} I_2(x_2).$$

Similarly, we know from the statement of the lemma that the existing move sequence satisfies these constraints:

$$\max_{x_2 \in X_2} I_1(x_2) < \min_{x_2 \in X_2} I_2(x_2) \leq \max_{x_2 \in X_2} I_2(x_2).$$

So we insert the two copies of the move $\tilde{y}_3$ as follows, to ensure that each cluster $(x_2, \tilde{y}_3)$ is solved:

$$\max_{x_2 \in X_2} I_1(x_2) < I_1(\tilde{y}_3) < \min_{x_2 \in X_2} I_2(x_2) \leq \max_{x_2 \in X_2} I_2(x_2) < I_2(\tilde{y}_3).$$

To ensure that $I_1(\tilde{y}_1) < I_2(\tilde{y}_1) < I_1(\tilde{y}_3) < I_2(\tilde{y}_3)$, we note that the above two constraints do not actually determine the ordering of $I_2(\tilde{y}_1)$ and $I_1(\tilde{y}_3)$. So we can pick an ordering where $I_2(\tilde{y}_1) < I_1(\tilde{y}_3)$, which will ensure that all clusters are solved. □

Lemma 14. *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 $I_1(x_3)$. As a side effect, this gadget also forces $I_2(x_2) < I_2(x_1)$, $I_2(x_2) < I_2(x_3)$, and*

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

Proof. Use a copy of the gadget from Lemma 13 to force $I_2(x_2) < I_2(x_1)$ and $I_2(x_2) < I_2(x_3)$. Let y_1, y_2, y_3 be three rows not used elsewhere in the construction. Make each cubie cluster in the set

$$\{(x_1, y_2), (x_1, y_3), (x_2, y_1), (x_2, y_3), (x_3, y_1), (x_3, y_2)\}$$

have the configuration depicted in Fig. 1(d). Make cubie clusters (x_1, y_1), (x_2, y_2), and (x_3, y_3) remain solved. Mark all of these cubie clusters important. Now we consider what form an ideal solution could take, given these restrictions.

First, consider the case where $I_1(x_2) < I_1(x_1)$ and $I_1(x_2) < I_1(x_3)$. Because of the configurations of cubie clusters (x_2, y_1) and (x_2, y_3), we know that

$$I_1(y_1) < I_1(x_2) < I_1(x_1), \qquad I_1(y_3) < I_1(x_2) < I_1(x_3),$$
$$I_2(y_1) < I_2(x_2) < I_2(x_1), \qquad I_2(y_3) < I_2(x_2) < I_2(x_3).$$

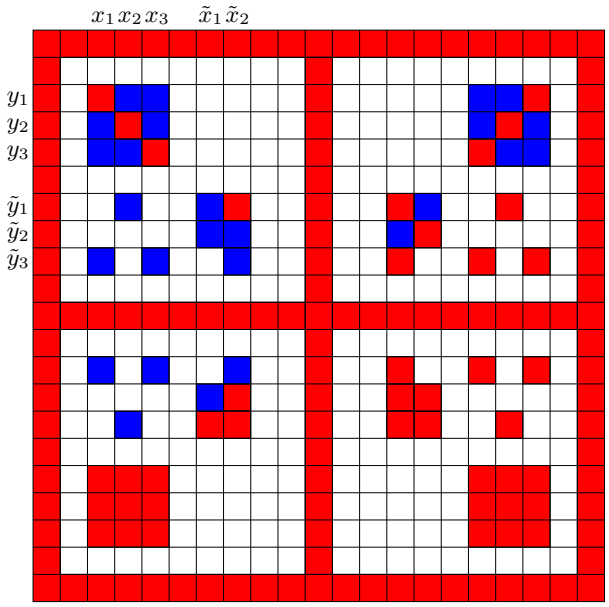


Fig. 9. A sample of the betweenness gadget from Lemma 14. Important cubies are red (solved) and blue (unsolved). Unimportant cubies are white. Any ideal solution must either have $I_1(x_1) < I_1(x_2) < I_1(x_3)$ or $I_1(x_3) < I_1(x_2) < I_1(x_1)$.

Given these constraints, and the requirement that the cubie cluster (x_1, y_1) remain solved, it must be that $I_2(y_1) < I_1(x_1)$. Similarly, because the cubie cluster (x_3, y_3) must remain solved, it must be that $I_2(y_3) < I_1(x_3)$. In order to solve the cubie clusters (x_1, y_3) and (x_3, y_1), it must be that $I_1(x_1) < I_2(y_3)$ and $I_1(x_3) < I_2(y_1)$. Therefore

$$I_1(x_1) < I_2(y_3) < I_1(x_3) < I_2(y_1) < I_1(x_1),$$

a contradiction. Hence this case cannot happen.

Now consider the case where $I_1(x_1) < I_1(x_2)$ and $I_1(x_3) < I_1(x_2)$. Then we know that the following inequalities hold:

$$I_1(x_1) < I_1(x_2) < I_2(x_2) < I_2(x_1) \quad \text{and} \quad I_1(x_3) < I_1(x_2) < I_2(x_2) < I_2(x_3).$$

Because of the configurations of cubie clusters (x_2, y_1) and (x_2, y_3), this sandwiching implies that

$$\begin{aligned} &I_1(x_1) < I_2(y_1) < I_2(x_1), && I_1(x_1) < I_2(y_3) < I_2(x_1), \\ &I_1(x_3) < I_2(y_1) < I_2(x_3), && I_1(x_3) < I_2(y_3) < I_2(x_3). \end{aligned}$$

To ensure that cubie cluster (x_1, y_1) still remains solved, we need $I_1(x_1) < I_1(y_1)$. Given the configuration of cubie cluster (x_1, y_3), we need $I_1(y_3) < I_1(x_1)$. To

ensure that cubie cluster (x_3, y_3) still remains solved, we need $I_1(x_3) < I_1(y_3)$. Given the configuration of cubie cluster (x_3, y_1), we need $I_1(y_1) < I_1(x_3)$. Thus

$$I_1(y_3) < I_1(x_1) < I_1(y_1) < I_1(x_3) < I_1(y_3),$$

a contradiction. Hence this case is also impossible.

Because neither of the two cases are possible, $I_1(x_2)$ must lie between $I_1(x_1)$ and $I_1(x_3)$, which is precisely what we wanted this gadget to enforce. We must also show that this gadget does not enforce any constraints other than the ones expressed in the lemma. Given any solution which satisfies the requirements given in the lemma, we must be able to insert the moves for the new rows and columns in such a way that all important clusters will be solved. The extra constraints on the existing move sequence ensure the following:

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

So we know that we can place the moves for the extra rows and columns used by the gadget from Lemma 13. We need only determine where to insert the moves for the three extra rows added by this gadget.

The constraints given in the statement of the lemma allow for four different possible orderings of all of the x_1, x_2, x_3 moves. We consider each case separately.

1. $I_1(x_1) < I_1(x_2) < I_1(x_3) < I_2(x_2) < I_2(x_1) < I_2(x_3)$. Then we insert the moves y_1, y_2, y_3 so that the following is a subsequence of the move sequence:

$$y_2, y_3, x_1, y_1, x_2, y_3, x_3, y_1, x_2, y_2, x_1, x_3.$$

2. $I_1(x_1) < I_1(x_2) < I_1(x_3) < I_2(x_2) < I_2(x_3) < I_2(x_1)$. Then we insert the moves y_1, y_2, y_3 so that the following is a subsequence of the move sequence:

$$y_2, y_3, x_1, y_1, x_2, y_3, x_3, y_1, x_2, y_2, x_3, x_1$$

3. $I_1(x_3) < I_1(x_2) < I_1(x_1) < I_2(x_2) < I_2(x_1) < I_2(x_3)$. Then we insert the moves y_1, y_2, y_3 so that the following is a subsequence of the move sequence:

$$y_1, y_2, x_3, y_3, x_2, y_1, x_1, y_3, x_2, y_2, x_1, x_3$$

4. $I_1(x_3) < I_1(x_2) < I_1(x_1) < I_2(x_2) < I_2(x_3) < I_2(x_1)$. Then we insert the moves y_1, y_2, y_3 so that the following is a subsequence of the move sequence:

$$y_1, y_2, x_3, y_3, x_2, y_1, x_1, y_3, x_2, y_2, x_3, x_1$$

□

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 14 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, there exists 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.*

Proof. We start with a single column $r_{center} \leq \lfloor n/2 \rfloor$. For each variable x_i in ϕ, we construct two columns $s_{x_i}, s_{\overline{x_i}} \leq \lfloor n/2 \rfloor$. We then add a copy of the gadget from Lemma 14 to ensure that $I_1(r_{center})$ lies between $I_1(s_{x_i})$ and $I_1(s_{\overline{x_i}})$, for each value of i. For each clause $c_j = y_1 \vee y_2 \vee y_3$, we add a new column t_j. Then we add a copy of the gadget from Lemma 14 to ensure that $I_1(t_j)$ lies between $I_1(s_{y_1})$ and $I_1(s_{y_2})$. Then we add one more copy of the gadget from Lemma 14 to ensure that $I_1(r_{center})$ lies between $I_1(t_j)$ and $I_1(s_{y_3})$.

Note that the additional constraints forced by the gadget from Lemma 14 mean that all of the following inequalities must hold:

$$\begin{aligned} I_2(r_{center}) &< I_2(s_{x_i}), & I_2(r_{center}) &< I_2(s_{\overline{x_i}}), \\ I_2(t_j) &< I_2(s_{y_1}), & I_2(t_j) &< I_2(s_{y_2}), \\ I_2(r_{center}) &< I_2(t_j), & I_2(r_{center}) &< I_2(s_{y_3}). \end{aligned}$$

We can satisfy all of these constraints by first performing the second move of r_{center}, then performing the second moves of all of the t columns, and finally by performing the second moves of all of the s columns. The other constraint imposed by the gadget from Lemma 14 can be satisfied by dividing up the moves into two sequential stages such that all of the variable, clause, and center moves are performed exactly once per stage. Hence, the additional constraints enforced by our gadget do not affect our ability to construct an ideal solution, as long as ϕ is a member of $\neq$-SAT.

To see why this reduction works, suppose that ϕ is a member of $\neq$-SAT. We must convert an assignment to the variables of ϕ to an ideal solution to the subset of the Rubik's Cube constructed above. As noted in the previous paragraph, we can choose an ordering of all of the second moves that satisfies the gadgets we have constructed. To arrange the first moves of all of the columns, we pick an ordering of the columns s corresponding to literals so that $I_1(s_y) < I_1(r_{center})$ for all true literals y and $I_1(r_{center}) < I_1(s_z)$ for all false literals z. The ordering of the literals themselves does not matter, and can be picked arbitrarily. This arrangement ensures that for each x_i, we have either $I_1(s_{x_i}) < I_1(r_{center}) < I_1(s_{\overline{x_i}})$, or $I_1(s_{\overline{x_i}}) < I_1(r_{center}) < I_1(s_{x_i})$; either way, there will be a way to correctly arrange the extra columns and rows used by the gadget from Lemma 14.

We then must pick times for the first move for each column t_j. Let $c_j = y_1 \vee y_2 \vee y_3$ be the clause corresponding to the column we are considering.

1. Consider the case where y_1 and y_2 are both true. Then we pick an arbitrary location for t_j between s_{y_1} and s_{y_2}, so that there is a way to correctly arrange the extra columns and rows used by the copy of the gadget that ensures that $I_1(t_j)$ lies between $I_1(s_{y_1})$ and $I_1(s_{y_2})$. This means that $I_1(t_j) < I_1(r_{center})$. Because y_1 and y_2 are both true, then y_3 must be false, and so $I_1(r_{center}) < I_1(y_3)$. Hence, there will be a way to correctly arrange the extra columns and rows used by the copy of the gadget from Lemma 14.
2. Now consider the case where y_1 and y_2 are both false. Then we pick an arbitrary location for t_j between s_{y_1} and s_{y_2}. Just as before, this satisfies the gadget from Lemma 14 which forces $I_1(t_j)$ to lie between $I_1(s_{y_1})$ and $I_1(s_{y_2})$. This also means that $I_1(r_{center}) < I_1(t_j)$. Because both y_1 and y_2 are false, y_3 must be true, and therefore $I_1(y_3) < I_1(r_{center}) < I_1(t_j)$. So once again, the gadget is satisfied.
3. Now consider the case where y_1 is true and y_2 is false, or vice versa. Then y_3 is either true or false. If y_3 is true, then $I_1(y_3) < I_1(r_{center})$, and so we pick the location for $I_1(t_j)$ to be just slightly larger than $I_1(r_{center})$, so that $I_1(y_3) < I_1(r_{center}) < I_1(t_j)$ and $I_1(t_j)$ lies between $I_1(s_{y_1})$ and $I_1(s_{y_2})$. Similarly, if y_3 is false, then $I_1(r_{center}) < I_1(y_3)$, and so we pick the location for $I_1(t_j)$ to be just slightly smaller than $I_1(r_{center})$, so that $I_1(t_j) < I_1(r_{center}) < I_1(y_3)$ and $I_1(t_j)$ lies between $I_1(s_{y_1})$ and $I_1(s_{y_2})$. In either case, the location for the first move t_j will satisfy both of the gadgets created using Lemma 14.

So if we have a solution to ϕ, then we also have an ideal solution.

Now we wish to prove the converse. Suppose that we have an ideal solution. Then we construct a solution to ϕ by setting x_i to be true if and only if $I_1(s_{x_i}) < I_1(r_{center})$. Because of the gadgets that we constructed, we know that all true literals y have the property that $I_1(s_y) < I_1(r_{center})$, while all false literals z have the property that $I_1(r_{center}) < I_1(s_z)$. To see why this works, consider a clause $c_j = y_1 \vee y_2 \vee y_3$. Assume, for the sake of contradiction, that all three literals in the clause are true. Then $I_1(s_{y_1}), I_1(s_{y_2}), I_1(s_{y_3}) < I_1(r_{center})$. Because our betweenness gadget is working correctly, we know that $I_1(t_j) < I_1(r_{center})$, as well. This means that $I_1(r_{center})$ does not lie between $I_1(t_j)$ and $I_1(s_{y_3})$, which means that our solution could not have been ideal. So our assumption must be wrong, and not all of the literals in the clause are true. A similar argument shows that not all of the literals in the clause are false. This means that the clause has at least one true literal and at least one false literal, and so ϕ is a member of $\neq$-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 will 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 $z = (n-1)-i$. If n is odd, then cubie cluster $(n-1)/2$ will consist of a single slice. This definition means that any short move affects the position and orientation of cubies in exactly one cubie cluster.

Lemma 15. *Given any $c_1 \times c_2 \times n$ Rubik's Cube configuration, and any cubie cluster t, the number of short moves affecting that cubie cluster in the optimal solution is at most $2^{2c_1c_2+8(c_1+c_2)} - 1$.*

Proof. Consider the subsequence of moves in the optimal solution which affect t. This should contain all of the long moves, and only those short moves which rotate one of the two slices in t. For notation purposes, we merge all consecutive long moves together into compound long moves $L_0, \ldots, L_k$, so that the sequence of moves is $L_0 \circ s_1 \circ L_1 \circ s_2 \circ \ldots \circ L_{k-1} \circ s_k \circ L_k$. For convenience, we define s_0 to be the identity function, so that we can write the sequence of moves as $s_0 \circ L_0 \circ s_1 \circ L_1 \circ s_2 \circ \ldots \circ s_k \circ L_k$.

We define the following to be the results of performing certain moves:

$$\begin{aligned}\textsc{Seq}(i,j) &= s_i \circ L_i \circ s_{i+1} \circ L_{i+1} \circ \ldots \circ s_{j-1} \circ L_{j-1} \circ s_j \circ L_j,\\ \textsc{LSeq}(i,j) &= L_i \circ L_{i+1} \circ \ldots \circ L_{j-1} \circ L_j.\end{aligned}$$

Assume for the sake of contradiction that there exist $i < j$ such that $\textsc{Seq}(0,i) \circ \textsc{LSeq}(i+1,k) = \textsc{Seq}(0,j) \circ \textsc{LSeq}(j+1,k)$. Composing by the inverse of $\textsc{LSeq}(j+1,k)$, we obtain $\textsc{Seq}(0,i) \circ \textsc{LSeq}(i+1,j) = \textsc{Seq}(0,j)$. Therefore

$$\begin{aligned}\textsc{Seq}(0,i) \circ \textsc{LSeq}(i+1,j) \circ \textsc{Seq}(j+1,k) &= \textsc{Seq}(0,j) \circ \textsc{Seq}(j+1,k)\\ &= \textsc{Seq}(0,k).\end{aligned}$$

Hence we can omit the moves $s_{i+1}, \ldots, s_j$ while still ending up with the correct configuration for t. This means that there exists a sequence of moves, shorter than the original, which brings the Rubik's Cube into the same configuration. But the original sequence was optimal. Therefore, our assumption must be wrong, and $\forall i < j$, $\textsc{Seq}(0,i) \circ \textsc{LSeq}(i+1,k) \neq \textsc{Seq}(0,j) \circ \textsc{LSeq}(j+1,k)$.

In both $\textsc{Seq}(0,i) \circ \textsc{LSeq}(i+1,k)$ and $\textsc{Seq}(0,j) \circ \textsc{LSeq}(j+1,k)$, the set of long moves is the same, so the configuration of all cubie clusters other than t must also be the same. Therefore, the results of those moves must differ in the configuration of the cubie cluster t. The cubie cluster with the greatest number of configurations is the cubie cluster which contains the ends of the Rubik's Cube (i.e., $z = 0$ and $z = n-1$), because of the additional information given by the exposed sides of the cube. For that cubie cluster, the total number of different configurations is $\leq 2^{2c_1c_2} \cdot 4^{4(c_1+c_2)} = 2^{2c_1c_2+8(c_1+c_2)}$. Each short move affecting t in the optimal solution must lead to a new configuration of t, and so the number of short moves must be $\leq 2^{2c_1c_2+8(c_1+c_2)} - 1$. □

Lemma 16. *There exists a sequence of long moves $\ell_1 \circ \ell_2 \circ \ldots \circ \ell_m$, where $m \leq (c_1c_2)! \cdot 2^{1+c_1c_2}$, such that:*

1. $\ell_1 \circ \ell_2 \circ \ldots \circ \ell_m$ *is the identity function; and*
2. *for every long sequence* L*, there exists some* i *such that* $\ell_i \circ \ell_{i+1} \circ \ldots \circ \ell_m = L$.

Proof. Every sequence of long moves causes a rearrangement of the cubies in the Rubik's Cube. However, there are no long moves which break up the boxes of cubies of size $1 \times 1 \times n$ — each box can be moved or rotated, but the relative positions of cubies within the box are always the same. There are $c_1 c_2$ such boxes, and each box can be oriented in two ways. This means that there can be no more than $(c_1 c_2)! \cdot 2^{c_1 c_2}$ reachable long configurations. We treat these long configurations as a graph, with edges between configurations that are reachable using a single long move. If we take a spanning tree of the graph and duplicate all edges of the tree, then we can find an Eulerian cycle which visits all of the nodes in the graph. If we start at the identity configuration and move along the cycle, then we have a path of $\leq (c_1 c_2)! \cdot 2^{1+c_1 c_2}$ long moves satisfying both of the above properties. □

Lemma 17. *Given any* $c_1 \times c_2 \times n$ *Rubik's Cube configuration, the number of long moves in the optimal solution is at most* $(c_1 c_2)! \cdot 2^{1+3c_1 c_2+8(c_1+c_2)}$.

Proof. Assume, for the sake of contradiction, that the number of long moves in the optimal solution is greater than $(c_1 c_2)! \cdot 2^{1+3c_1 c_2+8(c_1+c_2)}$. Then we can construct another solution with the same number of short moves as the optimal, and fewer long moves. Let $\ell_1 \circ \ell_2 \circ \ldots \circ \ell_m$ be the sequence of long moves satisfying Lemma 16. Let L be the sequence of long moves in the optimal solution, and let i be the index such that $\ell_i \circ \ell_{i+1} \circ \ldots \circ \ell_m = L$. We choose the sequence of long moves in our constructed solution to be

$$(\ell_i \circ \ell_{i+1} \circ \ldots \circ \ell_m) \circ \underbrace{(\ell_1 \circ \ell_2 \circ \ldots \circ \ell_m) \circ \ldots \circ (\ell_1 \circ \ell_2 \circ \ldots \circ \ell_m)}_{2^{2c_1 c_2+8(c_1+c_2)}-1 \text{ times}}.$$

This long move sequence has the same result as L. We must also specify how to interleave the short moves with the long move sequence. For a fixed long move sequence, the arrangement of short moves for one cubie cluster does not affect any other cubie cluster. Consequently, if we can correctly interleave the short moves for one cubie cluster, we can correctly interleave the short moves for all cubie clusters.

Pick an arbitrary cubie cluster. Consider the subsequence of moves in the optimal solution which affect said cubie cluster. For notation purposes, we merge all consecutive long moves together into compound long moves $L_0, \ldots, L_k$, so that the sequence of moves is $L_0 \circ s_1 \circ L_1 \circ s_2 \circ \ldots \circ L_{k-1} \circ s_k \circ L_k$. We place the short moves into the above sequence of long moves starting with s_k. Let a be the index such that $\ell_a \circ \ldots \circ \ell_m = L_k$. We insert s_k into the kth repetition of $(\ell_1 \circ \ldots \circ \ell_m)$ between ℓ_{a-1} and ℓ_a. This ensures that the sequence of long moves occurring after k will be equivalent to L_k.

In general, say that we have placed $s_{i+1}, \ldots, s_k$ in the $(i+1)$st through kth repetitions of $(\ell_1 \circ \ldots \circ \ell_m)$. Say we want to place s_i in the ith repetition. Let b

be the index such that s_{i+1} was placed between ℓ_{b-1} and ℓ_b. Let a be the index such that:

$$\ell_a \circ \ldots \circ \ell_m = L_i \circ (\ell_1 \circ \ldots \circ \ell_{b-1})^{-1} = L_i \circ (\ell_{b-1}^{-1} \circ \ell_{b-2}^{-1} \circ \ldots \circ \ell_2^{-1} \circ \ell_1^{-1})$$

This ensures that the sequence of long moves between s_i and s_{i+1} will be:

$$\ell_a \circ \ldots \circ \ell_m \circ \ell_1 \circ \ldots \circ \ell_{b-1} = L_i \circ (\ell_1 \circ \ldots \circ \ell_{b-1})^{-1} \circ (\ell_1 \circ \ldots \circ \ell_{b-1}) = L_i$$

Hence, if the number of long moves in the optimal solution is greater than $(c_1c_2)! \cdot 2^{1+3c_1c_2+8(c_1+c_2)}$, we can create a solution with the same number of short moves and fewer long moves. This means that we have a contradiction. □

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.*

Proof. By Lemma 17, we know that the total number of long moves in the optimal solution is at most $(c_1c_2)! \cdot 2^{1+3c_1c_2+8(c_1+c_2)}$, which is constant. We also know that there are a total of $c_1 + c_2$ possible long moves. Hence, the total number of possible sequences of long moves is constant, so we can enumerate all of these in time $O(1)$.

For each of the sequences of long moves, we want to find the optimal solution using that sequence of long moves. Because the long moves are fixed, we can calculate the short moves for each cubie cluster independently. To calculate the short moves for some fixed cubie cluster, we note that between two sequential long moves, there are at most four different ways to rotate each of the two slices in the cubie cluster, for a total of at most sixteen possible combinations of short moves.

For a given cubie cluster, we have to consider $\leq 16^{1+(c_1c_2)!\cdot 2^{1+3c_1c_2+8(c_1+c_2)}}$ possible combinations. This is constant, so we can try all possibilities to see if they solve the cubie cluster. We can pick the shortest of those. If we perform this operation for all cubie clusters, we will have an optimal solution for this particular sequence of long moves. If we calculate this for all sequences of long moves, then we can pick the overall optimal solution by taking the sequence of minimum length. □

7 Conclusion and Open Problems

In this paper, we presented several new results. First, we introduced a technique for parallelizing the solution to two types of generalized Rubik's Cubes. As a result, we showed that the diameter of the configuration space for these two types of Rubik's Cubes is $\Theta(n^2/\log n)$. In addition, we showed that it is NP-hard to find the shortest sequence of moves which solves a given subset of the cubies in an $n \times n \times 1$ Rubik's Cube. Finally, we showed that there exists a polynomial-time algorithm for solving Rubik's Cubes with dimensions $c_1 \times c_2 \times n$, where $c_1 \neq n \neq c_2$.

Our results leave several questions open. The most obvious questions concern the NP-hardness result: whether it can be modified to show the NP-hardness of optimally solving the whole $n \times n \times 1$ Rubik's Cube, and whether it can be further modified to show the NP-hardness of optimally solving the whole $n \times n \times n$ Rubik's Cube. The other questions concern approximation algorithms. In particular, is there a constant-factor polynomial-time approximation algorithm for finding an approximately optimal solution sequence from a given configuration? The analogous question for the $n^2 - 1$ puzzle has a positive answer [21]. The parallelism techniques we introduced for the diameter results seem to be central to developing such an approximation algorithm.

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. The GAP Group. GAP System for Computational Discrete Algebra. http://www.gap-system.org/.
28. V-CUBE. V-cube: the 21st century cube. http://www.v-cubes.com/.
29. Oskar van Deventer. Overlap cube 2x2x23. shapeways design. http://www.shapeways.com/model/96696/overlap_cube_2x2x23.html.
30. Panayotis Verdes. Cubic logic toy. European Patent EP 1 599 261 B1, May 2007.