Twenty-Six Moves Suffice for Rubik's Cube

Daniel Kunkle[*]
College of Computer Science
Northeastern University
Boston, MA 02115 / USA
kunkle@ccs.neu.edu

Gene Cooperman[*]
College of Computer Science
Northeastern University
Boston, MA 02115 / USA
gene@ccs.neu.edu

ABSTRACT

The number of moves required to solve any state of Rubik's cube has been a matter of long-standing conjecture for over 25 years — since Rubik's cube appeared. This number is sometimes called "God's number". An upper bound of 29 (in the face-turn metric) was produced in the early 1990's, followed by an upper bound of 27 in 2006.

An improved upper bound of 26 is produced using 8000 CPU hours. One key to this result is a new, fast multiplication in the mathematical group of Rubik's cube. Another key is efficient out-of-core (disk-based) parallel computation using terabytes of disk storage. One can use the precomputed data structures to produce such solutions for a specific Rubik's cube position in a fraction of a second. Work in progress will use the new "brute-forcing" technique to further reduce the bound.

Categories and Subject Descriptors: I.1.2 [Symbolic and Algebraic Manipulation]: Algebraic algorithms

General Terms: Algorithms, Experimentation

Keywords: Rubik's cube, upper bound, permutation groups, fast multiplication, disk-based methods

1. INTRODUCTION

Over the decades, short solutions to Rubik's cube have provided a fascination — both for researchers in techniques of search and enumeration and for hobbyists. For researchers, Rubik's cube serves as a well-known challenge problem against which otherwise diverse methods can be compared. In 1982, Singmaster and Frey [2] ended their book on *Cubik Math* with the conjecture that "God's number" is in the low 20's.

> No one knows how many moves would be needed for "God's Algorithm" assuming he always used the fewest moves required to restore the cube. It has been proven that some patterns must exist that require at least seventeen moves to restore but no one knows what those patterns may be. Experienced group theorists have conjectured that the smallest number of moves which would be sufficient to restore any scrambled pattern — that is, the number of moves required for "God's Algorithm" — is probably in the low twenties.

This conjecture remains unproven today. At the time of this conjecture, the best known bounds were a lower bound of 17 and an upper bound of 52 [2]. The current best lower bound is 20 [7]. Before this work, the best known upper bound was 27 [5]. Here, we improve that bound to 26.

Note that in all cases, we consider a *move* to be any quarter or half turn of a face of the cube, also known as the *face-turn metric*. We do not consider the alternative *quarter-turn metric*, which defines a half-turn to be two moves.

We present a new, algebraic approach that concentrates on analysis of cosets for the corresponding mathematical group. The novelty is based on a combination of:

- a subgroup chain of length two based on the square subgroup (order 663,552)
- a new fast multiplication (requiring less than 100 nanoseconds) of either a symmetrized coset or a symmetrized group element by a generator (see Sections 3, 4.1 and 5 for definitions);
- efficient parallel computation using the aggregate bandwidth of parallel disks for 7 TB of intermediate storage. (The aggregate bandwidth is comparable to the bandwidth of a single RAM subsystem.)
- an efficiently computable perfect hash function of the set of *symmetrized cosets* (see Section 4.1 for definitions), and efficient computation of its inverse; and
- a compact representation of the coset graph data structure using four bits per state to encode 1.4×10^{12} states.

The foundation of the method is to determine the maximal distance from the identity in both the square subgroup and the corresponding coset graph. At heart, the computation is simply breadth-first search using the 18 generators of Rubik's cube (including squares and inverses of generators). The out-of-core computation is required for the construction of the coset graph involving over 65 trillion cosets.

The problem is reduced in space and time through the use of the 48 symmetries of Rubik's cube (generated by the 24 symmetries of a geometric cube, plus a geometric inversion). These symmetries are generator-preserving automorphisms in their action on the group of Rubik's cube. We can define both equivalence classes of group elements (*symmetrized group elements*) and equivalence classes of cosets (*symmetrized cosets*) under the automorphism. This reduces the size of the coset graph to approximately 1.4 trillion symmetrized cosets.

[*]This work was partially supported by the National Science Foundation under Grants ACIR-0342555 and CNS-06-19616.

The paper is organized as follows. Section 2 briefly reviews some related work. Section 3 presents the overall algorithm from a high level. Section 4 presents background and some basic concepts. In particular, this includes the definitions of symmetrized group element and symmetrized coset. Section 5 describes the fast multiplication algorithm, along with the perfect hash function. Section 6 presents the details of finding a relatively tight upper bound on the distance to the identity element of all elements of a symmetrized coset. Section 7 presents details on the computations and final results.

2. RELATED WORK

One approach to finding bounds on solutions to Rubik's cube would be to produce the entire Cayley graph for the corresponding group. Cooperman, Finkelstein and Sarawagi used this method to show that 11 moves suffice for Rubik's $2 \times 2 \times 2$ cube [1]. For the full $3 \times 3 \times 3$ Rubik's cube, this is not feasible, since it has over 4.3×10^{19} states.

The first published upper bound was 52. Discovered by Thistlethwaite [2], it was based on solving the cube in a series of four steps, corresponding to a chain of subgroups of length four. The four steps were proven to have worst case lengths of 7, 13, 15, and 17, for the total of 52.

In 1992, this algorithm was improved by Kociemba [3] to use a subgroup chain of length two. In 1995, Reid [6] proved the worst case for the two steps was 12 and 18, for a total upper bound of 30. Further analysis showed that the worst case never occurs, and so a bound of 29 was shown. This bound was further refined by Radu [5] in 2006 to 27, which was the best upper bound before this work.

Besides work into methods with provable worst cases, several optimal solvers with no worst case analysis have been developed. The method developed by Kociemba and analyzed by Reid has a natural extension that guarantees optimal solutions. Korf [4] used similar techniques to optimally solve ten random cube states, one in 16 moves, three in 17 moves, and the remaining six in 18 moves.

3. OVERVIEW OF APPROACH

The group of Rubik's cube is decomposed into a chain of length two, using the *square subgroup* (the subgroup generated by using only half-turns of the faces) as the intermediate subgroup. The square subgroup has only $663,552$ elements, and there are approximately 6.5×10^{13} cosets in Rubik's group. This is in contrast to previous related approaches, which used subgroups that provided subproblems of nearly equal size.

The overall strategy for an improved upper bound for Rubik's Cube has the following three phases. First, in Section 3.1, we produce a graph of the symmetrized subgroup (symmetrized Cayley graph), with subgroup elements as nodes and the generators as edges, and show that no element has a distance of more than 13 from the identity. Second, In Section 3.2, we produce a graph of the symmetrized cosets (symmetrized Schreier coset graph), and show that no coset has a distance more than 16 from the trivial coset. These first two steps provide an initial upper bound of 29. Finally, in Section 6, we efficiently find still tighter bounds, by examining the distance from group elements in the symmetrized coset to the trivial group element, yielding the final upper bound of 26.

3.1 Construction of symmetrized Cayley subgroup graph

Because of the small size of the square subgroup, only 15,752 after reduction by symmetries, these computations are negligible when compared to the corresponding computations for the cosets.

First, we constructed the Cayley graph of the square subgroup by breadth-first search, using the square generators. This computations took only seconds, even using a simple implementation on a single computer. Then, we found the optimal solution for all elements of the square subgroup, where we allowed any of the 18 generators to be used. This was done using a bidirectional search for each of the subgroup elements, using a day of CPU time.

We chose to use bidirectional search for this case, which proceeds in two phases. First, we performed a forward search to depth 7 using all of the generators. Then, we performed one backward search for each of the 15,752 elements of the square subgroup. These backwards searches required anywhere from milliseconds to a few hours in the worst case. Overall, this optimization took less than one day, requiring no parallelization.

Table 1 in Section 7 shows the distribution of elements for the square subgroup, both for just the square generators and when allowing all generators of the cube group.

3.2 Construction of symmetrized Schreier coset graph

The construction is essentially a queue-based implementation of breadth-first search. The complexity in the algorithm is primarily due to the necessity of reducing the search space by the 48 symmetries of the cube, and of using disk.

The primary data structure is an almost perfectly dense hash array of 1.5×10^{12} entries, corresponding the the range of hash indices for all symmetrized cosets. The entries hold a four bit value describing the level at which the corresponding symmetrized coset occurs, for a total array size of 685 GB.

Algorithm 1 describes how this array is filled with the level at which every symmetrized coset occurs in the breadth-first search. It uses an iterative process with two phases, generating and merging, with one such iteration per level. As an optimization, the breadth-first search is initially conducted using only main memory, as in the traditional case, and switches to the parallel disk-based version once RAM limitations are met (in this case, at level 9).

4. NOTATION AND BASIC CONCEPTS

4.1 Group theory definitions

We review the formal mathematical definitions. Recall that a group G is a set with multiplication and an identity e ($eg = ge = g$), inverse ($gg^{-1} = g^{-1}g = e$), and an associative law ($(gh)k = g(hk)$). A *permutation* of a set Ω is a one-to-one and onto mapping from Ω to Ω. Composition of mappings provides the group multiplication, and the group inverse is the inverse mapping. A *permutation group* G is a subset of the permutations of a set Ω with the above operations. A *subgroup* $H < G$ is a subset H that is closed under group operations. A group G has *generators* $S \subseteq G$, written $G = \langle S \rangle$, if any element of G can be written as a product of elements of S and their inverses. The *order* of the group is the number of elements in it, $|G|$.

Given a group G and a subgroup $H < G$, a coset of H is the set $Hg = \{hg\colon\ h \in H\}$. A subgroup $H < G$ partitions the group into cosets. The set of all cosets is written G/H. The *conjugate* of g by h is defined by $h^g \overset{\text{def}}{=} g^{-1}hg$. $N < G$ is *normal* in G if $\forall n \in N, g \in G, n^g \in N$.

An *automorphism* α of a group G is a one-to-one and onto mapping of G such that for $g_1, g_2 \in G$, one has $\alpha(g_1g_2) = \alpha(g_1)\alpha(g_2)$. The informal idea of symmetries of Rubik's cube has its formal analogue in automorphisms.

A *Cayley graph* of a group G with generators S is a directed graph whose vertices are the elements of G and whose directed edges, (g_1, g_2), satisfy $g_1s = g_2$ for some edge label $s \in S$. Since our chosen generating set for Rubik's cube is preserved under inverses, the Cayley graph of Rubik's cube can also be considered an undirected graph. A *Schreier coset graph* of a group G with generators S and subgroup $H < G$ is a graph whose vertices are the elements of G/H and whose edges, (Hg_1, Hg_2), satisfy

Algorithm 1 Construct Symmetrized Schreier Coset Graph

```
 1: Initialize array of symmetrized cosets with all levels set to unknown (four bits per coset).
 2: Add trivial coset to array; set level $\ell$ to 0.
 3: while previous level had produced new neighbors, at next level do
 4:   {Generate new elements from the current level}
 5:   Let a segment be those nodes at level $\ell$ among $N$ consecutive elements of the array.
 6:   Scan array starting at beginning.
 7:   while we are not at the end of the array, extract next segment of array and do
 8:     for each node at level $\ell$ (representing a symmetrized coset) do
 9:       for each generator do
10:         Compute product by fast multiplication.
11:         Compute hash index of product.
12:         Save hash index in bucket $b$, where $b$ is the high bits of the hash index. Note, we only save the low order bits of the hash index not encoded by the bucket number. (This value fits in four bytes.)
13:         If bucket $b$ is full, transfer it (write it) to a disk file for bucket $b$.
14:       end for
15:     end for
16:     Transfer all buckets to corresponding disk files.
17:   end while
18:   {Now merge buckets into array of symmetrized cosets.}
19:   for each bucket $b$ on disk do
20:     Load portion of level array corresponding to bucket $b$ into main memory.
21:     for each buffered element on disk for this bucket (read in large chunks) do
22:       Look up corresponding level value in array.
23:       If a value already exists for the element, it is a duplicate. Otherwise, set its level to $\ell$.
24:     end for
25:     Write portion of level array back to disk.
26:   end for
27:   Increment level $\ell$.
28: end while
```

$Hg_1s = Hg_2$ for some edge label $s \in S$.

Symmetrized group elements and symmetrized cosets.
While the above definitions are standard, we extend them to symmetrized group elements and symmetrized cosets. Let A be a group of automorphisms of a group G. A *symmetrized group element* g^A for $g \in G$ is the set $\{\alpha(g)\colon\ \alpha \in A\}$.

Given a subgroup $H < G$, let A be a group of automorphisms of G that also preserve H ($\forall h \in H, \alpha \in A, \alpha(h) \in H$). The *symmetrized coset* Hg^A is the set of elements $\{h\alpha(g)\colon\ h \in H, \alpha \in A\} = \cup_{\alpha\in A} H\alpha(g)$. (Note $\alpha(Hg^A) = Hg^A\ \forall\alpha \in A$.)

Let A be a group of automorphisms of G. Assume that A preserves the generating set S of G. Then the *symmetrized Cayley graph* for (G, S, A) is the directed graph with vertices $\{g^A\colon\ g \in G\}$ and with edges (g_1^A, g_2^A) satisfying for some edge label $s \in S$: $\forall g_1' \in g_1^A$, $g_1's \in g_2^A$.

Without loss of generality, it suffices to consider only edge labels satisfying $g_1s \in g_2^A$ for a distinguished group element $g_1 \in g_1^A$. Note that the edge label in a symmetrized Cayley graph is not unique since we could equally well consider $\alpha(g_1')\alpha(s) \in \alpha(g_2)$ as defining that edge. Hence, the edge (g_1^A, g_2^A) has edge label s such that $g_1s = g_2$ and also edge label $s' = \alpha(s)$ for some $\alpha \in A$, such that $\alpha(g_1)\alpha(s) = \alpha(g_2)$. For any element $g_1's \in g_2^A$ satisfying $g_1' \in g_1^A$, there is an $\alpha \in A$ with $\alpha(g_1') = g_1$ and so the edge $((g_1')^A, (g_1's)^A) = (g_1^A, (g_1\alpha(s))^A)$.

Similarly, assuming that $G = \langle S\rangle$, $H < G$, and A preserves S, one defines the *symmetrized Schreier coset graph* for (G, H, S, A) as the directed graph with vertices $\{Hg^A\colon\ g \in G\}$ and with edges (g_1^A, g_2^A) satisfying $Hg_1' \in Hg_1^A$ and $Hg_1s \in Hg_2^A$. As before, in order to find all neighbors of g_1^A in a symmetrized Cayley graph, it suffices to choose any distinguished element $g' \in Hg_1^A$, and the set of neighbors is

$$\{(H(g')^A, H(g's)^A)\colon\ s \in S, g's \notin H(g')^A\}.$$

Note that for identity $e \in G$, the trivial coset $He = H$ and the trivial symmetrized coset $He^A = H$ are equal. For our work, we require the following property:

> Let $g' \in Hg^A$. Then the distance from the symmetrized coset Hg^A to the trivial symmetrized coset H in the symmetrized Schreier coset graph is the same as the distance from the coset Hg' to the trivial coset H in the Schreier coset graph.

The property is easy to prove. If there is a word w' with distinguished element $g'' \in Hg^A$ such that $g''w' \in He^A = H$, then there is an $\alpha \in A$ with $\alpha(g'') = g'$, and therefore $g'\alpha(w') = \alpha(g'')\alpha(w') \in H$. Since the automorphisms α preserve S, $\alpha(w')$ is a word in S of length d, from Hg' to the trivial coset in the Schreier coset graph.

Perfect hash function.
Next, a *perfect hash function* is a hash function that produces no collisions. Hence it is one-to-one. Section 5 describes efficient perfect hash functions both for certain classes of symmetrized cosets and for symmetrized group elements. There is an efficiently computable perfect hash function for symmetrized cosets of our chosen subgroup of Rubik's group. While this perfect hash function is not minimal, it is nearly so. Furthermore, it has an efficiently computable *inverse hash function*.

4.2 Rubik's cube definitions

We assume the reader has seen a Rubik's cube, and we provide this description solely to fix terminology according to standard conventions [2].

A Rubik's cube is built from 26 *cubies*, each able to make restricted rotations about a core of Rubik's cube. A *face* of Rubik's cube is a side. Each face is divided into 9 *facelets*, where each of the 9 facelets is part of a distinct cubie. A cubie is either an *edge cubie* (two visible facelets), a *corner cubie* (three visible facelets), or a *center cubie* (one visible face, in the center of a side). The facelets are similarly *edge facelets*, *corner facelets*, or *center facelets*.

The states of Rubik's cube can be considered as permutations on 48 facelets (the 24 corner facelets and the 24 edge facelets). The center facelets are considered to be fixed, and all rotations of Rubik's cube are considered to preserve a fixed orientation of the cube in 3 dimensions. The *home position* or solved position of Rubik's cube is one in which all facelets of a face are the same color, and (for the sake of specificity) the blue face is downwards. We will speak interchangeably about an *element of Rubik's group*, a *state*, or a *position* of Rubik's cube. Similarly, we will speak interchangeably about the *home position of Rubik's cube* or the *identity element of Rubik's group*. Hence, a position of the cube is identified with a group element that permutes the facelets from the home position to the given position.

The *moves* of Rubik's group are conventionally denoted U, U^{-1}, U^2, D, D^{-1}, D^2, R, R^{-1}, R^2, L, L^{-1}, L^2, F, F^{-1}, F^2, B, B^{-1}, and B^2. Mnemonically, U stands for a clockwise quarter turn of the "up" face, and similarly, D, R, L, F and B stand for "down", "right", "left", "front" and "back", respectively. Any of the above moves will move exactly 20 facelets. These moves also make up the generators of the Rubik's cube group, G. Standard techniques, such as Sims's algorithm for group membership, show that the order of Rubik's group is $|G| = 43,252,003,274,489,856,000$ (approximately 4.3×10^{19}).

The generators of the *square subgroup* are given by

$$Q = \langle U^2, D^2, R^2, L^2, F^2, B^2 \rangle$$

Standard techniques show that the order of the square subgroup is $|S| = 663,552$ (approximately 6.6×10^5). The *index*, or number of cosets, is $[G : S] = |G|/|S| = 65,182,537,728,000$ (approximately 6.5×10^{13}).

4.3 Symmetries (Natural Automorphisms) of Rubik's Cube

For Rubik's cube, we desire a subgroup of 48 automorphisms that preserve the set of generators: the *natural automorphisms* of Rubik's cube. Each automorphism can be identified with a *symmetry of a geometric cube*: either one of 24 rotations of the entire cube or a rotation followed by an inversion of the cube. An inversion of the cube maps each corner of the cube to the opposite corner.

A rotation of the cube maps the natural generators of Rubik's cube to generators. An inversion of the cube maps generators to inverse generators (clockwise quarter turns to counter-clockwise quarter turns). These 48 symmetries of the cube are known to preserve the natural generators of Rubik's group, and no other automorphisms of Rubik's group do so.

5. FAST GROUP MULTIPLICATION IN THE PRESENCE OF SYMMETRIES

Next, the method of fast multiplication is presented. The method breaks up a group into smaller subgroups (called *coordinates* by Kociemba [3]). The chosen subgroups are tailored for fast multiplication both within the square subgroup (group generated by squares of generators) and multiplication of the cosets by generators. Each subgroup is small enough so that the derived table-based computations fit inside CPU cache.

We will separately consider the *edge group* (the action on edge facelets) and the *corner group* (the action on corner facelets). The two actions are "linked". This will be discussed in later subsections. First, we provide the fundamental basis for our result.

5.1 Decomposition into Smaller Subgroups and Fast Multiplication

The following easy result provides the basis for our fast multiplication.

LEMMA 1. *Let G be a group with $G > H = QN$ and $Q \cap N = \{1\}$. Then given a set of canonical coset representatives of G/H, an element $g \in G$ uniquely defines q, $\bar{g}$ and n as follows.*

$$g = q\bar{g}n, \text{ where } q \in Q, \bar{g} \text{ is the canonical coset representative of } Hg, \text{ and } n \in N.$$

PROOF. Clearly, $g = h\bar{g}$ for a uniquely defined $h \in H$ and $\bar{g}$ the coset representative of Hg. The equation $h = qn'$ then uniquely defines q and n' for $q \in Q$ and $n \in N$. Define n' by $n = n'^{\bar{g}}$. Then $g = h\bar{g} = qn'\bar{g} = q\bar{g}n$. □

Lemma 1 provides the basis for a perfect hash function for G and Q, since g and the corresponding coset Qg can be encoded by ϕ_1 and ϕ_2 as follows.

$$\text{For } g = q\bar{g}n \text{ as above}, \quad \phi_1(g) = (q, \bar{g}, n) \qquad (1)$$
$$\text{and} \quad \phi_2(Qg) = (\bar{g}, n). \qquad (2)$$

Given perfect hash functions for Q, G/H and N, the above equations yield perfect hash functions for G and for G/Q through a mixed radix encoding. The importance of this particular encoding is that it adapts well to an algorithm for fast multiplication.

If the group N is also normal, then there is an efficient multiplication of a triple $(q, \bar{g}, n)$ and a pair $(\bar{g}, n)$. Recall that the *conjugate* of a group element h by g is defined as $h^g = g^{-1}hg$. Recall that a subgroup $N < G$ is a *normal subgroup* if $\forall n \in N$, and $g \in G$, $n^g \in N$. This immediately implies $ng = gn^g$ and $n_1^g n_2^g = (n_1 n_2)^g$ for $n, n_1, n_2 \in N$ and $g \in G$.

Next, consider $g = q\bar{g}n$ as above. In addition, assume that N is a normal subgroup. Let $\bar{g}s$ decompose into $q'\overline{\bar{g}s}n'$ by Lemma 1, where $\overline{\bar{g}s}$ is the canonical coset representative of $H\bar{g}s$. Note also that $H\bar{g}s = Hgs$ implies $\overline{\bar{g}s} = \overline{gs}$ for $\overline{gs}$ the canonical coset representative of Hgs. Then the product of g by a generator s can be expressed as follows.

LEMMA 2. *Let $Q < G$ and let N be a normal subgroup of G. Let $Q \cap N = \{1\}$. Let $g, s \in G$. Assume the decompositions $g = q\bar{g}n$ and $\bar{g}s = q'\overline{\bar{g}s}n'$, as given by Lemma 1. Then the following holds.*

$$\text{If } g = q\bar{g}n \text{ and } \bar{g}s = q'\overline{\bar{g}s}n', \text{ then} \quad gs = (qq')\overline{gs}(n'n^s) \quad (3)$$
$$\text{and} \quad Qgs = Q\overline{gs}(n'n^s). \quad (4)$$

Providing that Q, N and $G/(QN)$ are sufficiently small, Lemma 2 provides the foundation for precomputing small tables of all the required products, for any given $s \in G$. Typically, we choose s to be a generator since there are few generators, and so the relatively small size of the tables is maintained.

Fast multiplication by an arbitrary group element, while using small tables, is also feasible. By Lemma 1, an arbitrary group element can be written as $g = q\bar{g}n$. Assuming that g is represented as the triple $(q, \bar{g}, n)$, one can successively treat each of q, $\bar{g}$ and n as the generator s for purposes of multiplication on the right.

5.2 Fast Multiplication of Coset by Generator for Edge Group

First, consider the edge group E, the restriction of Rubik's group to act only on edge facelets. Similarly, let M_E be the restriction of the generators of Rubik's cube, M, to elements that act only on the edges. Let Q_E be the restriction of the square subgroup $Q = \langle \{s^2 \mid s \in M\} \rangle$ to act only on the edges. Hence, $Q_E = \langle \{s^2 \mid s \in M_E\} \rangle$.

Let N_E be the normal subgroup of E that fixes the edge cubies setwise, but allows the two facelets of an edge cubie to be transposed. There are 12 edge cubies, but it is not possible in Rubik's cube to transpose the facelets of an odd number of edge cubies. Group theoretic arguments then show the order of the group N_E to be 2^{11}.

It is easy to show that N_E is normal in G since for $n \in N_E$ and $g \in G$, $g^{-1}ng$ may permute the cubies according to g^{-1}, but n fixes the cubies, and g then brings the cubies back to their original position. So, $g^{-1}ng \in N$ and so N_E is normal.

We describe the method for fast multiplication of a coset of E/Q_E by a generator from M_E. We define $H_E = Q_E N_E$. The method depends on a subgroup chain

$$E > H_E > N_E, \qquad \text{for } N_E \text{ normal in } E,$$
$$H_E \stackrel{\text{def}}{=} Q_E N_E, \qquad \text{and } Q_E \cap N_E = \{1\}.$$

For the remainder of this section, we often omit the subscripts of M_E, H_E, N_E and Q_E, since we will always be concerned with the action on edges.

By Lemma 1, for any coset $Qg \in E/Q$, $Qg = Qr_1r_2$. So, Qg is represented as a pair (r_1, r_2) for r_1 a canonical coset representative in E/H and $r_2 \in N$. Given a generator s of E, equation 4 is

Edge Tables	Size	Inputs	Output
Table 1a	$1564 \times 18 \times 2\text{B}$	r_1, s	$H\overline{r_1 s} \in E/H$ for $\overline{r_1 s}$ a canonical coset representative of E/H
Table 1b	$1564 \times 18 \times 2\text{B}$	r_1, s	$\bar{r}_2 \overset{\text{def}}{=} n^{\overline{r_1 s}} \in N$, where n is defined by setting $h \overset{\text{def}}{=} \overline{r_1 s}(r_1 s)^{-1} \in H$ and uniquely factoring $h = \bar{q}n$ for $\bar{q} \in Q$, $n \in N$
Table 2	$2048 \times 18 \times 2\text{B}$	r_2, s	$r_2^s \in N$
Logical op's		r_2, r_2'	$r_2 r_2' \in N$ (using addition mod 2 on packed fields)

Figure 1: Edge table for fast multiplication

Edge Tables	Size	Inputs	Output
Table Aut	$1564 \times 18 \times 1\text{B}$	$r_{1,e}, s$	$\alpha \in A$ for $\overline{\alpha(r_1 s)}$ a canonical coset rep. of H in E (We choose α such that $\overline{\alpha(r_1 s)} = \min_{\beta \in A} \overline{\beta(r_1 s)}$.)
Table 1a (coset rep.)	$1564 \times 18 \times 2\text{B}$	$r_{1,e}, s$	$H\overline{\alpha(r_1 s)} \in E/H$ for α defined in terms of r_1 and s by Table Aut. (Note that $H^A = H$.)
Table 1b (N)	$1564 \times 18 \times 2\text{B}$	$r_{1,e}, s$	$\bar{r}_2 \overset{\text{def}}{=} n'^{\overline{\alpha(r_1 s)}} \in N$, where $h' \overset{\text{def}}{=} \alpha(r_1 s)\overline{\alpha(r_1 s)}^{-1} \in H$ and $h' = \bar{q}'n'$ for $\bar{q}' \in Q$, $n' \in N$ for α defined in terms of r_1 and s by Table Aut
Table 2	$2048 \times 18 \times 2\text{B}$	$r_{2,e}, s$	$r_2^s \in N$
Table 5	$2048 \times 48 \times 2\text{B}$	$n_e \in N, \alpha$	$\alpha(n) \in N$, where α is the output of Table Aut n is defined by $n = r_2^s$ (output of Table 2 for edges)
Logical op's		$r_{2,e}, r'_{2,e}$	$r_2 r_2' \in N$ (using addition mod 2 on packed fields)

Figure 2: Edge table for fast multiplication of symmetrized coset by generator

used below to multiply the pair (r_1, r_2) by s and return a new pair, $(r_1', r_2') = (\overline{r_1 s}, \bar{r}_2(r_2^s))$.

$$\begin{aligned} &\text{Let } r_1 s = \bar{q}\,\overline{r_1 s}\,\bar{r}_2, \\ &\qquad \text{where } \bar{q} \in Q, \bar{r}_2 \in N, \text{ and } \overline{r_1 s} \text{ is the} \\ &\qquad \text{canonical coset representative of } Qr_1 s. \qquad (5) \\ &\text{Then } Qr_1 r_2 s = Qr_1 s(r_2^s) = Q\overline{r_1 s}(\bar{r}_2(r_2^s)) \qquad (6) \end{aligned}$$

Given (r_1, r_2) and a generator s, one can compute (r_1', r_2') such that $Qr_1 r_2 s = Qr_1' r_2'$ primarily through table lookup. Figure 1 describes the necessary edge tables.

Note that the logical operations can be done efficiently, because N is an *elementary abelian* 2-group. This means that the group N is isomorphic to an additive group of vectors over a finite field of order 2. In other words, multiplication in N is equivalent to addition in $\text{GF}(2)^{11}$, the 11-dimensional vector space over the field of order 2. Addition in the field of order 2 can be executed by "exclusive or". Hence, it suffices to use bitwise "exclusive or" over 11 bits for group multiplication in N_E.

5.3 Extension to Group Action on Corners

For corners, we use the same logic as previously, but with the corner group C, the restriction M_C of the generators to corners, and the restriction Q_C of the square group to corners. Let N_C be the subgroup of C that fixes in position the corner cubies, but allows the facelets of the corner cubie to be permuted. There are 8 corner cubies, but a standard group-theoretic algorithm shows that within the subgroup C, the number of group elements fixing all corner cubies is only 3^7.

As before, we drop the subscripts for readability. Hence, the tables of Figure 1 can be reinterpreted as pertaining to corners. However, there is a small difference. Since N_C is an elementary abelian 3-group, multiplication in N_C is equivalent to addition over $\text{GF}(3)^7$.

5.4 Generalization to Fast Multiplication over Symmetrized Cosets

Assume that the automorphism group A acts on edge facelets and corner facelets, separately preserving edge and corner facelets. Assume also that A preserves the subgroups Q, N and $H = QN$. Therefore, A also maps the projections of Q, N and $H = QN$ into edges and into corners.

Assume that the symmetrized coset Qg^A is uniquely represented as $Q(r_1 r_2)^A$, where r_1 is the canonical representative of a symmetrized coset Hg^A with $H = QN$, and $r_2 \in N$, as described in Lemma 1.

The subscript e, below, indicates the restriction of a permutation to its action only on edges. Similarly, the subscript c is for corners. The subscripts are omitted where the meaning is clear.

$$\begin{aligned} &\text{For edges,} \\ &\quad Q\alpha(r_{1,e} r_{2,e} s) = Q\alpha(r_1 s)\alpha(r_2^s) = Q\overline{\alpha(r_1 s)}(\bar{r}_2 \alpha(r_2^s)), \\ &\quad \text{where } \overline{\alpha(r_1 s)} \text{ defined by Table 1a for edges,} \\ &\quad \text{where } \alpha \text{ is chosen to minimize } \overline{\alpha(r_1 s)}, \\ &\qquad \bar{r}_2 \text{ defined by Table 1b for edges, etc.} \qquad (7) \end{aligned}$$

$$\begin{aligned} &\text{However for corners,} \\ &\quad Q\alpha(r_{1,c} r_{2,c} s) = Q\alpha(\overline{r_1 s}\,(\bar{r}_2(r_2^s))) = \\ &\quad Q\alpha(\overline{r_1 s})\alpha(\bar{r}_2(r_2^s)) = Q\overline{\alpha(\overline{r_1 s})} n^{\alpha(\overline{r_1 s})}\alpha(\bar{r}_2(r_2^s)), \\ &\quad \text{where } \alpha \text{ is chosen as in equation 7,} \\ &\quad \text{where } \overline{r_1 s} \text{ defined by Table 1a,} \\ &\quad \overline{\alpha(\overline{r_1 s})} \text{ defined by Table 4a,} \\ &\quad n^{\alpha(\overline{r_1 s})} \text{ defined by Table 4b for corners,} \\ &\quad \bar{r}_2 \text{ defined by Table 1b,} \\ &\quad r_2^S \text{ by Table 2 for corners, etc.} \qquad (8) \end{aligned}$$

Note that the choice of $\alpha \in A$ for edges above depends on there being a unique such automorphism that minimizes $\overline{\alpha(r_1 s)}$. In fact, this is not true in about 5.2% of cases for randomly chosen r_1 and s. These unusual cases can be easily detected at run-time, and additional tie-breaking logic is generated. We proceed to describe tables for fast multiplication for the common case of unique $\alpha \in A$ minimizing $\overline{\alpha(r_1 s)}$, and discuss the tie-breaking logic later.

The tables that implement the above formulas follow. While it is mathematically true that we can simplify $\overline{\alpha(\overline{r_1 s})}$ into $\overline{\alpha(r_1 s)}$, we often maintain the longer formula to make clear the origins of that expression, which is needed for an implementation. As before, the subscripts e and c indicate the restriction of a permutation to its action only on edges and only on corners. Figures 2 and 3 describe the following edge tables, among others.

Ideally, one would use only the simpler formula and tables for edges, and copy that logic for corners. Unfortunately, this is not

Corner Tables	Size	Inputs	Output
Table 1a	$420 \times 18 \times 2$B	$r_{1,c}, s$	$H\overline{r_1 s} \in C/H$ for $\overline{r_1 s}$ a canonical rep. of a coset of C/H
Table 1b	$420 \times 18 \times 2$B	$r_{1,c}, s$	$\bar{r}_2 \overset{\text{def}}{=} n'^{\overline{r_1 s}} \in N$, where n' is defined by setting $h \overset{\text{def}}{=} r_1 s \overline{r_1 s}^{\,-1} \in H$ and uniquely factoring $h = \bar{q}n'$ for $\bar{q} \in Q$, $n' \in N$
Table 2	$2187 \times 18 \times 2$B	$r_{2,c}, s$	$r_2^s \in N$
Table 4a (coset rep.)	$420 \times 48 \times 2$B	$H\overline{r_{1,c}s} \in C/H, \alpha$	$H\overline{\alpha(\overline{r_1 s})} \in C/H$, where $H\overline{r_1 s}$ is the output of Table 1a, and α is the output of Table Aut on edges
Table 4b (N)	$420 \times 48 \times 2$B	$H\overline{r_{1,c}s} \in C/H, \alpha$	$n^{\overline{\alpha(\overline{r_1 s})}} \in N$, where $H\overline{r_1 s}$ is the output of Table 1a, and n is defined by setting $h = \alpha(\overline{r_1 s})\, \overline{\alpha(\overline{r_1 s})}^{\,-1} \in H$, and uniquely factoring h into qn for $q \in Q$, $n \in N$
Table 5	$2187 \times 48 \times 2$B	$n_c \in N, \alpha$	$\alpha(n) \in N$, where α is the output of Table Aut on edges n defined by computing $\bar{r}_2 = n'^{\overline{r_1 s}}$ (as in Table 1b), and r_2^s computed as in Table 2, and $\bar{r}_2 r_2^s$ computed by logical op's on corners
Logical op's		$r_{2,c}, r'_{2,c}$	$r_2 r'_2 \in N$ (using addition mod 3 on packed fields)

Figure 3: Corner table for fast multiplication of symmetrized coset by generator

possible. We must choose a representative automorphism $\alpha \in A$ for purposes of computation. We choose α based on the projection $r_{1,e}$ of r_1 into E (action of r_1 on edges). Hence, Tables 1a and 1b for edges take input r_1 and s, then compute α as an intermediate computation, then return $H\overline{\alpha(r_1 s)}$. A similar computation for corners is not possible, because the intermediate value α depends on $r_{1,e}$ and not on the corresponding element of the corner group $r_{1,c}$.

Tie-breakers: when the minimizing automorphism is not unique.

Table Aut in the previous table for edges defines an automorphism α that minimizes $\overline{\alpha(r_1 s)}$. Unfortunately, there is not always a unique such α. In such cases, one needs a tie-breaker, since different choices of α will in general produce different encodings (different hash indices).

For each possible value of $\overline{\alpha(r_1 s)}$, with α chosen to minimize the expression, we precompute the stabilizer subgroup $B \leq A$ defined by

$$B = \{\beta \in A\colon\ \overline{\beta(\overline{\alpha(r_1 s)})} = \overline{\alpha(r_1 s)}\}$$

we use the formulas and additional table below to find the unique $\beta \in B$ such that the product $\alpha\beta$ minimizes the edge pair result $(r'_{1,e}, r'_{2,e})$. Where even this is not enough to break ties, we compute the full encoding, while trying all possible tying automorphisms. This latter situation arises in only 0.23% of the time, and does not contribute significantly to the time. The tables of Figure 4 suffice for these computations.

For edges,

$$\begin{aligned}
&Q\beta(\alpha(r_{1,e}r_{2,e}s)) = Q\beta(\alpha(r_1 s))\beta(\alpha(r_2^s)) = \\
&Q\beta(\overline{\alpha(r_1 s)})\left(\beta\left(\bar{r}_2\alpha(r_2^s)\right)\right) = Q\overline{\alpha(r_1 s)}\bar{r}'_2\left(\beta\left(\bar{r}_2\alpha(r_2^s)\right)\right), \\
&\text{ where } \overline{\alpha(r_1 s)} \text{ defined by Table 1a for edges,} \\
&\text{ where } \alpha \text{ is chosen to minimize } \overline{\alpha(r_1 s)}, \\
&\text{ and } \beta \in A \text{ satisfies } Q\overline{\beta(\overline{\alpha(r_1 s)})} = Q\overline{\alpha(r_1 s)} \\
&\text{ and } \beta(\overline{\alpha(r_1 s)}) = \overline{\alpha(r_1 s)}\bar{r}'_2 \ (r'_2 \text{ defined in Table 3}), \\
&\text{ and } \bar{r}_2 \text{ defined by Table 1b for edges.}
\end{aligned} \tag{9}$$

However for corners,

$$\begin{aligned}
&Q\beta(\alpha(r_{1,c}r_{2,c}s)) = Q\beta(\alpha(\overline{r_1 s}\,(\bar{r}_2(r_2^s)))) = \\
&Q\beta(\alpha(\overline{r_1 s}))\beta(\alpha(\bar{r}_2(r_2^s))) = \\
&Q\overline{\beta(\alpha(\overline{r_1 s}))}n^{\overline{\beta(\alpha(\overline{r_1 s}))}}\beta(\alpha(\bar{r}_2(r_2^s))) \\
&\text{ where } \alpha \text{ and } \beta \text{ are chosen as in equation 9,} \\
&\text{ and other quantities based on} \\
&\text{ the previous Corner Tables using } \alpha\beta.
\end{aligned} \tag{10}$$

Table 1c is implemented more efficiently by storing the elements of each of the possible 98 subgroups of the automorphism group, and having Table 1c point to the appropriate subgroup $B \leq A$, stabilizing $r_{1,e}, s$.

5.5 Optimizations

In the discussion so far, we produce the encoding or hash index of a group element based on an encoding of the action of the group element on edges, along with an encoding of the action of the group element on corners. We can cut this encoding in half due to parity considerations.

Consider the action of Rubik's cube on the 12 edge cubies and the 8 corner cubies, rather than on the facelets. We define the *edge parity* of a group element to be the parity (even or odd) in its action on edge cubies. (Recall that the parity of a permutation is odd or even according to whether the permutation is expressible as an odd or even number of transpositions.) The *corner parity* is similarly defined.

The edge and corner parity of a symmetrized coset, Hg^A, are well-defined, and are the same as the edge and corner parity of g. This is so because $H = QN$, and elements of Q and N have even edge parity and even corner parity. Parity is unchanged by the action of an automorphism.

For Rubik's cube, the natural generators have the edge parity equal to the corner parity. So this property extends to all group elements, and hence to all symmetrized cosets Hg^A. Therefore, our encoding can assume that edge and corner parities of symmetrized cosets are equal. The size of the corresponding hash table is thus reduced by half.

Nearly Minimal Perfect Hash Function.

If we were to only use cosets instead of symmetrized cosets (no automorphism), then the perfect hash function that we have described implicitly would also be a *minimal perfect hash function*. However, there are examples for which $Q\alpha(g) = Qg$ for α not the identity automorphism.

A computation demonstrates that the perfect hash function of Section 5.4 with the addition of the parity optimization has an efficiency of 92.3%. In fact, we compute that there are

$$1,471,074,877,440 \approx 1.5 \times 10^{12} \approx |G|/|Q|/44.3$$

symmetrized cosets. The ratio 44.3/48 yields our efficiency ratio of 92.3%. Further details are omitted due to lack of space.

5.6 Fast Multiplication in Square Subgroup

There is a similar algorithm for fast multiplication in the square subgroup, which is omitted due to lack of space.

Edge Tables	Size	Inputs	Output
Table Mult Aut	$48 \times 48 \times 1$B	α, β	the product $\overline{\alpha\beta \in A}$
Table 1c (A)	$1564 \times 18 \times 1$B	$r_{1,e}, s$	$\{\beta \in A\colon\ \beta(\overline{\alpha(r_1 s)}) = \overline{\alpha(r_1 s)}\}$
Table 3 (N)	$2048 \times 48 \times 2$B	$H\overline{r_{1,c}s} \in C/H, \alpha$	$\bar{r}_2' \overset{\text{def}}{=} n''^{\overline{\beta(r_1)}} \in N$, where β taken from Table 1c, and
			where $h'' \overset{\text{def}}{=} \beta(r_1)\overline{\beta(r_1)}^{\,-1} \in H$, and $h'' = \bar{q}''n''$, for $\bar{q}'' \in Q, n'' \in N$

Figure 4: Edge table for fast multiplication of symmetrized coset by generator, adjusted to break ties

6. "BRUTE FORCE" UPPER BOUNDS ON SOLUTIONS WITHIN A COSET

6.1 Goal of Brute Forcing Cosets

Having constructed the Schreier coset graph, one wishes to test individual cosets, and prove that all group elements of that coset are expressible as words in the generators of length at most u. Hence, u is the desired upper bound we wish to prove.

Recall that G is the group of Rubik's cube, and Q is the square subgroup. Consider a coset Qg at a level ℓ (distance ℓ from the home position, or identity coset, in the coset graph). Let d be the diameter of the subgroup Q. For any group element $h \in Qg$, clearly its distance from the identity element is at least ℓ and at most $\ell + d$. We describe a computation to produce a finer upper bound on the distance of any $h \in Qg$ from the identity element.

Because our subgroup Q (the square subgroup) is of such small order, it is even feasible to simply apply an optimal solver to each element of a coset. If the optimal length words for each element is always less than or equal to u, then we are done. However, we will present a more efficient technique, which can scale to millions of cosets.

For simplicity, we first assume that we are not applying any symmetry reductions using the automorphism group. So, each coset contains 663,552 elements, just as does the square subgroup.

6.2 Basic Algorithm

Note that for the coset Qg, there can be many paths in the coset graph from the identity coset to Qg. In terms of group theory, there are multiple words, w_1, $w_2, \ldots$, where each word is a product of generators of Rubik's group, and $Qw_1 = Qw_2 = \ldots = Qg$. Note that in general, the words are distinct group elements: $w_1 \neq w_2$, etc. Nevertheless, $w_1 w_2^{-1} \in Q$. This is the key to finding a refined upper bound.

Next, suppose our goal is to demonstrate an upper bound u, for $\ell \leq u < \ell + d$. Let $\operatorname{dist}(q)$ denote the distance from an element $q \in Q$ to the identity element in the Cayley graph of G with the original Rubik generators. Let Q_u be $\{q \in Q \mid \operatorname{dist}(q) \leq u\}$, the subset of Q at distance from the identity at most u.

Next, consider $Q_k g \overset{\text{def}}{=} \{qg \mid q \in Q_k\}$. Assume that the words w_i are of length d in the generators of Rubik's group, and that $Qw_1 = Qw_2 = \ldots = Qg$. Note that for all elements of $Q_k w_1$, there is an upper bound, $k + d$. Similarly, for all elements of $Q_k w_2$, there is an upper bound, $k + d$. Therefore, the elements of $Q_k w_1 \cup Q_k w_1$ have an upper bound of $k + d$. More compactly,

$$\operatorname{dist}(Q_k w_1 \cup Q_k w_2) \leq k + d$$

More generally, for w_i a word in the generators of G, let $\operatorname{len}(w_i)$ be the length of that word. Then

$$\operatorname{dist}(Q_{k+d-\operatorname{len}(w_1)} w_1 \cup Q_{k+d-\operatorname{len}(w_2)} w_2) \leq k + d$$

since the length of any word in $Q_{k+d-\operatorname{len}(w_1)} w_1$ is at most $(k + d - \operatorname{len}(w_1)) + \operatorname{len}(w_1) = k + d$ and similarly for w_2.

Define the complement $Q_j' \overset{\text{def}}{=} Q \setminus Q_j$. We can now write:

$$Q'_{k+d-\operatorname{len}(w_1)} w_1 \cap Q'_{k+d-\operatorname{len}(w_2)} w_2 \supseteq \{h \in Qg \mid \operatorname{dist}(hg) > k+d\}$$

Clearly, the above equation can be generalized to the intersection of multiple words, w_1, $w_2, \ldots$.

$$\cap_i Q'_{k+d-\operatorname{len}(w_i)} w_i = \emptyset \Longrightarrow \forall i, \operatorname{dist}(Qw_i) = \operatorname{dist}(Qg) \leq k + d$$

Finally, for purposes of computation, Algorithm 2, below, captures these insights.

Algorithm 2 Coset Upper Bound

Input: a subgroup Q of a group G, a coset Qg; a desired upper bound ℓ; and a set of words $h_1, h_2, \ldots$ in generators of G such that $Qgh_i = Q$.

Output: a demonstration that all elements of Qg have solutions of length at most ℓ or else a subset $S \subseteq Qg$ such that all elements of $Qg \setminus S$ are known to have solutions of length at most ℓ.

1: Let $k = \ell - \operatorname{len}(g)$. Let $U_0 = \{q \in Q \mid \operatorname{len}(q) > k\} \subseteq Q$. Then $(Q \setminus U_0)g$ is the subset of elements in the coset Qg which are known to have solutions of length at most ℓ. The set $U_0 g$ is the "unknown set", for which we must decide if they have solutions of length ℓ or less.

2: For each $i \geq 1$, let $U_i = U_{i-1} \setminus \{q \in U_{i-1} \mid \operatorname{dist}(qgh_i) \leq \ell - \operatorname{len}(h_i)\}$. (Note that $qgh_i \in Q$). By $\operatorname{dist}(qgh_i)$, we mean the shortest path in the full set of generators of G. If $\operatorname{dist}(qgh_i) \leq \ell - \operatorname{len}(h_i)$, then qg has a solution of length at most ℓ. The solution for qg is given by a path length $\operatorname{len}(h_i)$ followed by a path of length $\ell - \operatorname{len}(h_i) = \operatorname{dist}(qgh_i)$.

3: If $U_i = \emptyset$ for some $i \leq j$, then we have shown that all elements of Qg have solutions of length at most ℓ. If $U_j \neq \emptyset$, then we have shown that all elements of $(Q \setminus U_j)g$ have solution length at most ℓ.

For purposes of implementation, note that $gh_i \in Q$. So, for $q \in Q$, qgh_i can be computed by fast multiplication within the subgroup Q.

6.3 Using Symmetries

We now generalize the method of the previous section to take account of reductions through symmetries.

First, note that for a symmetrized coset with representative coset Qg and for a natural automorphism α, $\operatorname{dist}(Qg) = \operatorname{dist}(Q\alpha(g))$. (This was demonstrated in Section 3.2). Furthermore, for any $h \in Qg$, $\operatorname{dist}(h) = \operatorname{dist}(\alpha(h))$.

From this, it is clear that any upper bound on the distance of elements in Qg from the identity will also hold for $Q\alpha(g)$. So, it suffices to determine upper bounds for a single representative of each automorphism class.

Finally, one must determine whether $hw_i^{-1} \notin Q'_{k+d-\operatorname{len}(w_i)}$. This can be done by maintaining a hash table mapping all elements $q \in Q$ to $\operatorname{dist}(q)$.

7. EXPERIMENTAL RESULTS

Our experimental results have proven that 26 moves suffice for any state of Rubik's cube. This was achieved in three steps: proving that all elements of the square subgroup are within 13 of the identity; proving that all cosets are within 16 of the trivial coset; and, refining the bound on the farthest cosets by brute force, reducing the bound by 3.

7.1 Square Subgroup Elements are within 13 of the Identity

Recall, from Section 3.1, the following two-step process for this computation. First, we constructed the Cayley graph of the subgroup by breadth-first search, using the square generators. Then, the distance for each of these elements from the identity, when allowing all generators of the full group, were determined using bidirectional search. All of these computations were done on a single computer in under a day.

Table 1 shows the distribution of element distances in the square subgroup, using either the square generators or the full set of generators.

Square Generators				All Generators			
Dist.	Elts	Dist.	Elts	Dist.	Elts	Dist.	Elts
0	1	8	1258	0	1	8	1871
1	1	9	2627	1	1	9	4093
2	2	10	4094	2	2	10	5394
3	5	11	4137	3	5	11	2774
4	18	12	2231	4	18	12	620
5	56	13	548	5	62	13	4
6	162	14	114	6	214		
7	482	15	16	7	693		
		Total	15752			Total	15752

Table 1: Distribution of elements in the square subgroup, after reduction by symmetries.

7.2 Cosets are within 16 of the Trivial Coset

The dominant time for our computations was in producing the symmetrized Schreier coset graph for the square subgroup in the group of Rubik's cube, as described in Section 3.2.

The computation used the DataStar cluster at the San Diego Supercomputer Center (SDSC). We used 16 compute nodes in parallel, each with 8 CPUs and 16 GB of main memory. For out-of-core storage, we used DataStar's attached GPFS (IBM General Parallel File System). We used up to 7 terabytes of storage at any given time, as a buffer for newly generated states in the breadth-first search. The final data structure, associating a 4-bit value with each symmetrized coset, used approximately 685 GB.

The computation required 63 hours, or over 8000 CPU hours. The fast multiplication algorithm allowed us to multiply a symmetrized coset by a generator at a rate between 5 and 10 million times per second, depending on the size of available CPU caches.

Table 2 shows the distribution of distances for cosets in the symmetrized Schreier coset graph.

Dist.	Elements	Distance	Elements	
0	1	9	80741117	$\approx 8.1 \times 10^7$
1	1	10	1028869318	$\approx 1.0 \times 10^9$
2	3	11	12787176355	$\approx 1.3 \times 10^{10}$
3	23	12	140352357299	$\approx 1.4 \times 10^{11}$
4	241	13	781415318341	$\approx 7.8 \times 10^{11}$
5	3002	14	421980213679	$\approx 4.2 \times 10^{11}$
6	38336	15	330036864	$\approx 3.3 \times 10^8$
7	490879	16	17	
8	6298864			
		Total	1357981544340	$\approx 1.36 \times 10^{12}$

Table 2: Distribution of symmetrized cosets of the square subgroup.

7.3 Brute Forcing of 3 Levels

Kociemba's Cube Explorer software [3] was used to show that all cosets at level 3 have solutions of length at most 14. To do this, it sufficed to analyze the elements at levels 12 and 13 from the square subgroup. Denoting the set of those subgroup elements S, and denoting all of the elements at level 3 in the cosets by T, we considered all pairwise products $S \times T$. There are $(620 + 4) \times 23 = 14{,}352$ such elements. Cube Explorer was run on each such element. This proves that for a coset at coset level $x > 3$, $(x - 2) + 13$ moves suffice. Combining this with the expected depth of $x = 16$ for the symmetrized Schreier coset graph yields an upper bound of $(x - 2) + 13 \leq 27$ moves for solutions to Rubik's cube.

We similarly showed that none of the elements in any of the 17 cosets at level 16 required more than 26 moves, again using Cube Explorer. In combination with the above, this demonstrates an upper bound of 26 moves for solutions to Rubik's cube.

7.4 Further Brute Forcing

Our continuing work is using the efficient brute forcing techniques given in Section 6 to further reduce the upper bound from 26 to 25 moves. We plan to achieve this by brute forcing all cosets at some early level by three moves.

Our current experiments indicate that there exist elements that can not be brute forced by three moves out to level 8. Directly considering all cases at level 9 is computationally expensive, as there are over 80 million level 9 cosets, each with 3398 unproven elements (corresponding to the last three levels of the square subgroup). Instead, we use the new brute forcing techniques for cosets at earlier levels, and "project" the remaining unproven elements to later levels.

So far, we have removed over 95% of the elements across all cosets at level 8. However, only about 10% of the cosets at level 8 have no remaining cases, and require additional brute forcing. We anticipate that, by using our efficient brute forcing technique, and significant computing power, we will be able handle the remaining cases at levels 8 and 9, and therefore prove that 25 moves suffice for Rubik's cube.

8. REFERENCES

[1] Gene Cooperman, Larry Finkelstein, and Namita Sarawagi. Applications of Cayley graphs. In *AAECC: Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, International Conference*, pages 367–378. LNCS, Springer-Verlag, 1990.

[2] Alexander H. Frey, Jr. and David Singmaster. *Handbook of Cubik Math*. Enslow Publishers, 1982.

[3] Herbert Kociemba. Cube Explorer. `http://kociemba.org/cube.htm`, 2006.

[4] Richard Korf. Finding optimal solutions to Rubik's cube using pattern databases. In *Proceedings of the Workshop on Computer Games (W31) at IJCAI-97*, pages 21–26, Nagoya, Japan, 1997.

[5] Silviu Radu. Rubik can be solved in 27f. `http://cubezzz.homelinux.org/drupal/?q=node/view/53`, 2006.

[6] Michael Reid. New upper bounds. `http://www.math.rwth-aachen.de/~Martin.Schoenert/Cube-Lovers/michael_re%id__new_upper_bounds.html`, 1995.

[7] Michael Reid. Superflip requires 20 face turns. `http://www.math.rwth-aachen.de/~Martin.Schoenert/Cube-Lovers/michael_re%id__superflip_requires_20_face_turns.html`, 1995.