

The exact query complexity of hidden subgroup detection for the symmetric group

Chad Brewbaker
Dept. of Electrical and Computer Engineering
Iowa State University
Ames, IA 50010
crb002@iastate.edu

September 17, 2008

Abstract

In this note we derive the exact query complexity of hidden subgroup detection for the symmetric group. Optimal query sets, while asymptotically similar to brute force search, are shown to yield significant savings for tractable problem sizes.

1 Introduction

This research was motivated by the following question.

“What is the optimal algorithm for generic symmetry detection?”

We formalize this problem by limiting the scope to detecting subgroups of the symmetric group, and make it “generic” by only allowing for an oracle that accepts or rejects a given permutation based on some criteria as long as the set of accepted permutations forms a group. Thus, we are now interested in the query complexity of hidden (non-trivial) subgroup detection for the symmetric group.

Far more efficient algorithms exist when we have a priori knowledge that the hidden subgroup corresponds to the automorphisms of a specific combinatorial object. However, even simple objects like graphs have no known polynomial time algorithms, nor has graph automorphism been shown to be NP-complete. According to [3] graph isomorphism and automorphism are in $O(\exp(\sqrt{cn \log n}))$. Furthermore, in [7] it was shown that hypergraph isomorphism and automorphism are in $O(c^n)$.

It should be noted that the query complexity of hidden subgroup detection for the symmetric group was touched upon in a graph automorphism

setting [1, 8] where permutations composed of disjoint concatenated two-cycles were used to generate asymptotic bounds.

The harder problem of constructing a generating set for a hidden subgroup was discussed from a computational learning theory perspective in [11].

This work should not be confused with research on “black box group algorithms” [2, 4, 12], where one is presented with a generating set of some group, while in our case no such information is given.

2 Preliminaries

In this paper permutations will be denoted as π . Permutation composition is defined where given a sequence of permutations $\pi_0 \times \pi_1 \times \dots \times \pi_k$ the permutation created by applying the permutations in the list from right to left, $\pi_0(\pi_1(\dots(\pi_k(x))))$, is their composition. For example if $\pi_0 = (0, 4)(1, 2, 3)$ and $\pi_1 = (0)(1, 2, 4)(3)$ then $\pi_0 \times \pi_1 = (0, 4, 2)(1, 3)$

The permutation π^k will be the composition of k copies of π . The set of permutations on n elements will be denoted S_n . A group of permutations, from here on referred to as a *group*, is a set of permutations $H \subseteq S_n$ that are closed under permutation composition.

The framework for our proof will be based on a graph optimization problem.

Definition 1 (MIN-DOMINATING-SET). *Given a graph with vertex set V , a MIN-DOMINATING-SET is the smallest set $D \subseteq V$ such that for each vertex $v \in V$, either $v \in D$ or there is a (directed) edge from a vertex in D to v .*

Also, we will use a special relation between two permutations where having one permutation in a group implies the presence of the other.

Definition 2 (Detection). *We say that a permutation π_i detects permutation π_j if $\pi_i \in (\pi_j)^k$ for some natural number k , i.e. π_i is in the cyclic group generated by π_j . Thus, if π_j is present in a group then π_i will also be present.*

Using the *detection* relation we can form a graph to describe our subgroup detection problem.

Definition 3 (Detection Graph of S_n). *Let G_n be a directed graph (without loops) whose vertices are labeled with a bijective mapping of the $n!$ permutations in S_n .*

Let π_i be the permutation label of vertex i , and π_j be the permutation label of vertex j .

Place an edge from π_i to π_j if π_i detects π_j .

Remove all edges adjacent to the identity permutation.

Figure 1 is a diagram of the detection graph for the set of permutations on 3 elements.

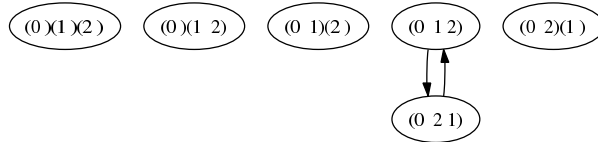


Figure 1: The detection graph of S_3 .

2.1 Structure of the Minimum Detection Sets

The following lemma ties the detection graph to our subgroup detection problem.

Lemma 1. *Any MIN-DOMINATING-SET for the detection graph of S_n provides a minimum set of queries for hidden subgroup detection.*

Proof. Follows from the definition of MIN-DOMINATING-SET. \square

In the general case computing MIN-DOMINATING-SET is NP-hard [6], however our proof below is constructive. During the preliminary analysis [10] was useful as it contains polynomial time pre-processing algorithms to simplify the search. It also might be of note that this graph written in lexicographic order of the underlying permutation labels seemed to exhibit exponential runtime behavior with NAUTY [9] version 2.2.

We would now like to show which sets of permutations form a MIN-DOMINATING-SET in the detection graph. Let H be the subgroup of S_n that we want to detect.

A particular class of permutations will be of interest.

Definition 4 (Disjoint Concatenated Prime Cycle Permutation (DCPCP)). *A DCPCP is a permutation composed of 1 or more disjoint prime length cycles of the same size.*

The following lemmas show that DCPCPs are necessary and sufficient detection sets.

Lemma 2. *Only DCPCPs detect DCPCPs.*

Proof. The cyclic group of a DCPCP contains only DCPCPs, so by the definition of the detection relation, nothing else can detect them. \square

Lemma 3. *Every permutation is detected by a DCPCP.*

Proof. If the permutation is a DCPCP, then by the previous lemma we are done. If not, we can extract a DCPCP that detects π . Calculate the least common multiple of its cycle lengths, i.e. the size of the cyclic group generated by π and denote this as k . Then take one of k 's prime factors which we will denote as i . The permutation $\pi^{\frac{k}{i}}$ will send all cycles with length not divisible by i to the identity, and will turn the other cycles into concatenated i cycles. Thus, we have extracted a DCPCP to detect π . \square

From the previous two lemmas we can derive the following following theorem.

Theorem 1. *A minimal detection set for S_n consists of one element from each DCPCP's cyclic group.*

Recall that an Abelian (commutative) permutation group is one where a composition of permutations can be made in any order without changing the result.

Corollary 1 (Abelian impotency). *Prior knowledge that our hidden subgroup H is Abelian is not enough to reduce the size of the minimal detection set.*

Proof. As DCPCPs generate Abelian groups, they must be detected anyway. \square

2.2 Enumeration of the Minimal Detection Sets

We will now count the number of elements in a minimum detection set for S_n .

This lemma will let us count the whole set of repeated prime size partitions that have size $\leq n$.

Lemma 4. *Let p index prime numbers $\leq n$. The number of distinct concatenated prime size partitions on n elements is*

$$\sum_p^n \sum_{i=1}^{\lfloor \frac{n}{p} \rfloor} 1.$$

Proof. The outer sum iterates over all primes $\leq n$, and the inner sum iterates over the number of partitions of this size that will fit into n . □

Now to fill in the partitions with permutations we can over-count and take all permutations on n elements composed of i disjoint p cycles.

Lemma 5. *The set of all permutations over n elements composed of i disjoint cycles of size p is*

$$\frac{n!}{(n - ip)!i!p^i}.$$

Proof. We will use a more general characterization from [5] attributed to Cayley.

“Let (a_1, a_2, \dots, a_n) be an n -tuple of nonnegative integers so that $\sum_{i=1}^n a_i \cdot i = n$. Then the number of n -permutations of type (a_1, a_2, \dots, a_n) is

$$\frac{n!}{a_1!a_2! \dots a_n! 1^{a_1} 2^{a_2} \dots n^{a_n}}.”$$

We can shorten this equation since our permutations are composed of p cycles and singleton cycles to

$$\frac{n!}{a_1!a_p! 1^{a_1} p^{a_p}}.$$

Set a_p to i to denote our i concatenated p -cycles, and set a_1 to $(n - ip)$ to denote the leftover elements that form singleton cycles,

$$\frac{n!}{(n - ip)!i! 1^{n-ip} p^i},$$

then extract the redundant 1^{n-ip} term to receive

$$\frac{n!}{(n-ip)!i!p^i}.$$

□

Finally, we only want one element for each cyclic group generated by a DCPCP.

Lemma 6. *Given the $p - 1$ DCPCPs in the same cyclic group, we can normalize this quantity to select a single member of this equivalence class by multiplying by*

$$\frac{1}{p-1}.$$

By applying the three previous lemmas we derive the following theorem.

Theorem 2 (Query Complexity). *Let p be prime numbers $\leq n$. The size of a minimal query set for nontrivial subgroup detection in S_n is*

$$\sum_p^n \sum_{i=1}^{\lfloor \frac{n}{p} \rfloor} \frac{n!}{(n-ip)!i!p^i(p-1)}.$$

The minimal detection sets for $n = 1, \dots, 10$ have sizes $\{0, 1, 4, 13, 41, 151, 652, 2675, 10579, 59071\}$.

Note that queries made from a minimal detection set are non-adaptive, i.e. the order in which they are made does not matter, so any program using them is inherently parallel.

When n is a prime number we can note that this quantity has a lower bound of $\Omega((n-2)!)$ by setting $n = p$ and $i = 1$. We of course always have the upper bound of $O(n!)$. Tight bounds would rely on better knowledge of the distribution of primes.

Using a minimum query set can yield significant savings over checking all nontrivial permutations, even for tractable problem sizes. This is illustrated in Table 1.

Table 1: Savings for small n over exhaustive testing.

n	Savings	Queries Eliminated
3	20.0%	1
4	43.5%	10
5	65.5%	78
6	79.0%	568
7	87.1%	4387
8	93.4%	37644
9	97.1%	352300
10	98.4%	3569728

3 Acknowledgements

Funding for this research was provided by US Dept. of Education GAANN fellowship. We would like to thank Srinivas Aluru for suggesting the research topic of protein folding which was the original application of this algorithm.

References

- [1] V. Arvind and Piyush P. Kurur. Graph isomorphism is in SPP. *Information and Computation*, 204:835–852, 2006.
- [2] V. Arvind and N. V. Vinodchandran. Solvable black-box group problems are low for PP. *Theoretical Computer Science*, 180:17–47, 1997.
- [3] Lázló Babai and Eugene M. Luks. Canonical labeling of graphs. *STOC*, pages 171–183, 1983.
- [4] Lázló Babai and Endre Szeméredi. On the complexity of matrix group problems. *IEEE FOCS*, 1984.
- [5] Miklós Bóna. *Combinatorics of Permutations*. Chapman and Hall, 2004.
- [6] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [7] Eugene M. Luks. Hypergraph isomorphism and structural equivalence of boolean functions. *STOC*, pages 652–658, 1999.

- [8] Ka Leung Ma. *In Solving the Dominating Set Problem: Group Theory Approach*. PhD thesis, Concordia University of Montreal, 1998.
- [9] Brendan McKay. Practical graph isomorphism. *Congressus Numerantium*, pages 45–87, 1981.
- [10] Rolf Niedermeier. *Invitation to Fixed Parameter Algorithms*. Oxford University Press, 2006.
- [11] N. V. Vinodchandran. *Counting Complexity and Computational Group Theory*. PhD thesis, Institute of Mathematical Sciences, Chennai, India, 1999.
- [12] N. V. Vinodchandran. Counting complexity of solvable black-box group problems. *SIAM Journal on Computing*, 33(4):852–869, 2004.