Contemporary Mathematics

Normalizer calculation using automorphisms

Alexander Hulpke

Abstract. We study how to reduce the calculation of the normalizer of a subgroup U in G to the calculation of centralizers and element conjugacy in G and calculations in the automorphism group $\mathrm{Aut}(U)$. Experimental run times show that this can be substantially faster than existing backtrack algorithms.

1. Introduction

A fundamental technique for the computation of subgroups in a permutation group, such as centralizer, normalizer, subgroup intersection or set stabilizer, is backtrack search. The idea for this stems from Sims's work on stabilizer chains [**Sim70**]. A description can be found in [**Ser03**] or [**HEO05**]. More recent developments using partition backtrack algorithms are described in [**Leo97**] or [**The97**].

In general, backtrack algorithms are of exponential complexity, though in practice the performance is often good. For example the computation of centralizers is considered to be easy in practice ([**Ser03**, p.205]). Furthermore, [**Luk93**] shows that centralizer, set stabilizer and subgroup intersection are polynomial time equivalent (and therefore also could be considered as "easy in practice", even though neither of them is known to be polynomial time).

Luks also shows that the computation of normalizers is at least as hard as computing centralizers, though it is not known whether the algorithmic complexity actually differs.

Recently, Luks and Miyazaki [**LM02**] have shown that the calculation of normalizers in permutation groups is polynomial time for groups with bounded composition factors. The complexity of the general case is unknown.

In practice – contrary to centralizer and its equivalents – normalizer computations are often hard with current backtrack-based algorithms. (A footnote on p. 121 in [**HEO05**] describes them as "among the most difficult problems in CGT".) This motivates the search for a reduction to easier problems.

To see why normalizer computations are hard consider the reductions possible in a backtrack search for $N = N_G(U)$ with $U \leq G \leq S_n$. First, for each coset of

2000 *Mathematics Subject Classification.* Primary 20B40; Secondary 20B05.

Key words and phrases. normalizer, computation, automorphism group, backtrack, permutation equivalent.

Supported in part by NSF Grant # 0633333.

the normalizer (or a subgroup of the normalizer) only one element has to be tested. If the index $[G : N]$ is large this method alone will not yield sufficient reductions.

In many cases it is possible to reduce this index by considering combinatorial or geometric invariants of the group U that is to be normalized (see for example [**RD04**], [**Hul05**, section 11], [**Miy06**]): the normalizer N must preserve the set of such invariants, thus the group G can be replaced by the stabilizer S of this invariant set in G and thus $N_G(U) = N_S(U)$. This is most fruitful if G is the full symmetric group.

This approach fails, however, if the subgroup U is very "symmetric" and has only little distinct permutational structure.

The second type of reduction employed is the actual problem-specific pruning of the backtrack tree: Sims's original work already suggests using the fact that $N_G(U)$ permutes the orbits of U on pairs of points. If U is elementary abelian or regular, however this approach often yields little improvement.

Another improvement, due to Holt [**Hol91**], uses the fact that elements of $N_G(U)$ induce automorphisms of U. In the case of a regular subgroup U this implies that any element of $N_G(U)$ is determined uniquely by specifying the images of two points. Furthermore [**Hol91**] suggests a reduction if G and U have a faithful permutation representation on a subset of the permutation domain: in this case the images of points in this domain determine the automorphism and thus determine the images of all other points.

However, if $U \leq S_n$ is elementary abelian but neither regular nor the intransitive direct product C_p^k, neither of these reductions might be applicable. In the author's recent work on classifying transitive permutation groups [**Hul05**] these problems frequently occurred.

2. Determining the normalizer from its action

As the computation of element centralizers (and the corresponding task of finding conjugating elements) seems to be so much easier in practice, the following approach for normalizing "small" or "known" subgroups looks promising:

Let Ω be a finite set, $G \leq S_\Omega$ be a permutation group and $U \leq G$. Then the normalizer $N_G(U)$ induces (by conjugation) automorphisms of U, and the kernel of this action is $C_G(U)$. Thus we can consider the quotient $N_G(U)/C_G(U)$ as a subgroup of $\mathrm{Aut}(U)$. We will assume that we know $\mathrm{Aut}(U)$ or can get it very cheaply. (For example this is the case if U is comparatively small or if it is abelian.)

Next, suppose we know the subgroup $A \leq \mathrm{Aut}(U)$ which is induced by $N_G(U)$. We then can compute, for every α in a generating set X of A, a conjugating element $g_\alpha \in G$ such that $u^\alpha = u^{g_\alpha}$ (see Section 3.3 for details). With this we get that

$$N_G(U) = \langle C_G(U), g_\alpha \rangle_{\alpha \in X}. \tag{1}$$

Assuming we know the subgroup A, this would reduce the normalizer computations to computations of centralizers and element conjugacies. Unfortunately, the exact determination of A seems to be hard in general as well. We therefore employ the following approximation:

Conjugacy in a permutation group preserves the cycle structure of elements. We thus partition the elements of U into classes $\{\mathcal{C}_i\}$ based on cycle structure. Let $B \leq \mathrm{Aut}(U)$ be the stabilizer of this partitioning. Clearly $A \leq B$.

If we consider $\mathrm{Aut}(U)$ as a permutation group on the (non-identity) elements of U, we can calculate B as an iterated set stabilizer. This again is a backtrack calculation, albeit of a different degree.

Using the conjugacy test from Section 3.3 for finding a conjugating element g_α, we can also check whether an element $\beta \in B$ is in fact contained in A. We thus can use a backtrack search over B (considering it as a permutation group and using [**HEO05**, §4.6.2]) to determine A. Doing so with the constructive test for normalizer induced automorphisms will automatically find conjugating permutations g_α for all the generators α that are needed in equation (1).

Essentially we are trading a normalizer backtrack search in G for a backtrack search in $\mathrm{Aut}(U)$ and an element conjugacy backtrack in G. If U (and thus $\mathrm{Aut}(U)$) is substantially smaller than G, this can be expected to yield shorter overall run time. The example runtimes in Section 6 support this claim.

We will describe the individual steps of such a calculation, including the choice of classes $\{\mathcal{C}_i\}$ in detail in the next section. However even from the short description the following question arises:

> Can one guarantee that $B = A$ or at least limit the index of A in B? (If so, one could bound the backtrack search for A in B which is potentially the worst-behaving part of the algorithm.)

We will study this question in Section 4.

3. Details of the algorithm

3.1. Normalizer classes. To obtain B we want to partition the elements of U into classes according to the action of $N_G(U)$. For this we observe the following:

- Conjugacy by $N_G(U)$ will join U-conjugacy classes of the same cardinality as $U \lhd N_G(U)$.
- Conjugacy by $N_G(U)$ preserves the cycle structure of elements.

In the first approximation we therefore consider the following classes $\{\mathcal{B}_i\}$ as the union of U-conjugacy classes: two elements $u_1, u_2 \in U$ are in the same $\mathcal{B}$-class if $\left|u_1^U\right| = \left|u_2^U\right|$ and u_1 has the same cycle structure as u_2. (If G is not transitive on Ω one can even separate by cycle structures when restricted to G-orbits of a particular length.)

To refine this partition, we observe that the class sums for the $N_G(U)$-classes in U form a subalgebra of the center of the group algebra for $N_G(U)$. We therefore calculate "structure constants" for the multiplication of the classes $\{\mathcal{B}_i\}$:

For $x \in \mathcal{B}_i$, let $z_{j,k}$ be the number of elements $y \in \mathcal{B}_j$ such that $xy \in \mathcal{B}_k$. We define the *signature* of x as the collection of values $z_{j,k}$ for all j, k.

Since the action of $N_G(U)$ induces automorphisms of U, elements with different signatures cannot be in the same class. This offers the possibility for a refinement by splitting up the class $\mathcal{B}_i$ according to element signatures; the same argument can be used to split up the corresponding class $\mathcal{B}_j$ using the same signatures. As $(xy)^{-1} = y^{-1}x^{-1}$ and elements and their inverses stay in the same class, it is sufficient to consider only ordered pairs i, j. We thus obtain (often finer) classes $\{\mathcal{C}_i\}_j$.

In principle this process can be iterated. In practice, however it turns out that further iterations do not typically yield improvements (that is, the classes turn out to be maximally refined after one iteration).

The calculations required in this step involve the multiplication of group elements and the identification of the class in which the result lies. Instead of multiplying permutations, it is faster to work with base images and to do class identifications based on hash values of base images.

3.2. Partition stabilizer. The next step is to calculate the automorphism group of U, for example following [**Sho28**] for abelian groups, [**ELGO02**] for p-groups, and [**CH03**] for non-solvable groups. Since we assume that U is small this calculation will be fast.

We then construct the permutation action of $\mathrm{Aut}(U)$ on the set $U^{\#}$ of non-identity elements of U as a subgroup of $S_{|U|-1}$. The classes $\mathcal{C}_i$ of elements obtained in the previous step thus correspond to sets of numbers. We arrange these sets in increasing order (typically the calculation of a set stabilizer is faster if the corresponding set is smaller) and iteratively compute the stabilizers of these sets using a partition backtrack algorithm [**Leo97**]. The last stabilizer then stabilizes all sets and therefore the (ordered) partition of U into classes; it is therefore equal to the group B.

In the special case that U is elementary abelian two improvements are possible: first, if any of the classes of U spans a subspace, the automorphism group can be reduced from $\mathrm{GL}_n(p)$ to a subspace stabilizer consisting of block matrices. Furthermore if the characteristic is different from 2, one can first consider the stabilizer under the projective action.

3.3. Normalizing elements. We now assume that we have obtained a subgroup $B \leq \mathrm{Aut}(U)$ which contains the group A of all automorphisms that are induced by $N_G(U)$, but might be larger. Assume that $U = \langle u_1, \ldots, u_n \rangle$. We calculate iteratively $C_G(u_1)$, $C_G(\langle u_1, u_2 \rangle) = C_{C_G(u_1)}(u_2)$,... etc. to obtain $C_G(U)$. The following lemma describes an element test for A which simultaneously produces elements $g_\alpha \in G$ inducing a particular automorphism α.

LEMMA 1. *For $\alpha \in \mathrm{Aut}(U)$ we have that α is induced by $N_G(U)$ if and only if there exists*

1. *$g_1 \in G$ such that $u_1^\alpha = u_1^{g_1}$*
2. *$g_2 \in C_G(u_1^{g_1})$ such that $u_2^\alpha = (u_2^{g_1})^{g_2}$.*

$\vdots$

m. *$g_m \in C_G(u_1^{g_1}, u_2^{g_1 g_2}, \ldots, u_{m-1}^{g_1 g_2 \cdots g_{m-1}})$ such that $u_m^\alpha = (u_m^{g_1 g_2 \cdots g_{m-1}})^{g_m}$.*

In this case the element $x = g_1 g_2 \cdots g_m$ is an element that induces α. Any other element inducing the same automorphism will differ from x only by an element of $C_G(U)$.

PROOF. Assume that x is as given. Then $u_i^x = u_i^{g_1 \cdots g_i}$. As $u_i^\alpha \in U$, the element x maps every generator of U into U and therefore normalizes the finite group U. Furthermore for any $u = u_{i_1} \cdots u_{i_k} \in U$, we have that

$$u^x = (u_{i_1} \cdots u_{i_k})^x = u_{i_1}^x \cdots u_{i_k}^x = u_{i_1}^\alpha \cdots u_{i_k}^\alpha = (u_{i_1} \cdots u_{i_k})^\alpha = u^\alpha.$$

Conversely, if α is induced by $y \in N_G(U)$ we can set $g_1 = y$, $g_i = 1$ for $i > 1$. Also in this case we have that $x \cdot y^{-1} \in C_G(U)$ as it fixes all generators of U. □

For performance reasons it can be advantageous to arrange the generators u_i in order of decreasing support (number of points moved). While this does not usually

result in a notable increase in the cost of a conjugacy test, it tends to decrease the size of the first centralizer, thus making the subsequent conjugacy tests easier. This is particularly relevant when computing normalizers in the full symmetric group.

4. Are all automorphisms induced?

An obvious question that arises at this point is whether indeed all automorphisms that stabilize the partition of U into classes are induced by the normalizer of U in G or, if not, what we can say about the index $[B : A]$.

One way to study this is via representation theory:

LEMMA 2. *Let G be a permutation group acting on Ω with the natural permutation matrix representation $\nu\colon G \to GL_{|\Omega|}(\mathbb{C})$ and $\alpha \in \mathrm{Aut}(G)$. The following are equivalent:*

a) *α preserves the cycle shape of every element.*
b) *α preserves the number of fixed points of every element*
c) *The (complex) representations ν and $\alpha\nu$ afford the same permutation character.*
d) *The representations ν and $\alpha\nu$ are equivalent (as complex representations).*

In this case we say that α is equidistributing.

PROOF. As complex representations are equivalent if and only if they afford the same character we only need to show that b) implies a). This follows, because for a prime p dividing an integer n every n-cycle of an element $g \in G$ becomes a set of p disjoint $\frac{n}{p}$-cycles in g^p. This gives rise to a recursive formula for the number of n-cycles in terms of the number of fixed points of powers of g. □

This lemma incidentally shows that instead of cycle structure it would have been sufficient to group elements according to their number of fixed points. However, this would have given initially larger classes $\mathcal{B}_i$ and led to longer runtime since the backtrack algorithm for set stabilizers runs faster if the set to be stabilized is smaller.

On the other hand, α is induced by $N_{S_n}(U)$ if the representations ν and $\alpha\nu$ are *permutation equivalent*, i.e., if they can be transformed into each other by a permutation of the basis vectors, or equivalently if they can be conjugated into each other by permutation matrices.

As the following examples show, these two concepts are not equivalent.

EXAMPLE 1. Let $U = \mathrm{PSL}_3(2)$, acting 2-transitively on 7 points. Then the rational classes of U (and thus also the cycle structures) are determined solely by the orders of elements. Thus the outer automorphism of order 2 is equidistributing. This automorphism can be considered as a duality between points and lines in the underlying projective geometry, so it cannot be induced by S_7.

EXAMPLE 2. For our second example let $G = S_8$ and let

$$U = \langle u_1 = (2,3)(6,8), u_2 = (2,6)(3,8)(5,7), u_3 = (1,4)(5,7)\rangle \leq G.$$

Then $U \cong C_2^3$, $|C_{S_8}(U)| = 16$, $|N_{S_8}(U)| = 64$. There exists an automorphism α of U that maps the generators of U as $u_1 \mapsto u_3 = (1,4)(5,7)$, $u_2 \mapsto u_3u_2u_1 = (2,6)(3,8)(5,7)$, and $u_3 \mapsto u_1 = (2,3)(6,8)$. This automorphism is equidistributing, but is not induced by $N_{S_8}(U)$.

As the first example shows, such behavior is often associated with the existence of interesting combinatorial structures. However – typically there are few interesting structures associated to a permutation group – one can expect this behavior to be infrequent.

For transitive groups the existence of an equidistributing automorphism α which is not induced by a permutation means that $\mathrm{Stab}_G(1)^\alpha$ is not a point stabilizer, i.e. G must have (at least) two classes of subgroups isomorphic to $\mathrm{Stab}_G(1)$. Thus the index $[B : A]$ counts classes of subgroups isomorphic to, but not conjugate to $\mathrm{Stab}_G(1)$. In general there are just a few classes of such subgroups, indicating that $[B : A]$ ought to be small.

We can prove equality $A = B$ in some cases, detailed below. Some of these cases are of practical relevance, as the groups are small in comparison to the degree and have comparatively little permutational structure to aid a backtrack search for the normalizer.

LEMMA 3. *Suppose that $U \leq S_n$ acts transitively with a cyclic point stabilizer (for example, if U is regular), and let $\alpha \in \mathrm{Aut}(U)$ be equidistributing. Then α is induced by $N_{S_n}(U)$.*

PROOF. Let $S = \mathrm{Stab}_U(1) = \langle g \rangle$. Then $S^\alpha = \langle g^\alpha \rangle$. As α is equidistributing, g^α and thus S^α has a fixed point and thus $S^\alpha \leq \mathrm{Stab}_U(\omega)$ for some point ω. As the sizes coincide we must have equality. Thus α maps a point stabilizer to a point stabilizer. By [**DM96**, Theorem 4.2B], α is therefore induced by conjugation in the symmetric group. □

A related question that has been studied in the literature, for example in [**PSZ78**] and [**Cam05**], is whether two permutation groups which have the same number of elements for any cycle structure of elements must be (permutation) isomorphic. (This has applications to the recognition of Galois groups, see e.g. [**Hul99**]). In this context Woltermann and Sehgal [**WS79**] obtained a result about uniqueness of such groups; the proof can be translated easily to yield the following result:

LEMMA 4. *Let U be a solvable $\frac{3}{2}$-transitive permutation group. Then every automorphism $\alpha \in \mathrm{Aut}(U)$ is induced by $N_{S_n}(U)$.*

PROOF. By [**Wie64**, Theorem 10.4] a solvable $\frac{3}{2}$-transitive group is either primitive or a Frobenius group. In either case it has a characteristic, regular normal subgroup N with $S = \mathrm{Stab}_U(1)$ a complement to N and all complements of N are conjugate [**Hup67**, Satz II.3.2, Satz V.8.3, Satz I.18.3]. As N is characteristic $N = N^\alpha$. Therefore S^α is a complement to N and thus conjugate to S. Thus S^α is a point stabilizer and by [**DM96**, Theorem 4.2B] α is induced by $N_{S_n}(U)$. □

The third important special case we consider is that of intransitive elementary abelian groups that occur as base groups for imprimitive permutation groups. (This is essentially the relevant case for the construction of transitive groups in [**Hul05**].)

LEMMA 5. *Let p be a prime and $m = \lfloor \frac{n}{p} \rfloor$. Then S_n has a subgroup $V \cong Z_p^m$ whose m orbits are $\{1, \ldots, p\}$, $\{p+1, \ldots, 2p\}, \ldots, \{(m-1)p+1, \ldots, mp\}$. Suppose that $U \leq V$ and that $\alpha \in \mathrm{Aut}(U)$ is equidistributing. Then α is induced by $N_{S_n}(U)$.*

PROOF. The action of V on each orbit is the regular action of Z_p. Because of this the cycle structure of each element of V is determined by the *number of orbits* on which it acts nontrivially.

We consider V as an m-dimensional vector space over $\mathbb{F}_p$ with basis

$$\{(1, 2, \ldots, p), (p+1, \ldots, 2p), \ldots\}.$$

The weight (defined as in coding theory to be the number of nonzero entries in a vector) of each element of V corresponds to its cycle structure, considered as a permutation: the weight equals the number of p-cycles.

Now suppose that $U \leq V$. Then B consists of the automorphisms that preserve the weight of each vector, i.e., it consists of weight preserving linear transformations.

By the theorem of MacWilliams [**Mac62**], [**HP03**, Theorem 7.9.4] this implies that every automorphism in B is a monomial transformation, that is an element of $\mathbb{F}_p^* \wr S_m$. Such elements however can be represented as elements of $S_p \wr S_m$ and thus as elements of S_n. □

5. Generalizations

The results of the previous section concentrating on elementary abelian groups might seem to be very weak. If we consider a general permutation group U however the following argument applies:

If U has a trivial radical the structure of U is very restricted and $[\mathrm{Aut}(U) : U]$ is small. We can therefore simply set $B := \mathrm{Aut}(U)$ and just run the backtrack search for $A \leq \mathrm{Aut}(U)$. (One can obtain the structure of $\mathrm{Aut}(U)$ easily by embedding into a direct product of wreath products [**CH03**].)

Otherwise, U has a nontrivial radical and therefore contains a characteristic elementary abelian subgroup $V \leq U$ and thus $N_G(U) \leq N_G(V)$. Setting $M := N_G(V)$ we have that $N_G(U) = N_M(U)$. As $V \lhd M$ we can compute this second normalizer in the factor group as $N_{M/V}(U/V)$, which provides a further reduction.

While we have considered permutation groups so far, the same same strategy can also be applied to the case of normalizing subgroups of $\mathrm{GL}_n(q)$. The best permutation representation for this group has degree at least $(q^n - 1)/(q - 1)$ which very quickly makes it infeasible to use such a permutation representation for a backtrack-type calculation. In effect therefore there exists no practical algorithm for the computation of subgroup normalizers.

On the other hand, the computation of normal forms of matrices yields an effective conjugacy test (as well as a determination of conjugating permutations). The calculation of module automorphisms [**Smi94**] provides an algorithm to compute element centralizers in $\mathrm{GL}_n(q)$.

Furthermore, even a single element centralizer will be comparatively small. This makes it feasible to then use a stabilizer-chain based approach [**But82**] for the calculation of conjugating elements within the centralizer and to determine iterated centralizers.

Therefore $\mathrm{GL}_n(q)$ fulfills all prerequisites for the algorithm proposed in this paper. Instead of cycle structure of elements, one can use normal forms for matrices. With this modification, the proposed algorithm makes it possible to compute the normalizer of somewhat small matrix groups in $\mathrm{GL}_n(q)$. Again it might be preferable to start by normalizing small characteristic subgroups for which the automorphism group can be determined easily.

6. Examples

As stated above, the algorithm proposed does not offer better complexity than the ordinary backtrack for a normalizer calculation. On the other hand, for the case of elementary abelian subgroups sometimes a dramatically better practical performance has been observed. This section will present some evidence for such a claim.

In the following description the "old" algorithm is the partition backtrack algorithm for the normalizer as implemented in GAP 4.4 [**GAP04**], following [**The97**]. It incorporates an initial reduction from S_n to a direct product of wreath products, following [**Hul05**, section 11].

The "new" algorithm is the author's GAP implementation of the automorphism-based approach of this paper for the case of elementary abelian subgroups.

We shall consider randomly generated elementary abelian subgroups of S_{30}. They were generated by picking a random p-element of the group and then repeatedly selecting random p elements that centralize the elements chosen so far until a group of the desired order was generated. It was possible that the process stopped before the desired order was reached if there were no further p-elements in the centralizer. In this case the attempt was abandoned and the construction started anew from scratch.

Table 1 summarizes the results of these experiments. The column entries are:

$|U|$: Order of the subgroups to be normalized.
Runs: Number of experimental runs. (The attempt was made to run up to 100 examples but some runs were interrupted by hand after spending substantially more time, as long as the results obtained up to that point appeared consistent.)
AvgOld: The average runtime for the "old" (backtrack-based) algorithm (in milliseconds).
AvgNew: The average runtime for the "new" (automorphism-based) algorithm (in milliseconds).
AvgRatio: The average ratio "old" to "new".
MinRatio: The minimal ratio "old" to "new".
MaxRatio: The maximal ratio "old" to "new".

Runtimes were calculated by GAP 4.4 on a 2.4GHz Pentium 4 under Linux and are given in milliseconds.

The cases in which runs were aborted by hand typically affected situations in which the "old" algorithm was performing particularly badly. As the terminated runs did not complete they were not included in the table but would have increased substantially the "worst case" factors.

Again, the results show that the new algorithm not only is faster in most cases, it also performs much less badly in the cases in which the old algorithm is superior than vice versa. In particular for larger subgroups that are still away from the theoretical maximum size for subgroups of S_{30} (e.g. 2^7, 3^6) the performance of the old algorithm is spectacularly worse.

The only cases in which the old algorithm is consistently better are subgroups of order 5^5, 5^6, 7^4 and 11^2. The reason for this is that the structure of the p-Sylow subgroups of S_{30} ($(5 \wr 5) \times 5$, 7^4, 11^2) very much restricts the possibilities for random subgroups of the given order: they will be almost always direct products

$\|U\|$	Runs	AvgOld	AvgNew	AvgRatio	MinRatio	MaxRatio
2^2	100	261	152	2.37	0.23	8.12
2^3	100	419	142	3.98	0.51	77.0
2^4	100	737	342	5.87	0.14	90.9
2^5	100	17546	174	206	0.66	15707
2^6	100	48698	236	491	0.67	12656
2^7	29	956058	448	3245	0.50	87103
3^2	100	426	182	3.73	0.31	11.8
3^3	100	3489	579	21.6	0.12	381
3^4	23	316277	834	2235	1.67	41826
3^5	13	376365	381	1202	1.86	12528
3^6	4	935833	1758	378	0.91	1103
5^2	100	2120	106	26.8	0.13	318
5^3	31	94583	195	477	0.70	4642
5^4	21	85266	1018	114	1.52	477
5^5	70	2176	23059	0.41	0.028	2.34
5^6	100	312	10770	0.029	0.019	0.047
7^2	100	575	109	5.81	0.51	27.7
7^3	100	457	328	1.43	0.30	3.09
7^4	100	139	1305	0.10	0.081	0.18
11^2	100	106	197	0.61	0.24	1.61

TABLE 1. Runtime Comparisons

$\|U\|$	Runs	AvgOld	AvgNew	AvgRatio	MinRatio	MaxRatio
5^5	20	737772	15425	186	1	648
7^3	20	11729	428	22	0.7	64

TABLE 2. Runtime Comparisons in S_{35}

of disjoint cycles. The stabilizer of this orbit partition, obtained as the first step of the normalizer algorithm is then very close to the normalizer, which leaves very little work for the backtrack search.

As soon as G is increased this second effect vanishes, and the "new" algorithm again performs better, as the times for normalizers in S_{35}, given in table 2 shows.

References

[But82] Gregory Butler, *Computing in permutation and matrix groups II: backtrack algorithms*, Math. Comp. **39** (1982), no. 160, 671–670.

[Cam05] Peter J. Cameron, *Partitions and permutations*, Discrete Math. **291** (2005), 45–54.

[CH03] John Cannon and Derek Holt, *Automorphism group computation and isomorphism testing in finite groups*, J. Symbolic Comput. **35** (2003), no. 3, 241–267.

[DM96] John D. Dixon and Brian Mortimer, *Permutation groups*, Graduate Texts in Mathematics, vol. 163, Springer, 1996.

[ELGO02] Bettina Eick, C.R. Leedham-Green, and E.A. O'Brien, *Constructing automorphism groups of p-groups*, Comm. Algebra **30** (2002), no. 5, 2271–2295.

[GAP04] The GAP Group, `http://www.gap-system.org`, GAP – *Groups, Algorithms, and Programming, Version 4.4*, 2004.

[HEO05] Derek F. Holt, Bettina Eick, and Eamonn A. O'Brien, *Handbook of Computational Group Theory*, Discrete Mathematics and its Applications, Chapman & Hall/CRC, Boca Raton, FL, 2005.

[Hol91] D. F. Holt, *The computation of normalizers in permutation groups*, J. Symbolic Comput. **12** (1991), no. 4-5, 499–516, Computational group theory, Part 2.

[HP03] W. Cary Huffman and Vera Pless, *Fundamentals of error-correcting codes*, Cambridge University Press, 2003.

[Hul99] Alexander Hulpke, *Techniques for the computation of Galois groups*, Algorithmic Algebra and Number Theory (B. H. Matzat, G.-M. Greuel, and G. Hiss, eds.), Springer, 1999, pp. 65–77.

[Hul05] ———, *Constructing transitive permutation groups*, J. Symbolic Comput. **39** (2005), no. 1, 1–30.

[Hup67] Bertram Huppert, *Endliche Gruppen I*, Grundlehren der mathematischen Wissenschaften, vol. 134, Springer, 1967.

[Leo97] Jeffrey S. Leon, *Partitions, refinements, and permutation group computation*, Proceedings of the 2nd DIMACS Workshop held at Rutgers University, New Brunswick, NJ, June 7–10, 1995 (Larry Finkelstein and William M. Kantor, eds.), DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, vol. 28, American Mathematical Society, Providence, RI, 1997, pp. 123–158.

[LM02] Eugene M. Luks and Takunari Miyazaki, *Polynomial-time normalizers for permutation groups with restricted composition factors*, Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation (Teo Mora, ed.), The Association for Computing Machinery, ACM Press, 2002, pp. 176–183.

[Luk93] Eugene M. Luks, *Permutation groups and polynomial-time computation*, Groups and Computation (Providence, RI) (Larry Finkelstein and William M. Kantor, eds.), DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, vol. 11, American Mathematical Society, 1993, pp. 139–175.

[Mac62] F. J. MacWilliams, *Combinatorial problems of elementary abelian groups*, Ph.d. thesis, Harvard University, 1962.

[Miy06] Izumi Miyamoto, *An improvement of* GAP *normalizer function for permutation groups*, Proceedings of the 31st International Symposium on Symbolic and Algebraic Computation held in Genova, July 9–12, 2006 (Jean-Guillaume Dumas, ed.), ACM Press, New York, 2006.

[PSZ78] Ann Scrandis Playtis, Surinder Sehgal, and Hans Zassenhaus, *Equidistributed permutation groups*, Comm. Algebra **6** (1978), no. 1, 35–57.

[RD04] Colva M. Roney-Dougal, *Conjugacy of subgroups of the general linear group*, Experimental Mathematics **13** (2004), no. 2, 151–163.

[Ser03] Ákos Seress, *Permutation group algorithms*, Cambridge University Press, 2003.

[Sho28] Kenjiro Shoda, *Über die Automorphismen einer endlichen Abelschen Gruppe*, Math. Ann. **100** (1928), 674–686.

[Sim70] Charles C. Sims, *Computational methods in the study of permutation groups*, Computational Problems in Abstract Algebra (John Leech, ed.), Pergamon press, 1970, pp. 169–183.

[Smi94] Michael J. Smith, *Computing automorphisms of finite soluble groups*, Ph.D. thesis, Australian National University, Canberra, 1994.

[The97] Heiko Theißen, *Eine Methode zur Normalisatorberechnung in Permutationsgruppen mit Anwendungen in der Konstruktion primitiver Gruppen*, Dissertation, Rheinisch-Westfälische Technische Hochschule, Aachen, Germany, 1997.

[Wie64] Helmut Wielandt, *Finite permutation groups*, Academic Press, 1964.

[WS79] Michael Woltermann and Surinder Sehgal, *Equidistributed $\frac{3}{2}$-transitive solvable permutation groups. I, II*, Comm. Algebra **7** (1979), no. 15, 1599–1643, 1645–1672.

Department of Mathematics, Colorado State University, Fort Collins, CO 80523
E-mail address: hulpke@math.colostate.edu