# An Improvement of GAP Normalizer Function for Permutation Groups*

Izumi Miyamoto
University of Yamanashi
4-3-11 Takeda Kofu 400-8511 Japan
imiyamoto@yamanashi.ac.jp

## ABSTRACT

In GAP system it takes unreasonably long time to compute the normalizers of some permutation groups, even though they are of small degree. The author gave an algorithm in [7, 8] to compute the normalizers of permutation groups and particularly it worked smoothly for transitive groups of degree up to 22. In 1999 GAP version 4 was released. Since then the GAP system has been improved and in 2004 GAP4r4 had a special function to compute the normalizers in the symmetric groups but it still has difficulties in computing the normalizers of some permutation groups. It has been also found that the author's algorithm in [7, 8] has difficulties in some groups of small degree but larger than 22. So the author will give two new programs improving the computation of normalizers of transitive permutation groups in the symmetric groups. One of them works comparatively smoothly for the transitive groups of degree up to 30.

## Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Manipulation**]: Algebraic Algorithms; G.2.1 [**Discrete Mathematics**]: Combinatorics; G.2.2 [**Discrete Mathematics**]: Graph Theory

## General Terms

Experimentation, Performance, Algorithms

## 1. INTRODUCTION

Among the computations of groups the algorithms for permutation groups have been well studied. In practice it may be rarely difficult to compute the normalizers of permutation groups but by our experiment using the GAP function Normalizer, it was found that among the 36620 transitive permutation groups of degree from 20 to 30 each of the normalizers of 755 groups in the symmetric groups cannot be computed within 10 hours. In [6] a polynomial-time algorithm for computing normalizers of permutation groups is shown under the condition that the groups in which the normalizers are computed have restricted composition factors. This algorithm is very complicated and it has not been implemented. In 2000, when the author presented the algorithm in [7, 8], 22 was the largest degree of transitive permutation groups in the data of the GAP library, while 30 is the largest now. The author's program in [7, 8] written in the GAP program language can compute the normalizers of the transitive groups of degree up to 22 in the symmetric groups smoothly but, in 14 cases of degree up to 30, cannot compute the normalizer within 10 hours. We will give two new programs written in the GAP language, one of which can compute the normalizer of any transitive permutation groups of degree up to 30 in the symmetric group within 30 seconds. The other one is faster to compute all these normalizers. In our experiments we also had to compute the normalizers of some subgroups of transitive groups and we found various subgroups of which normalizers are rather harder to compute than the given transitive groups. Such hard groups have not been well specified yet. So we do not check how our new programs work in groups of larger degree except the examples written in [7, 8] but we mainly restricted our interest to the transitive groups of degree up to 30 in the present paper.

Let $G$ and $K$ be permutation groups on a set $\Omega$ of $n$ points. The normalizer of $G$ in $K$ is defined by $\mathsf{Norm}(K, G) = \{k \in K | k^{-1}Gk = G\}$. Let $Sym(n)$ denote the symmetric group of degree $n$. In [7, 8] the author used GAP version 3. Now it is version 4r4 and has a special function DoNormalizerSA to compute the normalizers of some imprimitive or intransitive groups in the symmetric groups. GAP also has a special function SubgpConjSymmgp computing a conjugating element between two subgroups in a symmetric group. Here we focused on the normalizers of transitive groups in the symmetric groups. Normalizers of intransitive groups are computed by a straightforward method considering the action on each orbit in [7, 8]. Such normalizers are treated in our programs by a similar method used in GAP function NormalizerParentSA and normalizers in non symmetric groups are computed by simply taking the intersections of those in symmetric groups with the non symmetric groups in order to apply our algorithm recursively.

Suppose that $G$ is imprimitive and has only one block of length $m$ containing some fixed point. Then $\mathsf{Norm}(Sym(n),$

*(Produces the permission block, copyright information and page numbering). For use with ACM_PROC_ARTICLE-SP.CLS V2.6SP. Supported by ACM.


*ISSAC'06*, July 9–12, 2006, Genova, Italy .

$G) \subseteq Sym(m) \wr Sym(n/m)$, where $Sym(m) \wr Sym(n/m)$ is an appropriate ***wreath product*** of $Sym(m)$ by $Sym(n/m)$. In GAP4r4 DoNormalizerSA invokes NormalizerParentSA to compute the wreath product $Sym(m) \wr Sym(n/m)$ and then computes the normalizer in this smaller group. In [7] it is proved that the normalizer is contained in the automorphism group of the association scheme formed by $G$ if $G$ is transitive. Here we give the definition of an association scheme.

*Definition. 1.* ([1](2.1)) Let $\Omega$ be a set of $n$ points and let $R_i (i = 0, 1, \cdots d)$ be subsets of $\Omega \times \Omega$. $(\Omega, R_i)$ is an association scheme ( or a homogeneous coherent configuration) if it satisfies that

- $R_0 = \{(x, x) | x \in \Omega\}$,
- $\Omega \times \Omega = R_0 \cup R_1 \cup \cdots R_d$ and $R_i \cap R_j = \emptyset$ if $i \neq j$,
- for all $R_i$ there exists $i^*$ in $\{0, 1, \cdots, d\}$ such that $\{(x, y) | (y, x) \in R_i\} = R_{i^*}$ and
- for all $R_i, R_j, R_k$ the number $p_{i,j,k} = \#\{z | (x, z) \in R_i, (z, y) \in R_j\}$ is constant whenever $(x, y) \in R_k$.

Readers may refer to [1] for details of association schemes and to [4, 5] for some computing results. However in this paper an association scheme is always formed by a transitive group $G$ and $\{R_0, R_1 \cdots, R_d\}$ is the set of the orbits of $G$ on $\Omega \times \Omega$, which we call 2-orbits. Then each of its automorphisms is a permutation on $\Omega$ preserving the 2-orbits as a whole, which means that it may move one 2-orbit to another. Both the automorphism group of the scheme and the normalizer of $G$ are computed by backtrack methods. So the algorithm in [7, 8] needs backtrack methods twice to compute normalizers. The wreath product $Sym(m) \wr Sym(n/m)$ is given as the automorphism group of a typical association scheme. So the GAP special function can be seen using only such typical association schemes to avoid a backtrack computation. Following this idea the author considered an algorithm using a lemma in [7] and not using association schemes. This algorithm will be called Algorithm NormA. The program of this algorithm will be also denoted by NormA. The aim of this algorithm is to attach a small program to the GAP function Normalizer to improve it to some extent because our program in [7, 8] computes normalizers faster than GAP in general or on average but much slower in some cases. As a result NormA computes faster than the program in [7, 8] in general but sometimes slower than GAP for the transitive groups of degree up to 30.

In the GAP function SubgpConjSymmgp, computing an element conjugating subgroups $H$ and $K$ in the symmetric group, it is considered that if $H$ is imprimitive and has only one block $B$ containing some fixed point, a conjugating element should move the block system $B^H$ to the corresponding block system of $K$. So in this function the action of $H$ on $B^H$ and the action of the setwise stabilizer $H_B$ on $B$ are computed to restrict the choice of the conjugating element. In another algorithm, which we will call Algorithm NormB, we use a block of the automorphism group of the association scheme formed by $G$ similarly in Proposition 1. The computing time varies in each experiment. NormB can compute the normalizer of any transitive group in the symmetric group of degree up to 30 within 30 second on average. The maximum computing time was about 1 minute in our experiments.

For our experiments we used computers under Linux with CPU Xeon 2.8GHz and 1GB memory. We used *ParGAP* [2] to speed up our experiments.

## 2. ALGORITHM

Let $G$ be a transitive permutation group on a set of $\Omega$ of $n$ points. We will compute the normalizer $N =$Norm$(Sym(n), G)$. As is noted in [7], it is easily seen that $N$ preserves the 2-orbits of $G$ on $\Omega \times \Omega$. So the normalizer $N$ of a transitive group $G$ is contained in the automorphism group $A$ of the association scheme formed by $G$. Hence any block $B$ of $A$ is also a block of both of $N$ and $G$. So if $A$ is imprimitive, we compute the action $\bar{G}$ of $G$ on the set of blocks $B^G$ and the action $G_B|B$ of $G_B$ on $B$, where $G_B$ is the the setwise stabilizer of $B$ in $G$. We define $\bar{A}$ and $A_B|B$ similarly.

PROPOSITION 1. *Let $\bar{N}' =$Norm$(\bar{A}, \bar{G})$ and let $N'_B =$ Norm $(A_B|B, G_B|B)$. Then $N$ is contained $(N'_B \wr \bar{N}') \cap A$, where the points of each block of $B^G$ are arranged so that $B = [b_1, b_2, \cdots, b_m]$ and $B^g = [b_1^g, b_2^g, \cdots, b_m^g]$ for some $g \in G$.*

PROOF. Clearly $\bar{N} \subseteq \bar{N}'$ and $N_B|B \subseteq N'_B$. Let $x \in N$ and suppose $B^{gx} = B^h$. Then there exists $n' \in N'_B \wr \bar{N}'$ such that $\bar{x}\bar{n}' = \bar{1}$ and that $b_i^{hn'} = b_i^g$ for $1 \leq i \leq m$ by the definition of wreath product. Suppose that $b_i^{gx} = b_j^h$. Then $(b_i^g)^{xn'} = b_j^g$. Since $b_i^{gxh^{-1}} = b_j$, there exists $k_g \in N'_B \wr \bar{N}'$ such that $k_g = xn'$ on $B^g$ leaving all points not in $B^g$ fixed. Then $xn'$ is the product of all such $k_g$. So we have $x \in N'_B \wr \bar{N}'$. $\square$

We use this algorithm recursively. We also use the following lemma which is the first step of the lemma in [7]. The lemma in [7] is a little complicated. So we will give an easy proof here. Lemma 3 is an elementary well-known lemma which is required for the next step of our algorithm. So we will also give it here.

LEMMA 2. *Suppose $G \subseteq K$. Let $O$ be a common orbit of $G$ and $K$, let $p \in O$ and let $K_p$ be the stabilizer of $p$ in $K$. Then Norm$(K, G)$ is generated by Norm$(K_p, G)$ and $G$, which implies that Norm$(K, G) = G$Norm$(K_p, G)$.*

PROOF. Let $x \in$Norm$(K, G)$. Then there exists $g \in G$ such that $p^x = p^g$. So $xg^{-1} \in$Norm$(K, G)_p =$Norm$(K_p, G)$. Therefore Norm$(K, G)$ is generated by Norm$(K_p, G)$ and $G$. In fact Norm$(K_p, G)$ normalizes $G$, so the last assertion Norm$(K, G) = G$Norm$(K_p, G)$ follows. $\square$

LEMMA 3. *Let $C$, $D$, $E$ and $F$ be groups. Suppose that $C = DE$ and $D \subseteq F$. Then $C \cap F = D(E \cap F)$.*

Here, if $K_p$ and $G_p$ also have a common orbit $O'$, Norm$(K_p, G_p) = G_p$ Norm$(K_{p,p'}, G_p)$, where $p' \in O'$ and $K_{p,p'}$ is the pointwise stabilizer of $p$ and $p'$. Then we can proceed to the second step as stated below. Since Norm$(K_p, G)$ normalizes $G_p$, Norm$(K_p, G) \subseteq$ Norm$(K_p, G_p)$. So

$$\begin{aligned}\mathrm{Norm}(K_p, G) &= G_p(\mathrm{Norm}(K_{p,p'}, G_p) \cap \mathrm{Norm}(K_p, G)) \\ &= G_p\mathrm{Norm}(K_{p,p'}, G),\end{aligned}$$

since $G_p \subseteq$Norm$(K_p, G)$. Hence

$$\begin{aligned}\mathrm{Norm}(K, G) &= G\mathrm{Norm}(K_p, G) \\ &= GG_p\mathrm{Norm}(K_{p,p'}, G) \\ &= G\mathrm{Norm}(K_{p,p'}, G)\end{aligned}$$

We may go forward to the next step, if $K_{p,p'}$ and $G_{p,p'}$ have a common orbit $O''$. In the second step, for instance, $\mathsf{Norm}(K_{p.p'}, G) \subseteq \mathsf{Norm}(\ K_{p,p'}, G_p) \subseteq \mathsf{Norm}(K_{p,p'}, G_{p,p'})$. So we may compute the first normalizer as the normalizer of $G$ in the second or third normalizer once they are computed.

In NormA we heuristically compute $\mathsf{Norm}(A_{p,\cdots,p''}, G_{p,\cdots,p''})$ in the final step in usual cases and we compute the normalizer of $G$ in this normalizer, because it is faster in most cases. We use various heuristics in NormA which we will not explain in detail. In NormB we use NormA with less heuristics, which will be denoted by NormA' and we will explain here the heuristics used in NormA'. If $G_{p,\cdots,p''}$ is an identity group, we seek $G_{p,\cdots,p'''}$ so that its moved points contains those of $A_{p,\cdots,p''}$ and then compute $\mathsf{Norm}(A_{p,\cdots,p''}, G_{p,\cdots,p'''})$. If $G$ is intransitive, NormalizerParentSA computes, for instance if $G$ has $l$ orbits of length $m$, the wreath product $Sym(m) \wr Sym(l)$ and the direct product of such wreath products. Furthermore if $m \leq 30$, by TransitiveIdentification actions of $G$ on these orbits are identified to some classified transitive groups and the normalizer of the actions are also computed. Then using this data, more restricted wreath products are constructed. In NormA' we apply NormalizerParentSA to $G_{p,\cdots,p''}$ to obtain the direct product of these wreath products and compute the normalizers above in the intersection of $A_{p,\cdots,p''}$ and this direct product. If any orbit of $G_{p,\cdots,p''}$ is of length at most 2, then we use $G_{p,\cdots,p''''}$ instead of $G_{p,\cdots,p''}$ in the above procedure, where $p''''$ is the previous point to $p''$. Here is a rough GAP-like code of NormB.

```
NormB:=function ( K, G )

  A := auto_group( association_scheme( G ) );
  b := AllBlocks( G );
      # all blocks containing the point 1
  if b = [  ]  then
    N := NormA'( K, G );
    return N;
  else
    B := b[k];
      # choose the k-th block which is maximal
    R := List( B^G , function ( B' )
          return
            RepresentativeAction( G, B[1], B'[1] );
          end );
    B^G := List( R, function ( g )
             return List( B, function ( p )
                        return p ^ g;
                    end );
             end );
      # rearrange the points of every B' in B^G by g

    a1 := Action( A, B^G, OnSets );
    g1 := Action( G, B^G, OnSets );
    n1 := NormB( a1, g1 );

    a2 := Action( Stabilizer( A, B, OnSets ), B );
    g2 := Action( Stabilizer( G, B, OnSets ), B );
    n2 := NormB( a2, g2 );

    W := WreathProduct( n2, n1 );
    perm := MappingPermListList( [ 1 .. n ],
                            Concatenation( B^G ) );
    W := W ^ perm;
      # make an appropriate wreath product
    N := NormA'( Intersection( A, W ), G );
    return Intersection( N, K );
  fi;
end;
```

**Table 1: Computing times of the Normalizers of Transitive Groups in $Sym(n)$, $20 \leq n \leq 30$**

| time range | DoNorm | AS | NormA | NormB |
|---|---|---|---|---|
| $* \leq$0.1sec | 10510 | 1829 | 125 | 5 |
| 0.1sec$< * \leq$0.2sec | 11728 | 7231 | 1220 | 33 |
| 0.2sec$< * \leq$0.5sec | 5433 | 22898 | 24260 | 1266 |
| 0.5sec$< * \leq$1sec | 2200 | 2973 | 9947 | 9831 |
| 1sec$< * \leq$2sec | 1098 | 629 | 646 | 22278 |
| 2sec$< * \leq$5sec | 1015 | 363 | 236 | 2442 |
| 5sec$< * \leq$10sec | 621 | 182 | 68 | 122 |
| 10sec$< * \leq$30sec | 834 | 232 | 39 | 643 |
| 30sec$< * \leq$1min | 381 | 126 | 14 | 0 |
| 1min$< * \leq$2min | 480 | 40 | 29 | 0 |
| 2min$< * \leq$5min | 486 | 30 | 25 | 0 |
| 5min$< * \leq$10min | 357 | 6 | 5 | 0 |
| 10min$< * \leq$30min | 348 | 9 | 4 | 0 |
| 30min$< * \leq$1h | 114 | 12 | 2 | 0 |
| 1h$< * \leq$2h | 63 | 15 | 0 | 0 |
| 2h$< * \leq$5h | 112 | 24 | 0 | 0 |
| 5h$< * \leq$10h | 85 | 7 | 0 | 0 |
| 10h$< *$ | 755 | 14 | 0 | 0 |

## 3. EXPERIMENTS

In GAP library [3] there is a list of transitive permutation groups TransitiveGroup$(n, k)$ up to degree $n = 30$. There exist 36620 transitive groups of degree $n$, $20 \leq n \leq 30$. We computed the normalizers of these groups $G$ in the symmetric groups $Sym(n)$ using three programs. The first one is the GAP special function DoNormalizerSA, the second is NormA which uses neither association schemes nor Proposition 1 but is heuristically finely tuned up, and the last is NormB explained in the previous section. DoNormalizerSA is abbreviated to DoNorm. In Table 1 we show the timings of these programs. We also show in Table 1 the timings of the program given in [7, 8] for reference, which is denoted by AS in the third column. The first column of Table 1 shows the time ranges and the remaining columns show the numbers of groups of which normalizer in the symmetric groups in each time range. NormA is the fastest to compute all the normalizers of the transitive groups of degree between 20 and 30. We note that DoNorm cannot compute each of the normalizers of 755 transitive groups within 10 hours. So we stopped computing in 10 hours. It takes 57 days for DoNorm to compute the other 35865 normalizers and it should take more than 1 year for DoNorm to compute all the 36620 normalizers. It takes 32 days, 10.5 hours and 17.4 hours for programs AS, NormA and NormB to compute all the 36620 normalizers respectively. In Table 2 we show the total computing time of each degree. In Table 3 the timings of the examples explained below are shown. The first and the second columns denote $n$ and $k$ of TransitiveGroup$(n, k)$ in the GAP library. In Table 4 how the computing time varies is shown for some groups by NormA. It took 70 seconds in trial

**Table 2: Computing times of the Normalizers of Transitive Groups of each degree $n$ in $Sym(n)$, $20 \leq n \leq 30$**

| $n$ | num | DoNorm | NormA | NormB |
|---|---|---|---|---|
| 20 | 1117 | 744min | 4.8min | 14min |
| 21 | 164 | 1951min | 0.6min | 1.3min |
| 22 | 59 | 60min (10) | 0.2min | 0.5min |
| 23 | 7 | 86sec | 0.6sec | 4.6sec |
| 24 | 25000 | 39h (26) | 3.2h | 9.5h |
| 25 | 211 | 3255min (6) | 1.2min | 2.2min |
| 26 | 96 | 10h (24) | 0.09h | 0.23h |
| 27 | 2392 | 200h (202) | 1.9h | 0.7h |
| 28 | 1854 | 263h (256) | 0.9h | 1.1h |
| 29 | 8 | 0.4sec | 1.4sec | 10.1sec |
| 30 | 5712 | 32day (231) | 0.18day | 0.23day |
| tot. | 36620 | 57day (755) | 0.44day | 0.71day |

Remark. (num) in the third column shows the number of groups not computed within 10 hours.

**Table 3: Some computing times of typical examples ( in seconds )**

| $n$ | $k$ | DoNorm | NormA | NormB |
|---|---|---|---|---|
| 28 | 1375 | $> 36000$ | 0.3 | 29 |
| 30 | 834 | 15 | 3087 | 1.2 |
| 30 | 841 | 16 | 3149 | 1 |
| 28 | 321 | 858 | 1746 | 4 |
| 27 | 1518 | $> 36000$ | 0.3 | 1 |

2 to compute the normalizer of TransitiveGroup$(30, 4912)$ by NormA, which was the longest in our experiments by NormA.

Example 1: Let $\Omega = \{1, 2, \cdots, n\}$. $G =$TransitiveGroup$(28, 1375)$. $|G| = 3111696$. Set $N =$Norm$(Sym(28), G)$. Then $|N| = 6223392$. $G$ has only one block of each of length 7 and 14 containing the point 1. DoNorm and NormA use the block of length 7. So $W = Sym(7) \wr Sym(4)$, but in NormB the block of length 14 is used. Then it is hard to compute Norm$(W, G)$ directly. Let $A$ be the automorphism group of the association scheme formed by $G$. Then $|A| = 5161930260480000$. $W_1$, $A_1$ and $G_1$ have a common orbit of length 6 containing the point 2. $G_{1,2}$ has orbits of length 7 and 14 and fixes the remaining points. In NormA we invoke NormalizerParentSA using $G_{1,2}$ to compute the normalizer of each orbit. Let $W'$ be the direct product of these groups. Then $|W_{1,2} \cap W'| = 8890560$ and Norm$(W_{1,2} \cap W', G_{1,2})$ is easily computed. From this normalizer we obtain $N$ in NormA. Let $B$ be the block of length 14. In NormB, Norm$(Sym(14), G_B|B)$ is computed. Here $G_B|B$ is transitive on $B$. So we compute the automorphism group of the association scheme formed by $G_B|B$ and it has a block of length 7. Thus this normalizer is computed by NormB recursively. Then we have $|A \cap N'| = 24893568$ and Norm$((A_{1,2} \cap N')_{1,2}, G_{1,2})$ easily. Now Norm$((A_{1,2} \cap N')_{1,2}, G)$ is computed as the normalizer of $G$ in Norm$((A_{1,2} \cap N')_{1,2}, G_{1,2})$ and consequently $N$ is generated this normalizer and $G$. We note that $|G_{1,2}| = 18522$.

Example 2: $G$ :=TransitiveGroup$(30, 834)$, $|G| = 14580$. Set $N =$Norm$(Sym(30), G)$. Then $|N| = 29160$. $G$ has only one block of each of length 3, 6 and 15 containing the point 1. In DoNorm the block of length 3 is used and so is in NormA. Set $W = Sym(3) \wr Sym(10)$. Then it take 15 seconds for DoNorm to compute $N =$Norm$(W, G)$. For NormA the stabilizers $W_1$ and $G_1$ have a common orbit of length 2 containing the point 2. So Norm$(W_{1,2}, G)$ and $G$ generate $N$ and in NormA we compute Norm$(W_{1,2}, G_{1,2})$ in order to obtain Norm$(W_{1,2}, G)$ as the normalizer of $G$ in Norm$(W_{1,2}, G_{1,2})$. But it takes about 50 minutes to compute Norm$(W_{1,2}, G_{1,2})$. Here we remark that in this case Norm$(W_{1,2}, G)$ and Norm$(W_{1,2}, G_1)$ are rather easily computed. Let $A$ be the automorphism group of the association scheme formed by $G$. Then $|A| = 2418647040$. In NormB the block $B$ of length 15 is used. Set $N' =$Norm$(A_B|B, G_B|B) \wr$Norm$(\bar{A}, \bar{B})$. Then $|A \cap N'| = 9447840$ and $(A \cap N')_1$ and $G_1$ has a common orbit of length 2 containing the point 2. So using Lemma 2 similarly as in NormA we compute Norm$((A \cap N')_{1,2}, G_{1,2})$. This normalizer is computed easily. Next we compute Norm$((A \cap N')_{1,2}, G)$ as the normalizer of $G$ in Norm$((A \cap N')_{1,2}, G_{1,2})$, which is also easy, and we obtain $N$ from Norm$((A \cap N')_{1,2}, G)$ and $G$. However it happens that Norm$(A_{1,2}, G)$ is also easily computed directly in this case. We note that $G_{1,2}$ is of order $3^5$ and has 9 orbits of length 3. A similar situation occurs in TransitiveGroup$(30.841)$.

Example 3: $G =$TransitiveGroup$(28, 321)$. In this case $|G| = 5376$, $|N| = 32256$ and $|A| = 192631799808$. $G$ has only one block $B$ of length 4 containing the point 1. So $W = Sym(4) \wr Sym(7)$. It is a little hard for DoNorm to compute Norm$(W, G)$. $W_1$, $A_1$ and $G_1$ have a common orbit of length 3 containing the point 2. Then it is a little harder for NormA to compute Norm$(A_{1,2}, G)$ and Norm$(W_{1,2}, G_{1,2})$. In NormB, $\bar{G} =$Action$(G, B^G)$ and $\bar{N}' =$Norm$(Sym(7), \bar{G})$ are computed. We have $|\bar{G}| = 7$ and $|\bar{N}'| = 42$. $G_B$ act on $B$ as an alternating group. So $N = Sym(4) \wr \bar{N}'$. However it happens that $A = N$ in this case. Then we compute Norm$(A_{1,2}, G_{1,2})$ and obtain Norm$(A_{1,2}, G)$ as the normalizer of $G$ in this normalizer easily. We note that $G_{1,2}$ is elementary abelian of order 64 and has 6 orbits of length 4.

Example 4: $G =$TransitiveGroup$(27, 1518)$. $|G| = 279936$. $|N| = 1679616$. $G$ has only one block of length 9 containing the point 1. So $W = Sym(9) \wr Sym(3)$. It is hard for DoNorm to compute Norm$(W, G)$. $W_1$ and $G_1$ have a common orbit of length 8 containing 2. In NormB we compute $N'$ as above and $|A \cap N'| = 483729408$. Then the normalizer is easily computed. $G_{1,2}$ has 2 orbits of length 9 and fixes remaining 7 points in $\Omega \backslash \{1, 2\}$. In NormA, NormalizerParentSA is invoked using $G_{1,2}$ to compute the normalizer of the action of $G_{1,2}$ on the orbit of length 9 and also to compute a element interchanging the two orbits of length 9. Let $W'$ be the group generated by them and the symmetric group on the 7 fixed points, which is small enough of order 1881169920. So the remaining computation goes smoothly. We note $|G_{1,2}| = 1296 = 2^4 \times 3^4$.

## 4. CONCLUDING REMARKS

As is seen in Table 1 in programs DoNorm, AS and NormA most normalizers are computed within 0.5 second, while in NormB most of them are computed between 1 and 2 seconds. In particular the GAP special function DoNorm computes most of them within 0.2 second. So if NormB is ten times slower than DoNorm in usual cases, it will be an unbearable defect of NormB for groups of large degree, since it may take longer time for DoNorm to compute such normalizers. Table

**Table 4: Some examples such that computing time varies in 3 trials by NormB ( in seconds )**

| $n$ | $k$ | 1 | 2 | 3 |
|---|---|---|---|---|
| 27 | 1542 | 1 | 43 | 42 |
| 28 | 1828 | 2 | 27 | 27 |
| 30 | 4099 | 26 | 27 | 3 |
| 30 | 5325 | 26 | 3 | 26 |
| 30 | 5623 | 27 | 2 | 27 |

| $n$ | $k$ | 1 | 2 | 3 |
|---|---|---|---|---|
| 28 | 1394 | 27 | 27 | 1 |
| 30 | 4092 | 26 | 2 | 28 |
| 30 | 4912 | 2 | 70 | 2 |
| 30 | 5495 | 27 | 2 | 26 |
| 30 | 5649 | 28 | 2 | 27 |

**Table 5: Some normalizers of groups of degree 64 and order 128 ( in seconds )**

| No. | DoNorm | AS | NormA | NormB |
|---|---|---|---|---|
| 1201 | 17 | 1 | 281 | 3 |
| 1202 | 15 | 1 | 1553 | 3 |
| 1203 | 10 | 1 | 46 | 3 |
| 1204 | 1687 | 1 | 9910 | 3 |
| 1205 | 209 | 2 | 19 | 3 |
| 1206 | 19 | 1 | 1644 | 3 |
| 1207 | 1755 | 8 | 86 | 10 |
| 1208 | 158 | 8 | 1442 | 10 |
| 1209 | 6 | 1 | 32 | 3 |
| 1210 | 117 | 8 | 1438 | 10 |
| 1211 | 12 | 1 | 8033 | 3 |
| 1212 | 2399 | 3 | 9463 | 5 |
| 1213 | 5 | 2 | 35 | 3 |
| 1214 | 785 | 1 | 8084 | 3 |
| 1215 | 643 | 2 | 2319 | 4 |
| 1216 | 88329 | 2 | ? | 3 |

**Table 6: Some normalizers of perfect groups in $S_n$** ( in seconds )

| order | No. | deg | DoNorm | AS | NormA | NormB |
|---|---|---|---|---|---|---|
| 979200 | 1 | 85 | 0.2 | 11 | 7 | 7 |
| 604800 | 1 | 100 | 0.3 | 5 | 4 | 24 |
| 647460 | 1 | 110 | 2 | 25 | 47 | 96 |
| 571704 | 1 | 168 | 15 | 32 | 428 | 1744 |
| 322560 | 23 | 192 | 1953 | 29 | 2888 | 1644 |
| 15600 | 1 | 208 | 16 | 49 | 1766 | 1013 |
| 322560 | 27 | 256 | 63 | 168 | 5686 | 6318 |

**Table 7: Some computing times of conjugating elements and normalizers ( in seconds )**

| $n$ | $k$ | conj. | Norm |
|---|---|---|---|
| 28 | 157 | 5472 | 9023 |
| 28 | 160 | 2405 | 5596 |
| 28 | 321 | 771 | 744 |
| 27 | 187 | 557 | 996 |
| 27 | 163 | 542 | 962 |
| 27 | 160 | 472 | 1876 |
| 28 | 392 | 419 | 1421 |

3 shows that there exist different groups of which normalizers are hard to compute for every programs. In this sense it may be difficult to say what program is the best one. However NormB seems best for groups of degree up to 30 or a little more. Some data of groups of large degree are shown in Tables 5 and 6. The groups listed in these tables are taken from Table 3 in [7] and Table 2 in [8]. Future work may be needed to determine whether NormB is adaptable to groups of higher degree. The examples of Table 6 can be computed quickly by DoNorm. But it is sure that there exist groups of which normalizers are not easily computed by DoNorm. It may be preferable to store in GAP the precomputed normalizer of each of the transitive permutation groups of small degree arising in the catalog, currently the transitive groups of degree at most 30. Then given a transitive group $G$ of small degree $n$, the normalizer of $G$ in $Sym(n)$ is obtained by simply finding a permutation which conjugates $G$ to the equivalent permutation group in the catalog. Table 7 lists execution times for determining the normalizer of $G$ by this approach. The computing time of finding conjugating elements may vary significantly depending on the conjugating element. Table 7 shows it takes more than 1 hour to compute a conjugating element between two permutation groups isomorphic to $\mathsf{TransitiveGroup}(28, 157)$ or to compute its normalizer.

## 5. REFERENCES


[1] E. Bannai and T. Ito. *Algebraic Combinatorics I: Association Schemes*. Benjamin/Cummings, Menlo Park, CA, 1984.

[2] G. Cooperman. *Parallel GAP/MPI (ParGAP/MPI)*, Version 1, *College of Computer Science, Northeastern University*, 1999, `http://www.ccs.neu.edu/home/gene/pargap.html`.

[3] The GAP Groups. GAP - groups, algorithms and programming, version 4. *Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany and School of Mathematical and Computational Sciences, Univ. St. Andrews, Scotland*, 2000.

[4] A. Hanaki. Data of association schemes, published at www. `http://kissme.shinshu-u.ac.jp/as/`, 1999∼.

[5] A. Hanaki and I. Miyamoto. Classification of association schemes of small order. *Discrete Math.*, 264:75–80, 2003.

[6] L.M. Luks and T. Miyazaki. Polynomial-time normalizers for permutation groups with restricted composition factors. In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, 176–183, 2002.

[7] I. Miyamoto. Computing normalizers of permutation groups efficiently using isomorphisms of association schemes. In *Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation*, 200–204, 2000.

[8] I. Miyamoto. Computing isomorphisms of association schemes and its applications. *J. Symbolic Comp.*, 32:133–141, 2001.