Computing Normalizers of Permutation Groups Efficiently using Isomorphisms of Association Schemes

Izumi Miyamoto
Department of Computer Science Yamanashi University
4-3-11 Takeda Kofu 400-8511, Japan
izumi@esi.yamanashi.ac.jp

ABSTRACT

This note presents an algorithm to speed up the computation of normalizers of permutation groups. It is an application of computation of isomorphisms of association schemes.

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

Algorithms

1. INTRODUCTION

In GAP4 there is a data of all transitive permutation groups of degree up to 22 [6]. These degrees seem to be small, but there are a couple of groups for which the computation of the normalizer in the symmetric group is hard either by GAP4 or by Magma2.5. In this note, for a permutation group G we present an algorithm to construct a group in which the normalizer of G is contained. Then computing in the group by a GAP4 command `Normalizer` [8, 7], we can obtain the normalizer in reasonable time. It takes about 10 seconds as total runtime in many of the hard cases stated above. The performance of our program is seen in the tables in section 5. Especially, we can compute all the normalizers of the 59 transitive permutation groups of degree 22 within about 280 seconds and compute all the normalizers of the 164 transitive permutation groups of degree 21 within about 20 minutes. Some results in larger degrees are also shown in section 5.

A. Hanaki and the author classified the association schemes with small number of vertices Ω up to isomorphism ([4], [5], [3]). For the definition and fundamental properties of association schemes, readers may refer to [1]. An association scheme used in this note is related to a transitive permutation group. Let G be a transitive permutation group on a set Ω. Here an association scheme is the orbits of G acting on $\Omega \times \Omega$. We simply call such orbits two-orbits of G. Let H be an another permutation group on Ω. Two association schemes associated with G and H are said to be isomorphic if and only if there is a permutation on Ω that moves the two-orbits of G to those of H. A permutation normalizing G clearly leaves the two-orbits of G invariant. This means that it may move one two-orbit to another but fixes them as a whole. Thus the normalizer of G is contained in the group consisting of all the permutations which give the isomorphisms from the association scheme associated with G to itself.

We use above fact in order to restrict the group where the normalizer of G is computed. The author wrote a program in GAP programming language [2] to compute normalizers of permutation groups using above argument. In the program we apply this argument repeatedly under some familiar condition stated in the lemma in section3 and finally use the command `Normalizer` of GAP4.

2. ASSOCIATION SCHEMES OF TRANSITIVE GROUPS

In this section we will introduce association schemes used in this note briefly. Let G be a transitive permutation group on a set of points Ω. An association scheme used in this note is the set of the orbits of G on $\Omega \times \Omega$. We note that the orbits correspond to the double cosets of G by the stabilizer of a point of Ω. We will denote such an orbit, as we called a two-orbit, by a matrix whose rows and columns are indexed by the set Ω. Let A_i be the matrix corresponding to the i-th two-orbit of G. Then the (p, q) entry of A_i is defined by

$$A_i(p,q) = \begin{array}{ll} 1 & \text{if the tuple } [p,q] \text{ is in the } i\text{-th two-orbit.} \\ 0 & \text{otherwise.} \end{array}$$

If there are $d+1$ two-orbits, then we have $d+1$ $\{0,1\}$-matrices $A_0, A_1, \cdots, A_d$. Let A_0 be the two-orbit containing $[p,p]$. Then the matrices have the following properties.

1. A_i has $\{0,\ 1\}$-entries and has constant column and row sum. A_0 =identity matrix.
2. $A_0 + A_1 + \cdots + A_d = J$(all 1 matrix)
3. For all i there exists i' such that ${}^tA_i = A_{i'}$.
4. $A_iA_j = \Sigma_{0\le k\le d}\ p_{ijk}A_k$ ($A_0,\ A_1, \cdots, A_d$ are a basis of the algebra generated by themselves.)

The last property follows that the double cosets are a basis of the subalgebra of the group algebra of G generated by themselves. We will write an association scheme by a matrix defined by $A = 0A_0 + 1A_1 + 2A_2 + \cdots + dA_d$ in section 4. Let H be a transitive group on Ω and let $B_0, B_1, \cdots, B_d$ be the association scheme of H, where B_0 is the identity matrix. Then the two association schemes of G and H are isomorphic if and only if there exist a permutation σ on the indices $\{1, 2, \cdots, d\}$ of the matrices and a permutation matrix P on Ω which satisfy

$$B_i = P^{-1} A_{i^\sigma} P \qquad \text{for } i = 1, 2 \cdots, d.$$

An isomorphism from an association scheme to itself will be called simply an isomorphism of the association scheme. Suppose that Q is the permutation matrix of a permutation which normalizes G. Then there exists a permutation τ on $\{1, 2, \cdots, d\}$ which satisfies that $Q^{-1}A_iQ = A_{i^\tau}$ for $i = 1, 2, \cdots, d$, since Q leaves the two-orbits of G invariant. Let N be the group consisting of the permutations on Ω each of which gives an isomorphism of the association scheme of G with some permutation σ on $\{1, 2, \cdots, d\}$. We will call N the group of isomorphisms of the association scheme. Our program computing the group of isomorphisms of an association scheme uses backtrack algorithms considering some properties of an association scheme mentioned above. We note that an automorphism of an association scheme is an isomorphism which leaves each A_i invariant and that G is contained in the automorphism group of the association scheme.

3. ALGORITHMS

Suppose that we want to compute a normalizer of a permutation group G on Ω. If G is transitive, let N be the group of the isomorphisms of the association scheme of G. If G is not transitive then each operation on an orbit of G gives an association scheme. We will consider all such association schemes. We call them the system of association schemes of G. Now we can compute the isomorphisms of each association scheme in the system. Moreover if two of the association schemes in the system are isomorphic to each other then we also compute a permutation which gives an isomorphism between them. We will consider the group generated by the isomorphisms of the association schemes in the system given by G and the set of isomorphisms between all the pairs of mutually isomorphic association schemes. We denote this group N. Then the normalizer of G is contained in N, since it leaves the set of the two-orbits of the actions of G on the orbits invariant. We call N the group of isomorphisms of the system of association schemes.

LEMMA 1. *Let K be a permutation group on Ω. Let F be a tuple $[p_1, p_2, \cdots, p_r]$ of points in Ω and let G^i be the stabilizer of the subset $[p_1, p_2, \cdots, p_i]$ of F as a tuple in G for $i = 1, 2, \cdots, r$. Let N^i be the group of isomorphisms of the system of association schemes of G^i on $\Omega \backslash [p_1, p_2, \cdots, p_i]$. Set $N^0 = N$, $G^0 = G$ and set $N^{\{0..i\}} = N^0 \cap N^1 \cap \cdots \cap N^i$. Suppose that $G^i \cap K$ is transitive on the orbit of $N^{\{0..i\}} \cap K$ containing the point p_{i+1} for $i = 0, 1, \cdots, r-1$. Then the normalizer of G in K is generated by $G \cap K$ and the normalizer of G in $N^{\{0..r\}} \cap K$.*

PROOF. Suppose that $r = 0$, that is, F is an empty set. Then it is clear that the normalizer of G in K is contained in $N \cap K$. So the lemma holds. Suppose that the normalizer of G in K is generated by $G \cap K$ and the normalizer of G in $N^{\{0..r-1\}} \cap K$. Consider the right coset of $N^{\{0..r-1\}} \cap K$ by its stabilizer of the point p_r. Since the subgroup $G^{r-1} \cap K$ of $N^{\{0..r-1\}} \cap K$ is transitive on the orbit of $N^{\{0..r-1\}} \cap K$ containing p_r, all the elements of a transversal of the right cosets can be chosen in $G^{r-1} \cap K$. This transversal is also a set of the representatives of the right cosets of the normalizer of G in $N^{\{0..r-1\}} \cap K$ by its stabilizer of p_r, since it contains $G^{r-1} \cap K$. So this normalizer is generated by the transversal and the normalizer of G in the stabilizer of p_r in $N^{\{0..r-1\}} \cap K$. Then the latter normalizer also normalizes G^r, since it fixes $[p_1, p_2, \cdots, p_r]$ pointwise, and so it is contained in N^r. This means that it is contained in $N^{\{0..r\}} \cap K$ and the lemma is proved by mathematical induction. □

We note that if G is not transitive and the actions of G on all the orbits are mutually isomorphic as permutation groups and if K is, for instance, the symmetric group then the lemma is of little use, because we can choose no points in the lemma and we have to compute the normalizer in N by the `GAP`-command `Normalizer`. Precisely, in our program we first check sizes of the orbits and the orders of the actions of G on the orbits and if the sizes and the orders are equal on certain two orbits then an isomorphism between them is computed as association schemes.

In the second example in the next section if K is the alternating group then $N \cap K$ is transitive but $G \cap K$ has two orbits isomorphic to each other. Therefore this lemma does not work well. So in such a case we had better compute the normalizer in the symmetric group where the lemma works well and then compute the intersection of K and the normalizer. On the contrary if K is substantially smaller than the symmetric group, then it would be better to compute the normalizer in $N \cap K$ directly whether the lemma works well or not. So if K does not contain G, we divided the way to compute the normalizer in our program by comparing the numbers of the orbits of G and $G \cap K$ heuristically.

If we use the sequence of points $[p_1, p_2, \cdots, p_r]$ in the lemma to compute normalizers then we compute isomorphisms of (systems of) association schemes $r+1$ times, where r is the number of the points in the sequence. In order to restrict the groups in which the normalizers are computed, it may not be wise to use the lemma as far as possible. Moreover heuristically, if the length of the orbit containing p_i is rather short, it does not seem to be efficient to use the point p_i. In any step of $i = 1, 2, \cdots, r$ there may exist more than one orbit satisfying Lemma 1. In such case we choose the point p_i in a longer orbit in our program. If the lemma works, the most part of computing time of our program is spent by the computation about the association schemes stated in the lemma.

4. EXAMPLES

Let K be the symmetric group of degree 14, let $p = (2, 8, 10, 13, 5, 4, 9, 6, 11, 7, 3)(12, 14)$ and let $G =$ `TransitiveGroup(14,23)`^{p} $=$ `Group`((8,9,11,10,13,14,12), (1,6,7)(2,5,4)(8,9,10)(11,14,13), (1,5)(2,7)(4,6)(8,14)(9,13)(10,11), (1,11,5,10)(2,9,7,13)(3,12)(4,8,6,14)) from `GAP` library. G is of order 588. Then the association scheme (the two-orbits) of the permutation group $G = G^0$ on $\Omega = \{1, 2, \cdots, 14\}$ is as follows.

$$\begin{pmatrix}
0&1&1&1&1&1&1&2&2&2&2&2&2&2\\
1&0&1&1&1&1&1&2&2&2&2&2&2&2\\
1&1&0&1&1&1&1&2&2&2&2&2&2&2\\
1&1&1&0&1&1&1&2&2&2&2&2&2&2\\
1&1&1&1&0&1&1&2&2&2&2&2&2&2\\
1&1&1&1&1&0&1&2&2&2&2&2&2&2\\
1&1&1&1&1&1&0&2&2&2&2&2&2&2\\
2&2&2&2&2&2&2&0&1&1&1&1&1&1\\
2&2&2&2&2&2&2&1&0&1&1&1&1&1\\
2&2&2&2&2&2&2&1&1&0&1&1&1&1\\
2&2&2&2&2&2&2&1&1&1&0&1&1&1\\
2&2&2&2&2&2&2&1&1&1&1&0&1&1\\
2&2&2&2&2&2&2&1&1&1&1&1&0&1\\
2&2&2&2&2&2&2&1&1&1&1&1&1&0
\end{pmatrix}$$

Here the rows and the columns of the matrix are indexed by $\Omega = \{1, 2, \cdots, 14\}$ and G has 3 two-orbits on Ω, which are distinguished by the entries 0, 1 and 2. ¿From this matrix we can see that G is imprimitive with two blocks of length 7 and the stabilizer of the blocks acts doubly transitively on each block. Then the group of isomorphisms of this association scheme $N = N^0$ is isomorphic to the wreath product $S_7 \wr S_2$, where S_n denotes the symmetric group of degree n. N^0 is transitive on Ω and the orbits of N^0_1, the stabilizer of 1 in N^0 are $\{1\}$, $\{2, 3, \cdots, 7\}$ and $\{8, 9, \cdots, 14\}$. Let $[p_1, p_2, p_3] = [1, 8, 9]$. Then we see that the two orbits of the groups G^i and N^i stated in Lemma 1 coincide with each other respectively for $i = 0$, 1, 2 and so they satisfy the assumption of Lemma 1. Here is the data of $G^1 = G_1$ and $G^2 = G_{1,8}$, the generators, the orbits and the systems of association schemes respectively. $G^1 = G_1 =$ `Group`((2,5)(3,6)(4,7)(8,14)(9,13)(10,11), (2,4,6)(3,5,7) (8,9,10) (11,14,13), (8,9,11,10,13,14,12)). $G^2 = G_{1,8} =$ `Group`((2,4,6)(3,5,7)(9,11,13)(10,12,14), (2,5)(3,6)(4,7)(9,12) (10,13)(11,14)) $\cong$ cyclic group of order 6. The orbits of G_1 are $\{1\}$, $\{2, 3, \cdots, 7\}$ and $\{8, 9, \cdots, 14\}$. The orbits of $G_{1,8}$ are $\{1\}$, $\{2, 3, \cdots, 7\}$, $\{8\}$ and $\{9, 10, \cdots, 14\}$. We construct an association scheme for each orbit of the groups. Each two-orbit of the action of the groups on the orbits is denoted by the entries with a same number in the following matrices respectively.

$$\begin{pmatrix}
8&&&\ast&&&&&&&\ast&&&\\
&0&1&2&3&4&5&&&&&&&\\
&5&0&1&2&3&4&&&&&&&\\
\ast&4&5&0&1&2&3&&&&\ast&&&\\
&3&4&5&0&1&2&&&&&&&\\
&2&3&4&5&0&1&&&&&&&\\
&1&2&3&4&5&0&&&&&&&\\
&&&&&&&6&7&7&7&7&7&7\\
&&&&&&&7&6&7&7&7&7&7\\
&&&&&&&7&7&6&7&7&7&7\\
\ast&&&\ast&&&&7&7&7&6&7&7&7\\
&&&&&&&7&7&7&7&6&7&7\\
&&&&&&&7&7&7&7&7&6&7\\
&&&&&&&7&7&7&7&7&7&6
\end{pmatrix}$$

$$\begin{pmatrix}
12&&&\ast&&&&&&&\ast&&&\\
&0&1&2&3&4&5&&&&&&&\\
&5&0&1&2&3&4&&&&&&&\\
\ast&4&5&0&1&2&3&&&&\ast&&&\\
&3&4&5&0&1&2&&&&&&&\\
&2&3&4&5&0&1&&&&&&&\\
&1&2&3&4&5&0&&&&&&&\\
&&&&&&&13&&&\ast&&&\\
&&&&&&&&6&7&8&9&10&11\\
&&&&&&&&11&6&7&8&9&10\\
\ast&&&\ast&&&&&10&11&6&7&8&9\\
&&&&&&&\ast&9&10&11&6&7&8\\
&&&&&&&&8&9&10&11&6&7\\
&&&&&&&&7&8&9&10&11&6
\end{pmatrix}$$

$G^3 = G_{1,8,9}$ is the identity group. Thus $N^1 =$`Group`((2,3,4,5, 6,7), (3,7)(4,6)) $\times S_7$, $N^2 =$ `Group`((2,3,4,5,6,7), (3,7)(4,6), (2,9)(3,10)(4,11)(5,12)(6,13)(7,14)) and $N^3 \cong S_{11}$. Hence $N^{\{0..3\}} = N^0 \cap N^1 \cap N^2 \cap N^3 =$ `Group`((2,3,4,5,6,7), (3,7) (4,6), (10,14)(11,13)). Therefore the normalizer of G is generated by G and the normalizer of G in $N^{\{0..3\}}$ which is computed by `GAP4`-command `Normalizer`. The program is written in `GAP` programming language and the total runtime is about 1 second. The runtime of the direct computation using `Normalizer` is about 57 seconds. Here we used a 300MHz MMX Pentium machine under Linux.

Next example is $K \cong S_{22}$ and $G =$`TransitiveGroup(22,48)` $\cong M_{11} \wr S_2$, where M_{11} denotes the Mathieu group of degree 11. G is imprimitive with two blocks of length 11 and the stabilizer of the blocks is isomorphic to $M_{11} \times M_{11}$ acting on each blocks 4-ply transitive. G is generated by (1,16,2,14,11,17,8,18,6,15,9,20)(3,22,10,19)(4,21,7,13,5,12) and (1,16,6,19,9,20,11,17,5,14,8,22,4,13,3,12)(2,18)(7,15,10, 21). Put $[p_1, p_2, \cdots, p_8] = [1, 12, 13, 2, 14, 3, 15, 4]$. The orbits of the stabilizer G_1 are $\{1\}$, $\{2, 4, 5, 10, 7, 8, 11, 6, 3, 9\}$ and $\{12, 13, 20, 15, 14, 22, 21, 18, 16, 19, 17\}$. The stabilizer $G_{1,12,13,2,14,3}$= `Group`((15,19,21,16)(17,22,18,20), (15,18,21, 17)(16,20,19,22), (4,5,10,8)(6,9,7,11), (4,7,10,6)(5,9,8,11)) $\cong$ $Q_8 \times Q_8$, a direct product of quaternion groups, and its orbits longer than 1 are $\{4, 5, 7, 10, 9, 11, 8, 6\}$ and $\{15, 19, 18, 21, 22, 20, 16, 17\}$. It acts regularly on them. The stabilizers G_1, $G_{1,12}$, $G_{1,12,13}$, $G_{1,12,13,2}$ and $G_{1,12,13,2,14}$ are doubly transitive on their orbits longer than 8. Any such association scheme in the system of association schemes given by those groups has a group of isomorphisms isomorphic to the symmetric group. Thus the permutation structure of G gives that $N^{\{0..6\}}$ is the direct product of the groups of isomorphisms of the two association schemes given by $G_{1,12,13,2,14,3}$ on its two orbits of length 8. On the other hand the regularity of $G_{1,12,13,2,14,3}$ implies that $N^{\{0..6\}}$ is the normalizer of $G_{1,12,13,2,14,3}$ in $S_8 \times S_8$ on its two orbits of length 8. This normalizer is computed following the algorithm in Lemma 1 in our program. Then $N^{\{0..8\}}$ is the stabilizer of $[15, 4]$ in $N^{\{0..6\}}$. The runtime of this example is about 6 seconds. It takes about 26066 seconds by direct computation by `Normalizer` in `GAP4`.

5. EXPERIMENTS

Our program is written in `GAP4` programming language, not a compiled code. We used a 300MHz MMX Pentium machine under Linux.

There are 59 transitive groups of degree 22. We computed the normalizers of all the transitive groups of degree 22 in S_{22}. It took about 280 seconds by our program and in maximum cases it took about 15 seconds to compute a normalizer. The isomorphisms of association schemes are computed in symmetric groups. So it may be easy to compute normalizers in the symmetric group by our method. We also computed the 59 normalizers in the alternating group. Then it took about 360 seconds.

Data of computation of some cases are shown in Table 1. The times are given in seconds. The first column shows the numbers of the transitive groups in `GAP` library. The second, third and fourth columns give the computing times by our method denoted by "as", `GAP4` and Magma2.5 respectively. In the cases of `TransitiveGroup(22,i)` with `i`= 15, 16, 17, 18, 19, 20, 24, 25, 31 and 40, we did not finished computing by `Normalizer` directly but we computed more than 100 times longer before interrupting the computation than by our method. All these groups are imprimitive with blocks of length 11. Among other transitive groups of degree 22 the worst case is `i`=22 in Table 1

Table 1
normalizer of `TransitiveGroup(22,i)` in S_{22}

i	as	GAP4	Magma2.5
59	3.6	0.7	0.2
58	2.4	0.9	0.2
57	7.5	1.2	0.2
56	7.4	0.5	0.2
55	7.4	0.5	0.2
54	6.8	0.5	0.2
53	13.5	0.5	0.2
52	12.4	1.0	0.2
51	13.3	0.5	0.3
50	12.7	0.5	0.8
49	13.0	0.5	0.3
48	8.2	26066.0	6596.2
47	5.4	0.5	10.8
46	4.9	0.5	1.4
45	4.9	0.4	2.9
44	9.1	28.2	922.1
43	9.2	30.0	15756.0
42	5.6	258.3	>17386.3
41	4.7	552.1	
40	6.1	>57783.4	
22	3.8	9611.8	

We also computed the normalizers of all the transitive groups of degree 21 in S_{21}. There are 164 transitive groups of degree 21. It took about 20 minutes to compute all the normalizers by our program. It took about 25 minutes to compute all the normalizers in the alternating group. In these cases it seemed hard to compute directly by `Normalizer` if a group has a block of length 7. Computing data of some cases are in Table 2.

Table 2
normalizer of `TransitiveGroup(21,i)` in S_{21}

i	as	GAP4	i	as	GAP4
116	6	1731	91	4	366
110	5	1730	88	4	10962
109	129	1770	87	4	20662
108	5	1753	85	4	521
106	6	1725	82	4	10896
102	5	8936	78	3	41
101	86	1742	77	3	29
97	5	1726	72	4	20764
95	5	1733			

Table 3
normalizers of perfect groups in S_n

order	No	name	degree	as	GAP4
443520		M_{22}	22	3	850
10200960		M_{23}	23	4	1025
244823040		M_{24}	24	5	1403
979200	1	$Sp_4(4)$	85	251	130
604800	1	J2	100	147	268
647460	1	L(2,109)	110	126	136
571704	1	SL(2,83)	168	389	–
322560	23		192	394	–
15600	1	SL(2,25)	208	771	–
322560	27		256	3168	–

Table 3 shows computing times for the normalizers of some perfect groups in the library of `GAP`. We ran `GAP4` with an option of memory 32MB and the symbol "–" in the table means that `GAP4` could not extend the workspace any more to accomplish the computation.

6. ABOUT COMPUTING ISOMORPHISMS

We note that if G is regular on Ω then the group of isomorphisms of the association scheme given by G is equal to the normalizer of G in the symmetric group on Ω, since the matrices A_0, A_1, $\cdots$, A_d of the association scheme are permutation matrices. As an extreme case we computed the normalizer of the regular representation of an elementary abelian group of order 64 in the symmetric group. It took 144 seconds by our method and 33 seconds by using `GAP`-command `Normalizer` directly. We also computed the cases of the regular groups of order 96. In most cases it took from 300 to 400 seconds to compute a normalizer by our program while it took from several to 100 seconds by direct use of `Normalizer`. There were a few cases that could not be computed in reasonable time by our program.

An association scheme is an algebraically combinatorial object. Among the properties of association schemes stated in Section 2, the fourth one is surely most algebraic. Our program computing isomorphisms of association schemes uses this property. The author gave an exposition about this fact at the 9th annual meeting of JSSAC held in this year. but there is no reference available for the present time. So we will note some remarks about this property here. If an association scheme is given by a regular representation of a group, then the fourth property follows the multiplication of the group. The normalizer of a permutation group leaves the two-orbits of same length invariant. So if a group has many two-orbits of same length, in our program we have to compute the element σ stated in Section 2 which permutes the two-orbits so many times in order to find all the isomorphisms without using this algebraic property. The regular case seems purely algebraic and as stated above the direct computation by `Normalizer` is usually 10 to 100 times as fast as using our program. Hence we computed the groups of isomorphisms of association schemes given by some transitive groups of degree 32 whose stabilizer of one point is of order 2 as another example. Every two-orbit of these groups is either of length 32 or 2×32 and we can construct many such groups from the groups of order 64 in `GAP` library. As before we used `Normalizer` to compute the normalizers in these groups of isomorphisms to obtain the normalizers in the symmetric group. We compared the total runtime of this method and the runtime of the direct computation by `Normalizer`. In most cases the runtimes of the above method were a few seconds and those of the direct computation were similar. But there were several cases that the direct computation was more than 10 times as slow as the above method. ¿From this aspect it is worth seeing the two-orbits as an association scheme.

7. ACKNOWLEDGMENTS

The author would like to thank the referee for his kind comments.

8. REFERENCES

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

[2] The `GAP` Group. *`GAP` – Groups, Algorithms and Programming, Version 4*. Lehrstuhl D f Mathematik, Rheinisch Westfälische Technisch Hochschule, Aachen, Germany and School of Mathematical and Computational Sciences, U. St.Andrews Scotland, 1997.

[3] A. Hanaki. *Data of association schemes, published at WWW*. http://math.shinshu-u.ac.jp/~hanaki, 1999.

[4] A. Hanaki and I. Miyamoto. Classification of association schemes with 16 and 17 vertices. *Kyushu J. Math.*, 52:383–395, 1998.

[5] A. Hanaki and I. Miyamoto. Classification of association schemes with 18 and 19 vertices. *Korean J. Comp. App. Math.*, 5:543–551, 1998.

[6] A. Hulpke. *Konstruktion transitiver Permutationsgruppen*. Dissertation, RWTH-Aachen, 1996.

[7] B. Mckay. Practical graph isomorphism. In *Proceedings of the Tenth Manitoba Conference on Numerical Mathematics and Computing, Vol. 1(Winnipeg, Man., 1980)*, pages 45–87. Congressus Numerantium 30, 1981.

[8] H. Theißen. *Eine Methode zur Normalisatorberechnung in Permutationsgruppen mit Anwendungen in der Konstruktion primitiver Gruppen*. Dissertation, RWTH-Aachen, 1997.