BULL. AUSTRAL. MATH. SOC. 20-04, 20D10
VOL. 40 (1989) [281–292]

SOME REMARKS ON THE COMPUTATION OF CONJUGACY CLASSES OF SOLUBLE GROUPS

M. MECKY AND J. NEUBÜSER

Dedicated with sincerest gratitude to Professor B. H. Neumann on the occasion of his eightieth birthday

Laue et al have described basic algorithms for computing in a finite soluble group G given by an AG-presentation, among them a general algorithm for the computation of the orbits of such a group acting on some set Ω. Among other applications, this algorithm yields straightforwardly a method for the computation of the conjugacy classes of elements in such a group, which has been implemented in 1986 in FORTRAN within SOGOS by the first author and in 1987 in C within CAYLEY. However, for this particular problem one can do better, as discussed in this note.

1. ORBITS OF SOLUBLE GROUPS

We recall briefly the set-up from [12], to which we refer for details. Let $G = G_0 > \ldots > G_n = \langle 1 \rangle$ be a composition series of G with factors $G_{i-1}/G_i = \langle g_i G_i \rangle$ of order some prime p_i, then $(g_1, \ldots, g_n)$ is a generating sequence, called a PAG sequence, with a defining set of relations

$$g_i^{p_i} = w_{ii}(g_{i+1}, \ldots, g_n) \text{ for } 1 \leqslant i \leqslant n,$$
$$[g_i, g_j] = w_{ij}(g_{j+1}, \ldots, g_n) \text{ for } 1 \leqslant j < i \leqslant n.$$

Each element can be expressed uniquely in the form

$$g = g_1^{\nu_1} \cdots g_n^{\nu_n} \quad \text{with } 0 \leqslant \nu_i < p_i.$$

If $\nu_i = 0$ for $i = 1, \ldots, k-1$ and $\nu_k \neq 0$, we call $\nu_k = \lambda(g)$ the *leading exponent* and $k = w(g)$ the *weight* of g. With respect to $(g_1, \ldots, g_n)$ each nontrivial subgroup $U \leqslant G$ has a unique "canonical generating sequence" $(u_1, \ldots, u_r)$, abbreviated CGS, with the following properties.

(1)	$(u_1, \ldots, u_r)$	is a PAG-sequence for U,
(2)	$w(u_i) > w(u_j)$	for $i > j$,
(3)	$\lambda(u_i) = 1$	for $i = 1, \ldots, r$,
(4)	$\nu_{w(u_i)}(u_j) = 0$	for $i \neq j$.

Received 30 January 1989

From any given generating set of a subgroup $U \leqslant G$, its CGS can be determined by a "non-commutative Gauß algorithm".

In [12] two "homomorphism principles" are recommended for the computation of orbits:

1. Reduction of orbit size using homomorphisms of G-sets ([**12**, 3.3, p.111–112]). In the context of the computation of conjugacy classes this can be done in the following form.

Let $N \triangleleft G$ and assume by induction that we know

(i) a set of representatives $\{h_1N, \ldots, h_kN\}$ of the conjugacy classes of G/N,

(ii) for each class representative $h_iN \in G/N$ a CGS of its centraliser $\overline{C}_i := C_{G/N}(h_iN)$.

Let C_i be the complete preimage of $\overline{C}_i$ in G. Then C_i acts by conjugation on the set h_iN of elements of G, and we obtain a full set of class representatives of G, with their centralisers in G, by computing for each i representatives $\{h_{i,1}, \ldots, h_{i,r_i}\}$ of the orbits under this action and their stabilisers in C_i (see [**12**, Section 6, p.124]).

This idea can be used recursively with any chain of normal subgroups in any group. Since in a straightforward approach one wants to get as small orbits as possible at each step, use of a chief series is then optimal. For soluble groups it is easy to determine a normal series with elementary abelian factors, and this is often good enough (see below).

2. Computation of orbits of a group G acting on a set Ω using the fact that orbits of a normal subgroup $N \triangleleft G$ are blocks for G. This leads to the "orbit algorithm for soluble groups" ([**12**, Sections 3.1, 3.2, p.111]).

In the implementations of an algorithm for the computation of classes of soluble groups mentioned above, the problem is thus reduced to a frequent application of 2. in the situation described in 1.; that is, the orbits of C_i on h_iN and stabilisers of representatives have to be determined. For this the orbit algorithm for soluble groups requires roughly $|h_iN| = |N|$ conjugations of elements, and since a conjugation is performed by collecting the word $g^{-1}hg$, this is the time-critical part of the method.

2. The use of affine operations

By induction, N can be assumed to be an elementary abelian p-group, and in this note we want to exploit this fact further.

So let N be an elementary abelian normal subgroup of G, let hN be a coset and C be the preimage of $\overline{C} := C_{G/N}(hN)$. Let $hn \in hN$, and $c \in C$, then $(hn)^c = h^c n^c = h[h,c]n^c$ with $[h,c] \in N$ and hence $(hn)^c = hn^c[h,c]$.

Via the one–one mapping $hn \mapsto n$ of $hN \to N$ we see that the action of C on the

set hN by conjugation is equivalent to the action of C on N described by

$$\alpha_c : \quad n \mapsto n^c[h,c].$$

Since $n \mapsto n^c$ is linear and $[h,c]$ is an element of N depending only on c (and h), the mapping α_c is an affine transformation of the $GF(p)$-vector space N onto itself.

These affine transformations can be computed quickly for the elements of a CGS of C, which is all that is needed of C in an orbit algorithm for soluble groups, and then the time-expensive conjugation of elements of hN can be replaced by the much faster application of such affine mappings to the elements of N. Let us call an implementation using only this replacement of actions the "first version".

However, the "affine action on N" yields more. Since $N \trianglelefteq C$, the orbits of N acting on N via α are blocks for the action of C. Now for $m \in N$ we have

$$\alpha_m : \quad n \mapsto n^m[h,m] = n[h,m],$$

that is, N acts on N via α as a group of "translations" by the elements of the subgroup $[h,N]$. The orbits of this action are the cosets of $[h,N]$ in N. Hence we can find the orbits of C acting on N by first forming $N/[h,N]$, which is very cheap, and then applying the orbit algorithm only to the action $\overline{\alpha}$ of C on this often much smaller set. As

$$\overline{\alpha}_c : \quad n[h,N] \mapsto n^c[h,c][h,N],$$

$\overline{\alpha}$ is affine again and has N in its kernel; hence it induces an affine action of $\overline{C} = C/N$ on $N/[h,N]$.

We make use of this remark in the implementation, but even more in the counting of classes described in Section 5.

Assume we have thus found representatives $n_1, \ldots, n_s$ of the affine action α of C on N, and that, from the use of the orbit algorithm for the induced action $\overline{\alpha}$ on $N/[h,N]$, we have, for each n_i, a CGS of the stabiliser S_i of $n_i[h,N]$. In terms of the action of C on N, this group S_i, of course, is the block stabiliser of the block $n_i[h,N]$ with representative n_i. Let $c \in S_i$, then there exists $m \in N$ with

$$(*) \qquad \alpha_c(n_i) = n_i^c[h,c] = n_i[h,m], \quad \text{hence} \quad [h,m] = [n_i,c][h,c].$$

As all commutators are in N, we get from this

$$[n_i, cm^{-1}][h, cm^{-1}] = 1, \quad \text{that is,} \quad \alpha_{cm^{-1}}(n_i) = n_i.$$

So in each coset $cN \in S_i/N$ there exist representatives which lie in the stabiliser of n_i in C, and these can be found by solving the system of linear equations determined by

$(*)$ for the unknown m's with given h, n_i, and c. Actually, if m_1 and m_2 both solve $(*)$, then $m_1 m_2^{-1} \in C_N(h)$. So, if $C_N(h)$ is determined once for dealing with the coset hN, it suffices to find just one solution of $(*)$ for each given n_i and c, which, by the way, only influence the "right hand side" of the inhomogeneous equations $(*)$.

In the acknowledgement we refer to the implementation of this idea as the "second version".

3. Some details of implementation and modifications of the method

3.1 The first description in [8] of a construction of the classes of a group G from those of a factor group G/N dealt with the simplest case, namely N being central of order p; as shown by applications, its implementation yielded a remarkably efficient program for p-groups. C. Leedham-Green proposed the use of the following generalisation of the (very trivial) central lemma of [8]:

Let G be a soluble group, N an elementary abelian, central subgroup of G, $h \in G$ and C the preimage of $C_{G/N}(hN)$. A system R of representatives of the C-classes in hN and the centraliser $C_G(h)$ are obtained as follows:

If $[h, C] = \langle 1 \rangle$, then $R = hN$ and $C_G(h) = \langle C, N \rangle$.

If $[h, C] \neq \langle 1 \rangle$ let K be a complement of $[h, C]$ in N, then $R = hK$. Let $(c_1 N, \ldots, c_m N)$ be a CGS of $C_{G/N}(hN)$; running through $c_m, \ldots, c_1$ in that order, choose a basis $B = \{b_1 = [h, c_{i_1}], \ldots, b_r = [h, c_{i_r}]\}$ of $[h, C] \leqslant N$. For $[h, c_{i_k}] \notin B$ let

$$[h, c_{i_k}] = b_1^{e_1^{(k)}} \ldots b_r^{e_r^{(k)}}.$$

Then
$$C_G(h) = \langle c_{i_k} \left(c_{i_1}^{e_1^{(k)}} \ldots c_{i_r}^{e_r^{(k)}} \right)^{-1}, N \mid k = r+1, \ldots, m \rangle.$$

The implementation of this proposal for p-groups in CAYLEY has indeed led, where applicable, to a further gain in efficiency. There is, of course, no reason to restrict its use to p-groups, so the present SOGOS implementation can test whether an elementary abelian factor is central, and if so, use Leedham-Green's lemma, which may be viewed as solving equation $(*)$ of Section 2 in the special case that N is central.

3.2 Another simplification is possible if N is cyclic (non central) of some order p, since then the "affine" operation reduces to calculation in $\mathbf{Z}/p\mathbf{Z}$.

3.3 As was pointed out in Section 1, each subgroup of a group given by a PAG sequence can *uniquely* be described by a CGS, which is most helpful if subgroups have to be compared. However, at intermediate steps of the algorithm, a description of the centralisers by a generating sequence satisfying only conditions (1) and (2) of the definition of a CGS suffices, a simplification that again contributes significantly to efficiency.

3.4 A soluble group may have different normal series with elementary abelian factors; experiments have shown that computing times for the classes strongly depend on the choice of the series, in particular on the sizes of their elementary abelian factors. No very definite statement has emerged, but as a rule of thumb, the use of "linear algebra" is preferable, that is, a big elementary abelian subgroup N is "good" for the calculation of the representatives in hN, if the index $N:[h,N]$ is small. Moreover, the small cost of the test of centrality of N as described in 3.1. is worth the gain obtained when the answer is positive. Central factors should, if possible, be moved to the end of the series, since the increase of the number of classes from them becomes costly in subsequent steps.

3.5 As in the p-group case described in [8], the algorithm allows variations:

(a) Since quite generally $|C_{G/N}(hN)| \leqslant |C_G(h)|$, the computation can be restricted to classes whose elements have centralisers of order below a given upper bound by controlling the growth of centraliser orders at each inductive step.

(b) For a given element $h \in G$, the centraliser of h and a "canonical" (of course only with respect to the given normal series and depending on the algorithm) representative h^* of the class of h can be found, as well as an element transforming h into h^*.

(c) Since h^* depends only on the class h^G (besides the normal series and the algorithm) but not on the particular element $h \in h^G$, it can serve in a variation in which conjugacy of two given elements h and h' is tested and a transforming element is determined if they are conjugate. This variation is needed, for example, in the computation of power maps for character tables as well as of the class-multiplication coefficients needed for their determination by the Dixon-Schneider method [16].

3.6 Using a CGS of a subgroup $U < G$, and a series of normal subgroups of G with elementary abelian factors passing through $M \triangleleft G$, the classes of M under conjugation by U can be found analogously.

For details on these points, as well as for implementations in SOGOS, see [13].

3.7 As a final variation of the general method we mention the possibility of computing the fusion of the classes of a subgroup H into the classes of G, stepping down from the fusion of the classes of HN/N into the classes of G/N. This method has been implemented by Thiemann [18] within SOGOS for use with an implementation of an algorithm [4] for the computation of characters of p-groups as induced characters, a version of which was communicated to us by C. Leedham-Green and A. Mann. In SOGOS a set of one-dimensional characters of subgroups is determined, from which all irreducible characters can be obtained by induction, and the fusion map for these

subgroups and the power map for the whole group is computed. These data are then handed over to the CAS system, where the induced characters are computed using the facilities of CAS [14] for handling cyclotomic integers.

4. Performance

We compare implementations for the following test groups:

G_1 is a group constructed by Glasby [3] by iterated split extension:

$$E_{3^9} \rtimes (E_{2^7} \rtimes (E_{3^3} \rtimes (Q_8 \rtimes S_3))),$$

where E_n is an extraspecial group of order n, Q_8 is the quaternion group and S_3 the symmetric group of degree 3. The following chief series (top down) was used: $2, 3, 2^2, 2, 3^2, 3, 2^6, 2, 3^8, 3$.

$S_4\, wr_6\, A_4$ is the wreath product of the symmetric group S_4 with the alternating group A_4, represented as permutation group of degree 6. Series used: $3, 2, 2^4, 2^3, 3^6, 2^{12}$.

$3^9 \rtimes (757 \rtimes 9)$, $3^{10} \rtimes (61 \rtimes 10)$, and $5^9 \rtimes (19 \rtimes 9)$ are split extensions of elementary abelian groups of order 3^9, 3^{10}, and 5^9, respectively, with parts of the normalisers of their respective Singer cycles. Series used: $3, 3, 757, 3^9$; $5, 2, 61, 3^{10}$, and $3, 3, 19, 5^9$, respectively.

$(S_4\, wr\, S_3) wr\, S_3$ is the iterated wreath product of S_4 by S_3 in its natural permutation representation. Series used: $2^3, 3, 2^4, 3^3, 2^6, 3^9, 2^{18}$.

group	order	max.factor	# classes	CAYLEY 1987	SOGOS 1988
G_1	$2^{11} \cdot 3^{13}$	$3^8 = 6561$	181	2496^{sec}	101^{sec}
$S_4\, wr_6\, A_4$	$2^{20} \cdot 3^7$	$2^{12} = 4096$	1900	4438^{sec}	474^{sec}
$3^9 \rtimes (757 \rtimes 9)$	$3^{11} \cdot 757$	$3^9 = 19683$	135	5379^{sec}	62^{sec}
$3^{10} \rtimes (61 \rtimes 10)$	$2 \cdot 5 \cdot 3^{10} \cdot 61$	$3^{10} = 59049$	219		116^{sec}
$5^9 \rtimes (19 \rtimes 9)$	$3^2 \cdot 5^9 \cdot 19$	$5^9 = 1953125$	11575		3715^{sec}
$(S_4\, wr\, S_3) wr\, S_3$	$2^{31} \cdot 3^{13}$	$2^{18} = 262144$	52195		23856^{sec}

Computing times are given in seconds for comparable implementations, all using J. Cannon's STACKHANDLER and the 1982 version of V. Felsch's preprocessor generated "from-the-right" collector [5] on the same machine, a MASSCOMP 5500 PEP. As they have been measured in multiuser mode, they should be considered accurate only within a margin of some %. For reasons of space, computation of the classes of

$(S_4 \, wr \, S_3) wr \, S_3$ could only be finished by removing each centraliser during the last step, once the corresponding class had been found.

We close this section with a rather technical remark.

In the last three cases the 1987 CAYLEY-implementation ran out of storage space in our environment. At least in the case of $3^{10} \rtimes (61 \rtimes 10)$, however, the 1986 SOGOS implementation did not, but finished, although only after 3200 seconds. As stated in the introduction, the two implementations use the same mathematical set-up, the different performance is due to a small, but crucial difference in the implementations of the orbit algorithm for soluble groups: while in the 1986 SOGOS implementation each orbit under the action of the whole group is constructed separately (as described in [12]), the 1987 CAYLEY implementation first finds *all* orbits of the first generator, then links them under the action of the second generator, and so on. While both methods have roughly the same time-complexity, the second can be critically more wasteful of space.

5. Counting classes

By the method of Section 2, for each class a representative and its centraliser are explicitly constructed. Of course the amount of output produced this way already limits the application, as for example demonstrated by a group G_2 that S. Glasby sent us in September 1988 as a challenge for our method. The factor group of G_2 by its centre of order 13 is the group $G_3 = N \rtimes (E_{7^3} \rtimes G_{48})$ where N is an elementary abelian minimal normal subgroup of order 13^{14} of G_3, E_{7^3} is extraspecial of order 7^3, and G_{48} is a group of order 48, which can be described as a non-split extension of the quaternion group of order 8 by S_3 or as the non-split double cover of S_4. A quick estimate shows that N alone contains more than 10^{11} classes. So at best one can ask to count them. On the other hand, classes outside N can be expected to be fairly big, hence there ought to be many fewer of them. In fact in a run that took about 16 hours, our implementation explicitly constructed all classes outside N. There are 373738 of them.

As we shall see, it is then only a hand computation to find from the data obtained in this computation with the help of the Cauchy-Frobenius-Burnside (CFB) Lemma that there are 239150784421 classes inside N, so that G_3 has altogether 239151158159 classes. Rather than going into detail with this example, we describe how in general the number of all classes of a soluble group G can be obtained fairly easily from explicit knowledge of the classes of G/N, where N is an elementary abelian normal subgroup.

As in Section 1, let N be an elementary abelian normal subgroup of G, and assume that we know

(i) a set of representatives $\{h_1 N, \ldots, h_k N\}$ of the conjugacy classes of G/N,

(ii) for each class representative $h_i N$ a CGS of its centraliser $\overline{C}_i = C_{G/N}(h_i N)$.

As explained in Section 2, in order to find the total number of classes of G, we have to find for each h_iN, $i = 1, \ldots, k$, the number of orbits of $\overline{C}_i$ on $N/[h_i, N]$. However by the CFB Lemma, the number b_i of these orbits is

$$b_i = \frac{1}{|\overline{C}_i|} \sum_{\overline{c} \in \overline{C}_i} f(\overline{c}),$$

where $f(\overline{c})$ is the number of fixed points of $\overline{c}$ in its affine action on $N/[h_i, N]$.

Clearly conjugate elements in $\overline{C}_i$ have the same number of fixed points and so

$$b_i = \frac{1}{|\overline{C}_i|} \sum_{\overline{c} \in R} \frac{f(\overline{c}) \cdot |\overline{C}_i|}{|C_{\overline{C}_i}(\overline{c})|} = \sum_{\overline{c} \in R} \frac{f(\overline{c})}{|C_{\overline{C}_i}(\overline{c})|}$$

where R is a system of representatives of the conjugacy classes of $\overline{C}_i$.

Now let $\overline{c} \in R$, that is, $\overline{c}$ is a coset of N, and $c \in \overline{c}$. Then the fixed points $n[h_i, N]$ of $\overline{c}$ on $N/[h_i, N]$ are the solutions of

$$(**) \qquad n[h_i, N] = n^c[h_i, c][h_i, N]$$

which for given h_iN and $\overline{c}$ is an inhomogeneous system of linear equations. Let $n_1[h_i, N]$ and $n_2[h_i, N]$ be two solutions of $(**)$, then

$$n_1 n_2^{-1}[h_i, N] = \left(n_1 n_2^{-1}\right)^c [h_i, N],$$

that is, the set of solutions of $(**)$ is either empty or a coset of $C_{N/[h_i,N]}(\overline{c})$. However $C_{N/[h_i,N]}(\overline{c})$ is easily found as the eigenspace with eigenvalue 1 of the linear operation of $\overline{c}$ on $N/[h_i, N]$.

The case of the classes in N in Glasby's group G_3 which was left out at the beginning of this section becomes particularly easy. With $h_1 = 1$, we have $[h_1, N] = 1$, hence $\overline{C}_1 = G/N$, and R is the system of representatives of the conjugacy classes of G/N. For each $\overline{c} \in R$ we have to find $|C_N(c)|$ for some $c \in \overline{c}$. Since $C_N(c) = \ker(1 - c)$ and $[c, N] = \operatorname{im}(1 - c)$, we have $|N| = |C_N(c)| \cdot |[c, N]|$, but the groups $[c, N]$ had already been computed in the course of finding the classes outside N, so that indeed the total number of classes in Glasby's group is now easily found. However, with an implementation of the complete counting method, this number can be found much faster, namely in 16 minutes.

The CFB Lemma yields only the total number of classes, but since in cases like Glasby's group, only groups $\overline{C}_i$ are involved in the actual counting, which are comparatively few (57) and small $\left(|C_i| \leqslant 7^3 \cdot 48\right)$, two different methods can be employed to count the classes by their lengths.

One is the use of the generalised CFB Lemma that is described by Klass [11]. It gives the number of orbits of a $\overline{C}_i$ of given length l in terms of the Möbius function on the subgroup lattice of $\overline{C}_i$ and of the numbers of (common) fixed points of subgroups of $\overline{C}_i$ of index $\geqslant l$.

The second is the use of the Burnside matrix of $\overline{C}_i$. We recall briefly the relevant facts. Let $U_1, \ldots, U_n$ be a set of representatives of the conjugacy classes of subgroups of a finite group H, and let Δ_{H/U_i} be the permutation representation of H on the coset space H/U_i. Then $\Delta_{H/U_1}, \ldots, \Delta_{H/U_n}$ is a set of representatives of the equivalence classes of transitive permutation representations of H. For any permutation representation Δ_Ω of H on a set Ω we therefore have

$$\Delta_\Omega \sim \bigoplus_{i=1}^{n} m_i \Delta_{H/U_i},$$

where m_i is the number of orbits of H on Ω, which afford a (transitive) permutation representation of H that is equivalent to Δ_{H/U_i}. Obviously the number of orbits of a given length l is then obtained as the sum of those m_i for which the index $H : U_i$ is equal to l.

Now if we restrict the representations Δ_Ω and Δ_{H/U_i} to a subgroup U_j and denote by $f_\Omega(U_j)$ and $f_{H/U_i}(U_j)$, respectively, the number of points that U_j leaves fixed in its action on Ω and H/U_i then the decomposition of Δ_Ω given above yields

$$(***) \qquad f_\Omega(U_j) = \sum_{i=1}^{n} m_i f_{H/U_i}(U_j).$$

Since this holds for $j = 1, \ldots, n$, this is an inhomogeneous system of n linear equations for the n unknown values m_i, once $f_\Omega(U_j)$ and the $f_{H/U_i}(U_j)$ are known.

Now $f_{H/U_i}(U_j)$ is the number of cosets $U_i g$ of U_i such that

$$(\forall h \in U_j)(U_i g h = U_i g) \Leftrightarrow (\forall h \in U_j)\left(g h g^{-1} \in U_i\right) \Leftrightarrow U_j \leqslant g^{-1} U_i g.$$

Therefore $$f_{ij} = f_{H/U_i}(U_j) = n_{ij} \cdot (N_H(U_i) : U_i),$$

where n_{ij} is the number of conjugates of U_i containing U_j.

If we order the representatives $U_1, \cdots, U_n$ of the classes of subgroups of H such that $|U_i| \leqslant |U_j|$ for $i < j$, the f_{ij} form a triangular matrix ($f_{ij} = 0$ for $i < j$), which is known as Burnside's table of marks [1].

In our application, this Burnside matrix can be calculated for each of the (comparatively few and small) groups $\overline{C}_i$ with the help of Felsch's subgroup lattice program [6], which has been incorporated into the SOGOS system. Since the affine operation

of $\overline{C_i}$ on $\Omega = N/[h_i, N]$ is known, together with the subgroup lattice of $\overline{C_i}$ the number $f_\Omega(U_j)$ can easily be determined for a representative of each class of conjugate subgroups of $\overline{C_i}$, so that the m_i can be determined from the system $(***)$ of linear equations, which is even already in triangular form.

In the case of Glasby's group we obtain the following results.

We describe a class by a pair of numbers, for example (21,42), of which the first is the order of an element in the class, the second the order of its centraliser, and we indicate by $6\times(21,42)$ that there are 6 such classes.

There are 48 classes in G_3/N for which the full preimage is a class in G_3 (and the order of the elements is the same in G_3/N and in G_3). We list these together

$$6\times(7,16464),\ 6\times(14,336),\ 6\times(21,42),\ 6\times(28,28),$$
$$6\times(28,56),\ 6\times(42,42),\ 12\times(56,56).$$

There are 9 classes in G_3/N for which the full preimages split into several classes in G_3. We list these ordered by the classes in G_3/N:

classes in G_3/N	classes in G_3
(1, 16464)	$1\times(1,\ 16464*13^{14}),\ 12\times(13,\ 21*13^{14}),\ 12\times(13,\ 8*13^{14}),$
	$12\times(13,\ 6*13^{14}),\ 12\times(13,\ 4*13^{14}),\ 172368\times(13,\ 3*13^{14}),$
	$28718\times(13,\ 2*13^{14}),\ 239150583286\times(13,\ 13^{14})$
(2, 336)	$1\times(2,\ 336*13^{6}),\ 12\times(2*13,\ 8*13^{6}),\ 12\times(2*13,\ 6*13^{6}),$
	$12\times(2*13,\ 4*13^{6}),\ 28718\times(2*13,\ 2*13^{6})$
(3, 42)	$1\times(3,\ 42*13^{6}),\ 24\times(3*13,\ 6*13^{6}),\ 344760\times(3*13,\ 3*13^{6})$
(4, 56)	$1\times(4,\ 56*13^{2}),\ 24\times(4*13,\ 8*13^{2})$
(4, 28)	$1\times(4,\ 28*13^{2}),\ 24\times(4*13,\ 4*13^{2})$
(6, 42)	$1\times(6,\ 42*13^{2}),\ 24\times(6*13,\ 6*13^{2})$
(7, 49)	$1\times(7,\ 49*13^{2}),\ 24\times(7*13,\ 7*13^{2})$
(8, 56)	$1\times(8,\ 56*13^{2}),\ 24\times(8*13,\ 8*13^{2})$
(8, 56)	$1\times(8,\ 56*13^{2}),\ 24\times(8*13,\ 8*13^{2})$

Note that the counting with the help of the Burnside matrix yields the number of classes of given length, but in general not at the same time a separation by the pairs

(order, class length). In our case this has easily been inferred using the fact that $|N|$ and the index $G_3 : N$ are coprime.

The determination of this table took less than 30 minutes in the same computing environment as described in Section 4.

6. A REMARK ON COLLECTORS

Very recently, Soicher has shown by counting elementary operations, that a "from-the-left" collector should be much faster on hard collections than a "from-the-right" collector, and Vaughan-Lee has reported that replacement of the Havas-Nicholson "from-the-right" collector [**10**] by a "from-the-left" collector written by him in the Canberra Nilpotent Quotient program leads to a speed-up by a factor ranging up to 14 in certain examples.

Felsch has written a Fortran version of Soicher's collector and has compared its performance with that of his preprocessor generated "from-the-right" collector. The results are rather puzzling: The square of the "maximal element" in Glasby's group G_1 (that is, the element with maximal exponents in its representation as a normal word in the PAG generating sequence) was formed about 800 times (!) faster by Soicher's collector; squaring 10 "randomly chosen" elements from Glasby's group worked between 10 and 100 times faster. However, the determination of the classes of Glasby's group took about 1.6 times as long (!!) with Soicher's collector. With other groups as well as with other algorithms from SOGOS similar observations were made [**7**]. The reasons for these surprising facts are not completely clear. Observations in these experiments seem to indicate that algorithms like the one for the determination of the classes, working down some series of factors closely linked to the PAG generating sequence of the soluble group, work most of the time with short and sparse words, thus not giving these new collectors a fair chance to show their power on long and dense words. So while with the present range of applicability perhaps not too spectacular a gain can be expected from the introduction of the new collectors into algorithms of the kind described here, this topic needs further investigation, in which also preprocessor generated "from-the-left" collectors will have to be considered, as well as the balance of the cost of repeated preprocessing against the gain brought by a code tailored for a particular group.

7. CONCLUDING REMARKS AND ACKNOWLEDGEMENT

The method of "affine operation" was first proposed by Pahlings and Plesken in [**15**]; the authors thank W. Plesken for pointing this out in a discussion which led to the "second version" and H. Pahlings for several most helpful discussions, in particular about the use of the Burnside matrix. The method of affine action has recently also been used by Glasby and Slattery in an implementation of an algorithm for the computation

of intersections of subgroups of soluble groups in CAYLEY [9]. We thank the authors for sending us a preprint. Further applications, in particular to the calculation of normalisers of subgroups and to complements and supplements of normal subgroups, are possible. We hope to be able to come back to these topics later on.

References

[1] W. Burnside, *Theory of groups of finite order*, 2nd ed (CUP, Cambridge, 1911).

[2] J. Cannon, 'An introduction to the group theory language, Cayley', in *Computational Group Theory: Proc. LMS Symposium, Durham 1982*, Editor M. Atkinson (Academic Press, Florida, 1984). 146–183.

[3] J. Cannon, (private communication, 1987).

[4] S. B. Conlon, 'Calculating characters of p-groups', *J. Sci Comput.*, (to appear). (University of Sydney 1987, Preprint).

[5] V. Felsch, 'A machine independent implementation of a collection algorithm for the multiplication of group elements', *SYMSAC, 1976 (ACM 1976)*. Editor R.D. Jenks 159–166.

[6] V. Felsch and G. Sandlöbes, 'An interactive program for computing subgroups', in *Computational Group Theory: Proc. LMS Symposium, Durham 1982*, Editor M. Atkinson (Academic Press, Florida, 1984). 137–143 .

[7] V. Felsch, 'Comparing different collectors in SOGOS', (Preliminary report, Lehrstuhl D für Mathematik, RWTH Aachen, 1988).

[8] V. Felsch and J. Neubüser, 'An algorithm for the computation of conjugacy classes and centralizers in p-groups' **72**: *Lecture Notes in Comput. Sci.* (Springer-Verlag, Berlin, Heidelberg, New York, 1979). 452–465 .

[9] S. P. Glasby and M. C. Slattery, 'Computing intersections and normalizers in soluble groups'. (preprint) .

[10] G. Havas and T. Nicholson, 'Collection', in *SYMSAC, 1976 (ACM, 1976)*, Editor R.D. Jenks. 9–14 .

[11] M. J. Klass, 'A generalization of Burnside's combinatorial lemma', *J. Combin. Theory Ser. A* **20** (1976), 273-278.

[12] R. Laue, J. Neubüser and U. Schoenwaelder, 'Algorithms for finite soluble groups and the SOGOS system', in *Computational Group Theory: Proc. LMS Symposium, Durham 1982*, Editor M. Atkinson (Academic Press, Florida, 1984). 105–135 .

[13] M. Mecky, 'SOGOS IV, Konjugiertenklassen': *Diplomarbeit* (Lehrstuhl D für Mathematik, RWTH Aachen, 1989).

[14] J. Neubüser, H. Pahlings and W. Plesken, 'CAS; design and use of a system for the handling of characters of finite groups', in *Computational Group Theory: Proc. London Math. Soc. Symposium, Durham 1982*, Editor M. Atkinson (Academic Press, Florida, 1984). 195–247 .

[15] H. Pahlings and W. Plesken, 'Group actions on Cartesian powers with application to representation theory', *J. Reine Angew. Math.* **380** (1987), 178-195.

[16] G. Schneider, 'Dixon's character table algorithm revisited', (Universität Essen and University of Sydney. preprint).

[17] *SOGOS Manual* (Lehrstuhl D für Mathematik, RWTH Aachen, 1982); revised version 1988.

[18] P. Thiemann, 'SOGOS III, Charaktere': *Diplomarbeit* (Lehrstuhl D für Mathematik, RWTH Aachen, 1987).

Lehrstuhl D für Mathematik
RWTH Aachen
D-5100 Aachen, Federal Republic of Germany