MATHEMATICS OF COMPUTATION
Volume 69, Number 232, Pages 1633–1651
S 0025-5718(99)01157-6
Article electronically published on May 24, 1999

CONJUGACY CLASSES IN FINITE PERMUTATION GROUPS VIA HOMOMORPHIC IMAGES

ALEXANDER HULPKE

ABSTRACT. The lifting of results from factor groups to the full group is a standard technique for solvable groups. This paper shows how to utilize this approach in the case of non-solvable normal subgroups to compute the conjugacy classes of a finite group.

1. INTRODUCTION

The determination of all conjugacy classes of a finite permutation group G has already been the subject of several studies [18, 2, 3, 5]. A first approach is to check random group elements for conjugacy until representatives for all conjugacy classes have been found. Despite the simplicity this approach works quite effectively if there are few and large conjugacy classes, for example if the group is almost simple. As soon as there are small conjugacy classes, however, the algorithm will not finish in reasonable time, because representatives from these classes will not be found. Thus one of the main objectives of any better conjugacy class algorithm has to be the creation of a list of group elements of sufficiently small size which contains representatives for all classes.

The inductive approach [3] considers elements as roots of their prime-order powers. Thus every element can be found in the centralizer of a p-element. These elements themselves will be found in the Sylow subgroups, which are usually smaller than the group and for which effective methods for the determination of conjugacy classes are known [8].

In some cases, however the sheer size of p-element centralizers and the number of rational classes in a Sylow subgroup may render this approach unusable. A typical example is the group

$$[\tfrac{1}{2}S_{11}^2]2 = \langle (1,2,3), (1,2,3,4,5,6,7,8,9,10,11), (1,2)(12,13)$$
$$(1,12)(2,13)(3,14)(4,15)(5,16)(6,17)(7,18)(8,19)(9,20)(10,21)(11,22)\rangle$$

of degree 22. In this group, for example, the element $(1,2,3)$ has a centralizer of size $2^{14}3^75^37^211$ and the 2-Sylow subgroup has 460 classes which fuse to only 70 classes of 2-elements in the full group.

Received by the editor November 17, 1997 and, in revised form, November 17, 1998.
1991 *Mathematics Subject Classification.* Primary 20-04, 20B40, 68Q40.
Key words and phrases. Conjugacy classes, permutation group, algorithm.
Supported by EPSRC Grant GL/L21013.

Definition 1. A group H which is the direct product of simple groups of the same type, $H \cong T \times \cdots \times T$ (T simple), is called *elementary*. If T (and H) are abelian, H is called elementary abelian.

If G possesses an elementary abelian normal subgroup N, the problem can be reduced substantially: provided the classes of G/N are known, the classes of G can be lifted from these by an affine action on N. This leads to an efficient algorithm to determine all conjugacy classes of solvable groups given by PC presentations [15]. As a PC presentation can be computed easily for solvable permutation groups [19], we will only have to consider non-solvable permutation groups further on.

In fact, the affine method from [15] does not require the factor group to be solvable, and so we can in principle even restrict to groups with no solvable normal subgroup. In [5] it is shown that groups of this type have a large normal subgroup that is a direct product. In [5], the authors then suggest parametrizing the classes in this normal subgroup and using a random search outside. Again, however, groups like $[\frac{1}{2}S_{11}^2]2$ tend to possess some very small classes outside the direct product normal subgroup (data to support this claim can be found in Section 11). For groups like these, the random search is likely to take a very long time. This makes it desirable not only to increase the normal subgroup as far as possible but also to develop another way to find the outside classes.

With this aim, we will try to further investigate the structure of groups with a non-abelian normal subgroup to an extent that we can almost parametrize their classes.

Our strategy will be as follows. We will determine a normal series of G with elementary factors: $G = N_0 > N_1 > \cdots > N_r = \langle 1 \rangle$, $N_{i-1}/N_i \cong T_i^{d_i}$, T_i simple. Section 2 recalls how to do this. As in the case of solvable groups we then proceed down this series, determining the conjugacy classes of G/N_i from those of G/N_{i-1}. If N_{i-1}/N_i is abelian, this can be done in the same way as for solvable groups [15]. This paper is therefore concerned only with the case of a non-abelian factor N_{i-1}/N_i. To attain our goal, we examine the extension theory of such factors more closely in Section 3, showing that these extensions can be built from subdirect products. Consequentially, Section 4 describes how to obtain the conjugacy classes of a subdirect product from the conjugacy classes of its factors.

Section 5 then shows how to deal with component permutations, and Section 8 shows how to obtain remaining conjugacy classes. Finally, in Section 10 we will briefly discuss implementational issues. A list of the symbols used in the main sections (Sections 3 to 8) is provided at the end.

1.1. **Subdirect products.** The decomposition of a group as subdirect product of two of its factor groups will be crucial to the procedure. The remainder of this section is therefore devoted to a short recall of its definition:

Suppose the group G has two normal subgroups N and M with $N \cap M = \langle 1 \rangle$. This gives rise to two projections, $\alpha\colon G \to G/N =: A$ and $\beta\colon G \to G/M =: B$. It is easily checked that the mapping $G \to A \times B$, $g \mapsto (g\alpha, g\beta)$ is an injective homomorphism, and we can therefore embed G in the direct product of its factor groups A and B. If we embed G this way, α and β are the homomorphisms induced by the two component projections of $A \times B$.

Conversely, a subgroup G of $A \times B$ whose images under the component projections $\alpha\colon A \times B \to A$ and $\beta\colon A \times B \to B$ are $G\alpha = A$ and $G\beta = B$ has two normal

subgroups $\ker\alpha$ and $\ker\beta$ which intersect trivially; the factor groups of those kernels are isomorphic to A and B. The factor group $G/\langle A, B\rangle$ maps to factors of both A and B; thus A and B have an isomorphic factor group.

In this situation we say that G is an *(inner) subdirect product* of its factors A and B; we write $G = A \perp B$. If A and B are given, a subdirect product is determined up to isomorphism by an isomorphism of factor groups of A and B, and any such isomorphism gives rise to a subdirect product:

Definition 2. Let A and B be groups, and let $D \lhd A$, $E \lhd B$ be such that there is an isomorphism $\chi\colon A/D \to B/E$. Denote the natural homomorphisms by $\delta\colon A \to A/D$ and $\epsilon\colon B \to B/E$. The *(outer) subdirect product* of A and B induced by χ is the subgroup

$$\{(a, b) \in A \times B \mid a(\delta\chi) = b\epsilon\} \leq A \times B.$$

(In the language of category theory, a subdirect product is the pull-back in the category of groups.)

A natural generalization is an *iterated* subdirect product which corresponds to a subgroup of a direct product of more than two groups whose component projections are all surjective. This corresponds to a set of normal subgroups whose intersection is trivial.

2. Computing a chief series

Given a permutation group G, we first have to obtain a normal series with elementary factors: let $H \lhd G$ be a normal subgroup and suppose that a normal series of G/H is known. (Initially, we know a normal series of G/G.) Using the methods for the computation of composition series [16, 14, 1], we obtain a subgroup $U \lhd H$ such that $T := H/U$ is simple. Let

$$L := \bigcap_{g \in G} U^g;$$

then $L \lhd G$ (as intersection of a G-orbit) and $L < H$. Additionally H/L is an iterated subdirect product of groups isomorphic T; thus H/L is elementary of type T. We then proceed with L in place of H until the trivial subgroup is reached.

It takes little further work to obtain a series of normal subgroups that cannot be refined any further. (Such a series is normally called a *chief series.*) To achieve this some normal factors H/L obtained so far might have to be refined. If H/L is elementary abelian this can be done using standard MeatAxe methods [17]. If H/L is not abelian, we claim that H/L is already a chief factor:

Lemma 3. *Let T be non-abelian simple and $D = T^d$. Then every normal subgroup of D is a direct product of some of the d constituent copies of T.*

Proof. Let $\langle 1\rangle \neq N \lhd D$ be a nontrivial normal subgroup and $1 \neq n \in N$. Considering N as a direct product of d copies of T, we write $n = (t_1, \ldots, t_d)$. Without loss of generality (renumber the components) we can suppose that $t_1 \neq 1$. As T is simple non-abelian, there is an $s \in T$ such that $t_1^s \neq t_1$. Then the quotient (we write a/b as a shorthand for $a \cdot b^{-1}$)

$$n^{(s,1,\ldots,1)}/n = (t_1^s/t_1, t_2/t_2, \ldots, t_d/t_d) = (t_1^s/t_1, 1, \ldots, 1)$$

has to be contained in N. By further conjugation we get elements $(r, 1, \ldots, 1)$ with r running through the T-class of $t_1^s/t_1 \neq 1$. The subgroup of T generated by these

elements is a nontrivial normal subgroup of T; thus it has to be the whole of T. Accordingly, $T_1 \leq N$

Iterated application of this argument shows that any normal subgroup contains every one of the d direct factors isomorphic to T on which it projects non-trivially. Thus it has to be a direct product of those factors. □

Lemma 4. *The factor H/L is a chief factor.*

Proof. Applying Lemma 3 with $D = H/L$, we see that U/L has to be the direct product of all but one of the direct factors. The conjugates of U then are direct products of other $(d-1)$-combinations of the direct factors. The conjugates form one orbit; thus G has to act transitively on the d factors. Thus any normal subgroup of G that contains one of these direct factors contains all. □

An equivalent approach is mentioned in [13], constructing the series upwards instead of downwards. By modifying the composition series algorithm directly, instead of applying it as a subroutine, further improvements are possible [4].

3. Extensions with non-abelian elementary kernel

In the next sections we will describe how to compute the classes of G/N_i based on knowledge of the classes of G/N_{i-1} in the case that N_{i-1}/N is elementary non-abelian. To simplify notation, however, we will (by proceeding to the factor group G/N_i) assume that N_i is trivial and that $N = N_{i-1}/N_i$ is an elementary minimal normal subgroup. (The fact that we are computing in a factor group will not pose a problem for the actual calculations; see Remark 8.)

Let G be a finite group with $N \lhd G$, $N \cong T^n$ for a non-abelian simple group T. Then we can write the elements of N as an n-component vector $(t_1, \ldots, t_n)$ with $t_i \in T$. We consider the conjugation action φ of G on N. Its kernel is the centralizer $C = C_G(N)$. The intersection of C and N consists of those elements of N that contain in every component an element of the centre $Z(T)$. As T is non-abelian simple, $Z(T)$ is trivial; thus $Z(N) = C \cap N = \langle 1 \rangle$. Figure 1 illustrates the situation.

Accordingly, G can be considered as a subdirect product of the factor group G/N with the factor G/C. This factor is isomorphic to the image $F := G\varphi$ of G under φ, which is a subgroup of $\mathrm{Aut}(N)$.

As seen in Lemma 3, every normal subgroup of N is the direct product of some of the n copies of T. Therefore these copies themselves form a class of normal subgroups of N that is permuted by every automorphism. We denote this permutation

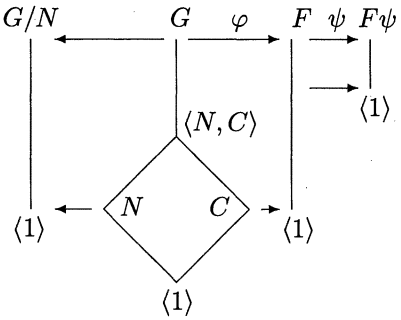

FIGURE 1. Structure of an extension with non-abelian kernel

action by ψ. An element of $\mathrm{Aut}(N)$ that fixes one component acts on this component like an element of $\mathrm{Aut}(T)$. The action of $\mathrm{Aut}(N)$ on N thus is composed from an $\mathrm{Aut}(T)$ action in the components and permutation of the components among each other. Conversely, every such composition forms an automorphism of N, so

$$\mathrm{Aut}(N) \cong \mathrm{Aut}(T) \wr S_n.$$

The base group of this wreath product is $\mathrm{Aut}(T)^n$, which in turn contains the elementary characteristic subgroup $E := T^n$. As $N \leq G$, we have in fact $E \leq F = G\varphi \leq \mathrm{Aut}(N)$. If N is a chief factor of G, by the same argument as in the proof of Lemma 4, the permutation image $F\psi$ is a transitive subgroup of S_n.

Remark 5. If $\mathrm{Out}(T) = \langle 1 \rangle$ we actually get $F = T \wr (F\psi)$. Otherwise, still, by the proof of Schreier's conjecture based on the classification of finite simple groups [10, Theorem 1.46] the index of $G\varphi$ in $\mathrm{Aut}(T) \wr (F\psi)$ will be fairly small, provided that n is not overly large.

If we take a transitive permutation representation of T that extends to $\mathrm{Aut}(T)$, we get a transitive action of $\mathrm{Aut}(T) \wr S_d$ on $\deg(T) \cdot d$ points (the natural imprimitive action). In this action, the image $G\varphi$ is a transitive subgroup of the wreath product, and its base group $M := \ker \psi$ is an iterated subdirect product of a group containing T and contained in $\mathrm{Aut}(T)$. (Groups of this type have been classified in [11].)

Every extension of a factor F with a kernel isomorphic to T^d thus is isomorphic to a subdirect product of F with a transitive imprimitive group of the described type. (One could also use the classification of those imprimitive groups together with a classification of subdirect products to get a classification of all extensions of a given factor group with elementary kernel of type T.)

4. Conjugacy classes of a subdirect product

The last section's analysis shows that subdirect products are building blocks of group extensions with non-abelian elementary kernel. Thus a description of their conjugacy classes is essential to describe the classes of extensions. In this section we consider a group M that is a subdirect product of its n projections $M\pi_i =: A_i$. We will use this procedure twice. The first time we will use it to compute the classes of the subgroup $M = \ker \psi \leq F$. Here all the A_i are isomorphic to a subgroup $A \leq \mathrm{Aut}(T)$. As we want F-classes, however, we will eventually have to modify the procedure. This will be described in Section 5.

The second use will be to compose the classes of G from the classes of G/N and F. In this situation $n = 2$ and the factor groups are usually not isomorphic.

We consider the iterated subdirect product M as a subgroup of the direct product $D = \overset{n}{\underset{i=1}{\times}} A_i$. If the conjugacy classes of the A_i are known (this will be the case either by induction or—when computing the classes of $M \leq F$—if the A_i are almost simple and so the classical random search quickly yields the classes), the classes of D are simply given by the Cartesian product of the sets of class representatives.

We will first refine these classes to the M-conjugacy classes on D and then drop those representatives which are not contained in M. This obviously yields the conjugacy classes of M. In the cases considered here, even when M is the subdirect product of more than two constituents, the index of M in D is not overly large due to Remark 5. So this approach is feasible.

We concentrate first on dealing with M-conjugacy. Let g be an element of D. As M projects surjectively on A_1, we can find $m_1 \in M$ such that $g^{m_1}\pi_1 \in A_1$ is the fixed representative of its class. Keeping this image fixed restricts further conjugation to $C_1 := (C_{A_1}(g^{m_1})\pi_1)\overline{\pi}_1^{-1}$, where the inverse $\overline{\pi}_1^{-1}$ denotes taking the full preimage. We then consider the image $g^{m_1}\pi_2$. By conjugation with $m_2 \in C_1$ we can let $(g^{m_1 m_2})\pi_2$ be a fixed representative of a $C_1\pi_2$-class in A_2. Leaving the first two projections images fixed, further conjugation is restricted to $C_2 := C_{A_2}(g^{m_1 m_2}\pi_2)(\pi_2|_{C_1}^{-1})$. We then proceed in a similar way for the other components, finally getting a "canonical" conjugate $g^{m_1 \cdots m_n}$ and its centralizer.

So, we can build M-class representatives component by component in the form $(r_1, r_2, \ldots, r_d)$, where r_i is a representative of a $\hat{C}_i := C_{i-1}\pi_i$-class on A_i. These $\hat{C}_i$-classes are refinements of the A_i-classes: if r is a representative of an A_i-class, representatives of the $\hat{C}_i$-classes therein are given by r^{s_j} with s_j running through a set of representatives of the double cosets $C_{A_i}(r)\backslash A_i/\hat{C}_i$. The appropriate centralizers of the form $C_{\hat{C}_i}(r^{s_j})$ can be obtained by intersecting the s_j-conjugate of the centralizer of r with $\hat{C}_i$. Taking the preimage under $\pi_i|_{C_{i-1}}$ then yields the next C_i.

If we want classes within M we have to drop all representatives that are not contained in M (as M is closed under M-conjugation). Whenever we encounter a partial representative $(r_1, \ldots, r_i)$ with $i \leq n$ which is not contained in the image $M(\pi_1, \ldots, \pi_i)$ of M when projected to the first i components, we can discard this representative; all its descendants will not lie in M and it thus will not give rise to any valid representative of an M-class on M.

This leads immediately to an algorithm that yields representatives of the M-classes on D.

To reduce the number of centralizer preimages to consider (and thus also the number of double cosets to be computed) we further observe that M contains n normal subgroups that act on only one component, namely

$$L_i := \bigcap_{j \neq i} \ker \pi_j.$$

They generate a direct product, the group $L := \langle L_i \mid 1 \leq i \leq n \rangle$. (In the situation of Section 3, we have for example $T^n = E \leq L$; thus L can be considered to be comparatively big.)

We can conjugate with elements of L in one component without disturbing any of the other components. Thus in our above consideration, we can replace C_i by $\langle C_i, L \rangle$ without introducing conjugation that would change previously chosen class representatives within some already considered projection images. The immediate benefit of this is not only that $\langle C_i, L \rangle$ is often bigger than C_i and thus fewer double cosets arise (and the computation will proceed much faster), but also that actually many of the groups $\langle C_i, L \rangle$ for different C_i's will coincide. This greatly reduces the number of double coset computations needed. (In the extreme case of a direct product, $\langle C_i, L \rangle \pi_{i+1}$ will be equal to A_{i+1}. Thus the double cosets degenerate to simple cosets, and our procedure just enumerates the cartesian product of class representatives.)

If we proceed in this way, of course we obtain (for example) a new C_2 as $C_2 := C_{A_2}(g^{m_1 m_2}\pi_2)(\pi_2|_{\langle C_1, L \rangle}^{-1})$. Thus the final stabilizer C_n is not the centralizer of the

representative, but the closure of the centralizer with L. But as computing images, preimages and double cosets in general is harder than computing centralizers, this loss of having to compute the centralizer separately is made up by the gain in the number of double coset computations needed.

By taking closures with L, many of the centralizers will coincide. We may take advantage of this to reduce the number of double coset computations.

The algorithm follows.

Algorithm 6. (Classes of an iterated subdirect product based on classes of the constituents) Let M, n, A_i, π_i, D and L be as defined previously. We denote the canonical embedding from A_i into D by ε_i, so we have $m = \prod_i m\pi_i\varepsilon_i$ for all $m \in M$.

Let number$_i$ be the number of classes and $\{r_{i,j}\}$ a set of class representatives of A_i. Let $C_{i,j} := C_{A_i}(r_{i,j})$. This algorithm computes a set of representatives of the conjugacy classes of M. (We denote lists by square brackets.)

1. [Initialization] Let *reps* $:= [1_M]$; *centralizers* $:= [M]$; *cent_index* $:= [1]$.
2. [Components loop] For i from 1 to n execute steps 3-17:
3. [New component] Let $\varpi := (\pi_1, \ldots, \pi_i)$; *Mproj* $:= M\varpi$ and *Lloc* $:= L\pi_i$. Let *new_reps* $:= [\,]$; *new_cent* $:= [\,]$; *new_cent_index* $:= [\,]$; *cent_images* $:= [\,]$; *cent_img_index* $:= [\,]$.
4. [Fuse centralizers that become the same in A_i] For j from 1 to $|$*centralizers*$|$ execute step 5
5. [Check whether two centralizers have the same image] Let $C :=$ *centralizers*$_j\pi_i$. If $C \in$ *cent_images* then let p be its position, otherwise add C to *cent_images* and let $p := |$*cent_images*$|$.
   In both cases add p to *cent_img_index*.
6. [Consider all previous centralizers] For j from 1 to $|$*cent_images*$|$ execute steps 7-16:
7. [Determine all representatives belonging to this centralizer image] Let $C :=$ *cent_images*$_j$. Let *select_cen* $:= \{k \mid$ *cent_img_index*$_k = j\}$ be the set of all centralizer indices with the same (enlarged) centralizer image. For k from 1 to *number*$_i$ execute steps 8-16:
8. [Double cosets] Let *dc* be a set of representatives of the double cosets $C_{i,k}\backslash A_i/C$. For all t from *select_cen* execute steps 9-14:
9. [Continue one partial representative] Let *cen* $:=$ *centralizers*$_t$. Let *select* $:= \{l \mid$ *cent_index*$_l = t\}$ be the set of indices of those representatives whose partial centralizer is *cen*.
   Let $\eta := \pi_i|_{(cen)}$, *new_cent_local* $:= [\,]$ and *new_cent_local_index* $:= [\,]$. For all d from *dc* execute for all s from *select* the steps 10-13:
10. [New representative] Let *elm* $:=$ *reps*$_s \cdot ((r_{i,k}^d)\varepsilon_i)$. If *elm* $\notin$ *Mproj* then go to step 13 as it may not lead to elements in M.
11. [Compute new centralizer] Let *newcen* $:= C\eta \cap \langle$*Lloc*$, C_{i,k}^d\rangle$. If *newcen* $\notin$ *new_cent_local* then add *newcen* to *new_cent_local* and let $p := |$*new_cent_local*$|$. Otherwise let p be the position of *newcen* in *new_cent_local*.
12. [Store the new element] Add *elm* to *new_reps* and add p to *new_cent_local_index*.
13. End the loops from step 9.
14. [Centralizer preimages] Let *shift* $:= [\,]$. For each l from 1 to $|$*new_cent_local*$|$ let P be the preimage of *new_cent_local*$_l$ under η. If $P \notin$ *new_cent* then add

P to *new_cent* and let $shift_l := |new\_cent|$. Otherwise assign the position of P in *new_cent* to $shift_l$.

15. [Move local centralizer indices to global] For each $l \in new\_cent\_local\_index$ add $shift_k$ to *new_cent_index*.
16. End the loops from step 6,7 and 8.
17. [End of component loop] Replace *centralizers* by *new_cent*, *cent_index* by *new_cent_index*, and *reps* by *new_reps*. End the loop of step 2.

When the algorithm terminates, the list *reps* contains representatives of the classes of M.

Remark 7. Of course – as seen before – the centralizer of the representatives can be computed componentwise as well by keeping true centralizers C_i' besides the C_i.

We note that in the case of a transitive action on the A_i the class representatives $r_{i,j}$ and centralizers $C_{i,j}$ only need to be computed once.

Remark 8. If M is a factor group of a larger group G (this happens when lifting the classes via subdirect products as described in Section 3) and if we don't yet have a faithful representation of M, we can instead of elements and subgroups of M work with their representatives, respectively full preimages in G. Tests for equality then must take place modulo the normal subgroup.

Remark 9. For two components, the test $elm \in Mproj$ in step 10 also can be replaced easily by checking whether $(reps_s)\pi_1$ and $(r_{i,k}^d)$ both project correctly to the common factor group that determines the subdirect product. (For more components, subdirect products become too complicated for such a test.)

Furthermore, the algorithm does not really compute in the first component, but just takes preimages of its classes/centralizers.

5. Fusion under component permutation

We now return to the situation of Section 3. Let $F \leq \mathrm{Aut}(T) \wr S_n$, with $M = \ker\psi \lhd F$. We aim to describe the conjugacy classes of F. This will be done in two steps: first we will show how to obtain representatives of the F-classes on M. This will be explained in this section.

In the second step (which will be carried out in Section 8) we will then describe how to get the remaining classes outside M.

The F-classes of elements in M are of course unions of M-classes. We therefore could simply compute the M-classes first and then fuse them under the action of F. In many cases, however, a substantial fusion takes place; this approach would first compute many M-classes and then do much work fusing them to comparatively few F-classes. We will therefore modify Algorithm 6 to construct representatives of F-classes in the first place by taking care of the component permutation induced by F.

We will suppose that we have obtained F by the action of G on a chief factor of G. So F acts transitively on the projections A_i of M, and therefore each A_i is isomorphic to a given group $A \leq \mathrm{Aut}(T)$. The action of F might further fuse classes of one A_i; this can be read off from the action of $S := (\mathrm{Stab}_{F\psi}(1))\overset{-1}{\psi}$ on the classes of A_1.

Remark 10. If M is complemented in F (for example by the Schur-Zassenhaus theorem), no such fusion might take place, because the further action of F on M is a pure component permutation.

To keep track of possible fusion, we now assign labels (which we call *colours*) to the classes of A, giving the same colour to classes if and only if they fuse under S. Furthermore we assign to every element of D a list of labels (which we call a *colour bar*) by listing the colours of the components classes:

$$(a_1, \ldots, a_n) \mapsto (\text{ColourOfClass}(a_1), \ldots, \text{ColourOfClass}(a_n)).$$

The permutation action of F will permute the colours in a bar while M stabilizes the bars. To obtain representatives of the F-classes on M it is thus sufficient to consider only such class representatives whose colour bars are from a set of representatives of the colour bars under the permuting action of $P := F\psi$. This in turn allows us to weed out all the representatives obtained in Algorithm 6 whose colour bars are not representatives.

We shall first consider the problem of computing representatives of the P-action on colour bars. In Section 7 we will then use the bar representatives to compute class representatives.

6. Action on colour bars

We observe that we can parametrize colour bars by the number of colours contained, by the frequency of the colours occurring, and finally by the contained colours themselves. (For example for $n = 5$ one class might consist of the bars containing three different colours, the first colour three times, the other colours both once; the colours being red, green and blue.) This parameterization is well suited to component permutation, because we can first compute representatives by number and frequency of the colour and can fill the actual colours in afterwards in arbitrary combinations.

If the acting group is the symmetric group this completely parameterizes the representatives, as we can sort each bar by "brightness" (i.e. an arbitrary total ordering) of the colours and the sorting permutation is contained in the symmetric group.

To deduce P-representatives from this we observe that if c is a colour bar and $S = \text{Stab}_{S_n}(c)$, all bars containing the same colours and the same frequencies are in bijection to the right cosets $S\backslash S_n$, as they form just one S_n-class. (Note that S is just a direct product of symmetric groups.) The P-orbits are in correspondence with the double cosets $S\backslash S_n/P$; we can obtain representatives by mapping c with a set of representatives for the double cosets.

This leads to the following algorithm:

Algorithm 11 (Representatives of colour bars). Let P be a permutation group on n points and *colours* a list of colours. This algorithm computes a set of representatives of the colour bars under the permuting action of P.

1. [Initialization] Let *allowed_colourbars* := [].
2. [Loop over the number of different colours in the bar] For *number* from 1 to min(n, *number*) execute steps 3–6 (we construct bars with exactly *number* colours).

3. [Colour combinations] Let *colour_comb* be a set of the $\binom{|colours|}{number}$ combinations of *number* entries from *colours*. For *pattern* running through a set of the ordered partitions of n with *number* cells execute steps 4–6 (the *pattern* indicates how often which colour will occur).
4. [Compute pattern stabilizer and double cosets] Let

$$expanded\_pattern := [\underbrace{1,\dots,1}_{pattern_1},\underbrace{2,\dots,2}_{pattern_2},\dots,\underbrace{number,\dots,number}_{pattern_{number}}].$$

Let S be the stabilizer in S_n of *expanded_pattern*. (This is the stabilizer of all bars that arrange their colours according to this pattern. Its computation is trivial as it is just a direct product of symmetric groups.)

Let *reps* be a set of representatives of the double cosets $S\backslash S_n/P$. For each $r \in reps$ execute step 5:

5. [Consider further permutations] Let *permuted_pattern* be the image of *expanded_pattern* under component permutation by r. For each *comb* from *colour_comb* add the bar

$$[comb(permuted\_pattern_1),\dots,comb(permuted\_pattern_n)]$$

to *allowed_colourbars*.

6. End the loops.

At this point, *allowed_colourbars* is a set of representatives.

We remark that in general n (and thus P) will be very small. Therefore this algorithm's runtime will be negligible in the overall cost and the potentially exponential algorithm need not be worried about.

The stabilizers of the colour bars in P are obtainable in step 5 as $S^r \cap P$.

Definition 12. We call those bars *unlucky* whose stabilizers are actually larger than M.

7. Representatives of the inner classes

In this section we will use the representatives of the colour bars to obtain representatives of the classes of M under F.

We assume that we have computed the fusion of A-classes under the action of $S := (\mathrm{Stab}_{F\psi}(1))^{\psi^{-1}}$ and thus obtained a list of colours and assignment of colours to classes. We also assume knowledge of a set of representatives of the colour bars under action of $P = F\psi$.

When constructing representatives of F-classes, representative tuples from M with a colour bar not in the list of bar representatives may be discarded. (There is an image under F whose colour bar is in the list; we will also construct such a representative. As M fixes all bars, this does not interfere with any labelling scheme for M-representatives.)

Thus in Algorithm 6 for each partial representative tuple $(r_1,\dots,r_{i-1})$ only representatives of those classes need to be added in the i-th position, whose colour is equal to an i-th colour in a bar starting with $\mathrm{Colourbar}(r_1,\dots,r_{i-1})$. Thus by attaching to each (partial) representative its (partial) colour bar we can simply restrict the number of representatives constructed while still covering all F-classes on M. The test for the permitted colours can take place in step 8 of Algorithm 6,

as at this point the colour bar of the image is already determined. We modify Algorithm 6 as follows:

Algorithm 13 (Classes of a subdirect product under further component fusion). We assume that *ColourOfClass* gives the colour for a class and that the list *allowed_colourbars* is the result from Algorithm 11. Furthermore we will keep a list *colourbar* to give the (partial) colourbars of the representatives.

3'. [New component] Let $\varpi := (\pi_1, \ldots, \pi_i)$; $Mproj := M\varpi$ and $Lloc := L\pi_i$. Let *new_reps* := []; *new_cent* := []; *new_cent_index* := []; *cent_images* := []; *cent_img_index* := []. Let *new_colourbar* := [].

7'. [Determine all representatives belonging to this centralizer image] Let $C := cent\_images_j$. Let $select\_cen := \{k \mid cent\_img\_index_k = j\}$ be the set of all centralizer indices with the same (enlarged) centralizer image. Let $select := \{k \mid cent\_index_k \in select\_cen\}$ be the set of indices of those representatives that might be extended here.

7a. [Determine the addable colours] Let *possible_colours* := { }. For $k \in select$ let $bar := colourbar_k$. Let

$$potential\_bars := \{b \in allowed\_colourbars \mid b_{[1,\ldots,i-1]} = bar\}$$

and let

$$possible\_colours := possible\_colours \cup \{b_i \mid b \in potential\_bars\}.$$

7b. [Run through the classes with correct colours] For those k from 1 to $number_i$ whose colour is in *possible_colours* execute steps 8-16.

9'. [Continue one partial representative] Let $cen := centralizers_t$. Let $select := \{l \mid cent\_index_l = t\}$ be the set of indices of those representatives whose partial centralizer is *cen*.

Let $\eta := \pi_i|_{(cen)}$, *new_cent_local* := [] and *new_cent_local_index* := []. For all d from dc execute for all s from *select* the steps 9a-13:

9a. [Test whether this colour may be added here]

Let $bar := colourbar_s \cup [ColourOfClass(k)]$.

If there is no bar $c \in allowed\_colourbars$ for which $c_{[1,\ldots,i]} = bar$ then go to step 13.

12'. [Store the new element] Add *elm* to *new_reps*, *bar* to *new_colourbar* and add p to *new_cent_local_index*.

17'. [End of component loop] Replace *cent_index* by *new_cent_index*, *colourbar* by *new_colourbar*, and *reps* by *new_reps*. End the loop of step 2.

Finally, among the representatives obtained by the modified algorithm, some further fusion by F may take place, but only among those representatives with the same colour bars. Two representatives $a, b \in M$ with the same colour bar c that are conjugate in F must be conjugate already in $\mathrm{Stab}_F(c)$. So conjugacy tests actually only need to be performed among those representatives with unlucky colour bars. In this situation the conjugacy test takes place in $\mathrm{Stab}_F(c)$, which is often a smaller group than F:

18. [Prepare for further F-fusion] Let *new_reps* := []; For every $bar \in colourbar$ execute step 19

19. [Fuse among classes with colour bar *bar*] Let *cand* be the set of those representatives in *reps* which have colour bar *bar*. Using conjugacy tests, compute

a set *fuse* of $\mathrm{Stab}_F(bar)$-representatives among the elements in *cand*. Append *fuse* to *new_reps*.

At the end of this loop *new_reps* contains representatives of the F-classes of elements in M.

8. Obtaining the outer classes

We now can compute the F-classes of elements in M. What remains to be done is the classes of elements of F not contained in M, the *outer classes*.

In contrast to the classes within M, F usually has very few outer classes (see the examples in Section 11). Some of these outer classes can be relatively small, however, and so a pure random search is not feasible.

To obtain these outer classes, we recall the general idea of [15]: Let $P := F/M$, and $\psi\colon F \to P$ the natural homomorphism. Let r be the representative of a class of P. Then representatives of those classes of F that map under ψ into the class of $r\psi$ can be found as representatives of the $Z := (C_P(r\psi))\psi^{-1}$-orbits on the coset rM. As for $z \in Z$ we have

$$(rm)^z = r^z m^z = r[r,z]m^z$$

with $[r,z] \in M$, this induces an action on M by

$$m \mapsto [r,z]m^z.$$

Here, in general, however, M is too big to pursue this actual action.

As in Section 4, we will now first restrict the conjugating group acting on D from Z to $M \leq Z$. As we may embed F in $\mathrm{Aut}(T) \wr S_n$, the action of M on M is a restriction of the action of M on D given by

$$d \mapsto [r,m]d^m. \tag{14}$$

Let $M = (m_1, \ldots, m_n) \in M$, $d = (a_1, \ldots, a_n) \in D$, $[r,m] =: ([r,m]_1, \ldots, [r,m]_n)$, the cartesian decomposition. Then the action (14) becomes

$$d \mapsto [r,m]d^m = ([r,m]_1 a_1^{m_1}, \ldots, [r,m]_n a_n^{m_n}).$$

This action thus is the composition of the action of the components A_i by

$$a_i \mapsto [r,m]_i a_i^{m_i}. \tag{15}$$

Now, to obtain representatives of the M-orbits on M under this action, we again proceed componentwise: we first compute the M-orbits on A_1. Then for each representative r_1 we compute the $\mathrm{Stab}_M(r_1)$ orbits on A_2, and so on. We thus obtain orbit representatives in the form $(r_1, \ldots, r_n)$. As above, to obtain the orbits on M we have to discard those representatives that are not contained in M (as r normalizes M, the action (14) fixes M set-wise), and we can simply discard all partial representatives $(r_1, \ldots, r_i)$ which are not contained in the projection image $M(\pi_1, \ldots, \pi_i)$.

As long as the size of the A_i is small we can simply run a standard orbit/stabilizer algorithm. For a large A_i (as in the example in the introduction for which $A_i \cong S_{11}$), however, this is not possible as we cannot store all elements. A standard technique in such cases is the use of *tadpoles* [7]. Let U be a subgroup of the acting group. Whenever elements are encountered a "standard" representative under action of U is computed (the iterative process usually used here gives rise

to the name "tadpole") and only this representative is stored. This permits us to reduce the number of elements to store roughly by $|U|$.

In our situation we will take U to be a subgroup that centralizes r. (Any larger group would change r and thus leave the operations domain.) Then for U the action (15) reduces to a simple conjugation action, and we can take the "standard" U-representatives to be computed representatives of the U-classes on A_i. (Again, as above, these classes can be computed by conjugating class representatives a with representatives of the double cosets $C_{A_i}(a)\backslash A_i/U$.) This permits us to compute the U-classes a priori and to simplify the orbit algorithm to fusion of the U-orbits. Section 9 explains how to proceed to obtain representatives of the M-orbits for these.

To get the largest possible subgroup we take U in each step (when computing possible representatives in the $(i+1)$-th component with entries $r_1, \ldots, r_i$ in the prior components) to be the intersection of the centralizer $C_F(r)$ of r with the acting group $\mathrm{Stab}_M(r_1, \ldots, r_i)$. Note that we have $|M|$ choices for r. To minimize the number of U-orbits, we try to maximize $C_F(r)$, which is fulfilled if the order of r is minimal possible. Let o be the order of $r\psi$. Then we need $r^o = ()$, which means that $\langle r \rangle$ intersects trivially with M.

Lemma 16. *For each cycle in the action of Mr on the components pick one component index. Let I be the set of these indices. If there is an $m \in M$ with $m\pi_i = (r^o)\pi_i$ for all $i \in I$ and $m\pi_j = ()$ for all $j \notin I$, then $\hat{r} := r/m$ is an element with $\hat{r}\psi = r\psi$ and $\hat{r}^o = ()$.*

Proof. By considering the orbits on the components separately, we can assume without loss of generality that r acts transitively and $I = \{1\}$. Let j be a point from the first component. Then $j^{(r^l)}$ $(l < o)$ is in a component on which m does act trivially. Accordingly

$$j^{(\hat{r}^o)} = j^{(rm^{-1}r\cdots rm^{-1})} = j^{(r^o/m)} = j$$

by the definition of m, as j is in the first component. So $\hat{r}^o$ acts trivially on the first component. As every point k is the image $k = j^{(\hat{r}^l)}$ $(l < o)$ for a j in the first component, we have similarly

$$k^{(\hat{r}^o)} = (j^{(\hat{r}^l)})^{(\hat{r}^o)} = j^{(\hat{r}^{l+o})} = (j^{(\hat{r}^o)})^{(\hat{r}^l)} = j^{(\hat{r}^l)} = k,$$

and thus $\hat{r}^o = ()$. □

If no suitable element of M can be found, we simply select an r with a large centralizer.

To continue in the next component we also need the stabilizer in U of the orbit representative (which will be the intersection of U with the stabilizer mentioned above). This stabilizer is simply (the full preimage of) the centralizer in U.

Remark 17. If U is not normal in G and several orbits of U get fused, however, these centralizers can be of different sizes. Therefore we sort the orbits of U in ascending size and pick always the smallest one as representative for starting the G-orbit. This allows us to continue in the next component with as large a U as possible.

Finally, we have to fuse the M-orbits to Z-orbits. As by Remark 5 $[F : M]$ is small, also $[Z : M]$ is small and thus not much fusion will take place. We also

observe that in general M is already quite big and has large orbits; thus only few classes have to be fused. Therefore this fusion test is done in a straightforward way by testing conjugacy of the elements obtained so far. (Obviously we can restrict this fusion test to elements with the same cycle structure and centralizer orders.)

This finishes the description of the non-abelian lifting step which computes the classes of G, provided the classes of G/N are known. To summarize: Assigning $F := G/C_G(N)$, we first compute the F-classes of M. This is done by Algorithm 13 and an additional fusion among representatives with the same unlucky colour bars. Afterwards we compute the F-classes outside M as in Section 8. Finally, we combine the classes of G/N and those of $F = G/C$ to classes of $G = (G/N) \leftthreetimes F$ using Algorithm 6. We then proceed inductively for the next chief factor.

9. An orbit-fusing algorithm

(This section uses its own notation to simplify variable names.) Let G be a (finite) group acting on Ω via the operation $\mu\colon \Omega \times G \to \Omega$. Furthermore let $U \leq G$ and $\nu := \mu|_U$ be the restriction of the operation. We assume that we know representatives $\{\omega_i\}$ of the U-orbits on Ω and their stabilizers, that we can find, for any $\delta \in \Omega$, in which U-Orbit it lies, and that we can compute $u \in U$ with $\delta^u = \omega_i$ for the suitable i. We want to obtain representatives and stabilizers of the G-orbits. As we already know the U-orbits, we use a slightly modified orbit algorithm that will always add a full U-orbit. However we want to avoid mapping all elements of Ω with generators of G but try to restrict ourselves (as far as possible, impossible in general) to mapping only the ω_i: we store which U-orbits ω_i^U are contained in the G-orbit. If we have mapped all ω_i with all generators of G and the product of orbit length (the sum of the $|\omega_i^U|$) and stabilizer size is smaller than $|G|$, we did not yet reach the full orbit or full stabilizer. In this situation we start mapping other points ω_i^u $(u \in U)$ until this condition is fulfilled.

Algorithm 18. (Fuse U-orbits under the action of $G \geq U$)

1. [Initialization] Let $orb := [\omega_1]$, $stab := \mathrm{Stab}_U(\omega_1)$ and $trans := [()]$, let $iterate := false$ and $pos := 1$
2. [Orbit loop] Let $\omega := orb[pos]$. For every generator gen of G execute steps 3-7.
3. [Pick up further elements to map] If $iterate$ then let u be a random element of U and set $gen := u \cdot gen$.
4. [Compute image and its canonical U-representative] Compute $img := \mu(\omega, gen)$. Let j be the index for which ω_j is in the U-orbit of img. Compute $v \in U$ with $\nu(\omega_1, v) = \omega_j$ and let $gen := gen \cdot v$. Let $transgen := trans[pos] \cdot gen$. (Now $transgen$ maps ω_1 to ω_j.)
5. [Did we extend the orbit?] If $\omega_j \in orb$ then go to step 6. Otherwise add ω_j to orb and add $transgen$ to $trans$. For each generator s of $\mathrm{Stab}_U(\omega_j)$ let $stab := \langle stab, transgen \cdot s \cdot transgen^{-1} \rangle$ Go to step 7.
6. [Extend stabilizer] Assume that $orb[i] = \omega_j$. Let $stab := \langle stab, transgen \cdot trans[i]^{-1} \rangle$
7. End the gen-loop of step 2
8. [New orbit index] Increment pos. If $pos > |orb|$ then set $pos := 1$ and $iterate := true$.
9. [Test for full orbit/stabilizer] If $|stab| \cdot \sum_{\omega \in orb} |\omega^U| \lneqq |G|$ then go to step 2.

When the algorithm terminates, *orb* contains those ω_i whose U-orbits form the G-orbit and $stab = \mathrm{Stab}_G(\omega_1)$. Iterated application gives all G-orbits.

Proof. Observe that the elements added in steps 5 and 6 all stabilize ω_1. By the orbit theorem the algorithm will terminate if we have found the full orbit and full stabilizer. By iterating we will finally (if the random function is really random) have mapped all orbit elements, so to show correctness it is sufficient to prove that the stabilizer generators added in step 6 finally will generate all Schreier generators of the stabilizer.

This is seen easily if we define the transversal $\mathcal{T}$ (which gives rise to all these Schreier generators) based on the computed transversal *trans*. That is, every element of $\mathcal{T}$ is of the form tu with $t \in trans$ and $u \in U$. The random element u in step 3 will finally give rise to all elements of $\mathcal{T}$.

Suppose that we map $t \cdot u \in \mathcal{T}$ with the generator *gen* and that the transversal element for the image is $t'u'$. (We have $t = trans[pos]$ and $t' = trans[i]$.) This gives rise to the Schreier generator $s := t \cdot u \cdot gen \cdot (u')^{-1}(t')^{-1}$. On the other hand, the element v computed in step 4 will have $t \cdot u \cdot gen \cdot v$ mapping the representative to an image corresponding to the transversal element t'. In step 6 then we add the element $s' := t \cdot u \cdot gen \cdot v \cdot (t')^{-1}$ to the stabilizer. As $u'v \in \mathrm{Stab}_U(\omega_1^{t'})$ and thus $v^{-1}(u')^{-1} = s_1 \cdot \cdots \cdot s_l$ for suitable generators s_i of $\mathrm{Stab}_U(\omega_1^{t'})$, we have that

$$s = t \cdot u \cdot gen(v \cdot v^{-1})(u')^{-1} \cdot (t')^{-1} = t \cdot u \cdot gen \cdot v((t')^{-1}t')(v^{-1} \cdot (u')^{-1})(t')^{-1}$$

$$= (t \cdot u \cdot gen \cdot v \cdot (t')^{-1})t'(s_1 \cdot \cdots \cdot s_l)(t')^{-1} = s' \cdot \prod_{i=1}^{l} s_i^{(t')^{-1}}.$$

The $s_i^{(t')^{-1}}$ have been added already in step 5. So we have finally $s \in stab$, which is what had to be proved. □

When picking the random element in step 3 it is sufficient to pick a u which is not in $U \cap U^{(gen^{-1})}$, because otherwise $u \cdot gen = gen \cdot \tilde{u}$ with $\tilde{u} \in U$ and so the image will not lie in a different U-orbit. If no such elements exist, the generator can be ignored when iterating.

Remark 19. As we know the size of the acting group and the orbit lengths, we may sometimes complete orbits without having computed all images: the length of the orbit must divide the order of the acting group. Suppose we have computed a partial orbit *orb* of length $l = \sum_{\omega_i \in orb} |\omega_i^U|$ and stabilizer subgroup *stab*. If for the smallest divisor d of $|G|$ which is not smaller than l ($d = \min\{e \mid |G| \mid e \geq d\}$) we have that $d \cdot |stab| = |G|$, we know that *stab* is the stabilizer and that the full orbit is of length d. If there is only one possibility to add up the sizes of remaining U-orbits to obtain $d - l$ these orbits have to be added to *orb* to obtain the full G-orbit. This can be particularly helpful if one small U-orbit is missing.

Usually the algorithm will finish with none or very few iterations and terminate much quicker than when running through all U-images $\nu(\omega_i, u)$ in a more orderly way.

If some of the U-orbits are small in average, it is relatively unlikely that they will be hit. Therefore it is preferable to pick ω_1 to be the representative with the smallest U-orbit. As a side effect, this permits one to start with the largest possible stabilizer and is consistent with the choice needed by Remark 17.

10. Implementational issues

The whole algorithm now stands as described in the introduction. We compute a series of normal subgroups with elementary factors: $G = N_0 > N_1 > \cdots > N_r = \langle 1 \rangle$, and inductively compute the classes of G/N_i from those of G/N_{i-1}. For an abelian N_{i-1}/N_i the algorithm of [15] is used, and in the non-abelian case the method described in Sections 3 to 8 is used.

10.1. **Factor representations.** The algorithm will require us to compute in factor groups. Following Remark 8, however, it is often not necessary to construct a concrete representation of the factor groups, but computation can proceed using coset representatives of full preimages of subgroups.

For the factor groups for which we need to compute a representation, however, small faithful permutation representations can easily be found:

The factor groups $F = G\varphi$ are obtained in Section 3 by action on chief factors. As shown there, F can be embedded into $\mathrm{Aut}(T) \wr S_n$. Using a faithful permutation representation for $\mathrm{Aut}(T)$ (such a representation is of relatively small degree, as T is simple non-abelian), we obtain a moderate degree permutation representation for the wreath product and thus for the factor F.

We now consider the lifting situation of Section 3 (G being a subdirect product of G/N and F) with G being a factor group of a (probably larger) group H. We thus want to defer computations in G to computations in H. As we can find a good permutation representation for F, the full preimage of $C = C_G(N)$ in H is just the kernel of the homomorphism $H \to F$, and is computable as such. We also may assume by induction that we already have class representatives and full preimages in H of the centralizers in G/N. Then (by Remark 8) we can use these preimages in Algorithm 6 and obtain representatives in H of class representatives for G. Similarly we obtain (preimages in H of) the centralizers in G by Remark 7 from the centralizers in G/N and F. Furthermore we may assume F to be the second component of the subdirect product. Then by Remark 9 all computations for the algorithm will take place in F, which has a good permutation representation.

This yields representatives for the classes and preimages of the centralizers, required inductively if G becomes a $\hat{G}/\hat{N}$ in the next step.

For abelian chief factors, the methods from [15] mainly require us to compute the affine action on the vector space. This can be computed without a factor representation by identifying cosets.

10.2. **Choice of the normal series.** The choice of the normal series can affect the algorithm's performance. In general the series should be chosen to have in each non-abelian step as small as possible a common factor group in the subdirect product of G/N with F. (The best case would be a direct product that is factor size 1, for which the classes are simply obtained as a cartesian product.)

Conversely, it might happen (especially in the first steps of the algorithm) that $C = 1$ and thus G/N is isomorphic to a factor of F. In this situation, of course, the classes of G are just the classes of F and G/N, and the fusion step is discarded.

11. Examples

The described algorithm has been implemented by the author in GAP 4 [9]. We give some results of this experimental implementation in Table 1. All runtimes are seconds on a 200MHz Pentium Pro machine running under Linux. The nonstandard

TABLE 1. Results of conjugacy class computations

Group	Size	deg	C	time_{std}	time
$2^{1+4+6}.A_8 \quad (< Co_2)$	$2^{17}3^2 \cdot 5 \cdot 7$	240	111	287	122
$2^{5+8}.(A_6 \times S_3) \quad (< Fi_{22})$	$2^{17}3^3 \cdot 5$	106	1920	401	355
$[\frac{1}{4}S_5^4]V_4 \underset{V_4}{\perp} A_6.V_4$	$2^{15}3^65^5$	30	1591	1161	708
$\frac{1}{2}[PGL_2(9)^3]S_3$(impr.)	$2^{12}3^75^3$	30	218	86	139
$\frac{1}{2}[PGL_2(9)^3]S_3$(prd.)	$2^{12}3^75^3$	1000	218	194	180
$\frac{1}{2}[S_5^3]S_3 \underset{S_3}{\perp} \frac{1}{2}[S_5^3]S_3$	$2^{17}3^75^6$	30	6115	—	2458
$M_{12}.2 \underset{C_2}{\perp} A_6.V_4 \underset{C_2}{\perp} S_9$	$2^{17}3^95^37 \cdot 11$	43	2286	2396	1177

Column "deg" gives the degree of the permutation representation used, "C" gives the total number of classes, column "time_{std}" is the runtime for the standard conjugacy class algorithm following [3, 20] as implemented in GAP 4, and column "time" the runtime for the author's implementation. A "—" indicates that the computation did not finish in three times the time used by the other algorithm.

TABLE 2. Results for subgroups of wreath products

Group F	$\lvert F\rvert$	deg	$[F{:}M]$	C	Out	part	time_{std}	time
$[\frac{1}{2}S_{11}^2]2$	$2^{16}3^85^47^211^2$	22	2	874	58	$2.5 \cdot 10^{-8}$	1061	363
$\frac{1}{2}[S_{11}^2]2$	$2^{16}3^85^47^211^2$	22	2	868	54	$1.3 \cdot 10^{-6}$	1026	944
$\frac{1}{2}[S_{12}^2]2$	$2^{20}3^{10}5^47^211^2$	24	2	1600	74	$1.3 \cdot 10^{-7}$	4248	2914
$\frac{1}{2}[A_5^4.2]S_4$	$2^{11}3^55^4$	20	24	118	68	$3.6 \cdot 10^{-5}$	66	70
$\frac{1}{2}[(L_7.2)^3]S_3$	$2^{12}3^47^3$	24	6	135	50	$2.9 \cdot 10^{-4}$	64	66
$[\frac{1}{2}S_6^4]D(4)$	$2^{18}3^85^4$	24	8	1645	513	$1.5 \cdot 10^{-9}$	2282	1158
$A_7 \wr C_4$	$2^{14}3^85^47^4$	28	4	1728	63	$5.2 \cdot 10^{-8}$	1330	428
$\frac{1}{2}[(L_9:3)^3]\mathcal{C}_3$	$2^63^45^311^3$	36	3	192	16	$7.5 \cdot 10^{-4}$	37	25
$[PSL_2(11)^3:2]C_3$	$2^73^45^311^3$	36	3	228	26	$3.7 \cdot 10^{-4}$	49	80

In addition to the data given in Table 1, column "Out" gives the number of classes outside M, and "part" the ratio of the smallest outer class within all outer classes.

names refer to constructions as explained in [6]. If solvable normal subgroups exist, the use of homomorphic images always improves the performance. The table gives only one example of this kind, as the behaviour of other such groups does not vary much from the given one and as the method used has been well-known for a couple of years; extensive runtime studies are given in [5]. Therefore all other examples selected do not have a solvable normal subgroup.

The group $\frac{1}{2}[PGL_2(9)^3]S_3$ is the same group isomorphism type given in two different permutation representations (imprimitive and product action). While the standard algorithm depends substantially on the degree of the representation, the new algorithm (which performs all hard calculations in the factor group F for which a small degree permutation representation is computed) is fairly independent of it.

The building blocks of the new part of the algorithm are the subgroups of wreath products, described as F in Section 3. We list a few extra results for a random selection of them in Table 2. Observe that, as promised, the number of outer classes usually is small compared with the total number of classes, but some classes are very

small, so a random search for outer classes as proposed in [5] will not necessarily work. We also see that these groups usually have very many classes in total, so even a modified random search along the lines of [12] is therefore not suitable.

A notable exception among the otherwise satisfactory runtimes is $\frac{1}{2}[S_{11}^2]2$, for which the new algorithm yields only a minimal improvement. In this group (the wreath product $S_{11} \wr 2$ has 3 normal subgroups of index 2 namely $[\frac{1}{2}S_{11}^2]2$, $\frac{1}{2}[S_{11}^2]2$ and S_{11}^2; see [6] for explanation of the names) the block action C_2 is not "bodily", that is, the group does not split over its normal subgroup $S_{11}^2.2$. Consequently, the centralizer $C_F(r)$ is relatively small and thus has many small orbits which are hard to fuse. However the loss in comparison to the standard algorithm is only due to the small size, as the similarly built $\frac{1}{2}[S_{12}^2]2$ shows.

In general, we deduce from comparison with the standard algorithm that the group must be large enough to make the new algorithm worth using; otherwise the setup of the series and the subdirect products takes too long.

On the other hand, the algorithm is capable of dealing with groups which the standard algorithm is incapable of, and may become useful in these cases.

12. Closing remarks

The algorithm presented in this note does not rely on G being a permutation group. All parts of the calculations which depend on the representation are kept in routines called (mainly: composition series, Double cosets and Homomorphisms). As soon as such routines are provided for other types of group representations, the algorithm becomes available there.

The parametrization of classes used here may also be used to run through the classes of a large group without storing all class representatives at once.

The author would like to thank Steve Linton for helpful comments on a first draft.

13. List of symbols

A_i constituent groups of M
C $C_G(N)$
C_i pre-image in M of centralizer in $M\varpi_i$
D direct product $\overset{n}{\underset{i=1}{\times}} A_i$
d element of D
E image $N\varphi \leq F$
F $G\varphi \cong G/C$
G Group for which we want to compute the conjugacy classes
L_i $= \bigcap_{j\neq i} \ker \pi_i < M$, subgroup acting on one component only
L $= \langle L_i \rangle$
M subdirect product, for example $\ker \psi$
m element of M
N elementary (minimal) normal subgroup of G
n T-"dimension" of N
P $F/M \cong F\psi$
r representative in F of P-class
S component stabilizer $(\mathrm{Stab}_{F\psi}(1))\overset{-1}{\psi}$
T simple group, $N \cong T^n$
U subgroup that centralizes r
Z full centralizer preimage $(C_P(r\psi))\overset{-1}{\psi}$
φ $: G \to F$, action of G on N
π_i $: M \to A_i$, component projection
ϖ_i $= (\pi_1, \ldots, \pi_i)$, projection on first i components
ψ $: F \to P$, action on the components of N.
$\overset{-1}{\psi}$ full preimage under ψ

REFERENCES

[1] Robert Beals and Ákos Seress, *Structure forest and composition factors for small base groups in nearly linear time*, Proceedings of the 24th ACM Symposium on Theory of Computing, ACM Press, 1992, pp. 116–125.

[2] Gregory Butler, *Computing in permutation and matrix groups II: Backtrack algorithms*, Math. Comp. **39** (1982), no. 160, 671–670. MR **83k:**20004b

[3] Greg[ory] Butler, *An inductive scheme for computing conjugacy classes in permutation groups*, Math. Comp. **62** (1994), no. 205, 363–383. MR **94c:**20011

[4] John Cannon and Derek Holt, *Computing chief series, composition series and socles in large permutation groups*, J. Symbolic Comput. **24** (1997), 285–301. MR **98m:**20009

[5] John Cannon and Bernd Souvignier, *On the computation of conjugacy classes in permutation groups*, Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (Wolfgang Küchlin, ed.), The Association for Computing Machinery, ACM Press, 1997, pp. 392–399.

[6] John H. Conway, Alexander Hulpke, and John McKay, *On transitive permutation groups*, LMS J. Comput. Math. **1** (1998), 1–8. (electronic) CMP 98:15

[7] Gene Cooperman and Michael Tselman, *Using tadpoles to reduce memory and communication requirements for exhaustive, breadth-first search using distributed computers*, Proceedings of ACM Symposium on Parallel Architectures and Algorithms (SPAA-97), ACM Press, 1997, pp. 231–238.

[8] Volkmar Felsch and Joachim Neubüser, *An algorithm for the computation of conjugacy classes and centralizers in p-groups*, Symbolic and Algebraic Computation (Proceedings of EUROSAM 79, An International Symposium on Symbolic and Algebraic Manipulation, Marseille, 1979) (Edward W. Ng, ed.), Lecture Notes in Computer Science, 72, Springer, Heidelberg, 1979, pp. 452–465. MR **82d:**20003

[9] The GAP Group, School of Mathematical and Computational Sciences, University of St Andrews, and Lehrstuhl D für Mathematik, RWTH Aachen, GAP – *Groups, Algorithms, and Programming, Version 4*, 1998.

[10] Daniel Gorenstein, *Finite simple groups*, Plenum Press, 1982. MR **84j:**20002

[11] Alexander Hulpke, *Konstruktion transitiver Permutationsgruppen*, Ph.D. thesis, Rheinisch-Westfälische Technische Hochschule, Aachen, Germany, 1996.

[12] Mark Jerrum, *Computational Polya theory*, Surveys in Combinatorics, 1995 (Stirling) (Peter Rowlinson, ed.), London Mathematical Society Lecture Note Series, vol. 218, Cambridge University Press, 1995, pp. 103–118. MR **96h:**05012

[13] William M. Kantor and Eugene M. Luks, *Computing in quotient groups*, Proceedings of the 22nd ACM Symposium on Theory of Computing, Baltimore, ACM Press, 1990, pp. 524–563.

[14] Eugene M. Luks, *Computing the composition factors of a permutation group in polynomial time*, Combinatorica **7** (1987), 87–89. MR **89c:**20007

[15] M[atthias] Mecky and J[oachim] Neubüser, *Some remarks on the computation of conjugacy classes of soluble groups*, Bull. Austral. Math. Soc. **40** (1989), no. 2, 281–292. MR **90i:**20001

[16] Peter M. Neumann, *Some algorithms for computing with finite permutation groups*, Groups – St Andrews 1985 (Edmund F. Robertson and Colin M. Campbell, eds.), Cambridge University Press, 1986, pp. 59–92. MR **89b:**20004

[17] Richard Parker, *The Computer Calculation of Modular Characters (the* MeatAxe*)*, Computational Group Theory (Michael D. Atkinson, ed.), Academic Press, 1984, pp. 267–274. MR **85k:**20041

[18] Charles C. Sims, *Determining the conjugacy classes of a permutation group*, Computers in Algebra and Number Theory (Garrett Birkhoff and Marshall Hall Jr., eds.), SIAM-AMS Proc., vol. 4, American Mathematical Society, Providence, RI, 1971, pp. 191–195. MR **49:**2901

[19] ———, *Computing the order of a solvable permutation group*, J. Symbolic Comput. **9** (1990), 699–705. MR **91m:**20011

[20] Heiko Theißen, *Methoden zur Bestimmung der rationalen Konjugiertheit in endlichen Gruppen*, Diplomarbeit, Lehrstuhl D für Mathematik, Rheinisch-Westfälische Technische Hochschule, Aachen, 1993.

SCHOOL OF MATHEMATICAL AND COMPUTATIONAL SCIENCES, UNIVERSITY OF ST. ANDREWS, THE NORTH HAUGH, UK-ST ANDREWS, FIFE KY16 9SS, SCOTLAND

E-mail address: ahulpke@dcs.st-and.ac.uk