CONSTRUCTING AUTOMORPHISM GROUPS OF p-GROUPS

Bettina Eick
Institut für Geometrie
Universität Braunschweig
Pockelsstrasse 14
38106 Braunschweig
Germany

beick@tu-bs.de

C.R. Leedham-Green
School of Mathematical Sciences
Queen Mary
University of London
London E1 4NS
United Kingdom

C.R.Leedham-Green@qmw.ac.uk

E.A. O'Brien
Department of Mathematics
University of Auckland
Private Bag 92019
Auckland
New Zealand

obrien@math.auckland.ac.nz

Abstract

We present an algorithm to construct the automorphism group of a finite p-group. The method works down the lower exponent-p central series of the group. The central difficulty in each inductive step is a stabiliser computation; we introduce various approaches designed to simplify this computation.

2001 *Mathematics Subject Classification.* Primary 20D15

1 Introduction

Given an arbitrary finite group, the computation of its automorphism group is a very difficult task. Pioneer work in this area was carried out by Felsch & Neubüser ([8] and [9]), whose algorithm used the output of their subgroup lattice program. In the early 1970s, Neubüser developed a technique to determine the automorphism group by considering its action on unions of certain conjugacy classes of the group; similar methods were used by Hulpke [14]. Cannon & Holt [5] present a new algorithm to answer this problem.

More efficient approaches are available to determine the automorphism groups of groups satisfying certain properties. Following the work of Shoda [27], Hulpke in 1997 implemented a practical method for finite abelian groups. Wursthorn [31] adapted modular group algebra techniques to compute the automorphism group of a p-group. Smith [30] introduced an algorithm for finite soluble groups; it has recently been further developed by Slattery.

The p-group generation algorithm of Newman [19] and O'Brien [21] can be modified to compute the automorphism group of a finite p-group as outlined in [24]. The algorithm proceeds by induction down the lower exponent-p central series of a given p-group P; that is, it successively computes $Aut(P_i)$ for the quotients $P_i = P/\mathcal{P}_i(P)$, where $(\mathcal{P}_i(P))$ is the descending sequence of subgroups defined recursively by $\mathcal{P}_1(P) = P$ and $\mathcal{P}_{i+1}(P) = [\mathcal{P}_i(P), P]\mathcal{P}_i(P)^p$ for $i \geq 1$. The exponent-p class of P is the length of its lower exponent-p central series.

The initial step of the algorithm returns $Aut(P_2) \cong \mathrm{GL}(d, p)$, where d is the rank of the elementary abelian group P_2. In the inductive step we determine $Aut(P_{i+1})$ from $Aut(P_i)$. For this purpose we introduce an action of $Aut(P_i)$ on a certain elementary abelian p-group M (the *p-multiplicator* of P_i). The main computation of the inductive step is the determination of the stabiliser in $Aut(P_i)$ of a subgroup U of M. If the stabiliser is obtained by constructing the corresponding orbit of U, the *length* of this orbit is a practical limitation. We recall the algorithm in its basic form in Section 3.

In this paper we introduce new features and refinements of the algorithm which extend significantly its range of application. In summary, we seek either to break up a single difficult stabiliser computation into smaller pieces or to replace the acting group by a proper subgroup which contains the required

stabiliser. An overview of our refinements is given in Section 4.

We first exploit the structure of the automorphism group $Aut(P_i)$. Since M is an elementary abelian p-group, $Aut(P_i)$ acts on M as a subgroup of the appropriate general linear group. Further, since $Aut(P_i)$ contains a known normal p-subgroup, it has a useful "hybrid" structure. In Sections 5 and 6, we introduce general-purpose techniques to simplify stabiliser computations under the action of matrix groups and hybrid groups respectively.

We next exploit the subgroup structure of the given p-group P. In Section 7, we present an algorithm to construct subgroups of P which on theoretical grounds are known to be characteristic. In Section 8, we discuss how to use group-theoretic invariants of subgroups of P to obtain other characteristic subgroups. We use these characteristic subgroups to replace the acting group by a proper subgroup which contains the required stabiliser.

Some of our reductions incur significant overheads and so we wish to invoke them only if an inductive step of the automorphism group algorithm would otherwise be too difficult. Consequently, we introduce in Section 9 a method to estimate *a priori* the length of an orbit. This method also provides random elements of the corresponding stabiliser.

The resulting refined algorithm is summarised in Section 10. It takes as input a p-group P and a set C of subgroups of P and returns a description of the subgroup A of the automorphism group $Aut(P)$ which stabilises each element of C. Thus, if C is empty or each subgroup of C is characteristic in P, the algorithm returns $Aut(P)$. The description is a generating set for A specified by its action on P. We identify a normal p-subgroup of A where A/N is a subgroup of the relevant linear group; we may also identify a soluble subgroup S of A containing N.

We have implemented the algorithm in MAGMA [3]; a variation is available as the AUTPGRP package [7], distributed with GAP [10]. In Section 11 we discuss aspects of our implementation and comment on its performance. We illustrate the algorithm in Section 12.

The ideas presented here have implications for other related tasks: computing the automorphism groups of finite (soluble) groups; generating (descriptions of) p-groups [21]; and deciding isomorphism between p-groups [23].

2 Background

2.1 Polycyclic generating sequences and bases

Let G be a soluble group with composition series

$$G = C_1 \rhd C_2 \rhd \cdots \rhd C_n \rhd C_{n+1} = 1.$$

Each factor C_i/C_{i+1} is cyclic of prime order p_i. If we choose $g_i \in C_i \setminus C_{i+1}$, then we obtain a *polycyclic generating sequence* $(g_1, \ldots, g_n)$ of G. Each $g \in G$ can be written uniquely as $g_1^{\epsilon_1} \ldots g_n^{\epsilon_n}$ for $0 \leq \epsilon_i < p_i$. We call $(\epsilon_1, \ldots, \epsilon_n)$ the *exponent vector* of g relative to $(g_1, \ldots, g_n)$. A polycyclic generating sequence of G allows efficient computations with subgroups and can be used readily to determine polycyclic generating sequences for factor groups of G. Such descriptions underpin most efficient algorithms for exploring soluble groups. See [29, Chapter 9] for more details. If G is a p-group, then we can use a chief series for G as composition series. A corresponding polycyclic generating sequence for G is known as a *base* – a natural generalisation of the concept of a basis of a vector space to a non-commutative situation. A *base* determines a *consistent power-commutator presentation* for G whose defining relations are of the form $g_i^p = g_{i+1}^{\beta(i,i,i+1)} \cdots g_n^{\beta(i,i,n)}$ and $[g_j, g_i] = g_{j+1}^{\beta(i,j,j+1)} \cdots g_n^{\beta(i,j,n)}$ for $1 \leq i < j \leq n$ where $\beta(i,j,k) \in \{0, \ldots, p-1\}$.

2.2 The p-covering group of a p-group

The *p-covering group* P^* of a p-group P is the largest elementary abelian, central Frattini extension of P. Thus, if $\psi : P^* \to P$ is the natural homomorphism of the extension and $M = \ker(\psi)$, then M is an elementary abelian p-group which is central in P^* and $M \leq \Phi(P^*)$. The kernel M is the *p-multiplicator* of P.

If P is a p-group described by a base, then using the method of [20], we can efficiently compute a power-commutator presentation for P^*. We also obtain an explicit homomorphism $\psi : P^* \to P$ and a base for its kernel M.

We introduce a connection between terms of the lower exponent-p central series and p-covering groups. For a proof see [21].

Theorem 2.1 *Let P be a p-group, let $P_i = P/\mathcal{P}_i(P)$ have minimal generating set $g_1, \ldots, g_d$, and let P_i^* be the p-covering group of P_i. Consider the natural*

epimorphisms $\psi : P_i^* \to P_i$ *and* $\gamma : P_{i+1} \to P_i$. *Let* g_j^* *and* $\overline{g_j}$ *be arbitrary preimages of* g_j *under* ψ *and* γ, *respectively. Then* $\epsilon : P_i^* \to P_{i+1} : g_j^* \mapsto \overline{g_j}$ *defines an epimorphism.*

Note that $M \leq \Phi(P_i^*)$ and thus $P_i^* = \langle g_1^*, \ldots, g_d^* \rangle$. If we have polycyclic generating sequences for P_i^* and P_{i+1}, we can compute $U = \ker(\epsilon)$. By construction $U \leq M$. Figure 1 illustrates these relationships.

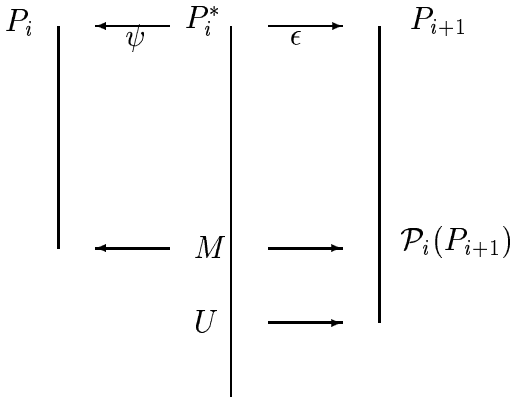

Figure 1: Relationship between factors and p-covering group

3 The basic algorithm

We now recall the basic algorithm to compute the automorphism group of a p-group as described in [24]. For background and proofs, we refer to [21].

The input of the algorithm is a p-group P described by a base. We assume that we can compute the exponent vector of an element relative to the given base. Thus, for example, P can be a permutation group or described by a power-commutator presentation. We describe an automorphism group by a set of generators and its order. In particular, the output of the algorithm returns such a description for $Aut(P)$.

The algorithm proceeds by induction down the lower exponent-p central series of P. Since $P_2 = P/\mathcal{P}_2(P)$ is elementary abelian, $Aut(P_2) \cong \mathrm{GL}(d, p)$. Now we assume by induction that we know $Aut(P_i)$ for some $i \geq 2$ and we seek a generating set of $Aut(P_{i+1})$. Let P_i^* be the p-covering group of P_i and M the corresponding p-multiplicator.

Theorem 3.1 *Each automorphism α of P_i extends to an automorphism α^* of P_i^* via the natural homomorphism $P_i^* \to P_i$. Moreover, α^* leaves M invariant and α induces an automorphism α_M of M, which depends only on α.*

We recall the explicit construction for the action on M. Let $m \in M$. Since $M \leq \Phi(P_i^*)$, we can write $m = w(g_1^*, \ldots, g_d^*)$ for some word w in the generating set $g_1^*, \ldots, g_d^*$ of P_i^* from Theorem 2.1. Let $h_i = (g_i^*)^{\psi\alpha} \in P_i$ and choose a preimage h_i^* in P_i^* under the natural epimorphism $\psi : P_i^* \longmapsto P_i$. We define $m^{\alpha_M} = w(h_1^*, \ldots, h_d^*)$.

From Theorem 2.1, we have an epimorphism $\epsilon : P_i^* \to P_{i+1}$ with kernel $U \leq M$. Using the action of $Aut(P_i)$ on M, we define the stabiliser $S = Stab_{Aut(P_i)}(U)$. Let T be the group of automorphisms of P_{i+1} which centralise $P_{i+1}/\mathcal{P}_i(P_{i+1})$.

Theorem 3.2 *Let $\nu : Aut(P_{i+1}) \to Aut(P_i)$ be the natural homomorphism. Then $T = \ker(\nu)$ and $S = \mathrm{im}(\nu)$. Hence $Aut(P_{i+1}) = TR$, where R is an arbitrary preimage of S under ν.*

It is straight-forward to construct M, U, and the action of $Aut(P_i)$ on M. To determine $Aut(P_{i+1})$, we must construct generators for each of S and T. The major work is in computing S; a generating set for T is readily obtained as follows.

Lemma 3.3 *Let P be a p-group with $\mathcal{P}_{c+1}(P) = 1$ and $c \geq 2$. Let $g_1, \ldots, g_d$ and $x_1, \ldots, x_l$ be minimal generating sets for P and $\mathcal{P}_c(P)$, respectively. Define*

$$\beta_{i,j} : P \to P : \begin{cases} g_i \mapsto g_i x_j \\ g_k \mapsto g_k \text{ for } k \neq i. \end{cases}$$

Then $\{\beta_{i,j} \mid 1 \leq i \leq d \text{ and } 1 \leq j \leq l\}$ is a base for the elementary abelian p-group of automorphisms of P centralising $P/\mathcal{P}_c(P)$.

The standard technique to compute generators for a stabiliser is to list the orbit of the subspace and, concurrently with its construction, calculate Schreier generators for the stabiliser. See for example [4, Chapter 13] for further details. If the orbit is small, this approach is very efficient. However, we usually face two problems: the orbit may be very large and the set of Schreier generators for the stabiliser is highly redundant. In Section 12 we discuss some examples which cannot be resolved using the basic algorithm.

4 Refinements to the algorithm

We now introduce a number of significant refinements to the basic algorithm. All are designed to minimise the lengths of the orbits which we must construct. We provide details of these refinements in the next four sections and summarise the resulting refined algorithm in Section 10.

Most significantly, we exploit the *hybrid* structure of the automorphism group. Observe, from Lemma 3.3, that the acting group $G = Aut(P_i)$ has a normal p-subgroup N, namely the centraliser in G of $V \cong P/\mathcal{P}_2(P)$, and G/N is a subgroup of $\mathrm{GL}(V)$. (In practice, we may be able to identify a larger normal p-subgroup of $Aut(P_i)$.)

We can also refine further the structure of G by computing the preimage S of the soluble radical of G/N. Hence, we use a series containing three distinct parts in G:

$$G \rhd S \rhd N \rhd 1$$

where S is soluble, N is a p-group, and both are normal subgroups of G.

We construct a canonical representative for the orbit of a subspace U of M under N *without* explicitly constructing its orbit and simultaneously construct the stabiliser of U in N. An important feature is that we can determine whether or not two subspaces are in the same orbit under N by constructing their canonical representatives.

We next ascend a composition series for S to determine the stabiliser of U under S. Since S is soluble, we have an efficient algorithm to construct both the orbit and stabiliser of U in S. Since both N and S are normal subgroups of G, the orbit of U under each is a block for G. This simple observation allows us to control the number of generators needed for the stabiliser of U under G. The hybrid structure of the automorphism group is retained during each iteration of the algorithm, and the final description returned identifies these components of the automorphism group.

The second important refinement arises from the observation that $Aut(P)$ acts naturally on P_i and thus induces a subgroup, say B_i, of $Aut(P_i)$. It suffices for the remaining inductive steps to supply a subgroup A_i of $Aut(P_i)$ which contains B_i. Obviously we wish to supply a group A_i which contains B_i as a subgroup of small index. We construct such subgroups A_i by considering the characteristic subgroup structure of both P_i and P. Thus the input at

the i-th inductive step may be a *proper* subgroup of $Aut(P_i)$.

Finally, we exploit the internal structure of the p-multiplicator M of P_i. Since M is elementary abelian, it is an $Aut(P_i)$-module and we use its submodule structure to minimise the lengths of the orbits constructed.

5 Stabilisers in matrix groups

Let $G \leq \mathrm{GL}(n,q)$, where $q = p^e$, and $U \leq V$ a u-dimensional subspace of the n-dimensional natural G-module V. We discuss how to determine $Stab_G(U)$.

Observe that if we construct $Stab_G(U)$ using the standard technique, then we must determine and store the orbit U^G. Each element of the orbit is a u-dimensional subspace described by a canonical basis. The dual space U^* of U is an $(n-u)$-dimensional subspace of V. Hence, if $u > n/2$, it requires less space to store the orbit of U^*, even though we achieve no reduction in orbit length.

5.1 Invariant subspaces

We consider a G-module composition series $V = V_1 > V_2 > \cdots > V_r > V_{r+1} = 0$ of V; that is, a series of G-invariant subspaces of V such that V_i/V_{i+1} is irreducible under the action of G. For $1 \leq j \leq i \leq r+1$ we define $U_{i,j} = (U + V_i) \cap V_j$. Since $U_{i,j+1} = U_{i,j} \cap V_{j+1}$ and $U_{i-1,j} = U_{i,j} + V_{i-1}$, we obtain

$$Stab_G(U_{i,j}) \leq Stab_G(U_{i-1,j}) \quad \text{and} \quad Stab_G(U_{i,j}) \leq Stab_G(U_{i,j+1}).$$

Lemma 5.1 *Let G act on Ω and let $w, v \in \Omega$ where $Stab_G(w) \leq Stab_G(v)$. Then $Stab_G(w) = Stab_{Stab_G(v)}(w)$.*

By Lemma 5.1 we can divide the computation of $Stab_G(U)$ into a sequence of stabiliser computations. We loop over the subspaces $U_{i,j}$ and stabilise each subspace in turn using the following algorithm.

- Initialise $H := G$.
- Mark $U_{i,i}$ for $1 \leq i \leq r+1$.
- while there is an unmarked subspace $U_{i,j}$:
  - Choose an unmarked subspace $U_{i,j}$ where both $U_{i,j+1}$ and $U_{i-1,j}$ are marked.

- Construct the action of H on $U_{i-1,j}/U_{i,j+1}$.
- Compute $H := Stab_H(U_{i,j}/U_{i,j+1})$.
- Mark $U_{i,j}$.

- end while;

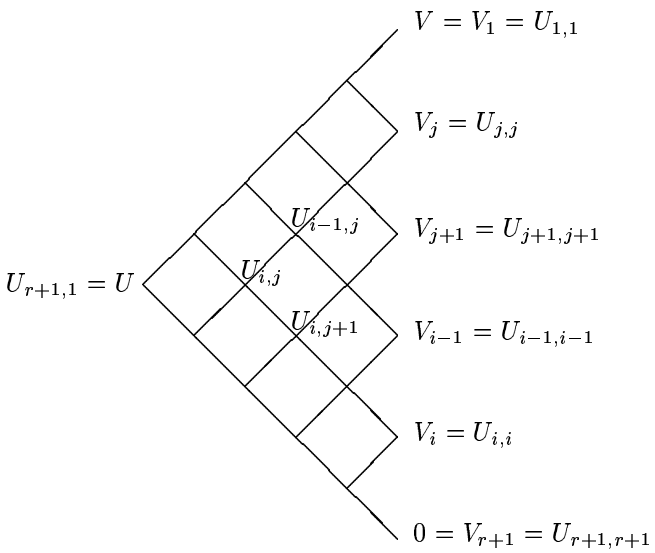

Figure 2: A lattice of subspaces of V

At any stage in the algorithm, H is the subgroup of G which stabilises all of the marked subspaces. This procedure will terminate with $H = Stab_G(U)$, since $Stab_G(U) = \cap_{i \geq j} Stab_G(U_{i,j})$. We compute $Stab_H(U_{i,j})$ only if H already stabilises both $U_{i,j+1}$ and $U_{i-1,j}$ for two reasons. First, if H did not stabilise $U_{i,j+1}$ or $U_{i-1,j}$, then we could find an intermediate subgroup between H and $Stab_H(U_{i,j})$ by computing the stabiliser of each of these two subspaces. Secondly, we can reduce the action of H to the smaller space $U_{i-1,j}/U_{i,j+1}$, by noting that $Stab_H(U_{i,j}/U_{i,j+1}) = Stab_H(U_{i,j})$.

Our description does not prescribe a *unique* path to $Stab_G(U)$. It may be most useful to choose a subspace $U_{i,j}$ having smallest non-trivial orbit among the eligible unmarked subspaces. We can estimate the orbit length of a subspace using the method of Section 9. If, at any step of the iteration, the orbit length of each of the unmarked subspaces is too large and H is a proper subgroup of G, we can recursively apply this approach to one of the factors $U_{i-1,j}/U_{i,j+1}$, in the hope that this factor is reducible under the action of H.

5.2 The stabiliser under a p-group

Let G be a p-subgroup of $\mathrm{GL}(n,q)$ where $q = p^e$; thus G is unipotent. Schwingel [25] presents an algorithm to construct a canonical representative $\overline{U}$ of the G-orbit of a subspace U of the natural G-module V. At the same time, the algorithm constructs a generating set for the stabiliser in G of $\overline{U}$ and an element t of G such that $U^t = \overline{U}$. The canonical representative is defined by an order relation on the orbit of U under G. Hence, we can decide whether two subspaces are in the same orbit under G by computing and comparing their canonical representatives.

Since we are interested only in subgroups of $\mathrm{GL}(n,p)$, we outline the algorithm only for the case $p = q$. We assume that we have a base $(g_1, \ldots, g_m)$ for the acting group G. Further, let $V = V_1 > V_2 > \cdots > V_{n+1} = 0$ be a maximal G-invariant flag in V. Then V has an ordered basis $(e_1, \ldots, e_n)$ with $e_i \in V_i \setminus V_{i+1}$ for $1 \leq i \leq n$. The canonical form of a subspace is not absolute, but depends on the chosen ordered basis.

Definition 5.2 *Let $X, Y \subseteq \{1, \ldots, n\}$. Then $X < Y$ if one of the following occurs:*

1. *$X \neq \emptyset$ and $Y = \emptyset$.*
2. *The least element of X is less than the least element of Y.*
3. *If X and Y have the same least element k, then $X \setminus \{k\} < Y \setminus \{k\}$.*

Hence, we obtain a total (or linear) ordering on the set of subsets of $\{1, \ldots, n\}$.

Let $v = \sum_{i=1}^{n} a_i e_i \in V$, and let $Z_v = \{i : 1 \leq i \leq n \text{ and } a_i = 0\}$. We define a partial ordering on V as follows: if $v, w \in V$ then $v < w$ if $Z_v < Z_w$. The canonical form of a vector v in its orbit under G is the unique least element of its orbit under this ordering. (Although we have only defined a partial order on V, this least element is unique because of the assumptions on G.)

Let $U \leq V$. Intersecting U with the V_j gives a maximal flag in U once duplicates have been removed. Clearly, $Stab_G(U)$ stabilises each subspace in this flag. In particular, $Stab_G(U)$ stabilises the one-dimensional space U_1 in this flag. We determine the canonical form and relevant stabiliser for U_1, and proceed by recursion.

Hence we reduce to the case where U has dimension 1. Let u be the non-zero element of U_1 with leading coefficient 1. Since G is unipotent, u^g has leading coefficient 1 as well. Thus $Stab_G(U)$ stabilises u and finding a canonical form for U reduces to finding a canonical form for u.

We now outline how to construct this canonical form for u. We define the *height* of $v \in V$ to be the integer j such that $v \in V_j \setminus V_{j+1}$, where the height of the zero vector is taken to be $n+1$. Further, for $g \in G$ we define the *height* $h_v(g)$ with respect to v to be the height of $v(g-1)$.

If $h_u(g) = n+1$ for every g in the given base $(g_1, \ldots, g_m)$ for G, then G centralises u and we are finished. If not, let j be the minimum of these heights. Let k be the largest integer between 1 and m such that $h_u(g_k) = j$; define $g = g_k$. Now for some integer l where $0 \leq l \leq p-1$, we have $u^{g^l} = \sum_{i=1}^{n} a_i e_i$ with $a_j = 0$; define $u_1 = u^{g^l}$. Remove g from the base of G. For each $x \neq g$ in the base with $h_{u_1}(x) = j$, determine an integer l_x where $0 \leq l_x \leq p-1$ such that $h_{u_1}(xg^{l_x}) > j$; replace x by xg^{l_x}. It can now be proved that the resulting sequence is a base for a subgroup G_1 of G which has the property that $h_{u_1}(x) > j$ for all $x \in G_1$, and further the canonical form of u with respect to G is the canonical form of u_1 with respect to G_1.

The cost (in field operations) of the resulting algorithm is dominated by the cost of adjusting the base elements of G of height j with respect to u. Since there are at most $\frac{1}{2}n(n-1)\log_p q$ elements in a base for G, the cost is $O(n^5 \log_p q)$ for each j; hence the cost of the algorithm is $O(n^6 \log_p q)$ for a space of dimension one, and $O(n^7 \log_p q)$ for an arbitrary U.

5.3 Local orbits

Let L be a subspace of U and let L^G be the orbit of L under G. The *local orbit* of L in U is $\mathcal{L} = \{T \in L^G \mid T \leq U\}$. (Similarly, if L is such that $U \leq L$, we can define the local orbit of L by $\{T \in L^G \mid U \leq T\}$.)

We first compute $H = Stab_G(L)$ and next construct a subgroup K of G that acts transitively on $\mathcal{L}$. Now $Stab_G(U) \leq HK$. Of course, to apply this strategy successfully, we must find a subspace L of U having small local orbit. A variation of this idea was used in [22].

6 Stabilisers in hybrid groups

We describe a general technique which uses normal subgroups of the acting group to speed up an orbit-stabiliser computation. A similar approach was introduced by Leedham-Green for finite soluble groups and exploited by Laue, Neubüser & Schoenwaelder [16]. Here we extend the idea to *hybrid* groups: insoluble groups which have a normal soluble subgroup.

6.1 Normal subgroups and blocks

Let G be a group acting on a set Ω and let $N \lhd G$. Recall that the orbits of N on Ω are blocks for the action of G on Ω; see, for example, [15, p. 147]. Hence G has an induced action on the set of orbits of N. Clearly, N acts trivially on this set of orbits and the induced action of G is an action of G/N.

Let $w \in \Omega$ and let $\delta = w^N$ denote the orbit of w under N. Consider the set stabiliser $Stab_{G/N}(\delta)$ in the induced action of G/N and let $Ng \in Stab_{G/N}(\delta)$. Then $w^g = w^x$ for some $x \in N$ and thus $gx^{-1} \in Stab_G(w)$. We denote by $\tilde{g} = gx^{-1}$ a lifting of g from the set stabiliser to the point stabiliser. The following is proved in [16].

Lemma 6.1 *Let G be a group acting on a set Ω and $N \lhd G$. For $w \in \Omega$, let $\delta = w^N$ and let $\{\delta_1, \ldots, \delta_s\}$ be the orbit of δ under the induced action of G/N.*

a) $w^G = \delta_1 \cup \cdots \cup \delta_s$.

b) $Stab_G(w)N/N = Stab_{G/N}(\delta)$ *and* $Stab_N(w) = Stab_G(w) \cap N \lhd Stab_G(w)$.

c) *If* $Stab_{G/N}(\delta) = \langle Ng_1, \ldots, Ng_t \rangle$, *then* $Stab_G(w) = \langle \tilde{g}_1, \ldots, \tilde{g}_t \rangle Stab_N(w)$.

Using Lemma 6.1, we divide the computation of w^G and $Stab_G(w)$ into two smaller orbit-stabiliser computations. First, we determine w^N and $Stab_N(w)$. Then we compute the orbit and stabiliser of w^N under the induced action of G/N. These two steps allow us to determine w^G and a generating set for $Stab_G(w)$.

If G has l generators, then a set of Schreier generators for $Stab_G(w)$ has cardinality $|w^G|(l-1)+1$. If l_1 and l_2 are the numbers of generators for G/N and N, respectively, then Lemma 6.1 provides a generating set for $Stab_G(w)$

of size $(|w^G|/|w^N|)(l_1 - 1) + |w^N|(l_2 - 1) + 2$, which is usually much smaller. Further, this generating set exhibits a generating set for the normal subgroup $Stab_N(w)$ which can be used to iterate this approach. Finally, constructing w^G is usually more efficient using the action on sets, as described in Lemma 6.1a).

6.2 Hybrid groups having soluble normal subgroups

If the acting group G has a soluble normal subgroup S, we can apply the "divide-and-conquer" strategy suggested by Lemma 6.1. Assume we know a composition series for S, so

$$G \rhd S = S_1 \rhd S_2 \rhd \cdots \rhd S_l \rhd S_{l+1} = 1$$

where $[S_i : S_{i+1}] = p_i$, a prime, for $1 \leq i \leq l$. Let $s_i \in S_i \setminus S_{i+1}$ and consider the polycyclic generating sequence $(s_1, \ldots, s_l)$ of S. Since S_i/S_{i+1} has prime order, we obtain two cases.

- S_i/S_{i+1} fixes $w^{S_{i+1}}$ and $Stab_{S_i}(w) = \langle \tilde{s_i} \rangle Stab_{S_{i+1}}(w)$.
- S_i/S_{i+1} moves $w^{S_{i+1}}$ and $Stab_{S_i}(w) = Stab_{S_{i+1}}(w)$.

Hence we can compute the orbit of w under S without visiting an element of w^S twice, and we also obtain a polycyclic generating sequence for $Stab_S(w)$. The remaining inductive step from $Stab_S(w)$ to $Stab_G(w)$ is now performed as suggested by Lemma 6.1.

7 Constructing characteristic subgroups

We introduce a method to construct subgroups of P which are known on theoretical grounds to be characteristic. In summary, we construct images and preimages of terms of the lower exponent-p central series under p-th power and commutation maps, and close the resulting collection of characteristic subgroups under these operations.

We are particularly interested in characteristic subgroups containing $\Phi(P)$, since these result in the most significant improvements for the automorphism group algorithm.

Write V_i for $\mathcal{P}_{i-1}(P)/\mathcal{P}_i(P) \leq P_i$, and call the image of a characteristic subgroup of P in this quotient a *characteristic subspace* of V_i. Our aim is to determine characteristic subspaces without using the action of the relevant automorphism group. If we know a list of characteristic subspaces of this kind, we can restrict our attention to the subgroup of $Aut(P_i)$ that stabilises the image in P_i of each of the corresponding subgroups.

In particular, we can begin our calculation with the subgroup of $\mathrm{GL}(V_1)$ that stabilises each known characteristic subspace of V_1. To determine this subgroup of $GL(V_1)$, we use an algorithm of Schwingel [25] to construct in polynomial time the subgroup A of $\mathrm{GL}(d,p)$ which stabilises each element of a set of subspaces of F_p^d. Hence the initial step of the automorphism group algorithm returns A rather than $\mathrm{GL}(d,p)$.

Our construction technique for characteristic subgroups uses linear algebra, and hence runs in polynomial time, given a bound to the number of characteristic subspaces we are prepared to construct. In fact, we search for characteristic subgroups Q satisfying $\mathcal{P}_{i+1}(P) \leq Q \leq \mathcal{P}_i(P)$ for all i, partly because they are needed to construct characteristic subgroups containing $\Phi(P)$, and partly because they play a useful role in Section 8.

For each i, let S_i be a set of characteristic subspaces of V_i. We initialise S_i to contain the two trivial subspaces of V_i. Let π_i be the map from V_i to V_{i+1} induced by p-th powering for p odd or $i > 1$, and let $c_{ij} : V_i \otimes V_j \to V_{i+j}$ be the map induced by commutation. For $p = 2$, we define π_1 as the squaring map from V_1 to $\mathcal{P}_2(P)/P'$. We now form the closure of S_i as follows.

1. (a) If $p = 2$ and $U \in S_1$, add the inverse image in V_2 of $\pi_1(U)$ to S_2; if $p = 2$ and $U \in S_2$, add $\pi_1^{-1}(\bar{U})$ to S_1, where $\bar{U}$ is the image of U in $\mathcal{P}_2(P)/P'$.

   (b) Otherwise, if $U \in S_i$, add $\pi_i(U)$ to S_{i+1} and if $i > 1$, add $\pi_{i-1}^{-1}(U)$ to S_{i-1}.

2. If $U_i \in S_i$ and $U_j \in S_j$, then add $c_{ij}(U_i \otimes U_j)$ to S_{i+j}.

3. If $U_i \in S_i$, and $U_{i+j} \in S_{i+j}$, then add to S_j the subspace of V_j consisting of those vectors $v \in V_j$ such that $c_{ij}(U_i \otimes v) \leq U_{i+j}$.

8 Fingerprinting

It may be that the characteristic subspaces constructed using the method of Section 7 do not reduce the size of the orbits sufficiently. However, they allow us to identify characteristic sections of P. We now seek to exploit them further.

Let $Y \leq X$ be characteristic subgroups of P where X/Y is an elementary abelian p-group. We consider the set of minimal subgroups of X/Y. We list these subgroups and their preimages in P under the natural homomorphism $X \to X/Y$. Now we partition the set of minimal subgroups of X/Y into subsets using properties of the preimages that must be $Aut(P)$-invariant. We call this process *fingerprinting the section* X/Y in P. Equivalently, we can consider maximal subgroups (or indeed all of the subgroups of an arbitrary fixed order) of X/Y. Of course, we can also fingerprint a section of P_i using properties that must be $Aut(P_i)$-invariant. Here we have the following advantages. If A is the subgroup of $Aut(P_i)$ which we have constructed in the previous stage, then we can restrict our attention to just *one* A-orbit and determine invariants for elements of this orbit only. Further, if we know a subgroup H of $Aut(P_{i+1})$, it suffices to compute invariants for a set of representatives of the H-orbits of subgroups only. Such an H could be obtained using the strategy outlined in Section 9.

Fingerprinting yields new conditions on $Aut(P)$; we discuss now how exploit these.

8.1 New characteristic subgroups

Let X/Y be an arbitrary $Aut(P)$-invariant elementary abelian section of P. We fingerprint X/Y and so obtain a partition of the set of minimal subgroups of X/Y. Each subset of the partition generates an $Aut(P)$-invariant subgroup of X/Y and its preimage is characteristic in P. We can now use the characteristic closure algorithm of Section 7 to generate images and preimages of the corresponding characteristic space.

8.2 Partition stabiliser

Let A be the subgroup of the automorphism group of $P_i = P/\mathcal{P}_i(P)$ constructed by the automorphism group algorithm in the previous stage. Then A

acts on $\mathcal{P}_j(P)/\mathcal{P}_{j+1}(P)$ for $1 \leq j < i$. We can use this action to identify an A-invariant section X/Y in one of these factors. We fingerprint this section and obtain a partition of the set of minimal subgroups of X/Y. The stabiliser in A of this partition can now replace A.

To determine this stabiliser, we first calculate the permutation action B of A on the set of minimal subgroups of X/Y, then compute the partition stabiliser in the permutation group B, and, finally, compute the preimage of the stabiliser under the natural homomorphism $\psi : A \to B$. It simplifies this preimage computation if the kernel of ψ is readily determined. Hence, for example, this approach is particularly effective if $X/Y = P_2$.

8.3 Enforcing new "characteristic" subgroups

As before, let A be the constructed subgroup of the automorphism group of $P_i = P/\mathcal{P}_i(P)$, and let X/Y be an A-invariant section of P_i. We partition the set of minimal subgroups of X/Y using properties of their preimages in P_{i+1}. Assume that we also know $H \leq A$ which lifts to a subgroup of $Aut(P_{i+1})$. Clearly, every part in the partition is a union of H-orbits. Suppose that some part consists of a single H-orbit and let L be the preimage in P of an element of this H-orbit.

We now compute the subgroup S of $Aut(P_{i+1})$ that stabilises L. We can construct S by adding L to the set of characteristic subgroups used in Section 7 and then re-running the automorphism group algorithm up to this step. (By adding L to the list of subgroups, we expect that this computation will be more efficient than the initial computation.) Now it is easy to deduce that $Aut(P_{i+1}) = SH$. If $\alpha \in Aut(P_{i+1})$, then $L^\alpha = L^\beta$ for some $\beta \in H$; thus $\alpha \cdot \beta^{-1}$ stabilises L and so is contained in S.

8.4 Refining the result of fingerprinting

Fingerprinting is most useful if we obtain new characteristic subspaces of $V_1 = P/\mathcal{P}_2(P)$, since we may then be able to reduce the stabiliser computation to a smaller subgroup of $\mathrm{GL}(d,p)$. The simplest method to obtain a new characteristic space of V_1 is to find a part in a partition resulting from fingerprinting a section of V_1 whose sum, or intersection, defines a proper non-trivial subspace of V_1. Hence, we want our partition to be as fine as possible.

Given an initial partition constructed as above, we seek to refine it by using lattice-theoretic invariants. For example, if $\mathcal{A}$ and $\mathcal{B}$ are (not necessarily distinct) parts, we can define a fingerprint function on $\mathcal{A}$ as follows. For each $U \in \mathcal{A}$, consider for each i the number $g_U(i)$ of subspaces W of $\mathcal{B}$ such that $U+W$ has dimension i. If the functions g_U are not constant on $\mathcal{A}$, this gives a refinement of $\mathcal{A}$ as a union of parts on which g_U is constant (as a function of U). There are many variations: taking intersections rather than sums; taking triples of spaces rather than pairs, *etc.*

8.5 Stabilisers of lattices

Recall that the primary function of fingerprinting is to produce sets of subgroups of P_2 which are setwise invariant under the action of $Aut(P)$. Consequently, we consider briefly the general problem of finding the subgroup of $\mathrm{GL}(V)$ that setwise stabilises a given set of subspaces of V. It is usually not easy to find generators for this subgroup. However, given such a set, one can try to refine it by using combinatorial properties, so as to find a simpler set S of subspaces of V with the property that any group normalising the original set must also normalise S. Here "simpler" means "having a larger normaliser", ruling out the trivial cases when the normaliser is the whole of $\mathrm{GL}(V)$.

The hope is that one could recognise the normaliser M of S as being one of the maximal subgroups of $\mathrm{GL}(V)$ given explicitly in Aschbacher's [1] classification of the maximal subgroups of $\mathrm{GL}(V)$. For example, if S is a single subspace, or a set of subspaces of the same dimension whose direct sum is V, then M is clearly of this form. In fact many of the maximal subgroups of $\mathrm{GL}(V)$ defined explicitly by Aschbacher are normalisers of lattices of subspaces of V.

This raises the question of whether a C9 maximal subgroup of $\mathrm{GL}(V)$ can be the normaliser of a set S of subspaces of V. Guralnick & Shareshian [11] prove that if so, either the lattice generated by S is the full subspace lattice of V, or V has even dimension $2n$ for some n and the non-trivial elements of S are subspaces of dimension n with the property that any two intersect trivially. Further they prove that if S does not satisfy these conditions, then the normaliser of S is a subgroup of a maximal subgroup M of $\mathrm{GL}(V)$ lying in another Aschbacher category (not C9) and M is the normaliser of some lattice of subspaces. The challenge then is to find such an M.

9 Estimating the orbit length

The refinements introduced may incur some overheads. Consequently, we wish to invoke them only if the orbit is so large that we cannot readily compute and store its elements. The "Birthday Paradox" permits us to estimate the orbit length.

Assume we have a large population and seek to estimate its size. We select an independent uniformly distributed random sample of the population, noting how large a sample is needed to obtain L coincidences for some specified L. Using the statistical model developed by Goodman [26, §4.3.1], we can then estimate the expected size N of the population. If n is the size of the sample selected, then N is $n^2/2L$. Confidence intervals for N are asymptotically distributed as χ-squared with $2L$ degrees of freedom. The coefficient of variation of N is $1/\sqrt{L}$ and the model appears to have practical validity if $L > 15$.

Theorem 9.1 *Let G act on a set Ω and let $w \in \Omega$. Assume we construct n elements of the orbit of w under G before obtaining L coincidences.*

- *The expected length N of the orbit is $n^2/2L$.*
- *An approximate $100(1-\alpha)\,\%$ confidence interval for N is*

$$2n^2/(z(\alpha/2) + \sqrt{4L-1})^2 < N < 2n^2/(-z(\alpha/2) + \sqrt{4L-1})^2$$

where $z(\alpha/2)$ is the $\alpha/2$ quantile on the standard $(0,1)$ normal distribution.

We construct random elements of the orbit of w under G by applying random elements of G to w. Moreover, this process produces random elements of the stabiliser of w: if $w^g = w^h$, then $gh^{-1} \in Stab_G(w)$. Hence it provides the basis for a Monte-Carlo algorithm to construct the stabiliser.

10 The algorithm

We now outline our algorithm to compute the automorphism group of a p-group P. We assume that the computation is difficult; in easier cases, various steps of the algorithm can be simplified.

We present our algorithm as a recursive function, `AutomorphismGroup`(P, C), where P is a p-group and C is a set of subgroups of P. The function computes the subgroup of $Aut(P)$ which stabilises each element of C. Thus, if C is empty or each subgroup in C is characteristic in P, the output is $Aut(P)$. We initialise C as described in Section 7. Assume P has a lower exponent-p central series of length c.

`AutomorphismGroup` (P, C)

1. Determine $A = \cap_{S \in C} N_{Aut(P_2)}(S\mathcal{P}_2(P)/\mathcal{P}_2(P)) \leq Aut(P_2) \cong \mathrm{GL}(d, p)$ using the algorithm described in [25]. We also obtain $N = O_p(A)$ and a base for N.

2. Determine (if possible) the preimage S in A of the soluble radical of A/N and hence a composition series of S/N in order to exploit the ideas of Section 6.

3. for i from 2 to c do

   (a) Compute P_i^* and determine M and U. Calculate the induced action of A on M in matrix form (Section 3).

   (b) Determine $Stab_N(U)$ and $\overline{U}$ (Section 5.2).

   (c) Call the function `BlockStabiliser`$(A, N, U, \overline{U})$ which we describe below. It can return three results.

      i. If the result is $B = Stab_A(U)$, continue as follows:
         - Assign $A = B$ and $N = Stab_N(U)$.
         - Lift A and N to groups of automorphisms of P_{i+1} (Section 3).
         - Extend the base for N by a base for T (Section 3).

      ii. If the result is a set D of subgroups of P, then the stabiliser computation is difficult, but we have found new characteristic subgroups of P. We recursively call `AutomorphismGroup` $(P, C \cup D)$.

      iii. If the result is "fail", then the stabiliser computation is too hard and we return "fail".

   end for;

4. Return $Aut(P) = A$.

The central step of the algorithm is the function `BlockStabiliser`, which seeks to determine the stabiliser of A acting on N-blocks of U. We use the method of Section 5.2 to represent N-blocks by canonical forms.

`BlockStabiliser`$(A, N, U, \overline{U})$

- Use the method of Section 9 to estimate the orbit size of A/N acting on N-canonical forms. If the expected size is small, then compute the stabiliser of U using the composition series of S/N and the methods of Section 6.
- Otherwise, construct a composition series for the A-module M and apply the methods of Section 5.1 to compute recursively stabilisers in the resulting sections. If an orbit is too large to be constructed directly, then apply the local subgroups strategy (Section 5.3) or fingerprinting (Section 8). Either may succeed in constructing the stabiliser B; the latter may provide a new set of characteristic subgroups.
- If neither strategy succeeds, we return "fail".

We can readily modify the algorithm to construct the outer automorphism group of P. In practice, the function returns a generating set for a supplement to the inner automorphism group of P and a generating set for the inner automorphism group. The generators are mappings acting on the underlying p-group P; the description also exhibits the hybrid structure of $Aut(P)$.

11 Implementation and performance

Our implementations of the algorithm in both GAP and MAGMA are publicly available.

We use the MEATAXE [13] to compute a composition series for a G-module as required in Section 5.1; it can readily construct such for spaces of very large dimension in time polynomial in the dimension of the G-module.

Our choice of fingerprints in Section 8 depends on the order of the preimage O in P of a subgroup of a characteristic section. Examples include the isomorphism type of O and the order of the commutator subgroup $[P, O]$. If O is small, the former can be determined using the IDGROUP function of Besche & Eick [2]. For larger preimages, we determine the ranks of the factors of their lower p-central series.

For our application of the "Birthday Paradox", we choose $\alpha = 0.05$. Thus $z(\alpha/2) = 1.96$ and we obtain with 95% confidence that

$$\frac{n^2}{50} < N < \frac{n^2}{12},$$

where N is the size of the orbit and n is the number of elements of the orbit we generate before obtaining at least 16 coincidences. Of course, we also bound the number of elements we are willing to construct; the precise value chosen depends on the size and nature of the orbit elements. We use the algorithm of [6] to construct random elements of a group described by a generating set, and so obtain random elements of the orbit.

On efficiency grounds, the soluble normal subgroup S computed in step (2) should be as large as possible. If A is soluble, then we initialise S to be A. Otherwise, if we can efficiently determine a permutation representation of A/N, then we compute its soluble radical using the methods of either [17] or [28] and we initialise S to be the preimage of this radical. If not, we initialise S to be N.

11.1 Comments on performance

A limiting factor for the basic algorithm in constructing the automorphism group of a p-pgroup P of Frattini rank d is that it must construct the orbit of a subspace under $\mathrm{GL}(d, p)$.

The most critical factor for the new algorithm is whether we can find a "useful" overgroup A of the subgroup B of $\mathrm{GL}(d, p)$ induced by $Aut(P)$. In practice, the overgroup A should either be soluble or a p-group, or B should have "small" index in A. Since we find such an overgroup by constructing characteristic subgroups of P, our ability to detect these is critical. Hence, the most difficult examples are those p-groups where we cannot find characteristic subgroups. Characteristic subspaces and the associated subgroup of $\mathrm{GL}(d, q)$ which fixes these can be obtained in time polynomial in d.

The determination of the stabiliser of a subspace under a unipotent subgroup of $\mathrm{GL}(n, q)$ has complexity $O(n^7 \log_p q)$. As we iterate the construction, we expect that the time taken to construct the stabiliser of a subspace under the normal p-subgroup will dominate. While this computation may in practice be expensive, its success is not limited by the *length* of the orbit, since we

construct only the canonical representative for an orbit. The work of Martin [18] provides some theoretical justification for this expectation.

The partition stabiliser approach of Section 8.2 is highly effective, particularly so if no characteristic subspaces are found, but it is limited to those cases where a permutation representation can be explicitly constructed for the action of a subgroup of the automorphism group on a characteristic section.

12 Some applications

We now present some examples which are beyond the range of the basic algorithm of Section 3.

12.1 An exponent-p class 2 example

Let P be the group of order 3^{13} having the following power-commutator presentation where all defining relations with trivial right-hand sides are omitted.

$$\{g_1, \ldots, g_{13} \mid [g_{10}, g_6] = g_{11}, \ [g_{10}, g_7] = g_{12}, \\ [g_2, g_1] = [g_4, g_3] = [g_6, g_5] = [g_8, g_7] = [g_{10}, g_9] = g_{13}\}$$

The lower exponent-p central series of P has length 2 and the factors $\mathcal{P}_1(P)/\mathcal{P}_2(P)$ and $\mathcal{P}_2(P)/\mathcal{P}_3(P)$ have ranks 10 and 3 respectively. We apply the algorithm of Section 10.

1. We find no characteristic subgroup in P. Hence $A \cong GL(10, 3)$ and N is trivial.

2. P_2^* is a group of order 3^{65}. The p-multiplicator M is elementary abelian of order 3^{55} and has U as a subgroup of index 3^3. The action of A is straight-forward to determine. We replace U by its dual.

3. There is one A-invariant submodule V in M of dimension 45. However, $U \leq V$ and so the methods of Section 5.1 do not yield a reduction of the orbit length. We restrict the computation to V and so compute with smaller matrices.

4. We generate 50000 elements of the orbit of U under A without obtaining any coincidences and conclude that the orbit is "large".

5. We fingerprint the minimal and maximal subgroups of $\mathcal{P}_2(P_3)/\mathcal{P}_1(P_3)$, obtaining two partitions of each. Following the strategy of Section 8.4, we obtain six non-trivial characteristic spaces of $P/\mathcal{P}_2(P)$.

6. The stabiliser A in $\mathrm{GL}(10,3)$ of each of these characteristic subspaces has order $2^{19} \cdot 3^{45} \cdot 5 \cdot 13$. Its largest normal p-subgroup N has order 3^{37}.

7. We now use the algorithms of Sections 5.1, 5.2 and 6 to deduce that the orbit of U under N has length 3^{15} and the orbit of canonical forms under A/N has length $2^6 \cdot 3^3 \cdot 13$, and to construct the stabiliser of U under A.

8. Since the group T has $10 \cdot 3$ generators, the automorphism group of P has order $2^{13} \cdot 3^{57} \cdot 5$.

Our MAGMA V2.6 implementation on a Sun UltraSPARC Enterprise 4000 server determined this automorphism group in 2670 seconds. This example is typical for groups of exponent-p class 2: it is usually difficult to find useful characteristic subgroups, the p-multiplicator M has only one invariant subgroup, and the normal subgroup N is trivial. Hence, fingerprinting is invoked in most cases.

12.2 An exponent-p class 7 example

In a study of 3-transposition groups, Hall & Soicher [12] consider the two groups H_i having finite presentations

$$\begin{aligned}\{a,b,c,d,e \;:\; & a^2, b^2, c^2, d^2, e^2, (ab)^3, (ac)^3, (ad)^3, (ae)^3,\\ & (bc)^3, (bd)^3, (be)^3, (cd)^3, (ce)^3, (de)^3,\\ & (a^b c)^3, (a^b d)^3, (a^c d)^3, (b^c d)^3, (a^b c^d)^3, (a^c b^d)^3,\\ & (a^d b^c)^3, (a^b e)^3, (a^c e)^3, (a^d e)^3,\\ & (b^c e)^3, (b^d e)^3, (c^d e)^3, (a^{(bc)} e)^3, (b^{(ca)} e)^3,\\ & (c^{(ab)} e)^3, (a^{(bd)} e)^3, (b^{(da)} e)^3, (d^{(ab)} e)^3, (a^{(cd)} e)^3,\\ & (c^{(da)} e)^3, (d^{(ac)} e)^3, (b^{(cd)} e)^3, (c^{(db)} e)^3, (d^{(bc)} e)^3,\\ & (c^{(dbab)} e)^3, (c^{(dab)} e)^3, (c^{(dba)} e)^3, (a^{(bcd)} e)^3,\\ & (a^{(bdc)} e)^3, (ab^{(edce)})^3, (ab^{(ecde)})^3, (ac^{(bed)^2})^3, r_i\}\end{aligned}$$

where $r_1 = a^{-1}a^{(bcd)^2}$ and $r_2 = a^{-1}a^{(bcd)^2}(cde)^2$. Each has derived group of index 2 and order 3^{29}. We constructed the automorphism group of each derived group. Since the details of the computations are identical, we report on that for P, the subgroup of index 2 in H_1.

The lower exponent-p central series of P has length 6. In Table 1, for each factor P_i with $i \in \{2, \ldots, 6\}$, we list the dimensions of the p-multiplicator M and of the relevant allowable subgroup U, the order of $Aut(P_i)$, and the length of the orbit of U under the action of $Aut(P_i)$.

i	P_i	dim M	dim U	$\lvert Aut(P_i)\rvert$	$\lvert O\rvert$
2	3^4	10	4	$2^9 \cdot 3^6 \cdot 5 \cdot 13$	1
3	3^{10}	24	21	$2^9 \cdot 3^{30} \cdot 5 \cdot 13$	$2^3 \cdot 5$
4	3^{13}	32	21	$2^6 \cdot 3^{42} \cdot 13$	3^{20}
5	3^{24}	35	34	$2^6 \cdot 3^{66} \cdot 13$	1
6	3^{25}	38	34	$2^6 \cdot 3^{70} \cdot 13$	3^{13}
	3^{29}			$2^6 \cdot 3^{73} \cdot 13$	

Table 1: The group of order 3^{29}

We find no characteristic subgroup and so $A \cong \mathrm{GL}(4,3)$. The inductive steps for $i = 2, 3$ are both easy, since the corresponding orbits are small. We consider in more detail the inductive step from P_4 to P_5. The input automorphism group $Aut(P_4)$ has order $2^6 \cdot 3^{42} \cdot 13$ and has normal 3-subgroup N of order 3^{36}.

1. We compute a power-commutator presentation for P_4^* which shows that M has order 3^{32} and U is a subgroup of index 3^{11}.

2. We now use the algorithms of Sections 5.1, 5.2 and 6 to deduce that the orbit of U under N has length 3^{20} and the orbit of canonical forms under A/N has length 1, and to construct the stabiliser of U under A.

An alternative approach is to determine an $Aut(P_4)$-composition series for M. The resulting composition series has length 18 and the non-trivial orbits have length 3^8, 3^6, 3^3 and 3^3.

In the inductive step from P_6 to P_7, we again use the algorithm of Section 5.2. It took 52 seconds to determine $Aut(P)$.

12.3 Wreath products

We consider the n-fold wreath product of C_p for various values of p and n. Since characteristic subgroups are easily found, the initial step of the algorithm reduces the group acting from $\mathrm{GL}(n+1,p)$ to a "computable" subgroup A. This reduction and the refinement of Section 5.1 suffice. In Table 2, we record the defining parameters p and n, the order of P, its exponent-p class c, and the order of the subgroup A.

p	n	$\lvert P\rvert$	c	$\lvert A\rvert$
2	5	2^{63}	32	6
2	7	2^{255}	128	168
3	3	3^{40}	27	48
5	3	5^{156}	125	1280
7	2	7^{57}	49	1512

Table 2: Some wreath product examples

Acknowledgements

This work was supported in part by the Marsden Fund of New Zealand via grant #9144/3368248. Eick and O'Brien acknowledge the financial support of the Alexander von Humboldt Foundation, Bonn. We are grateful to M.F. Newman for helpful comments.

References

[1] M. Aschbacher. On the maximal subgroups of the finite classical groups. Invent. Math. **1984**, *76*, 469–514.

[2] Besche, Hans Ulrich; Eick, Bettina. The groups of order $q^n \cdot p$". Comm. Algebra **2001**, *29* (4), 1759–1772.

[3] Bosma, Wieb; Cannon, John; Playoust, Catherine. The Magma algebra system I: The user language. J. Symbolic Comput. **1997**, *24*, 235–265.

[4] Butler, G. *Fundamental Algorithms for Permutation Groups*; Lecture Notes in Comput. Sci.; Springer-Verlag: Berlin, Heidelberg, New York, 1991; Vol. 559.

[5] Cannon, John; Holt, Derek F. Automorphism group computation and isomorphism testing in finite groups. Preprint, 2001.

[6] Celler, Frank; Leedham-Green, Charles R.; Murray, Scott H.; Niemeyer, Alice C.; O'Brien, E.A. Generating random elements of a finite group. Comm. Algebra **1995**, *23*, 4931–4948.

[7] Eick, Bettina; O'Brien, E.A. AutPGroup – A GAP 4 package. See [10].

[8] Felsch, V.; Neubüser, J. Über ein Programm zur Berechnung der Automorphismeng ruppe einer endlichen Gruppe. *Numer. Math.* **1968**, *11*, 277–292.

[9] Felsch, V.; Neubüser, J. On a programme for the determination the automorphism group of a finite group. In *Computational Problems in Abstract Algebra*, Oxford, 1967; Pergamon Press: Oxford, London, Edinburgh, 1970; 59–60.

[10] The GAP Group. GAP – *Groups, Algorithms, and Programming, Version 4.2*. (`http://www.gap-system.org`), 2000.

[11] Guralnick, Robert M.; Shareshian, John. Subgroups of $\mathrm{GL}_n(q)$ which fix sublattices of $\mathcal{B}_n(q)$. Preprint, 2001.

[12] Hall, J.I.; Soicher, L.H. "Presentations of some 3-transposition groups", Comm. Algebra **1995**, *23*, 2517–2559.

[13] Holt, Derek F.; Rees, Sarah. Testing modules for irreducibility. J. Austral. Math. Soc. Ser. A **1994**, *57*, 1–16.

[14] Hulpke, Alexander. *Konstruktion transitiver Permutationsgruppen*. PhD thesis, RWTH Aachen, 1996.

[15] Huppert, B. *Endliche Gruppen I*; Grundlehren Math. Wiss.; Springer-Verlag, Berlin, Heidelberg, New York, 1967; Vol. 134.

[16] Laue, R.; Neubüser, J; Schoenwaelder, U. Algorithms for finite soluble groups and the SOGOS system. In *Computational Group Theory*, Durham, 1982; Academic Press: London, New York, 1984; 105–135.

[17] Luks, Eugene M.; Seress, Ákos. Computing the fitting subgroup and solvable radical for small-base permutation groups in nearly linear time. In *Groups and computation*, II; New Brunswick, NJ, 1995; DIMACS Ser. Discrete Math. Theoret. Comput. Sci., *28*; Amer. Math. Soc., Providence, RI, 1997; 169–181.

[18] Martin, Ursula. Almost all p-groups have automorphism group a p-group. Bull. Amer. Math. Soc. (N.S.) **1986**, *15*, 78–82.

[19] Newman, M.F. Determination of groups of prime-power order. In *Group Theory*, Canberra, 1975; *Lecture Notes in Math. 573*; Springer-Verlag, Berlin, Heidelberg, New York, 1977; 73–84.

[20] Newman, M.F; O'Brien, E.A. Application of computers to questions like those of Burnside, II. Internat. J. Algebra Comput. **1996**, *6*, 593–605.

[21] O'Brien, E.A. The p-group generation algorithm. J. Symbolic Comput. **1990**, *9*, 677–698.

[22] O'Brien, E.A. The groups of order 256. J. Algebra **1991**, *143* (1), 219–235.

[23] O'Brien, E.A. Isomorphism testing for p-groups. J. Symbolic Comput. **1994**, *17*, 133–147.

[24] O'Brien, E.A. Computing automorphism groups of p-groups. In *Computational Algebra Number and Number Theory*, Sydney, 1992; Kluwer Academic Publishers, Dordrecht, 1995; 85–92.

[25] Schwingel Ruth. *Two matrix group algorithms with applications to computing the automorphism group of a finite p-group*, PhD thesis. Queen Mary and Westfield College, University of London.

[26] Seber, G.A.F. *The estimation of animal abundance and related parameters*, 3rd Edition. Edward Arnold; London, Sydney, 1982.

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

[28] Sims, Charles C. Computing the order of a solvable permutation group. J. Symbolic Comput. **1990**, *9*, 699–705.

[29] Sims, Charles C. *Computation with finitely presented groups.* Cambridge University Press, 1994.

[30] Smith, Michael J. *Computing automorphisms of finite soluble groups.* PhD thesis, Australian National University, 1994.

[31] Wursthorn, Martin. Isomorphisms of modular group algebras: An algorithm and its application to groups of order 2^6. J. Symbolic Comput. **1993**, *15*, 211–227.