Application of computers to questions like those of Burnside, II

M.F. Newman E.A. O'Brien

Abstract

We show how automorphisms can be used to reduce significantly the resources needed to enforce laws in p-groups. This increases the extent to which Burnside groups with prime-power exponent can be studied in detail. For example, we describe how to construct power-conjugate presentations for the restricted Burnside groups $R(5,4)$ and $R(3,5)$ which have orders 2^{2728} and 5^{2282} respectively. We also describe how to determine the exponent of a p-group and report on relevant features of the current implementation of an algorithm to compute power-conjugate presentations.

1991 *Mathematics Subject Classification* (*Amer. Math. Soc.*): 20-04, 20D15, 20F05.

1 Introduction

The purpose of this paper is to describe some of the improvements made to the ANU p-Quotient Program which have significantly extended its capacity to handle computations with Burnside groups beyond those described in the paper of Havas & Newman (1980) and the monographs of Vaughan-Lee (1993) and Sims (1994).

Given the solution of the Restricted Burnside Problem by Zel'manov (1991), the basic Burnside question becomes: what is the order of $R(d,e)$, the largest finite d-generator group of exponent e? The methods we use apply to the case when e is a prime-power p^m. This should be thought of as a test question; it is really more important to compute, and make readily available for use, consistent power-conjugate presentations (see Section 2) for these groups or at least significant quotients of them. We will describe the main algorithmic advances which have been made recently (beyond the accounts above). We use the resulting implementation to obtain:

- the order and a consistent power-conjugate presentation for the largest finite 5-generator group, $R(5,4)$, with exponent four which has order 2^{2728};

The second author thanks the Alexander von Humboldt Foundation, Bonn, for its support.

- a consistent power-conjugate presentation for the largest finite 3-generator group, $R(3,5)$, with exponent five which has order 5^{2282}.

The main advance is the use of some of the automorphisms of the Burnside groups to reduce the number of instances of the relevant power word which need to be calculated to ensure that the exponent law holds. Another important feature is the use of consequences of the exponent law both to control the size of intermediate presentations and reduce the computation time. Moreover one can use information gathered in the process of computing these consistent power-conjugate presentations to make it easier to regenerate them. This helps ameliorate the problem of making the presentations available for others to use. The ANU p-Quotient Program also incorporates, and in part improves on, ideas which have already been reported in the literature:

- the improvements to consistency enforcing and exponent enforcing suggested by Vaughan-Lee (1984);
- the use of collection from the left (Leedham-Green & Soicher, 1990; Vaughan-Lee, 1990a).

In Section 2 we recall both important features of power-conjugate presentations and the basic algorithm for computing such presentations. Much of this material is discussed in Havas & Newman (1980), Sims (1994, Chapter 11) and Vaughan-Lee (1993, Appendix B). In Section 3 we present an algorithm for determining the exponent of a p-group. In Section 4 we discuss using automorphisms to help construct power-conjugate presentations. In Section 5 we discuss using consequences of an exponent law to control the size of intermediate presentations. We discuss aspects of the implementation in Section 6. In Section 7 we report on the use of our implementation to construct consistent power-conjugate presentations for $R(3,5)$ and $R(5,4)$, and illustrate how the use of automorphisms permits us to reduce significantly the number of words whose relevant powers need to be made trivial. In Section 8 we describe how information obtained about the group can be used to allow cheap regeneration of the presentation. All times reported in the paper are in (rounded) CPU seconds obtained on a Sparc Server 10/51, having 128 MB RAM.

2 The basic algorithm

Finite groups of prime-power order may be described using presentations which are commonly known as *power-conjugate* (or *power-commutator*) presentations. The generating set is a finite set $\{a_1, \ldots, a_n\}$. The defining relations are:

$$\begin{aligned} a_i^p &= \prod_{k=i+1}^{n} a_k^{\beta(i,k)},\ 0 \le \beta(i,k) < p\,,\ 1 \le i \le n\,, \\ a_j^{a_i} &= a_j \prod_{k=j+1}^{n} a_k^{\beta(i,j,k)},\ 0 \le \beta(i,j,k) < p\,,\ 1 \le i < j \le n\,. \end{aligned}$$

These presentations have proved to be of central importance in allowing effective computation with p-groups; see Sims (1994, Chapter 9) for further discussion. Every group of order p^n has a power-conjugate presentation on n generators.

A critical feature of a power-conjugate presentation is that every element of the presented group may be written as a *normal word* $a_1^{\alpha_1}a_2^{\alpha_2}\dots a_n^{\alpha_n}$ where each α_i is an integer and $0 \leq \alpha_i < p$. Moreover there are mechanical procedures for collecting an arbitrary word in the generators to a normal word equivalent to it using the power-conjugate presentation. If every element has a unique normal form, then the power-conjugate presentation is *consistent*. Collection using a consistent power-conjugate presentation provides a solution to the word problem.

A power-conjugate presentation for a finite p-group may be constructed using a p-quotient algorithm. In theory, one can deal with groups described in various ways. We consider finite presentations, possibly combined with exponent laws. The first such algorithm was described by Macdonald (1974). Havas & Newman (1980) describe an algorithm which provides the foundation for the one in common use today. Holt (1984) provides an algorithm to compute such a presentation for a permutation group.

Our algorithm uses a variation of the lower central series known as the *lower exponent-p central series*. This is the descending sequence of subgroups

$$G = P_0(G) \geq \dots \geq P_{i-1}(G) \geq P_i(G) \geq \dots$$

where $P_i(G) = [P_{i-1}(G), G]P_{i-1}(G)^p$ for $i \geq 1$. If $P_c(G) = 1$ and c is the smallest such integer then G has *exponent-p class c*. A group with exponent-p class c is nilpotent and has nilpotency class at most c. In this paper the class of a group refers to its exponent-p class.

In its default mode, the algorithm takes as input a finite presentation $\{X \,|\, \mathcal{R}\}$, a prime p and a positive integer c, and yields a consistent power-conjugate presentation for the largest class c p-quotient of the group defined by $\{X \,|\, \mathcal{R}\}$. It works class by class. That is, having computed a consistent power-conjugate presentation $\mathcal{P}$ for the largest class k quotient P it goes on to compute one for the largest class $k+1$ quotient. Let d be the generator number of P; then P can be represented as F/R where F is a free group of rank d. The largest class $k+1$ quotient is a homomorphic image of the *p-covering group* $P^* = F/[R, F]R^p$ of P. The steps of the algorithm are:

- add new generators (tails) to $\mathcal{P}$ – corresponding to a generating set for $R/[R, F]R^p$;
- compute the other relations needed to define a power-conjugate presentation for the p-covering group on this extended generating set (compute tails);
- make the resulting presentation consistent;
- impose the relations in $\mathcal{R}$.

If the description of the group includes the condition that its exponent is p^m, then one has the additional step, which, in our context, is of central importance:

- compute and factor out the subgroup of P^* generated by the p^m-powers.

The final step of the algorithm eliminates the redundancies which arise among the new generators from consistency, imposition of defining relations, and exponent enforcement. Suppose that t new generators are added and that r independent relations are found between them. Then a consistent power-conjugate presentation for the largest class $k+1$ quotient has $t-r$ more generators than one for the largest class k quotient. Eliminating r of the added generators using the relations amounts to solving a system of r linear equations in t unknowns over the field of p elements. This step will be considered in more detail in Section 5.

The ANU p-Quotient Program provides access to our implementation of this algorithm. Various strategies for *collection* exist; a general discussion can be found in Sims (1994, §9.4). Inspired by the investigations of Leedham-Green & Soicher (1990), Vaughan-Lee (1990a) developed and implemented an algorithm to carry out collection from the left and also demonstrated that it performs significantly better for Burnside groups. His implementation is modelled on that developed by Havas & Nicholson (1976). In 1991, he and O'Brien further refined the implementation of this algorithm. In particular, they introduced tests to reduce the possibility of (integer) overflow and polished other parts of the implementation. The program uses the consistency tests of Wamsley (1974) as improved by Vaughan-Lee (1984).

Let G be a d-generator p-group of order p^n. The consistent power-conjugate presentations constructed have additional structure so that, for example, $\{a_1, \ldots, a_d\}$ is a generating set for G. For each a_k in $\{a_{d+1}, \ldots, a_n\}$, there is at least one relation whose right hand side is a_k. (We store the relations as commutators rather than conjugates.) One of these relations is taken as the *definition* of a_k. The power-conjugate presentations also have a *weight* function associated with them: a generator is assigned a weight corresponding to the stage at which it is added and this weight is extended to all normal words in a natural way. More formally, a function, ω, is defined on the generators of the power-conjugate presentation according to the following rules:

(i) $\omega(a_i) = 1$ for $i = 1, \ldots, d$;

(ii) if the definition of a_k is $a_i^p = a_k$, then $\omega(a_k) = \omega(a_i) + 1$;

(iii) if the definition of a_k is $[a_j, a_i] = a_k$, then $\omega(a_k) = \omega(a_j) + \omega(a_i)$.

Note that $\omega(a_n)$ is the class of G.

In our implementation, the construction of the tails is carried out in two parts to provide added flexibility. The first stage adds the appropriate new generators or tails and takes little time. For example, in computing a consistent power-conjugate presentation for $R(5, 4)$ at class 11, adding the new generators for the 10 692 relations $[a_j, a_i] = \ldots$ with $i \leq 5$ which are not definitions takes less than 0.1 seconds. However, the second stage where the rest of the tails, or the right hand sides of other relations which are needed for collection, are computed is quite time consuming. In Section 5 we state theorems which guarantee that some of these tails are trivial and hence need not

be computed; in this way, we obtain significant time reductions. A detailed description of the method of constructing tails can be found in Newman, Nickel & Niemeyer (1995).

3 Determining the exponent of a p-group

The study of Burnside groups requires a practical algorithm for verifying that exponent laws hold or, alternatively, enforcing exponent laws. Essentially, the same methods can be used to determine the exponent of a p-group.

As shown in Sims (1994, p. 563), the Higman Lemma provides a basis for a practical algorithm. It states that a group of class c described by a power-conjugate presentation has exponent p^m if every normal word of weight at most c has order dividing p^m. Moreover, since the powers of an element of order p^m have order dividing p^m, it suffices to consider normal words with leading coefficient 1.

For example, there is a finite presentation with 4 defining generators and a relation set of 1016 fourth powers which defines a group with a largest class 10 quotient of order 2^{422}. In this case we have to consider 376 727 normal words; if the fourth power of (the element defined by) each of these is trivial, then the group has exponent four — and is $R(4,4)$.

Vaughan-Lee (1984) reduced the number of necessary normal words using the following observations:

- since all the elements in a conjugacy class have the same order, it suffices to consider one element in each conjugacy class;
- if the normal closure of a generator a having order p has class at most $p-1$ and certain weight conditions hold on u, then $(ua)^{p^m} = u^{p^m}$ and so ua need not be considered.

The algorithm uses filters based on these ideas to reduce the list of normal words by filtering out some conjugates and some which satisfy this normal closure condition. For example, it uses the resulting filtered list of 143 134 words to verify that the above presentation defines a group of exponent four. For $R(5,4)$, verification is still a serious bottle-neck — there are 161 117 868 normal words in the putative power-conjugate presentation and the program would compute the fourth power of 83 905 543 words. We describe in Section 4 how the use of automorphisms allows us to get much more practical lists of words to test. They consist of fewer words and a larger proportion of words whose fourth powers are fast to calculate.

The general purpose algorithm for computing the exponent of a finite group first determines representatives of the conjugacy classes and then takes the least common multiple of their orders. The large number of conjugacy classes in p-groups frequently poses problems for this algorithm.

We now describe how the p-quotient algorithm can be used to determine the exponent of a p-group. This method, not surprisingly, is much more efficient than the general purpose approach. The basic idea is that the exponent of a p-group can be

computed at the same time as a consistent power-conjugate presentation is being computed. Suppose that P is the largest class k quotient of the given group, G, and Q is the largest class $k+1$ quotient. Suppose further that we have computed a consistent power-conjugate presentation for Q and know that the exponent of P is e. Clearly the exponent of Q is either e or pe. Thus it suffices to compute the e-th powers in Q for the filtered list of normal words of the consistent power-conjugate presentation for Q. If one of these is non-trivial, the exponent of Q is pe and we can stop calculating. If they are all trivial, then, by the Higman Lemma and the Vaughan-Lee filters, Q has exponent e. It would be useful to find even sharper criteria that work for p-groups in general. Since the program can also compute automorphism groups, the ideas used in the context of Burnside groups could be extended.

Since the exponent of a p-group of class at most $p-1$ is the maximum of the orders of the defining generators, it is particularly easy to compute the exponent for p-groups of class at most $p-1$. For a proof of this result, see M. Hall (1959, Chapter 12). If the class of the group is larger than $p-1$, we use the Higman Lemma and the Vaughan-Lee filters to get a list of words whose p^m-th powers suffice to determine the exponent of the p-covering group, P^*. These are applied to the relevant quotient Q of P^*. Of course, once one finds an element of order greater than the exponent of P, we know the exponent of Q; otherwise Q and P have the same exponent. In summary:

1. Compute the maximum, e, of the orders of the defining generators of G.
2. If c is at most $p-1$, return e as the exponent of G.
3. Otherwise, construct the largest p-quotient of G having class $p-1$.
4. For each class $k = p, \ldots, c$, first construct the largest class k quotient of the group which satisfies the defining relations of G. Compute the e-th powers of the words in the filtered list of normal words. If one of these is non-trivial, stop processing this list and update e to have value pe. Return e as the exponent of the class k quotient.

Versions of this algorithm are implemented in both GAP and Magma and perform extremely well. For example, Magma takes 4 seconds to determine that the class 10 5-quotient of the free group on 2 generators, a group of order 5^{520}, has exponent 5^{10}. Our algorithm also forms the basis of one to find the exponent of finite nilpotent groups, since the exponent of each Sylow p-group can be found using this strategy.

4 Using automorphisms

All the elements in an automorphism class of a group have the same order, so to determine or verify the exponent of a group it suffices to evaluate the order of one element in each automorphism class. It may not be easy, or worthwhile in cost terms, to determine the full automorphism group. However, as we report, it can pay to use

a subgroup of automorphisms and consider (at least) one element from each class of elements relative to this group of automorphisms.

The groups $R(d, e)$ are relatively free (see Hanna Neumann, 1967, p. 19) so every mapping from a minimal generating set into the group lifts to an endomorphism. Since $G = R(d, e)$ has $G/\mathcal{P}_1(G)$ as an elementary abelian quotient of rank d, every element of $\mathrm{GL}(d, p)$ lifts to an automorphism of G. We take a generating set of $\mathrm{GL}(d, p)$ and consider the group of automorphisms which results from lifting this. With this group acting, every class of elements of G which does not lie in $\mathcal{P}_1(G)$ contains an element of the form $a_d u$ where $u \in \mathcal{P}_1(G)$. Consider the list of normal words provided by the Higman Lemma and the Vaughan-Lee filters. Take the sublist consisting of those elements which have the form $a_d u$. It suffices to take this sublist because the p^m-th power subgroup of the p-covering group is the subgroup generated by the closure under the automorphisms of the p^m-th powers of the words in this sublist. Not only is this list shorter but it consists of elements whose relevant powers are, on average, much faster to calculate. At first sight, we could choose any one of $\{a_1, \ldots, a_d\}$ as the leading term; however, the interaction of the various filters produces the shortest filtered list if we choose a_d.

In the case of the class 11 quotient of $R(5, 4)$ this approach gives a list of 3 503 197 elements. It is possible to make the list significantly shorter by considering the classes of elements that do not lie in $\mathcal{P}_2(G)$ or later terms of the central series. In verifying that the group presented by our putative presentation for $R(3, 5)$ has exponent five, we considered classes of elements that do not lie in $\mathcal{P}_3(G)$. This reduces the length of the list of normal words from 2 099 401 to 24 062.

5 Use of consequences

Another useful way of reducing the space and time requirements of computing power-conjugate presentations for Burnside groups is to use some of the consequences of the exponent law.

For the cases we considered, it is important to control the size of the intermediate presentations generated when we compute tails of various weights. The elimination process might require a lot of space especially towards the end of a calculation when the number r of relations found is more than half the number t of new generators added. A rough calculation suggests that it could need at least space for $(t/2)^2/p$ words. Thus, in the class 12 step of the calculation for $R(5, 4)$, where 10 692 new generators are added and 10 652 independent relations are found, 50 MB additional space might be needed for the elimination. Consequences of the exponent condition can be used to give the system of equations a block structure which can be used to reduce dramatically space requirements. We outline this below.

For example, in groups of prime exponent we have the $(p-1)$-Engel congruence that $[y, (p-1)x]$ is a product of commutators of weight at least $p+1$ (see Vaughan-Lee,

1993, 2.4.8). We also use the multilinear form of this identity: namely,

$$\sum_{\sigma \in perm\{1,\ldots,p-1\}} [y, x_{1\sigma}, \ldots, x_{(p-1)\sigma}]$$

is a product of commutators of weight at least $p+1$ with each of $y, x_1, \ldots, x_{p-1}$ as an entry. (This is implicit in the proof of 2.4.8.)

For instance, the class 11 quotient of $R(3,5)$ has order 5^{1855}; its 5-covering group has order 5^{5565} and the class 12 quotient has order 5^{2133}. We use instances of the 4-Engel congruence, with y replaced by generators all having weight one of 5 through 8 and x replaced by generators of weight 1, to generate a total of 657 redundancies among the new generators; their automorphism closure yields 2061 additional redundancies. The cost of generating the consequences is 70 seconds; calculating their closure takes a further 160 seconds. The closure of a single instance each of the exponent law and of consistency provides the remaining 714 redundancies.

For groups of exponent four we use four results:

(1) for $d \geq 3$ the class of a d-generator group is at most $3d-2$ (Gupta & Newman, 1974; also see Vaughan-Lee, 1993, 6.4.1);

(2) every commutator of weight at least 6 in which an element occurs four times as an entry is trivial (Vaughan-Lee, 1993, 6.3.20);

(3) the commutator $[a,b,c,x,x,x]$ is a product of commutators with entries from $\{a,b,c,x\}$ each with at least two entries a, two b's, two c's and three x's (strong form of Vaughan-Lee, 1993, 6.3.8);

(4) for $d \geq 3$ the $(3d-3)$-th term of the lower central series has rank $2\binom{d+1}{3} + \binom{d+1}{2}$ (Vaughan-Lee, 1993, 6.5.2 and Lemma 6.5.4).

Result (1) ensures that we can stop the computation at the end of class $3d-2$. Result (2) is used in two ways. First, it is used to impose a block structure on the equation system. For example, the consistent power-conjugate presentation for the class 11 quotient of $R(5,4)$ has 110 generators of weight 11 and 264 generators of weight 10; so the tails step adds 550 new generators of weight 12 and 1210 generators of weight 11. Among the new generators of weight 12 there are 200 which have an entry occurring 4 times and 12 whose last 3 entries are the same — these are trivial by result (3). In the context where sufficient automorphisms are included to induce the full general linear group on the class 1 quotient, the automorphism closure of these $200+12$ generators has dimension 490. Thus the equation system has a block of 490 equations in 550 unknowns. The new generators of weight 11 which have an entry occurring 4 times or which have the same last 3 entries generate under closure 1145 additional relations. Continuing in this way provides further blocks for each weight. The complete system can now be readily solved using these subsystems. Second, result (2) is used in calculating tails. In the same example there are 335 610 tails to calculate

at weight 12. Calculating these in the usual way takes 900 seconds. By (2) the $[a_j, a_i]$ with at least four entries the same are trivial. They are easy to recognise and filter out; the remaining 86 892 are processed in 240 seconds.

Result (4) ensures that it suffices to verify the consistency and exponent of the presentation for the class $(3d-4)$-quotient. For $d = 5$ this reduces the number of fourth powers to evaluate to 6 599 245 in a smaller class context; as reported in Section 4, we use automorphisms to reduce the number of powers to 3 503 197.

6 The implementation

The implementation of the p-quotient algorithm described in Havas & Newman (1980) was written in Fortran. (At that time, it was called the nilpotent quotient program.) A part of this implementation was available in the computational algebra system, CAYLEY (see Cannon, 1984). A translation of this code to C was carried out by machine in 1987. In 1991, O'Brien used some of this code as one component in developing a new C implementation.

The resulting program is now known as the ANU p-Quotient Program. It contains about 22 000 lines of (commented) code; about 8000 of these implement the p-quotient algorithm. The program also offers access to implementations of the following:

- the p-group generation algorithm described in Newman (1977) and O'Brien (1990);
- an algorithm to decide isomorphism of p-groups described in O'Brien (1994);
- an algorithm to compute the automorphism group of a p-group described in O'Brien (1995).

The program is menu-driven and provides two levels of control. The Basic Menu is designed for routine use; the Advanced Menu provides high levels of control to an experienced user. The program is available both as a stand-alone by anonymous FTP from ftpmaths.anu.edu.au and:

- as a share package with GAP (see Schönert *et al.*, 1994);
- as part of Magma (see Bosma & Cannon, 1994);
- as part of Quotpic (see Holt & Rees, 1993).

Many of the features of the program are accessible via these systems.

The Advanced Menu provides options for collecting words and evaluating commutators in the defining and power-conjugate presentation generators. A user may also supply automorphisms, which can be used in enforcing the exponent law. Matrices which describe the action of these automorphisms on sections of the central series of the group can be produced and used (in GAP or Magma, for example) to analyse module structure. The exponent law option allows the user to construct:

(1) the filtered list of normal words obtained by applying the Higman Lemma and the Vaughan-Lee filters;

(2) the shorter list obtained by assuming the relevant general linear group is induced on the class 1 quotient;

(3) lists obtained where each word has a specific user-supplied prefix and each additional letter in the word has a weight lying between user-supplied bounds.

An important practical feature in closing relations under the action of automorphisms is the order in which the consequences of an exponent law are processed. These relations are stored in a queue whose entries are then closed under the action of the induced automorphisms. We found that the time taken to close is reduced if we first process all relations which have "short" length and process the longer relations only when all short relations are processed. The difference in cost can be attributed to the time taken to echelonise the longer relations. The *queue factor* of a relation is the ratio of its length to the maximum number of generators (the number of tails added) for that class. The program first processes those relations whose queue factor is less than a selected value. We used queue factors ranging from 0.05 through 0.15 for the computations reported here.

7 Computing presentations

Here we report on the use of the ANU p-Quotient Program in computing consistent power-conjugate presentations for various Burnside groups.

7.1 Groups of exponent 5

A consistent power-conjugate presentation for $R(2,5)$, a group of order 5^{34}, can be computed using the default implementation in 6 seconds. Sims (1994, §11.7) reports in some detail on aspects of this computation.

Let $W(3,5)$ be the free 3-generator Lie algebra in the variety of Lie algebras determined by $5x = 0$ and the series of multilinear identities $K_n = 0$ introduced by Vaughan-Lee (1985). Havas, Newman & Vaughan-Lee (1990) show that $W(3,5)$ has dimension 2282 and class 17. Vaughan-Lee (1990b) used this result to prove that the order of $R(3,5)$ is 5^{2282}. Let $L(3,5)$ be the 3-generator Lie ring associated with the lower central series of $R(3,5)$, and let $N(a,b,c)$ be the normal subgroup of $R(3,5)$ generated by commutators with multiweight (d,e,f) where $d > a$ or $e > b$ or $f > c$. He constructed a consistent power-conjugate presentation for $R(3,5)/N(5,6,6)$, a group of order 5^{2180}, and used this to deduce that $L(3,5)$ is equal to $W(3,5)$ and hence established the order of $R(3,5)$.

In 1991, we computed for the first time a consistent power-conjugate presentation for $R(3,5)$ itself using the ANU p-Quotient Program, where we supplied sufficient automorphisms to induce the action of the general linear group $\mathrm{GL}(3,5)$ on the class 1

quotient. We found that computing the extensions of a generating set of automorphisms to act on the group is the most expensive part of the computation. We investigated the impact of two generating sets on the performance of the algorithm. The first was the "standard" generating set:

$$\begin{array}{llllllll} \alpha_1: & a_1 & \longmapsto & a_1^2\,, & \alpha_2: & a_1 & \longmapsto & a_1^4 a_3\,, \\ & a_2 & \longmapsto & a_2 & & a_2 & \longmapsto & a_1^4\,, \\ & a_3 & \longmapsto & a_3 & & a_3 & \longmapsto & a_2^4\,. \end{array}$$

The second was the following:

$$\begin{array}{llllllll} \alpha_1: & a_1 & \longmapsto & a_1 a_2\,, & \alpha_2: & a_1 & \longmapsto & a_2\,, \\ & a_2 & \longmapsto & a_2 & & a_2 & \longmapsto & a_3\,, \\ & a_3 & \longmapsto & a_3 & & a_3 & \longmapsto & a_1^2\,. \end{array}$$

The time taken to compute extensions for the elements of the second set was about 40% of the cost for the first. We considered a few other generating sets without obtaining further improvement.

Task	$R(3,5)$	$R(5,4)$
Tails	10 592	3711
Automorphism extensions	24 290	1080
Elimination	2025	3908
Engel evaluation	1526	—
Closure	2132	636
Total	40 788	9348

Table 1: Times to construct power-conjugate presentations

In Table 1 we report the time taken to construct a power-conjugate presentation for $R(3,5)$. When the presentation for the class 17 quotient was constructed, we verified that it is consistent in 10 300 seconds. Finally, as reported in Section 4, we verified that it has exponent five by evaluating 24 062 normal words in 196 000 seconds. The maximum workspace used was about 150 MB.

7.2 Groups of exponent 4

Completely routine use of the ANU p-Quotient Program yields a consistent power-conjugate presentation for $R(4,4)$ and a proof in 2470 seconds. The class 8 quotient of this presentation can be shown to be consistent in 5 seconds and to have exponent four in 28 seconds by evaluating 11 533 fourth powers.

A consistent power-conjugate presentation for $R(5,4)$ was first computed by Newman in 1989 using a special purpose version of the original Fortran implementation. Features of the prime 2 were used to reduce the amount of space required. A consistent power-conjugate presentation for $R(5,4)$ can now be readily computed using the methods described here and in Table 1 we report timings for this task. It takes 3000 seconds to verify that the class 11 quotient is consistent. We verified that the group has exponent four by evaluating 3 503 197 normal words in 37 000 seconds. The maximum workspace used was about 40 MB.

Since Sanov (1940) has proved that all groups of exponent four are locally finite, $R(d,4)$ is the largest d-generator group of exponent four (this is often denoted $B(d,4)$).

8 Regenerating presentations

Once a consistent power-conjugate presentation for a group has been determined, one can read off from the output files, including the presentation, produced in the process information which makes regenerating the presentation cheaper (also noted in Newman, 1993).

For example, the process of generating $R(4,4)$ shows that 1016 fourth powers are enough to define the group. Using these powers together with a set of 213 instances of consistency yields a consistent power-conjugate presentation in 95 seconds. Imposing those relations is much quicker than using list (1) or even list (2). In this case using automorphisms and consequences does not give significant further improvements. However for R(5,4) they do play an important role, particularly in ensuring relatively cheap solution of the underlying linear equation systems.

This approach also has the advantage for the groups considered in this paper that one does not need to store the consistent power-conjugate presentation because it can be regenerated reasonably quickly. For example, the data file for $R(5,4)$ is about 22 MB while a commented input file is about 14 KB. This makes transmission much easier, and has the additional advantage that it is possible to see that only 4th powers and their consequences were used to generate the presentation. This last point is important because there is no way of recognising this from the presentation itself.

References

Wieb Bosma and John Cannon (1994), *Handbook of* Magma *functions*. School of Mathematics and Statistics, Sydney University.

John J. Cannon (1984), "An Introduction to the Group Theory Language, Cayley", *Computational Group Theory*, (Durham, 1982), pp. 145–183. Academic Press, London, New York.

N.D. Gupta and M.F. Newman (1974), "The nilpotency class of finitely generated groups of exponent four", *Proc. Second Internat. Conf. Theory of Groups*, Lecture Notes in Mathematics, **372**, (Canberra, 1973), pp. 330–332. Springer-Verlag, Berlin, Heidelberg, New York.

Marshall Hall, Jr. (1959), *The Theory of Groups*. Macmillan Co., New York.

George Havas and M.F. Newman (1980), "Application of computers to questions like those of Burnside", *Burnside Groups*, Lecture Notes in Math., **806**, (Bielefeld, 1977), pp. 211–230. Springer-Verlag, Berlin, Heidelberg, New York.

George Havas, M.F. Newman and M.R. Vaughan-Lee (1990), "A nilpotent quotient algorithm for graded Lie Rings", *J. Symbolic Comput.*, **9**, 653–664.

George Havas and Tim Nicholson (1976), "Collection", SYMSAC '76, *Proc. ACM Sympos. symbolic and algebraic computation*, (New York, 1976), pp. 9–14. Association for Computing Machinery, New York.

D.F. Holt (1984), "The Calculation of the Schur Multiplier of a Permutation Group", *Computational Group Theory*, (Durham, 1982), pp. 307–318. Academic Press, London, New York.

Derek F. Holt and Sarah Rees (1993), "A graphics system for displaying finite quotients of finitely presented groups", *Groups and Computation*, Amer. Math. Soc. DIMACS Series, **11**, (DIMACS, 1991), pp. 113–126.

C.R. Leedham-Green and L.H. Soicher (1990), "Collection from the left and other strategies", *J. Symbolic Comput.*, **9**, 665–675.

I.D. Macdonald (1974), "A computer application to finite p-groups", *J. Austral. Math. Soc. Ser. A*, **17**, 102–112.

Hanna Neumann (1967), *Varieties of Groups*. Springer-Verlag, Berlin, Heidelberg, New York.

M.F. Newman (1977), "Determination of groups of prime-power order", *Group Theory*, Lecture Notes in Math., **573**, (Canberra, 1975), pp. 73–84. Springer-Verlag, Berlin, Heidelberg, New York.

M.F. Newman (1993), "Groups of exponent 8 are different", *Bull. London Math. Soc.*, **25**, 263–264.

M.F. Newman, Werner Nickel and Alice C. Niemeyer (1995), "Descriptions of groups of prime-power order", submitted.

E.A. O'Brien (1990), "The p-group generation algorithm", *J. Symbolic Comput.*, **9**, 677–698.

E.A. O'Brien (1994), "Isomorphism testing for p-groups", *J. Symbolic Comput.*, **17**, 133–147.

E.A. O'Brien (1995), "Computing automorphism groups of p-groups", Wieb Bosma and Alf van der Poorten (Eds.), *Computational Algebra Number and Number Theory*, (Sydney, 1992), pp. 83–90. Kluwer Academic Publishers, Dordrecht.

I.N. Sanov (1940), "Solution of Burnside's problem for exponent 4", *Leningrad State Univ. Ann.*, **10**, 166–170.

Martin Schönert *et al.* (1994), GAP – *Groups, Algorithms and Programming*. Lehrstuhl D für Mathematik, RWTH, Aachen.

Charles C. Sims (1994), *Computation with finitely presented groups*. Cambridge University Press.

M.R. Vaughan-Lee (1984), "An Aspect of the Nilpotent Quotient Algorithm", *Computational Group Theory*, (Durham, 1982), pp. 76–83. Academic Press, London, New York.

M.R. Vaughan-Lee (1985), "The restricted Burnside problem", *Bull. London Math. Soc.*, **17**, 113–133.

M.R. Vaughan-Lee (1990a), "Collection from the left", *J. Symbolic Comput.*, **9**, 725–733.

M.R. Vaughan-Lee (1990b), "Lie Rings of groups of prime exponent", *J. Austral. Math. Soc. Ser. A*, **49**, 386–398.

M.R. Vaughan-Lee (1993), *The Restricted Burnside Problem* (2nd edition). London Math. Soc. Monographs, (N. S.), **5**. Oxford University Press, Oxford.

J.W. Wamsley (1974), "Computation in nilpotent groups (theory)", *Proc. Second Internat. Conf. Theory of Groups*, Lecture Notes in Math., **372**, (Canberra, 1973), pp. 691–700. Springer-Verlag, Berlin, Heidelberg, New York.

E.I. Zel'manov (1991), "Solution of the Restricted Burnside problem for 2-groups", *Math. Sb.*, **182**, 568–592.

Centre for Mathematics and its Applications
School of Mathematical Sciences
Australian National University
Canberra, ACT 0200
Australia
newman@maths.anu.edu.au

Lehrstuhl D für Mathematik
RWTH
Templergraben 64
52062 Aachen
Germany
obrien@math.rwth-aachen.de