Computing normal subgroups

Alexander Hulpke*

School of Mathematical and Computational Sciences
University of St. Andrews
The North Haugh, St. Andrews, Fife KY16 9SS,
United Kingdom

ahulpke@dcs.st-and.ac.uk
http://www-gap.dcs.st-and.ac.uk/~ahulpke

Abstract

This note presents a new algorithm for the computation of the set of normal subgroups of a finite group. It is based on lifting via homomorphic images.

1 Introduction

Next to a composition series, the structure of a finite group is reflected by the lattice of its normal subgroups. It indicates, for example, possible epimorphic images or decompositions as (sub)direct products. Furthermore it contains all characteristic subgroups. Contrasting this structural prominence, however, the problem never seems to have been considered seriously in the literature and the best algorithm known to the author is the naïve approach to take normal closures of conjugacy classes. This requires a priori knowledge of the conjugacy classes. The alternative approach presented here avoids this requirement and performs much quicker for larger groups (even if the classes are known for free). This will permit to compute the normal subgroups in reasonable time even for fairly large groups, for which for example the conjugacy classes could not be computed. Evidence for this is given in the last section.

The algorithm described below does not explicitly refer to a groups representation, but requires several auxiliary routines which in turn may make use of the representation. It can be applied whenever all these routines can be applied for a given group. For all those required routines (element test, composition series/chief series, normal intersection, computation of presentations) efficient algorithms are known and implemented for permutation groups [19] or groups given by a polycyclic presentation.

2 Setup

Assume that G is a finite group for which we want to determine the normal subgroups. We start with a chief series for G. Such a series can be computed [13, 11] by forming the intersection of conjugates in a composition series of G (this yields a series of normal subgroups) and refining elementary abelian subfactors using the MeatAxe [18]. For permutation groups alternatively the composition series algorithm can be modified to compute normal subgroups in the first place [3].

Using the homomorphism principle we compute the normal subgroups of G inductively along this series, in a step from N_i to $N_{i+1} < N_i$ computing the normal subgroups of G/N_{i+1} from those of G/N_i. After the last step for $N_{i+1} = \langle 1 \rangle$ this yields the normal subgroups of G.

Remark 1. Computing representations for factor groups is not only very hard in the general case; there are situations where all faithful representation of a factor are much "larger" than the representation of G we started with [7, 17]. To overcome this problem we always compute with representatives in G instead of elements of a factor group and replace subgroups of a factor by their full preimages in G. The element order of $N_i g$ in the factor can be obtained as the smallest k such that $g^k \in N_i$. The readers should convince themselves that all calculations performed in the next two sections are possible with representatives and preimages as well and yield representatives respectively preimages of the results in the factor group. (Note, however, that in many cases it is still possible to compute in reasonable time a faithful permutation representation of reasonable degree for a factor group if it was required. We don't require it in this paper.)

It is sufficient to describe a single lifting step. In this factor $H = G/N_{i+1}$ the image $M = N_i/N_{i+1}$ is a minimal normal subgroup, that is there is no proper subgroup $\langle 1 \rangle < K < M$ such that K is normal in H. To simplify notation we assume without loss of generality that $N_{i+1} = \langle 1 \rangle$ and $G = H$.

If B is a normal subgroup of G which does not contain M, then $B \cap M = \langle 1 \rangle$ (as the intersection is a normal subgroup contained in M) and $A = \langle M, B \rangle$ is a normal subgroup containing M. B is a normal complement to M in A.

Assuming by induction that all normal subgroups A containing M are known a priori, it is therefore sufficient to show how for a given $A > M$ all possible B can be determined. This will give rise to all normal subgroups of G via a loop over all these subgroups.

It is well known [6, Thm. 4.3.A(iii)] that as a minimal normal subgroup M must be the direct product of copies of a simple group T. Let n denote the number of direct factors: $M = \overset{n}{\underset{i=1}{\times}} T_i$, $T_i \cong T$. Here we have to distinguish two cases,

*Supported by EPSRC Grant GL/L21013

depending on whether T is abelian or not.

3 Abelian case

If T is abelian, M is an elementary abelian group – in other words a vector space over a finite field. A basis $m_1, \ldots, m_n$ of M corresponds to a series of subgroups $U_i = \langle m_j \mid j \leq i \rangle$ with

$$\langle 1 \rangle = U_0 < U_1 < \cdots < U_{n-1} < U_n = M.$$

Element tests in these subgroups permit to decompose an element of M with respect to the chosen basis.

If a normal complement B to M in A exists, A is the direct product of M with B and M therefore must be central in A. In this case all complements to M in A are parametrized (if they exist) by the group of *1-Cocycles*:

$$Z^1(A/M, M) := \{\gamma \colon A/M \to M \quad \mid \quad (fg)\gamma = (f\gamma)^g(g\gamma) \text{ for all } f, g \in A/M\}$$

If C is a fixed complement, the complement corresponding to $\gamma \in Z^1$ is of the form

$$\{c \cdot ((cM)\gamma) \mid c \in C\}.$$

If a presentation for A/M is known, it is possible to compute C as the solution of an inhomogeneous system of linear equations [5]. If no complements exist, this system has no solutions. Z^1 arises from the corresponding homogeneous system. A presentation for A/M can either be computed directly from a representation for A/M or by first computing a presentation for A and adding words for generators of M. For permutation groups a presentation can be computed either via a stabilizer chain [4] or from a composition series [1, 14]. The latter usually yields a better presentation and can also be modified to yield a presentation for A/M without the need to compute a permutation representation of this group first.

The complements parametrized by Z^1 are all normal in A but not necessarily normal in G. We therefore have to discard those complements that are not normal in G.

Problems may occur if Z^1 is large, that is there are many subgroups of A that complement M. If this happens A is usually close to abelian. The worst case certainly is if A is elementary abelian:

Lemma 2. *If A is elementary abelian, $|A| = p^d$ and $|M| = p^n$ there are $(p^n)^{d-n}$ complements to M in A.*

Proof. Suppose we have a basis $\mathfrak{B}$ for M and want to extend it to a basis of A. There are $p^d - p^k$ vectors in A which are not in a given k-dimensional subspace, therefore we can extent $\mathfrak{B}$ in $x = (p^d - p^n)(p^d - p^{n+1}) \cdot \cdots \cdot (p^d - p^{d-1})$ possible ways. The added vectors always span a complement (of dimension $d - n$) but there are $y = |\mathrm{GL}_{d-n}(p)| = (p^{d-n} - 1)(p^{d-n} - p) \cdot \cdots \cdot (p^{d-n} - p^{d-n-1})$ possible bases for every complement. Taking the quotient x/y yields the number of complements. □

Usually only very few of these complements are invariant under G. In the case of an elementary abelian A with $(p^n)^{d-n} > 20$ we therefore don't use Z^1. Instead we compute the lattice of all G-submodules of A using the algorithm in [16]. Unless A is central in G (in this case there simply is a large number of normal subgroups in G) this usually yields but a few submodules. Of these we take those submodules which intersect trivially with M and have the right size.

For a solvable A, the same approach can be used with the invariant subgroups algorithm of [10] generalizing the submodule computation [16]. (The size restriction to potential complements can be applied to avoid creating subgroups too small to be a complement.) In the examples considered so far this could always avoid running through a large Z^1.

To obtain finally all normal subgroups of G, we perform the complement computation for all A in which M is central.

4 Nonabelian case

We now consider the case of nonabelian T and M. As a normal complement, B centralizes M.

The centre of M is trivial. Therefore B is the centralizer in A of M. We can obtain all B by intersecting all A with $C = C_G(M)$.

Remark 3. The system GAP [9] contains a nearly-linear time algorithm to compute centralizers of normal subgroups in permutation groups. Here however G may be a proper factor group for which we do not necessarily have a faithful permutation representation. Therefore we cannot simply use this centralizer algorithm but will describe another method that also works in factor groups when computing with repesentatives and preimages.

To compute this centralizer, we proceed as follows: G acts transitively on $\{T_1, \ldots, T_n\}$, because M is a minimal normal subgroup. Let $S := N_G(T_1)$. As the orbit of T_1 under G is of length n, this normalizer (whose full preimage is simply the normalizer of the preimage of T_1) has index n and therefore can be computed quickly by either a specific normalizer routine or even a simple Orbit-Stabilizer algorithm.

Lemma 4. *Let $t \in T_1$, $D = C_S(t)$ and $E := \bigcap_{g \in G} D^g$. Then $C = C_G(M) = E$.*

Proof. Obviously C centralizes T_1 and therefore $C \leq D$. As $C \lhd G$ we have $C \leq \bigcap_{g \in G} D^g = E$. Vice versa observe that for $s \in S$ we have $D^s = C_S(t^s)$. As T is simple, the t^s generate T_1 and therefore $E \leq C_G(T_1)$. For every other k there is a $g \in G$ such that $T_1^g = T_k$ and thus $D^g = C_{N_G(T_k)}(t^g)$. Again the t^g run for different choices of g through a class of T_k and therefore $E \leq C_G(T_k)$ for all k. As $M = \langle T_k \rangle$, we have $E \leq C$. □

As T is simple and nonabelian, it contains involutions (elements of order 2) [8]. As every element of even order has a power which is an involution, a random search will very quickly find an involution $t \in T_1$. It is shown in [12] that if T is not of Lie type of characteristic 2 the probability for a random element being of even order is at least 1/4.

For the computation of $C_S(t)$, we use the following generalization of the "dihedral group trick" [15]:

Lemma 5. *Let G be a group and $t \in G$ be an involution. For every $g \in G$ let*

$$o_t(g) := \left|tg^{-1}tg\right| = \left|t \cdot t^g\right|.$$

a)(J. Bray,*[2]) For $g \in G$ we have*

$$z_t(g) := \underbrace{g^{-1}tgtg^{-1}t\cdot\cdots}_{2\cdot o_t(g)-1 \text{ letters}} \in C_G(t).$$

b)For $g \in C_G(t)$ we have $z_t(g) = g^{-1}$, that is $z_t\colon G \to C_G(t)$ is surjective.
c)(R. Parker*) If $o_t(g)$ is odd, z_t is injective on the coset $C_G(t)\cdot g$. If g is taken from this coset with uniform distribution, $z_t(g)$ is from $C_G(t)$ with equal distribution.*

Proof. a) We have

$$1 = \left(tg^{-1}tg\right)^{o_t(g)} = \underbrace{tg^{-1}tg\cdot tg^{-1}tg\cdot\cdots\cdot tg^{-1}tg}_{4\cdot o_t(g) \text{ letters}}$$

and therefore

$$t = \underbrace{g^{-1}tgtg^{-1}t\cdot\cdots\cdot tg^{\pm}tg^{\mp}}_{=z_t(g)}\cdot t\cdot \underbrace{g^{\pm}tg^{\mp}t\cdot\cdots\cdot tg^{-1}tg}_{=z_t(g)^{-1}}$$

b) For $g \in C_G(t)$ we have $o_t(g) = 1$.
c) Let $c \in C_G(t)$. Then $o_t(g) = o_t(cg)$ and thus

$$\begin{aligned} z_t(cg) &= g^{-1}\cdot c^{-1}tc\cdot gtg^{-1}\cdot c^{-1}tc\cdot gt\cdot\cdots\cdot tg^{-1}c^{-1} \\ &= g^{-1}tgtg^{-1}tgt\cdot\cdots\cdot tg^{-1}c^{-1} = z_t(g)c^{-1}. \end{aligned}$$

Thus for $c \in C_G(t)$ arbitrary, $z_t(c^{-1}\cdot z_t(g)\cdot g) = c$. □

The $z_t(s)$ for a few random $s \in S$ will usually generate $C_S(t)$. To ensure that the elements found indeed generate the full centralizer we use the homomorphism $\varphi\colon M \to T_1 = T$. For permutation groups the routines for composition series applied to M (respectively its full preimage if M is a factor group) yield a faithful small degree representation for $M/\ker\varphi$ as well as this homomorphism effectively.

We compute $m := [T : C_T(t\varphi)]$. As T is a simple group in a small degree permutation representation this is done quickly.

If a data base of simple groups is available one can alternatively identify the isomorphism type of T (for example via the size, which often, but not always permits to identify the type) which usually can be done very quickly. If the size of an involution centralizer is determined uniquely by the group (for example if there is only one class of involutions) this size might be taken, eliminating the need to compute $C_T(t\varphi)$. For practical purposes however it seems that the database lookup takes longer than the calculation of $C_T(t\varphi)$ unless T is exceedingly large.

The conjugation action of S induces (as S stabilizes T_1) via φ automorphisms of T. These may fuse the class of $t\varphi$ with k other classes of the same size. Therefore we have $[S : C_S(t)] = k\cdot m$ for a natural number k. If we know k we know the index of $C_S(t)$ and thus know when we have reached the full centralizer.

Remark 6. This fusion of involution classes happens rarely. A search over the character table library in GAP4 [9] found just 4 simple groups for which such automorphisms exist, namely $Sp_4(4)$, $O_8^+(2)$, $O_8^+(3)$ and $F_4(2)$.

To avoid unnecessary conjugacy tests, we proceed as follows to compute $C_S(t)$:

1. Compute $m := [T : C_T(t\varphi)]$.
2. For 10 random elements $s \in S$ compute $z_t(s)$. Let U be the subgroup generated by these $z_t(s)$. If $[S : U] = m$ we know that $k = 1$. Stop.
3. Let $orb := [t]$. For every element $o \in orb$ execute steps 4 and 5.
4. For every generator s of S execute step 5.
5. Let $i := t^s$. If for every $o' \in orb$ we have that $i\varphi$ is not conjugate under T to $o'\varphi$, then add i to orb.
6. End the loops of step 4 and 5. Let $k = |orb|$.
7. Let $U := \langle U, z_t(s)\rangle$ for a random $s \in S$.
8. If $[S : U] > k\cdot m$ go to step 7.

The loops in steps 3-5 implement a standard orbit algorithm on conjugacy classes, given by preimages of their representatives. They compute the orbit of the class of $t\varphi$.

The computation stops when U equals $C_S(t)$. It finishes because we will encounter all elements of $C_S(t)$ by lemma 5 b). If a is the minimum possible number of generators for $C_S(t)$, usually the same number of random elements suffice to generate $C_S(t)$.

Once $C_S(t)$ has been obtained we compute $C = C_G(M)$ by lemma 4. We then compute the intersections of C with all normal subgroups $A > M$ to obtain the normal subgroups of G. Note that the intersection of normal subgroups in permutation groups is possible in nearly-linear time [19, Lemma 4.6].

Again, it should be stressed that if G is a proper factor of a larger group no representation of G is needed, but all calculations can be performed using preimages of subgroups and representatives of elements.

5 Examples

The algorithm has been implemented by the author in GAP4 [9]. Table 1 gives runtimes (seconds on a 200MHz PentiumPro machine under Linux) for various permutation groups in comparison to the naïve class closure algorithm mentioned in the first section.

In the example $Sp_4(4).4$ the class of the involution chosen during the example run in the normal subgroup $Sp_4(4)$ fuses under the full group. This gives an example for class fusion in the nonabelian case.

The examples show that the new algorithm is almost always performs better than the naïve algorithm, even when neglecting the time to compute the conjugacy classes. The performance gain becomes larger if the group has relatively few normal subgroups compared to the number of its conjugacy classes. Extreme cases are almost simple groups.

For larger groups already the calculation of the classes can take very long or be completely unfeasible. The new algorithm then still computes the normal subgroups without problems.

A shortcoming remains the case of a large Z^1 in the abelian case. So far, however, I have not encountered groups in which this case could not be treated using the invariant subgroups approach [10] mentioned.

I would like to thank R. Parker and R. Wilson for the suggestion to use lemma 5. I am also indebted to the anonymous referee for several helpful remarks.

G	$\lvert G\rvert$	deg	normal	Cl	$t_{\text{ old}}$	t
$\mathrm{Syl}_2(Co_3)$	1024	276	153	61	6+257	148
$\mathrm{Syl}_2(Co_3)$	1024	pc	153	61	6+101	57
$[S_4(6c)^2]2$	1152	12	8	20	2+3	3
$2^4 \times S_5$	1920	13	441	112	21+881	42
$\frac{1}{2}[2^6]L(6):2$	3840	12	5	23	6+2	9
$2.2^7.3^2.S_3 \quad (\leq Co_3)$	27648	pc	13	45	8+11	2
$2.2^7.3^2.S_3 \quad (\leq Co_3)$	27648	72	13	45	10+25	6
$[\frac{1}{2}S_3^5]D(5)$	38880	15	5	48	5+8	2
$3_+^{1+4}:4S_5 \quad (\leq McL:2)$	58320	275	33	10	23+46	31
$S_3 \wr A_5$	466560	15	7	72	36+19	14
$3_+^{1+4}:2_-^{1+4}.S_5 \quad (\leq Co_2)$	933120	1080	7	50	55+229	58
$\frac{1}{2}[A_5^3:2]S_3$	1296000	15	4	50	17+4	2
$S_5 \times S_5 \times S_5$	1728000	15	38	343	110+516	32
$Sp_4(4).4$	3916800	170	4	30	15+14	5
$(2_+^{1+6} \times 2^4).A_8 \quad (\leq Co_2)$	41287680	240	6	111	—	30
$(2 \times 2_+^{1+8}:U_4(2)):2 \quad (\leq Fi_{22})$	53904160	1024	6	115	395+299	312
$2^{6+8}:(A_7 \times S_3) \quad (\leq Fi_{23})$	247726080	4535	8	133	—	274
$\frac{1}{2}[A_5^4.2]S_4$	311040000	20	5	118	60+24	6
$\frac{1}{2}[M_{12}^4.2]S_4$	$1.958 \cdot 10^{21}$	96	5	—	—	46

Column "deg" gives the degree (or "pc" for groups given by a pc presentation), "normal" the number of normal subgroups and "Cl" the number of classes. The times are given in seconds, $t_{\text{ old}}$ for the naïve algorithm (separated in conjugacy class computation + normal subgroups computation) and t for the new algorithm. A dash (—) indicates that the calculations did not finish in reasonable time.

Table 1: Example runtimes

References

[1] Babai, L., Goodman, A. J., Kantor, W. M., Luks, E., and Pálfy, P. P. Short presentations for finite groups. *J. Algebra 194* (1997), 97–112.

[2] Bray, J. *Symmetric Presentations of Sporadic Groups and Related Topics*. PhD thesis, The University of Birmingham, 1997.

[3] Cannon, J., and Holt, D. Computing chief series, composition series and socles in large permutation groups. *J. Symb. Comput. 24* (1997), 285–301.

[4] Cannon, J. J. Construction of defining relators for finite groups. *Discrete Math. 5* (1973), 105–129.

[5] Celler, F., Neubüser, J., and Wright, C. R. B. Some remarks on the computation of complements and normalizers in soluble groups. *Acta Appl. Math. 21* (1990), 57–76.

[6] Dixon, J. D., and Mortimer, B. *Permutation Groups*, vol. 163 of *Graduate Texts in Mathematics*. Springer, Heidelberg, 1996.

[7] Easdown, D., and Praeger, C. E. On minimal faithful permutation representations of finite groups. *Bull. Austral. Math. Soc. 38* (1988), 207–220.

[8] Feit, W., and Thompson, J. G. Solvability of groups of odd order. *Pacific J. Math. 13* (1963), 775–1029.

[9] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4*. Lehrstuhl D für Mathematik, RWTH Aachen, Germany and School of Mathematical and Computational Sciences, U. St. Andrews, Scotland, 1997.

[10] Hulpke, A. Computing subgroups invariant under a set of automorphisms. *submitted* ().

[11] Hulpke, A. *Konstruktion transitiver Permutationsgruppen*. PhD thesis, Rheinisch-Westfälische Technische Hochschule, Aachen, Germany, 1996.

[12] Isaacs, I. M., Kantor, W. M., and Spaltenstein, N. On the probability that a group element is p-singular. *J. Algebra 176*, 1 (1995), 139–181.

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

[14] Kantor, W. M., and Seress, Á. Permutation group algorithms via black box recognition algorithms. In *Groups '97 Bath/St. Andrews* (to appear), C. M. Campbell, E. F. Robertson, and G. C. Smith, Eds., Cambridge University Press.

[15] Linton, S. A. The art and science of computing in large groups. In *Proceedings of CANT '92* (1995), W. Bosma and A. J. van der Poorten, Eds., Kluwer, pp. 91–109.

[16] Lux, K., Müller, J., and Ringe, M. Peakword Condensation and Submodule Lattices: An Application of the Meat-Axe. *J. Symb. Comput. 17* (1994), 529–544.

[17] Neumann, P. M. Some algorithms for computing with finite permutation groups. In *Groups – St. Andrews*

1985 (1986), E. F. Robertson and C. M. Campbell, Eds., Cambridge University Press, pp. 59–92.

[18] Parker, R. The Computer Calculation of Modular Characters (the MeatAxe). In *Computational Group theory* (1984), M. D. Atkinson, Ed., Academic press, pp. 267–274.

[19] Seress, Á. Nearly linear time algorithms for permutation groups: an interplay between theory and practice. *Acta Appl. Math.* (1998), to appear.