Computing Normalizers in Permutation p-Groups

Eugene M. Luks *
Computer and Information Science
University of Oregon

Ferenc Rákóczi *
Computer and Information Science
University of Oregon

Charles R. B. Wright
Department of Mathematics
University of Oregon

Abstract

Let G and H be subgroups of a finite p-group of permutations. We describe the theory and implementation of a polynomial-time algorithm for computing the normalizer of H in G. The method employs the imprimitivity structure and an associated canonical chief series to reduce to linear problems with fast solutions. An implementation in GAP exhibits marked speedups over general-purpose methods applied to the same groups. There are analogous procedures and timings for the problem of testing conjugacy of subgroups of p-groups, and implementations are planned. It is an easy matter, also, to extend the application to general nilpotent groups.

1 Introduction and Related Work

This paper contains a contribution to the collection of methods for computing normalizers in permutation groups, a problem on which, as Holt [Ho] has commented, "there seems to be almost unlimited scope for possible improvements." In general, the normalizer problem in permutation groups is not known to be in polynomial time and, in fact, its complexity is of great interest because of its relation to the problem of testing graph isomorphism (see, e.g., [Lu3]). Hence, it is not surprising that existing implementations have exponential running time in the worst case. However, it appears that these implementations remain exponential even for nilpotent groups, for which the normalizer problem is known to be in polynomial time [KL]. (In fact, normalizers are computable in polynomial time even for solvable groups [Lu2].) With this in mind, we describe a normalizer algorithm for p-subgroups of S_n which has worst case timing of $O(n^4)$. An implementation in GAP [Sc] has resulted in substantial speedups over the GAP library function (see Section 7) on permutation domains of moderate size.

Thus, our point of view is that if the groups under consideration are known to possess special properties, such as nilpotency, then one can hope to exploit that knowledge to devise normalizer algorithms that are faster than the generic ones applicable to wider classes of groups. We consider the problem of computing $N_G(H)$, the normalizer of H in G, in the particular setting in which G and H are given as subgroups of a finite p-group of permutations. Although we address only the p-group case here, it is easy to extend the methods to a nilpotent permutation group $\langle G, H\rangle$.

In comparison with other available algorithms that apply specifically to nilpotent or solvable groups, notably the Glasby-Slattery method [GS], we emphasize that our goal is to exploit not only the nilpotency of our groups but also the fact that their elements multiply as members of a permutation group. The method of [GS] applies more generally to groups given by, or convertible to, power-commutator presentations and then requires collection methods to multiply elements. However, it is not merely the fast multiplication in the permutation domain that contributes to the efficiency of our approach (and we do recognize that collection is often quite fast). We make essential use of natural actions of p-groups on structure (imprimitivity) forests and, more significantly, of a data structure that admits efficient linear methods in the (elementary abelian) quotients of a particular normal series. (See, e.g., [BC] and [GHLSW] for other examples of exploitation of imprimitivity systems of p-groups to get at the group structure; [LM] shows another, indispensable, use of linear algebra in dealing with such groups.)

In a future paper, we will elaborate on implementations of analogous methods for dealing with the problem of testing conjugacy of groups H_1 and H_2 under G when $\langle G, H_1, H_2\rangle$ is nilpotent. In the base linear algebra problem, the extension of our method only involves loss of homogeneity of a linear system. We note, however, that the procedure exhibits the same computational complexity as the normalizer method. Incorporated in the conjugacy test is an algorithm for finding centralizers of subgroups.

The paper is organized as follows. We give a generic description of the main algorithm and associated subroutines for computing certain maximal subgroups. Then we present the data structures, matrices and linear methods used to implement the algorithms, and we describe some of the technical details of the implementations. The paper ends with a discussion of timing and implementation experience.

2 The Main Algorithm

Throughout, p will be a prime, G and H will be subgroups of a finite group K of order a power of p, and our goal will be to compute $N_G(H)$. Since K is a p-group, it has a central chief series, $K = K_0 \rhd K_1 \rhd \cdots \rhd K_L = 1$, with $K_j \trianglelefteq K$ for each j and with each factor K_{j-1}/K_j of order

*Research supported in part by NSF grant CCR-9013410.

ISAAC 94 - 7/94 Oxford England UK
© 1994 ACM 0-89791-638-7/94/0007..$3.50

The Normalizer Algorithm

{**input:** *Subgroups G and H of a group K.*
A normal series $K = K_0 \rhd \cdots \rhd K_L = 1$ of K.
$H_i = K_i \cap H$ for $i = 0, \ldots, L$.}
{**output:** $N_G(H)$.}
begin
{*Initialize the ambient normalizer M.*}
$M := G$
$\{M = N_G(H_L).\}$
for $i := L - 1$ **downto** 0 **do**
  $\{M = N_G(H_{i+1})\}$
  for $j := i$ **to** $L - 1$ **do**
    $\{M = N_G(H_iK_j) \cap N_G(H_{i+1})\}$
    $M := N_M(H_iK_{j+1})$
    $\{M = N_G(H_iK_{j+1}) \cap N_G(H_{i+1})\}$
  $\{M = N_G(H_i)\}$
$\{M = N_G(H)\}$
return M
end.

Figure 1.

p. We suppose a series like this to be given. Later in the paper we will explicitly construct such a series suited to our needs. Define $H_i := K_i \cap H$ for $i = 0, \ldots, L$. The plan of the main algorithm is to compute normalizers of a sequence of subgroups H_iK_j of K, starting with a subgroup that is obviously G-normal and ending with $H_0K_L = (K_0 \cap H)K_L = H$. Figure 1 gives a preliminary version. Statements in braces { } are assertions about the values of variables at stages of the execution where the statements appear. Here and in subsequent algorithms, **endfor** and **endif** statements are implied by the indentation.

The assertions in braces are immediate, since $H_L = 1$, $N_G(H_iK_i) = N_G(K_i) = G$, $H_iK_L = H_i$, $N_G(H_i) \leq N_G(H_{i+1})$ and $H_0 = H$.

This algorithm does not require G and H to be p-groups, nor does it require special properties of the normal series for K. These conditions will come into play later, when we compute $N_M(H_iK_{j+1})$. Note in passing that the overall architecture of our algorithm differs from that in [GS] only by having the inner and outer loops interchanged. At this level of discussion that difference is inconsequential.

The preliminary algorithm just given can be modified to omit some cases in which M cannot change. If $H_i = H_{i+1}$, i.e., if H *avoids* K_i/K_{i+1}, then we can skip the inner loop for i, and if $HK_j = HK_{j+1}$, i.e., if H *covers* K_j/K_{j+1}, then we can skip the inside step for j. In the actual implementation that we describe below, we will still carry out the step for j in the covering case in order to update additional data for $j + 1$. The factors K_s/K_{s+1} that H covers and avoids can be determined initially, and that information can be used to reduce subsequent computations.

The case in which K is a p-group and $K_0 \rhd \cdots \rhd K_L$ is a chief series is especially good; H either covers or avoids each chief factor, and the covering-avoiding information can be obtained as a byproduct of other computations that we will be making. In the p-group case we also know that $K_0 \rhd \cdots \rhd K_L$ is a central series, whence $H_{L-1} \leq K_{n-1} \leq Z(K)$ so $N_G(H_{L-1}) = G$, and also $[H_i, G] \leq [K_i, G] \leq K_{i+1}$ so $N_G(H_iK_{i+1}) = G$. Figure 2 shows the resulting streamlined algorithm.

The Normalizer Algorithm for p-Groups

{**input:** *Subgroups G and H of a finite p-group K.*
A chief series $K = K_0 \rhd \cdots \rhd K_L = 1$ of K.
$H_i = K_i \cap H$ for $i = 0, \ldots, L$.
$\Lambda = \{s \in \{0, \ldots, L-1\}: H$ covers $K_s/K_{s+1}\}$}
{**output:** $N_G(H)$.}
begin
$M := G$
for $i := L - 2$ **downto** 0 **and** $i \in \Lambda$ **do**
  for $j := i + 1$ **to** $L - 1$ **and** $j \notin \Lambda$ **do**
    $M := N_M(H_iK_{j+1})$
return M
end.

Figure 2.

To carry out the algorithm we must be able to compute $N_M(H_iK_{j+1})$ assuming the following five conditions:

(a) M normalizes H_{i+1};

(b) M normalizes H_iK_j;

(c) $0 \leq i < j < L$;

(d) $i \in \Lambda$, so $HK_i = HK_{i+1}$, and $[H_i : H_{i+1}] = p$;

(e) $j \notin \Lambda$, so $H_j = H_{j+1}$, and $[HK_j : HK_{j+1}] = p$.

It follows from (a)—(e) that the group $V := H_iK_j/H_{i+1}K_{j+1}$ is elementary abelian of order p^2 and is acted upon by M, which centralizes both $H_iK_j/H_{i+1}K_j$ and $H_{i+1}K_j/H_{i+1}K_{j+1}$. To get the stabilizer in M of the 1-dimensional subspace $H_iK_{j+1}/H_{i+1}K_{j+1}$ of V we use the following.

Proposition 1. *If (a)—(e) hold and if $h_{i+1} \in H_i \backslash H_{i+1}$, then the map $\theta: M \longrightarrow K_j/K_{j+1}$ given by*

$$[h_{i+1}, g] \in H_{i+1}\theta(g)$$

is a well-defined homomorphism. Its kernel, $N_M(H_iK_{j+1})$, has index 1 or p in M.

Proof. The natural action of M on V induces a matrix representation

$$g \mapsto \begin{pmatrix} 1 & \theta(g) \\ 0 & 1 \end{pmatrix}$$

relative to the basis $\{h_{i+1}H_{i+1}K_{j+1}, k_jH_{i+1}K_{j+1}\}$ for V. Clearly, $\theta: M \longrightarrow \mathbb{Z}_p$ is a homomorphism, and $\ker(\theta) = N_M(H_iK_{j+1})$. (Our thanks to a referee for this short argument.) ∎

To use the proposition, we must have elements $h_{i+1} \in H_i \backslash H_{i+1}$, which we obtain from generating sets for H and K.

The group K contains elements $z_1, \ldots, z_L$ with $K_{i-1} = \langle z_i, \ldots, z_L \rangle$ for $i = 1, \ldots, L+1$. Section 4 describes the construction of such a sequence for K in an important special case. For now, suppose that a canonical generating sequence $z_1, \ldots, z_L$ of this sort has been chosen for K.

If X is a subgroup of K, then $X = X \cap K_0 \trianglerighteq \cdots \trianglerighteq X \cap K_L = 1$ is a central series for X with each factor of order 1 or p. We call a sequence $x_1, \ldots, x_t$ of generators for X an *induced generating system* (IGS) for X (relative to $K_0 \rhd \cdots \rhd K_L$) in case the subgroups $\langle x_i, \ldots, x_t \rangle$ for $i = 1, \ldots, t+1$ are the distinct subgroups in the chain $X \cap K_0 \trianglerighteq \cdots \trianglerighteq X \cap K_L$. If $x_1, \ldots, x_t$ is an IGS for X, then $|X| = p^t$, and for each chief factor K_{j-1}/K_j covered by X there is a unique i with $K_{j-1} = \langle x_i \rangle K_j$.

We will use an IGS to describe the subgroup M in the implementation of the normalizer algorithm. Thus the group G is given in the input by an IGS for G, and the algorithm returns an IGS for $N_G(H)$. If the permutation group G is initially given by a set S of generators, we can construct an IGS for G by sifting S against the canonical generating sequence $z_1, \ldots, z_L$ for K.

It will be convenient to describe the input subgroup H by a sequence $h_1, \ldots, h_L$ that is not quite an IGS. If $H_{i-1} = H_i$, i.e., if H avoids K_{i-1}/K_i, let $h_i := 1$. Otherwise, choose $h_i \in H_{i-1} \backslash H_i$ so that $h_i K_i = z_i K_i$. Thus, for Λ as in the algorithm,

$$\Lambda = \{i \in \{0, \ldots, L-1\} : h_{i+1} \neq 1\} \quad \text{and}$$
$$h_{i+1} \equiv z_{i+1} \bmod K_{i+1} \quad \text{for} \quad i \in \Lambda.$$

One can construct such a sequence $h_1, \ldots, h_L$, which we call a *strong generating system* (SGS) for H, by sifting a set of generators for H against $z_1, \ldots, z_L$. In our implementation, $h_1, \ldots, h_L$ and the IGS for G are computed by a modification of Knuth's [Kn] organization of the Sims-Schreier methods. Note that the sequence $h_1, \ldots, h_L$ is only computed once.

3 Updating the Normalizer

We use facts from Section 2 to describe an algorithm for updating $N_G(H_{i+1})$ to $N_G(H_i)$. Consider a fixed i in Λ, and let $h := h_{i+1} \in H_i \backslash H_{i+1}$ be the corresponding member of an SGS for H relative to the canonical generating sequence $z_1, \ldots, z_L$ for K. For $j = i, \ldots, L$ let $M_j = N_G(H_{i+1}) \cap N_G(H_i K_j)$. Thus $M_i = N_G(H_{i+1})$, $M_L = N_G(H_i)$, and the instruction $M := N_M(H_i K_{j+1})$ in the main algorithm replaces M_j by M_{j+1}. To compute M_{j+1} from M_j we maintain three sequences: an IGS $m_1, \ldots, m_t$ for M_j, a sequence $x_1, \ldots, x_t$ of members of H_{i+1}, and a sequence $\phi(1), \ldots, \phi(t)$ of exponents in $\mathbf{Z}_p = \{0, 1, \ldots, p-1\}$.

Figure 3 gives the expanded algorithm for computing M_{j+1} from M_j. To show that this algorithm produces the correct result, we verify the statements in braces. Since $h = h_{i+1} \in K_i$, we have $[h, m_k] \equiv 1 \equiv x_k \bmod K_{i+1}$ on first entry into the main loop. We verify the last comment by considering cases.

Suppose first that $j \notin \Lambda$ and that $s = \max\{k : \phi(k) \neq 0\}$. We have $[h, m_k] \equiv x_k z_{j+1}^{\phi(k)} \bmod K_{j+1}$ for $1 \leq k \leq s$. Proposition 1 applies, since $j \notin \Lambda$. We have $\theta(m_k) = z_{j+1}^{\phi(k)} K_{j+1}$, so $\theta(m_k m_s^{\alpha(k)}) = \theta(m_k)\theta(m_s)^{\alpha(k)} = z_{j+1}^{\phi(k)+\alpha(k)\phi(s)} K_{j+1} = K_{j+1}$, whence $m_k m_s^{\alpha(k)} \in M_{j+1}$. For $k > s$ the values of m_k and x_k are simply moved to m_{k-1} and x_{k-1}. Since $\theta(m_s) \neq 1$, $m_s \notin \ker\theta = M_{j+1}$, a maximal subgroup of M_j. Thus the sequence given in the original notation by $m_1 m_s^{\alpha(1)}, \ldots, m_{s-1} m_s^{\alpha(s-1)}, m_{s+1}, \ldots, m_t$ is an IGS for M_{j+1} in this case.

Update from $N_G(H_{i+1})$ to $N_G(H_i)$
{input: *An IGS $m_1, \ldots, m_t$ for $N_G(H_{i+1})$.*}
{output: *An IGS for $N_H(H_i)$.*}
begin
{*Initialize.*}
for $k := 1$ **to** t **do**
  $x_k := 1 \in H_{i+1}$
  $\phi(k) := 0 \in \mathbf{Z}_p$
for $j := i+1$ **to** $L-1$ **do**
  {$(m_1, \ldots, m_t)$ *is an IGS for* M_j,
   $x_1, \ldots, x_t \in H_{i+1}$,
   and $[h_{i+1}, m_k] \equiv x_k \bmod K_j$ *for* $k = 1, \ldots, t$}
  for $k := 1$ **to** t **do**
    Compute $\phi(k) \in \mathbf{Z}_p$ with
    $x_k^{-1}[h_{i+1}, m_k] \equiv z_{j+1}^{\phi(k)} \bmod K_{j+1}$
  if $j \notin \Lambda$ **then**
    if $\phi(k) \neq 0$ for some k **then**
      $t := t-1$
      $s := \max\{k : \phi(k) \neq 0\}$
      for $k := 1$ **to** $s-1$ **do**
        Solve $\phi(s)\alpha(k) + \phi(k) = 0$ for $\alpha(k) \in \mathbf{Z}_p$
        $m_k := m_k m_s^{\alpha(k)}$
        $x_k := (x_s m_s^{-1})^{\alpha(k)} x_k m_s^{\alpha(k)}$
      for $k := s$ **to** t **do**
        $m_k := m_{k+1}$
        $x_k := x_{k+1}$
  else {$j \in \Lambda$}
    for $k := 1$ **to** t **do**
      $x_k := x_k h_{j+1}^{\phi(k)}$
  {$(m_1, \ldots, m_t)$ *is an IGS for* M_{j+1},
   $x_1, \ldots, x_t \in H_{i+1}$,
   and $[h_{i+1}, m_k] \equiv x_k \bmod K_{j+1}$ *for* $k = 1, \ldots, t$}
return $(m_1, \ldots, m_t)$
end.

Figure 3.

We know that x_s and x_k are in H_{i+1}, which is normalized by M_j. Hence

$$(x_s m_s^{-1})^{\alpha(k)} x_k m_s^{\alpha(k)} \in H_{i+1}(m_s^{-1})^{\alpha(k)} x_k m_s^{\alpha(k)} = H_{i+1}.$$

Now to show that $[h, m_k m_s^{\alpha(k)}] \equiv (x_s m_s^{-1})^{\alpha(k)} x_k m_s^{\alpha(k)} \bmod K_{j+1}$ we may assume that $K_{j+1} = 1$, so that $z := z_{j+1} \in Z(K)$. Then $[h, m_k] = x_k z^{\phi(k)}$, so $m_k^h = m_k x_k^{-1} z^{-\phi(k)}$. It follows that

$$\begin{aligned}(m_k m_s^{\alpha(k)})^h &= m_k x_k^{-1} z^{-\phi(k)} (m_s x_s^{-1} z^{-\phi(s)})^{\alpha(k)} \\ &= m_k x_k^{-1} (m_s x_s^{-1})^{\alpha(k)} z^{-\phi(k)-\phi(s)\alpha(k)} \\ &= m_k m_s^{\alpha(k)} m_s^{-\alpha(k)} x_k^{-1} (m_s x_s^{-1})^{\alpha(k)},\end{aligned}$$

so $[m_k m_s^{\alpha(k)}, h] = ((x_s m_s^{-1})^{\alpha(k)} x_k m_s^{\alpha(k)})^{-1}$, as desired.

Suppose next that $j \notin \Lambda$ but that $\phi(k) = 0$ for every k. Then $M_j = M_{j+1}$, and already $[h, m_k] \equiv x_k \bmod K_{j+1}$ for each k.

In the final case, with $j \in \Lambda$, $M_j = M_{j+1}$ again and $m_1, \ldots, m_t$ is still an IGS. Moreover, since $j \in \Lambda$, $h_{j+1} K_{j+1} = z_{j+1} K_{j+1}$ and thus $[h, m_k] \equiv x_k h_{j+1}^{\phi(k)} \bmod K_{j+1}$ with $x_k h_{j+1}^{\phi(k)} \in H_{i+1}$ since $j \geq i+1$.

The execution of this algorithm requires computing products, powers and commutators in K. The algorithm also requires computing "leading coefficients" $\phi(1), \ldots, \phi(t)$ (in the sense of [Sc]) relative to the canonical generating sequence for K. In the next section we describe a data structure that permits these coefficients to be calculated rapidly, and in Section 5 we discuss their calculation.

4 The Linear Structure

In this section we consider a finite p-group K, acting on a rooted tree in a way that produces a normal series for K with elementary abelian factors. The unique refinement of the series to a chief series for K corresponds to a sequence of K-invariant flags in the factors, viewed as $\mathbb{Z}_p$-vector spaces, and the matrices that describe the bases associated with the flags provide easy membership tests for subgroups in the chief series.

If K is any finite group of permutations it is possible to construct a *structure forest* for K [LM, Lu1] consisting of rooted trees, one for each orbit of K, such that in a given tree the children of the root correspond to maximal blocks of imprimitivity, and the subtree rooted at the child corresponding to a block is the structure tree for the restriction to that block of its setwise stabilizer. This construction can be carried out essentially as efficiently as finding imprimitivity systems [At] (see also [GHLSW] for additional comments on the construction in p-groups). In case K is a p-Sylow subgroup of S_{p^t}, the repeated wreath product $K = C_p \wr C_p \wr \ldots \wr C_p$ of t groups of order p, the structure forest consists of a single full p-ary structure tree.

In this paper, G and H are subgroups of a p-group K of permutations, so it is possible to compute a structure forest for $\langle G, H \rangle$; the general implementation of our algorithm begins by constructing such a forest, using imprimitivity information from $\langle G, H \rangle$. For this exposition, however, we will assume that the forest consists of a single tree. (The extension to the general case involves a straightforward reformulation of the normal series in K; for example, one can view the disjoint trees in a vertical list, redefining "layers" accordingly in the account below.) Thus, we let $n = p^t$ and suppose that G and H are given as subgroups of the p-Sylow subgroup K of S_n, acting as automorphisms on a full p-ary rooted tree Γ with n leaves.

The nodes of Γ form layers, on each of which K acts transitively. For $r = 0, 1, \ldots, t$ let F_r be the subgroup of K fixing each of the p^r nodes at depth r, which we label so that the children of the root are labeled $0, \ldots, p-1$ and the children of a node labeled k are labeled $kp, \ldots, (k+1)p-1$. Then $K = F_0 > F_1 > \cdots > F_t = 1$, and each group F_r is normal in K. Let τ_r in F_r be the cyclic permutation $(0, \ldots, p-1)$ of the p children of the first node at depth r, permuting the subtrees rooted at those children but otherwise leaving the subtrees unchanged. Then the conjugates of τ_r under K permute the children of the other nodes at depth r, so F_r/F_{r+1} is elementary abelian, generated by τ_r and its K-conjugates. Indeed, $K = \langle \tau_0, \tau_1, \ldots, \tau_{t-1} \rangle$, $F_r = \langle \tau_r, \tau_{r+1}, \ldots, \tau_{t-1} \rangle^K$ for each r, and K acts linearly on the $\mathbb{Z}_p$-vector space $V_r := F_r/F_{r+1}$, which has a basis consisting of p^r conjugates of τ_r under K (mod F_{r+1}). To refine the series $K = F_0 \rhd \cdots \rhd F_t = 1$ to a chief series for K we must find for each r a basis $b_0, \ldots, b_{p^r-1}$ for V_r such that every subspace $\langle b_s, b_{s+1}, \ldots, b_{p^r-1} \rangle$ is $\langle \tau_0, \ldots, \tau_{r-1} \rangle$-invariant.

Proposition 2. *For each $r = 1, \ldots, t$ there is a $p^r \times p^r$ matrix $\mathbf{B}_r$ with the following properties.*

(a) The rows $\mathbf{b}_0, \ldots, \mathbf{b}_{p^r-1}$ of $\mathbf{B}_r$ form a $\mathbb{Z}_p$-basis for $V_r = \mathbb{Z}_p^{p^r}$.

(b) For $s = 0, \ldots, p^r - 1$ the subspace $V_r^{(s)}$ of V_r spanned by $\{\mathbf{b}_s, \ldots, \mathbf{b}_{p^r-1}\}$ is invariant under $\tau_0, \ldots, \tau_{r-1}$

(c) The inner products of the rows of $\mathbf{B}_r$ satisfy

$$\mathbf{b}_i \cdot \mathbf{b}_j \equiv \begin{cases} 0 & \bmod p \text{ for } i+j \geq p^r \\ (-1)^i & \bmod p \text{ for } i+j = p^r - 1. \end{cases}$$

(d) $\mathbf{B}_r = \mathbf{B}_r^{-1}$.

Moreover, the subspaces $V_r^{(s)}$ in (b) are uniquely determined by their invariance property.

Proof. We first exhibit a matrix $\mathbf{B}_1$ satisfying (a)–(d) for $r = 1$. Then we build $\mathbf{B}_2, \ldots, \mathbf{B}_t$ from $\mathbf{B}_1$ by using Kronecker products of matrices.

To begin with, $V_1 = \mathbb{Z}_p^p$, with standard basis $\mathbf{e}_0, \ldots, \mathbf{e}_{p-1}$, relative to which conjugation by the p-cycle τ_0 has the permutation matrix

$$\mathbf{C} := \begin{pmatrix} 0 & 1 & & & 0 \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ 1 & 0 & \ldots & 0 & 0 \end{pmatrix}.$$

From now on we will index rows and columns of matrices beginning with 0. Define the $p \times p$ integer matrix $\mathbf{B}$ with rows $\mathbf{b}_0, \ldots, \mathbf{b}_{p-1}$ by

$$\begin{aligned} \mathbf{b}_0 &:= [1 \quad 0 \quad \ldots \quad 0] = \mathbf{e}_0 \quad \text{and} \\ \mathbf{b}_i &:= \mathbf{b}_0(\mathbf{I} - \mathbf{C})^i \quad \text{for} \quad 1 \le i < p, \end{aligned}$$

where $\mathbf{I}$ denotes the $p \times p$ identity matrix. Then $\mathbf{b}_i = \sum_k (-1)^k \binom{i}{k} \mathbf{e}_k$.

We have

$$\mathbf{B}^2 = \mathbf{I},$$

since $(\mathbf{B}^2)_{ij} = \sum_k (-1)^{k+j} \binom{i}{k}\binom{k}{j} = \sum_k (-1)^{k+j} \binom{i}{j}\binom{i-j}{k-j} = \binom{i}{j} \sum_s (-1)^s \binom{i-j}{s}$, which yields $(\mathbf{B}^2)_{ii} = 1$ and $(\mathbf{B}^2)_{ij} = 0$ if $j \ne i$.

The matrix $\mathbf{B}(\mathbf{I} - \mathbf{C})\mathbf{B}^{-1}$ has an especially nice form. For $0 \le i < p-1$,

$$\mathbf{e}_i(\mathbf{B}(\mathbf{I} - \mathbf{C})\mathbf{B}^{-1}) = \mathbf{b}_i(\mathbf{I} - \mathbf{C})\mathbf{B}^{-1} = \mathbf{e}_0(\mathbf{I} - \mathbf{C})^{i+1}\mathbf{B}^{-1}$$

$$= \mathbf{b}_{i+1}\mathbf{B}^{-1} = \mathbf{e}_{i+1}\mathbf{B}\mathbf{B}^{-1} = \mathbf{e}_{i+1}.$$

For the last row, since $\mathbf{C}^p = \mathbf{I}$,

$$\begin{aligned} &\mathbf{e}_{p-1}\mathbf{B}(\mathbf{I} - \mathbf{C})\mathbf{B}^{-1} \\ &= \mathbf{b}_{p-1}(\mathbf{I} - \mathbf{C})\mathbf{B}^{-1} \\ &= \mathbf{e}_0(\mathbf{I} - \mathbf{C})^p\mathbf{B}^{-1} \\ &= \mathbf{e}_0(-1)^{p-1}\left(-\mathbf{I} + \sum_{k=0}^{p}\binom{p}{k}(\mathbf{C} - \mathbf{I})^k - (\mathbf{C} - \mathbf{I})^p\right)\mathbf{B}^{-1} \\ &= (-1)^{p-1}\mathbf{e}_0\left(\sum_{k=1}^{p-1}(-1)^{-k}\binom{p}{k}(\mathbf{I} - \mathbf{C})^k\right)\mathbf{B}^{-1} \\ &= (-1)^{p-1}\sum_{k=1}^{p-1}(-1)^k\binom{p}{k}\mathbf{b}_k\mathbf{B}^{-1} \\ &= (-1)^{p-1}\sum_{k=1}^{p-1}(-1)^k\binom{p}{k}\mathbf{e}_k. \end{aligned}$$

Thus

(1)

$$\mathbf{B}(\mathbf{I} - \mathbf{C})\mathbf{B}^{-1} = \begin{pmatrix} 0 & 1 & & & 0 \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ 0 & (-1)^p\binom{p}{1} & \ldots & -\binom{p}{p-2} & \binom{p}{p-1} \end{pmatrix}.$$

In particular,

$$\mathbf{B}(\mathbf{I} - \mathbf{C})\mathbf{B}^{-1} \equiv \begin{pmatrix} 0 & 1 & & & 0 \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0 & 1 \\ 0 & 0 & \ldots & 0 & 0 \end{pmatrix} \bmod p,$$

so that, taken mod p, the rows of $\mathbf{B}$ form a basis for a τ_0-invariant flag in V_1; i.e., $\langle \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{p-1}\rangle\mathbf{C} \subseteq \langle \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{p-1}\rangle$ for $i = 0, \ldots, p-1$.

Moreover, $(\mathbf{B}\mathbf{B}^T)_{ij} = \sum_k \mathbf{B}_{ik}\mathbf{B}_{jk} = \sum_k \binom{i}{k}\binom{j}{k} = \binom{i+j}{i}$, so $(\mathbf{B}\mathbf{B}^T)_{ij} \equiv 0 \bmod p$ if $i + j \ge p$ and $(\mathbf{B}\mathbf{B}^T)_{ij} \equiv (-1)^i \bmod p$ if $i + j = p - 1$. Hence

$$(2) \qquad \mathbf{B}\mathbf{B}^T \equiv \begin{pmatrix} & & & & 1 \\ & * & & -1 & \\ & & \iddots & & \\ & -1 & & 0 & \\ 1 & & & & \end{pmatrix} \bmod p.$$

Thus $\mathbf{B}_1 := \mathbf{B}$ satisfies (a)–(d) for $r = 1$.

Now consider $r > 1$. For $0 \le j < r$, the mapping τ_j permutes the nodes $0, 1, \ldots, p^r - 1$ of Γ at depth r by

$$a \longrightarrow \begin{cases} a + p^{r-j-1} \bmod p^{r-j} & \text{for } 0 \le a < p^{r-j} - 1 \\ a & \text{for } p^{r-j} \le a < p^r. \end{cases}$$

The corresponding $p^r \times p^r$ permutation matrix $\mathbf{T}_{jr}$ has the form

$$\mathbf{T}_{jr} := \begin{pmatrix} 0 & \mathbf{I}_1 & & & 0 \\ & 0 & \mathbf{I}_1 & & 0 \\ & & \ddots & \mathbf{I}_1 & \vdots \\ \mathbf{I}_1 & & & 0 & 0 \\ 0 & 0 & \ldots & 0 & \mathbf{I}_2 \end{pmatrix}$$

for suitable identity matrices $\mathbf{I}_1$ and $\mathbf{I}_2$, which we can describe using Kronecker, i.e., tensor, products of matrices.

Recall that the Kronecker product of an $m \times n$ matrix $\mathbf{A}$ and an $s \times t$ matrix $\mathbf{B}$ is the $ms \times nt$ matrix $\mathbf{A} \otimes \mathbf{B}$ defined by

$$(\mathbf{A} \otimes \mathbf{B})_{\alpha s+\beta, \gamma t+\delta} = \mathbf{A}_{\alpha\gamma}\mathbf{B}_{\beta\delta}$$

for $0 \le \alpha < m$, $0 \le \gamma < n$, $0 \le \beta < s$, $0 \le \delta < t$. It follows that $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{A}\mathbf{C}) \otimes (\mathbf{B}\mathbf{D})$ for matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ of compatible sizes. Denote by $\mathbf{A}^{\otimes r}$ the r-fold Kronecker product $\mathbf{A} \otimes \cdots \otimes \mathbf{A}$, with the convention that $\mathbf{A}^{\otimes 0}$ is the 1×1 identity matrix. Thus $\mathbf{I}^{\otimes r}$ is the $p^r \times p^r$ identity matrix.

To describe the matrix $\mathbf{T}_{jr}$ for the transformation induced by τ_j, let $\mathbf{D}$ be the $p \times p$ matrix with $\mathbf{D}_{ij} = [i = 0][j = 0]$ (here and later we use the notational convention that $[predicate]$ has the value 1 if *predicate* is true and the value 0 if *predicate* is false). Thus $\mathbf{D}$'s only nonzero entry is a 1 in the upper left corner. Then

$$\mathbf{T}_{jr} - \mathbf{I}^{\otimes r} = \mathbf{D}^{\otimes j} \otimes (\mathbf{C} - \mathbf{I}) \otimes \mathbf{I}^{\otimes(r-j-1)} \qquad \text{for} \quad 0 \le j < r,$$

where $\mathbf{C}$ is the cyclic permutation matrix from the case $r = 1$ above.

We want a $p^r \times p^r$ matrix $\mathbf{B}_r$ whose rows form a basis defining a flag in V_r that is invariant under conjugation by $\tau_0, \ldots, \tau_{r-1}$, i.e., a $\mathbf{B}_r$ such that $\mathbf{B}_r(\mathbf{T}_{jr} - \mathbf{I}^{\otimes r})\mathbf{B}_r^{-1}$ is strictly upper-triangular for $0 \le j < r$. It turns out that $\mathbf{B}^{\otimes r}$ has the correct rows, but in the wrong order.

Let $\mathbf{P}_r$ be the matrix of the permutation

$$\pi_r : \alpha_{r-1}p^{r-1} + \cdots + \alpha_1 p + \alpha_0 \longrightarrow \alpha_0 p^{r-1} + \cdots + \alpha_{r-2}p + \alpha_{r-1}$$

of $\{0, \ldots, p^r - 1\}$, where $0 \le \alpha_i < p$ for each i. Then $\mathbf{P}_r^2 = \mathbf{I}^{\otimes r}$. Moreover, it is easy to check that

$$\mathbf{P}_r(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_r)\mathbf{P}_r = \mathbf{A}_r \otimes \cdots \otimes \mathbf{A}_1$$

for $p \times p$ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_r$, and hence that $\mathbf{P}_r\mathbf{B}^{\otimes r} = \mathbf{B}^{\otimes r}\mathbf{P}_r$. Define

$$\mathbf{B}_r := \mathbf{P}_r\mathbf{B}^{\otimes r}.$$

Then $\mathbf{B}_r^2 = \mathbf{P}_r\mathbf{B}^{\otimes r}\mathbf{P}_r\mathbf{B}^{\otimes r} = (\mathbf{B}^{\otimes r})^2 = (\mathbf{B}^2)^{\otimes r} = \mathbf{I}^{\otimes r}$, so $\mathbf{B}_r$ satisfies (d).

Moreover, we have

$$\mathbf{B}_r\mathbf{B}_r^T = \mathbf{P}_r\mathbf{B}^{\otimes r}(\mathbf{B}^T)^{\otimes r}\mathbf{P}_r^T = \mathbf{B}^{\otimes r}\mathbf{P}_r\mathbf{P}_r^T(\mathbf{B}^T)^{\otimes r}$$

$$= \mathbf{B}^{\otimes r}(\mathbf{B}^T)^{\otimes r} = (\mathbf{B}\mathbf{B}^T)^{\otimes r}.$$

Since $(\mathbf{B}\mathbf{B}^T)_{ij} \equiv 0 \bmod p$ for $i + j \ge p$ and $(\mathbf{B}\mathbf{B}^T)_{ij} \equiv (-1)^i \bmod p$ for $i + j = p - 1$,

$$(3) \qquad (\mathbf{B}_r\mathbf{B}_r^T)_{ij} \equiv \begin{cases} 0 & \bmod p \text{ for } i + j \ge p^r \\ (-1)^i & \bmod p \text{ for } i + j = p^r - 1. \end{cases}$$

Thus $\mathbf{B}_r$ satisfies (c).

Furthermore, for $0 \le j < r$

$$\begin{aligned} (4)\ \mathbf{B}_r(\mathbf{T}_{jr} - \mathbf{I}^{\otimes r})\mathbf{B}_r^{-1} &= \mathbf{P}_r\mathbf{B}^{\otimes r}\left(\mathbf{D}^{\otimes j} \otimes (\mathbf{C} - \mathbf{I}) \otimes \mathbf{I}^{\otimes(r-j-1)}\right)\mathbf{B}^{\otimes r}\mathbf{P}_r \\ &= \mathbf{P}_r\left[\left(\mathbf{B}^{\otimes j}\mathbf{D}^{\otimes j}\mathbf{B}^{\otimes j}\right) \otimes (\mathbf{B}(\mathbf{C} - \mathbf{I})\mathbf{B}) \right. \\ &\qquad \left. \otimes \left(\mathbf{B}^{\otimes(r-j-1)}\mathbf{I}^{\otimes(r-j-1)}\mathbf{B}^{\otimes(r-j-1)}\right)\right]\mathbf{P}_r \\ &= \mathbf{P}_r\left[(\mathbf{B}\mathbf{D}\mathbf{B})^{\otimes j} \otimes (\mathbf{B}(\mathbf{C} - \mathbf{I})\mathbf{B}) \otimes \mathbf{I}^{\otimes(r-j-1)}\right]\mathbf{P}_r \\ &= \mathbf{I}^{\otimes(r-j-1)} \otimes (\mathbf{B}(\mathbf{C} - \mathbf{I})\mathbf{B}) \otimes (\mathbf{B}\mathbf{D}\mathbf{B})^{\otimes j}. \end{aligned}$$

Since the matrix $\mathbf{B}(\mathbf{C}-\mathbf{I})\mathbf{B}$ is strictly upper-triangular mod p by (1), so is $\mathbf{B}_r(\mathbf{T}_{jr} - \mathbf{I}^{\otimes r})\mathbf{B}_r^{-1}$. Thus the rows of $\mathbf{B}_r$ yield a basis for a flag in V_r that is invariant under $\tau_0, \ldots, \tau_{r-1}$. That is, $\mathbf{B}_r$ satisfies (a) and (b).

We can say substantially more. Indeed, we know that

$$(\mathbf{I}^{r-j-1})_{\alpha,\delta} = [\alpha = \delta],$$

$$(\mathbf{B}(\mathbf{I}-\mathbf{C})\mathbf{B})_{\beta\epsilon} = [\epsilon = \beta + 1] + [\beta = p-1][\epsilon > 0](-1)^{p+1+\epsilon}\binom{p}{\epsilon},$$

$$\text{and} \quad (\mathbf{B}\mathbf{D}\mathbf{B})_{\gamma\zeta} = \mathbf{B}_{\gamma 0}\mathbf{B}_{0\zeta} = 1 \cdot [\zeta = 0].$$

Hence (4) yields

$$\begin{aligned} (5)\ &(\mathbf{B}_r(\mathbf{I}^{\otimes r} - \mathbf{T}_{mr})\mathbf{B}_r^{-1})_{\alpha p^{r-j} + \beta p^{r-j-1} + \gamma, \delta p^{r-j} + \epsilon p^{r-j-1} + \zeta} \\ &= (\mathbf{I}^{\otimes(r-j-1)})_{\alpha\delta}(\mathbf{B}(\mathbf{I} - \mathbf{C})\mathbf{B})_{\beta\epsilon}((\mathbf{B}\mathbf{D}\mathbf{B})^{\otimes j})_{\gamma\zeta} \\ &= [\alpha = \delta] \cdot \left([\epsilon = \beta + 1] + [\beta = p - 1][\epsilon > 0](-1)^{p+1+\epsilon}\binom{p}{\epsilon}\right) \cdot \\ &\qquad [\zeta = 0] \end{aligned}$$

for $0 \le \alpha < p^j$, $0 \le \delta < p^j$, $0 \le \beta < p$, $0 \le \epsilon < p$, $0 \le \gamma < p^{r-j-1}$ and $0 \le \zeta < p^{r-j-1}$. To complete the proof, we must show uniqueness of the flags. Denote the rows of $\mathbf{B}_r$ by $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{p^r-1}$. For $0 < i < p^r$ let p^m be the highest power of p that divides i; say $i = p^m s$ with s prime to p. Write $i - 1 = \alpha p^{m+1} + \beta p^m + \gamma$ with $0 \le \alpha$, $0 \le \beta < p$, and $0 \le \gamma < p^m$. Then $p^m s = i = \alpha p^{m+1} + \beta p^m + 1 + \gamma$. Hence $1 + \gamma \equiv 0 \bmod p^m$, so $1 + \gamma = p^m$ and thus $s = \alpha p + \beta + 1$ with $\beta + 1 < p$. Hence $i = \alpha p^{m+1} + (\beta p^m + 1 + \gamma) < (\alpha + 1)p^{m+1}$. From (5) we have, since $\beta \ne p - 1$,

$$\begin{aligned} &(\mathbf{B}_r(\mathbf{I}^{\otimes r} - \mathbf{T}_{r-m-1,r})\mathbf{B}_r^{-1})_{i-1, \delta p^{m-1} + \epsilon p^m + \zeta} \\ &= (\mathbf{B}(\mathbf{I}^{\otimes r} - \mathbf{T}_{mr})\mathbf{B}_r^{-1})_{\alpha p^{m+1} + \beta p^m + \gamma, \delta p^{m+1} + \epsilon p^m + \zeta} \\ &= [\delta = \alpha][\epsilon = \beta + 1][\zeta = 0] \\ &= [\delta = \alpha][\alpha p + \epsilon = s][\zeta = 0] \\ &= [\delta = \alpha][\delta p + \epsilon = s][\zeta = 0] \\ &= [\delta p^{m+1} + \epsilon p^m + \zeta = i], \end{aligned}$$

so

$$\begin{aligned} &\mathbf{b}_{1-i}(\mathbf{I}^{\otimes r} - \mathbf{T}_{r-m-1,r}) \\ &\quad = \mathbf{e}_{i-1}\mathbf{B}_r(\mathbf{I}^{\otimes r} - \mathbf{T}_{r-m-1,r})\mathbf{B}_r^{-1}\mathbf{B}_r \\ &\quad = \mathbf{e}_i\mathbf{B}_r = \mathbf{b}_i. \end{aligned}$$

For $0 \le s < p^r$ the subspaces $V_r^{(s)} := \langle \mathbf{b}_s, \mathbf{b}_{s+1}, \ldots \rangle$ form a flag in V_r invariant under $\mathbf{T}_{0,r}, \ldots, \mathbf{T}_{r-1,r}$. Suppose that U is a subspace of V_r invariant under $\mathbf{T}_{0,r}, \ldots, \mathbf{T}_{r-1,r}$ that contains (mod p) an element $\mathbf{u} := \mathbf{b}_s + \alpha_{s+1}\mathbf{b}_{s+1} + \cdots \in \mathbf{b}_s + V_r^{(s+1)}$ with $s + 1 < p^r$. Then $\mathbf{b}_{s+1} = \mathbf{b}_s(\mathbf{I}^{\otimes r} - \mathbf{T})$ for some $\mathbf{T} \in \{\mathbf{T}_{0,r}, \ldots, \mathbf{T}_{r-1,r}\}$, so U contains $\mathbf{u}(\mathbf{I}^{\otimes r} - \mathbf{T}) \in \mathbf{b}_{s+1} + V_r^{(s+2)}$. By finite induction, U contains $\mathbf{b}_{p^r-1}$, and hence also, working back up, $\mathbf{b}_{p^r-2}, \ldots, \mathbf{b}_{s+1}, \mathbf{b}_s$. Thus $U = V_r^{(j)}$ for some j. It follows that the invariant subspaces $V_r^{(s)}$ are unique. ∎

We have just seen how to construct $\mathbf{b}_i$ from $\mathbf{b}_{i-1}$. The next proposition shows an easy way to construct the $\mathbf{b}_i$'s from the bottom up, as fast as they can be written, starting with the last row of $\mathbf{B}_r$, which is $(1, 1, \ldots, 1)$. Here we denote the j-th component of $\mathbf{b}_k$ by $(\mathbf{b}_k)_j$.

Proposition 3. *If $0 \le i < p^r$ and if p^m is the highest power of p that divides i, then $(\mathbf{b}_{i-1})_j$ is congruent mod p to*

$$\begin{aligned} (\mathbf{b}_i)_j \quad & \text{for} \quad 0 \le j < p^{r-m-1} \\ (\mathbf{b}_{i-1})_{j-p^{r-m-1}} + (\mathbf{b}_i)_j \quad & \text{for} \quad p^{r-m-1} \le j < p^{r-m} \\ (\mathbf{b}_{i-1})_{j-p^{r-m}}, \quad & \text{for} \quad p^{r-m} \le j < p^r. \end{aligned}$$

Proof. Write

$$i = \alpha_{r-1}p^{r-1} + \cdots + \alpha_m p^m$$

with $0 \le \alpha_k < p$ for all k and $\alpha_m \ne 0$. Let π be the permutation introduced above that reverses p-ary digits and has matrix $\mathbf{P}_r$. Then

$$\pi(i) = \alpha_m p^{r-m-1} + \alpha,$$

with $0 \le \alpha < p^{r-m-1}$. We have

$$i - 1 = \alpha_{r-1}p^{r-1} + \cdots + (\alpha_m - 1)p^m + (p^m - 1),$$

so

$$\pi(i-1) = (p^m - 1)p^{r-m} + (\alpha_m - 1)p^{r-m-1} + \alpha.$$

Define γ, δ and β by

$$j = \gamma p^{r-m} + \delta p^{r-m-1} + \beta$$

with $0 \le \gamma < p^m$, $0 \le \delta < p$ and $0 \le \beta < p^{r-m-1}$.

Then

$$\begin{aligned} (\mathbf{B}_r)_{i-1,j} &= (\mathbf{P}_r\mathbf{B}^{\otimes r})_{i-1,j} = \mathbf{B}^{\otimes r}_{\pi(i-1),j} \\ &= \mathbf{B}^{\otimes m}_{p^m-1,\gamma} \cdot \mathbf{B}_{\alpha_m-1,\delta} \cdot \mathbf{B}^{\otimes(r-m-1)}_{\alpha,\beta} \\ &\equiv 1 \cdot \mathbf{B}_{\alpha_m-1,\delta} \cdot \mathbf{B}^{\otimes(r-m-1)}_{\alpha,\beta} \mod p, \end{aligned}$$

because the last row of $\mathbf{B}^{\otimes m}$ consists of 1's mod p. Since γ is irrelevant here, we have established the case $p^{r-m} \le j$ of the proposition.

By a similar argument,

$$(\mathbf{B}_r)_{i,j} \equiv 1 \cdot \mathbf{B}_{\alpha_m,\delta} \cdot \mathbf{B}^{\otimes(m-r-1)}_{\alpha,\beta} \mod p,$$

so that

$$\begin{aligned} &(\mathbf{B}_r)_{i-1,j} - (\mathbf{B}_r)_{i,j} \\ &\equiv (\mathbf{B}_{\alpha_m-1,\delta} - \mathbf{B}_{\alpha_m,\delta}) \cdot \mathbf{B}^{\otimes(m-r-1)}_{\alpha,\beta} \\ &\equiv (-1)^\delta \left[\binom{\alpha_m - 1}{\delta} - \binom{\alpha_m}{\delta} \right] \cdot \mathbf{B}^{\otimes(m-r-1)}_{\alpha,\beta} \mod p. \end{aligned}$$

If $0 \le j < p^{r-m-1}$, then $\delta = 0$ so $(\mathbf{B}_r)_{i-1,j} - (\mathbf{B}_r)_{i,j} \equiv 0$ as claimed.

Finally, if $\delta > 0$ then

$$\begin{aligned} = (-1)^\delta \left[\binom{\alpha_m - 1}{\delta} - \binom{\alpha_m}{\delta} \right] &= (-1)^\delta \binom{\alpha_m - 1}{\delta - 1} \\ &= \mathbf{B}_{\alpha_m-1,\delta-1}, \end{aligned}$$

which establishes the proposition for $p^{r-m-1} \le j < p^{r-m}$, since

$$(\mathbf{B}_r)_{i-1,j-p^{r-m-1}} \equiv 1 \cdot \mathbf{B}_{\alpha_m-1,\delta-1} \cdot \mathbf{B}^{\otimes(m-r-1)}_{\alpha,\beta} \mod p. \quad \blacksquare$$

5 Computing $\phi(k)$ and Testing Membership

Proposition 2(c) gives an easy method for determining membership in the subgroups K_j and for computing the coefficients $\phi(k)$ required by the update algorithm.

In the main loop of the algorithm we are given elements $x_k^{-1}[h_{i+1}, m_k]$ in K_j and must find constants $\phi(k) \in \{0, 1, \ldots, p-1\}$ such that

$$x_k^{-1}[h_{i+1}, m_k] \equiv z_{j+1}^{\phi(k)} \mod K_{j+1}.$$

Given j, the first step is to compute r such that $F_r \ge K_j \ge K_{j+1} \ge F_{r+1}$, i.e., such that K_j and K_{j+1} correspond to K-invariant subspaces $\langle \mathbf{b}_s, \mathbf{b}_{s+1}, \ldots \rangle$ and $\langle \mathbf{b}_{s+1}, \ldots \rangle$ of $V_r := F_r/F_{r+1}$, with z_{j+1} corresponding to $\mathbf{b}_s$. For convenience, assume that $F_{r+1} = 1$, and let $\mathbf{v} := x_k^{-1}[h_{i+1}, r_k]$. Then $\mathbf{v} = \phi(k)\mathbf{b}_s + \mathbf{u}$ with $\mathbf{u} \in \langle \mathbf{b}_{s+1}, \ldots \rangle = K_{j+1}$. By Proposition 2(c),

$$\mathbf{b}_{p^r-s-1} \cdot \mathbf{v} = \phi(k)\mathbf{b}_{p^r-s-1} \cdot \mathbf{b}_s \equiv (-1)^s \phi(k) \mod p,$$

so to compute $\phi(k)$ we need only take the dot product of $\mathbf{v}$ with an appropriate row of $\mathbf{B}_r$. We also get a test for membership in K_j, since, for $\mathbf{u} \in V_r$,

$$\mathbf{u} \in \langle \mathbf{b}_s, \mathbf{b}_{s+1} \ldots \rangle \quad \text{iff} \quad \mathbf{u} \cdot \mathbf{b}_\beta^T = 0 \text{ for } \beta \ge p^r - s.$$

On each pass through the main loop of the update algorithm the value of j increases by 1. Thus the test vectors for V_r run through $\mathbf{b}_{p^r-1}, \mathbf{b}_{p^r-2}, \ldots, \mathbf{b}_0$. By Proposition 3, the complete matrix $\mathbf{B}$ need not be stored in order to implement the algorithm.

6 Complexity

There are two parts to the algorithm, each of them with a worst-case complexity of $O(n^4)$. The first part is to obtain the chief series for H, which is essentially a Sims-Schreier procedure, as organized by Knuth [Kn]. The gain in the analysis comes from the fact that we have fewer than n/p subgroups in the chain, each having p cosets, while in Knuth's analysis we have n in the worst case for both of these factors.

In the second part (finding the normalizer) there are two nested loops. Each of them can be $O(n)$ long. There is usually an $O(n)$ check or $O(n^2)$ maintenance work when the normalizer does not change, and there is an $O(n^3)$ maintenance work when the normalizer shrinks, which cannot happen more than n/p times. Thus the whole time is still $O(n^4)$.

7 Implementation and Experiments

We have implemented the algorithm in GAP [Sc]. The program occupies about 400 lines of code.

Comparing the running times with GAP's built-in normalizer algorithm, it appears that, while the running times of our algorithm are fairly predictable, the ones for GAP's algorithm depend more on the structure of the groups involved. In the cases that we investigated (for example, 2-, 3-, 5- and 7-groups on ≤ 125 points), our algorithm was up to several hundred times faster on some examples. We saw slower times only in some small examples; here the overhead in setting up our matrices, etc., was not worth the effort.

We also compared the timings with those obtained by GAP's AgGroup normalizer function (which is based upon the method of [GS]). For this we first applied the GAP function AgGroup to the ambient group $\langle G, H \rangle$, getting the embedding of G and H in the AG group via the PreImage function. After determining $N_G(H)$ as an AG-group, we used the Image function to lift the answer back to a permutation group.

In the tables below, we report some sample results run on a Sun SPARC-2 (in multiuser mode) for p-subgroups of S_{100}, when $p = 2$ and 3. We use the notation $\ell(K)$ to denote the length of a composition series for the p-group K, i.e., $\ell(K) = \log_p |K|$. The running times are in seconds of cpu-time as reported by the GAP function Runtime (and rounded to the nearest second). In instances where the other methods (listed as GAP and AG, respectively) required more than 20× that of our linear method (listed as LIN) we indicate only an asterisk (*) in the table; in some instances we simply interrupted the process when this time ratio was exceeded. The groups were generated by between 1 and 4 random elements of a random Sylow p-subgroup of S_{100}. In some instances, we forced $H \leq G$ (indicated by $\ell(G) = \ell(\langle G, H \rangle)$) and a few trials involved the full Sylow subgroup for G ($\ell(G)$ is then 97 when $p = 2$ and 48 when $p = 3$).

$p = 2$

$\ell(G)$	$\ell(H)$	$\ell(\langle G, H \rangle)$	$\ell(N_G(H))$	*Running Times*		
				LIN	GAP	AG
97	5	97	34	19	*	*
97	67	97	72	138	*	861
85	40	90	43	104	*	1974
85	35	85	45	89	*	994
77	3	83	19	57	148	1012
40	85	90	39	113	526	*
3	77	83	1	68	35	1353

$p = 3$

$\ell(G)$	$\ell(H)$	$\ell(\langle G, H \rangle)$	$\ell(N_G(H))$	*Running Times*		
				LIN	GAP	AG
48	3	48	15	7	*	137
48	32	48	36	27	*	149
40	3	44	7	10	*	*
36	21	42	22	16	28	178
21	36	42	19	17	*	267
3	40	44	2	14	12	207

Note that the instances in these tables where the general GAP normalizer has the better timing correspond to the case of a cyclic, and fairly small, G.

As suggested by a referee, we ran additional tests to determine the time spent in various procedures by the LIN and AG methods. One of the objectives was to determine the timings essentially for multiplication by "collection" (in the AG setting) versus multiplication in the permutation setting, since the structures of the underlying methods (ours compared to that of [GS]) are analogous. In some instances, the time in AG for just this part was closer to, or even better than, the overall time for our method. Rarely, however, did it match the timing for just the corresponding segment of our method (excluding, for example, the time to build a structure forest and to determine a chief series for $\langle G, H \rangle$). For instance, the "pure" part of the AG method for the first group in the $p = 2$ table took 56 seconds; the analogous part of the LIN program took 9 seconds. For the second group, the two methods had very similar timings for these segments: 94 seconds for the AG method and 99 seconds for LIN. For the other groups in that table, the AG timings were always twice, or more, those for LIN. While there must be groups where the AG method will be substantially superior, our "random" selections did not include any.

Ongoing investigations will include an implementation in Magma [CP].

References

[At] M.D. Atkinson, *An algorithm for finding the blocks of a permutation group,* Math. Comp., 29 (1975), 911–913.

[BC] G. Butler and J.J. Cannon, *On Holt's algorithm,* J. Symb. Comp., 15 (1993), 229–233.

[Bu] G. Butler, *Computing normalizers in permutation groups,* J. Algorithms, 4 (1983), 163–175.

[CP] J. Cannon and C. Playoust, An Introduction to MAGMA, School of Mathematics and Statistics, U. of Sydney, 1993.

[GHLSW] Z. Galil, C.M. Hoffmann, E.M. Luks, C.P. Schnorr, A. Weber *An $O(n^3 \log n)$ deterministic and an $O(n^3)$ Las Vegas isomorphism test for trivalent graphs,* J. ACM, 34 (1987), 513–531.

[GS] S.P. Glasby and M.C. Slattery, *Computing intersections and normalizers in soluble groups,* J. Symb. Comp., 9 (1990), 637–651.

[Ho] D.F. Holt, *The computation of normalizers in permutation groups,* J. Symb. Comp, 12 (1991), 498–516.

[KL] W.M. Kantor and E.M. Luks, *Computing in quotient groups,* Proc. 22nd ACM Symposium on Theory of Computing, 1990, 524–533.

[Kn] D.E. Knuth, *Notes on efficient representation of perm groups,* Combinatorica, 11 (1991), 57–68.

[Lu1] E.M. Luks, *Parallel algorithms for permutation groups and graph isomorphism,* Proc. 27th IEEE FOCS, 1986, 292–302.

[Lu2] E.M. Luks, *Computing in solvable matrix groups,* Proc. 33rd IEEE Symp. on the Foundations of Comp. Sci., 1992, 111–120.

[Lu3] E.M. Luks, *Permutation groups and polynomial-time computation,* in Groups and Computation, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 11, Amer. Math. Soc, 1993, ed. L. Finkelstein and W. Kantor, 139–175.

[LM] E.M. Luks and P. McKenzie, *Parallel computation in solvable permutation groups* J. Comp. Syst. Sci., 37 (1988), 39-62

[Sc] M. Schönert et al., GAP, Groups, Algorithms and Programming, Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, 3rd edition, 1993.