MATHEMATICS OF COMPUTATION, VOLUME 29, NUMBER 131
JULY 1975, PAGES 911–913

An Algorithm for Finding the Blocks of a Permutation Group

By M. D. Atkinson

Abstract. An efficient algorithm for finding the blocks of imprimitivity of a group from generating permutations is described and justified.

One of the most economical ways of storing a description of a group on a digital computer is to store permutations which generate the group. The penalty for this economy is the difficulty of answering specific questions about the group although, in the case of groups of degree up to about 200, a method due to Sims [1] is very convenient. For groups of larger degree, one can at least (without much difficulty) decide whether the group is transitive. If it is, the next thing one usually wants to know is whether it is primitive. This paper gives an algorithm to decide whether the group generated by given permutations is primitive. In the case of an imprimitive group, the algorithm produces a nontrivial block system.

Let $\Omega = \{1, 2, \ldots, n\}$ be the (finite) set permuted by permutations $g_1, g_2, \ldots, g_m$ which generate a transitive group G. The main part of the algorithm is the following process P_ω which calculates (when $\omega \neq 1$) the smallest block which contains $\{1, \omega\}$ and the block system containing this block. P_ω calculates a function f: $\Omega \longrightarrow \Omega$ and so, in a computer implementation, storage space must be reserved for the values of f. In addition, storage space has to be reserved for a collection C of symbols; symbols of Ω are added to and deleted from C during the process. Apart from storage space required for a few working variables, this is all that is necessary; however, at the end of the paper, I will indicate how P_ω may be made faster, using some additional storage to make Ω into a singly linked list.

The Process P_ω, $\omega \neq 1$.

1. Initially C is empty and, for all $\alpha \in \Omega$, $f(\alpha)$ is set equal to α.
2. Add ω to C and set $f(\omega) = 1$.
3. Delete a symbol β from C and calculate $\alpha = f(\beta)$.
4. Set an integer j equal to 0.
5. Increase j by 1 and calculate $\gamma = \alpha g_j$ and $\delta = \beta g_j$.
6. If $f(\gamma)$ and $f(\delta)$ are equal, go to 9.
7. Ensure $f(\delta) < f(\gamma)$ by interchanging γ and δ if necessary.
8. Redefine those values $f(\epsilon)$ of f which are equal to $f(\gamma)$ giving them the new value $f(\delta)$ and add the old value of $f(\gamma)$ to C.
9. If $j < m$, go to 5.
10. If C is nonempty, go to 3.

Received April 24, 1972; revised June 19, 1974.
AMS (MOS) subject classifications (1970). Primary 20B99; Secondary 20–04

11. Stop.

In the description of P_ω, I have used the symbol f for a function which changes its values throughout the process; and from the description, the reader should have no difficulty in constructing a computer program to carry out P_ω. However, in order to justify that P_ω is an algorithm that produces the required result, I shall adopt a different approach which does not rely on arguments concerned with time varying functions. I thank Professor C. C. Sims for suggesting this method of exposition, improvements 3 and 4 at the end of the paper, and many other helpful comments.

Let f_0 be the initial function f; $f_0(\omega) = 1$ and $f_0(\alpha) = \alpha$ if $\alpha \neq \omega$. Let $f_1, f_2, \ldots, f_r = \bar{f}$ be the variants of f defined by step 8. Associated with each function f_i is a partition Π_i of Ω, each part of Π_i consisting of all the symbols on which f_i takes a fixed value. Because of step 8, Π_{i+1} is obtained from Π_i by replacing two parts of Π_i by their union; in particular, Π_i is a refinement of Π_{i+1} (i.e., every part of Π_i is contained in a part of Π_{i+1}).

For any partition Π of Ω, let $\Pi(\alpha)$ denote the part of Π which contains α. Clearly $f_0(\alpha) \in \Pi_0(\alpha)$ for all $\alpha \in \Omega$; and because of step 8, it is also evident that $f_i(\alpha) \in \Pi_i(\alpha)$ for all $\alpha \in \Omega$ and $i = 1, 2, \ldots, r$. This establishes

LEMMA 1. (a) *If* $f_i(\alpha) = f_i(\beta)$ *then* $f_j(\alpha) = f_j(\beta)$, $j = i, \ldots, r$,

(b) $f_i^2 = f_i$, $i = 0, 1, \ldots, r$.

LEMMA 2. (a) $\alpha \geqslant f_0(\alpha) \geqslant f_1(\alpha) \geqslant \ldots \geqslant \bar{f}(\alpha)$,

(b) *a point* β *belonged to* C *if and only if* $\beta \neq \bar{f}(\beta)$,

(c) *if* β *belonged to* C, *then there exists* $\alpha < \beta$ *with* $\bar{f}(\alpha) = \bar{f}(\beta)$ *and* $\bar{f}(\alpha g_j) = \bar{f}(\beta g_j)$, $j = 1, \ldots, m$.

Proof. (a) Step 1 ensures that $\alpha \geqslant f_0(\alpha)$ and step 7 ensures that $f_i(\alpha) \geqslant f_{i+1}(\alpha)$.

(b) Points β are added to C in steps 2 and 8. In step 2, $\beta = \omega$ and $\omega > f_0(\omega) = 1 = \bar{f}(\omega)$. In step 8, $\beta = f_i(\gamma)$ for some i and γ; then $f_i(\beta) = \beta = f_i(\gamma)$ and $f_{i+1}(\beta) < f_i(\beta)$. Conversely, if $\beta > \bar{f}(\beta)$, then either $\beta = \omega$ belonged to C or we can choose i with $\beta = f_i(\beta) > f_{i+1}(\beta)$; then clearly $f_i(\beta) = \beta$ belonged to C.

(c) Let α be the point defined in step 3 when β is deleted from C. Then $\alpha = f_i(\beta) < \beta$ for some i. Moreover $f_i(\alpha) = f_i^2(\beta) = f_i(\beta)$ and so $\bar{f}(\alpha) = \bar{f}(\beta)$. Finally, after step 8 for a given j, $f_k(\alpha g_j) = f_k(\beta g_j)$ for some k and so $\bar{f}(\alpha g_j) = \bar{f}(\beta g_j)$.

LEMMA 3. $\bar{\Pi} = \Pi_r$ *is invariant under* G.

Proof. It is sufficient to prove that each g_j preserves $\bar{\Pi}$. Suppose that there exists $\theta, \phi \in \Omega$, $\theta < \phi$, with $\bar{f}(\theta) = \bar{f}(\phi)$ but $\bar{f}(\theta g_j) \neq \bar{f}(\phi g_j)$. Choose ϕ minimal subject to this. Then $\bar{f}(\phi) = \bar{f}(\theta) \leqslant \theta < \phi$ and so ϕ belonged to C. Hence there exists $\alpha < \phi$ with $\bar{f}(\alpha) = \bar{f}(\phi)$ and $\bar{f}(\alpha g_j) = \bar{f}(\phi g_j)$. Since $\bar{f}(\alpha) = \bar{f}(\theta)$ and $\alpha < \phi$, the minimal choice of ϕ ensures that $\bar{f}(\alpha g_j) = \bar{f}(\theta g_j)$. Thus that $\bar{f}(\phi g_j) = \bar{f}(\alpha g_j) = \bar{f}(\theta g_j)$; a contradiction.

Thus $\Delta = \bar{\Pi}(1)$ is a block of G containing 1 and ω. As G is transitive and $\bar{\Pi}$ is G-invariant, $\bar{\Pi}$ is the block system containing Δ.

LEMMA 4. Δ *is the smallest block containing* 1 *and* ω.

Proof. Let Δ_1 be the smallest block containing 1 and ω so that $\Delta_1 \subseteq \Delta$. Let $\hat{\Pi} = \{\Delta_1^g \mid g \in G\}$. Then $\hat{\Pi}$ is a partition of Ω; we now prove that each Π_i is a refinement of $\hat{\Pi}$. This is clearly true if $i = 0$. Assume now that $i > 0$, Π_i is a refinement of

$\hat{\Pi}$ and consider a part of Π_{i+1}. Such a part is either a part of Π_i or the union of two parts of Π_i of the form $\Pi_i(f_i(\gamma)) \cup \Pi_i(f_i(\delta)) = \Pi_i(\gamma) \cup \Pi_i(\delta)$ where $\gamma = \alpha g_j$, $\delta = \beta g_j$ and $\Pi_i(\alpha) = \Pi_i(\beta)$. By an inductive assumption, $\hat{\Pi}(\alpha) = \hat{\Pi}(\beta)$. Then

$$\hat{\Pi}(\gamma) = \hat{\Pi}(\alpha)g_j = \hat{\Pi}(\beta)g_j = \hat{\Pi}(\delta) \supseteq \Pi_i(f_i(\gamma)) \cup \Pi_i(f_i(\delta)).$$

This completes the induction and we have $\Delta = \bar{\Pi}(1) = \Pi_r(1) \subseteq \hat{\Pi}(1) = \Delta_1$. Hence $\Delta = \Delta_1$.

To verify that G is primitive, we have to run the process P_ω with $\omega = 2, 3, \ldots, n$. If some P_ω produces a nontrivial block system, then generators $\bar{g}_1, \ldots, \bar{g}_m$ for the group induced on the blocks may be obtained by taking the points of $f(\Omega)$ as the permuted symbols and defining $f(\alpha)\bar{g}_j = f(\alpha g_j)$.

There are several ways in which the algorithm can be made faster:

(1) Keep a record of the size of $\Pi(1)$ (the set of symbols β with $f(\beta) = 1$) during P_ω. If it becomes larger than the greatest divisor of n, then Ω is the only block containing 1 and ω.

(2) If the P_ω, $\omega = 2, 3, \ldots$, are performed in this order and we find a symbol $\gamma \in \Pi(1)$ with $1 < \gamma < \omega$, then again Ω is the only block containing 1 and ω.

(3) Ω can be given the structure of a singly linked list in which the parts of the current partition defined by f are represented by circular sublists; the uniting of two parts is then easily done by manipulating pointers and without scanning the whole of Ω.

(4) If C is stored in such a way that its elements are not overwritten when they are deleted, then when we exit from the process, we will have a list of all the elements which have been in C. These are the only symbols α for which $f(\alpha)$ has been changed from its initial value α, and so we may use C to perform step 1 of the run of the next process more quickly.

(5) If generators for H, the stabilizer of the point 1, are known (for example, they would be known if G arose as the result of a coset enumeration) then we can calculate the orbits on Ω under H and only run P_ω taking a representative ω from each orbit.

In general, the computation time rises as the square of the degree. I have programmed the algorithm in Fortran on an ICL System 4 (a machine logically similar to the IBM System 360 and with a cycle time of 0.75 μs) and have treated many generating sets. Two reasonably representative examples are:

(i) the group generated by $(1, 2, 3, \ldots, 998)$ and $(1, 2, 3)(4)(5) \ldots (998)$–found to be primitive in less than 1 second;

(ii) the group generated by $(1, 2, 3, \ldots, 499)(500, 501, \ldots, 998)$ and $(1, 2)(3, 4) \ldots (997, 998)$–found to be imprimitive in 20 seconds. The reason that (i) is much faster than (ii) is because (2) above works very well in this.

Department of Computing Mathematics
University College
Cardiff, Wales

1. C. C. SIMS, "Computational methods in the study of permutation groups," *Computational Problems in Abstract Algebra* (Proc. Conf., Oxford, 1967), Pergamon Press, Oxford, 1970, pp. 169–183. MR 41 #1856.