A COMPUTATIONAL TOOLKIT FOR FINITE PERMUTATION GROUPS

J. Cannon*
Rutgers University and University of Sydney

1 INTRODUCTION

Over the past two decades a range of powerful algorithms has been designed and implemented for computing with permutation groups. For example, if G is a permutation group acting faithfully on the finite set Ω, and X is a set of generators for G, then it is possible to compute the order of G even in situations where the cardinality of Ω is 10,000 or more.

In this paper we outline most of the algorithms that are currently available giving, where possible, an indication as to their efficiency. Implemented in a group theory package such as Cayley (Cannon [15]) these techniques have found wide application to problems arising from both inside and outside group theory.

The availability of these techniques together with the classification of finite simple groups (Gorenstein [23]) mean that it is now possible to devise computer programs which, given as input a set of generators for a permutation group G, will output a description of the structure of G. The exact form of such a description will depend upon G and can range from the names of the composition factors of G to the name of G itself (in cases where G is a member of some class of groups that has an explicit name). Algorithms for performing such an analysis have been developed for 2-transitive groups by Cameron, Cannon and Neumann [11] and for arbitrary permutation groups by Cannon and Neumann [20]. Since these algorithms utilize most of the known permutation group algorithms, this paper may be regarded as setting the stage for the work described in those papers.

2 TRANSITIVITY, REGULARITY AND PRIMITIVITY

Permutation group algorithms divide fairly naturally into

*This research was supported by the Australian Research Grants Scheme and the National Science Foundation.

those that require a representation of the set of elements of the group and those that do not. In this section the latter class of algorithms will be discussed. Throughout the paper we let G denote a permutation group which acts faithfully on the finite set Ω of cardinality n. The symmetric group on Ω will be denoted by S_Ω. We suppose that G is generated by the set $X = \{g_1,\ldots,g_m\}$.

The following simple algorithm can be used to compute the orbit of an element of Ω.

Algorithm ORBIT: Given a group G that acts faithfully on a set Ω, and a generating set $X = \{g_1,\ldots,g_m\}$ for G, construct the orbit Δ of $\alpha \in \Omega$.

i) [Initialize] $\Delta \leftarrow \{\alpha\}$; $Q \leftarrow \{\alpha\}$.

ii) [Take the next unprocessed point from the queue Q]. If Q is the null set, stop. Otherwise, let β be some element of Q; $Q \leftarrow Q - \{\beta\}$, $i \leftarrow \emptyset$.

iii) [Form the image of β under the next generator]. $i \leftarrow i + 1$; If $i > m$, go to (ii); $\gamma \leftarrow \beta g_i$. If $\gamma \notin \Delta$, $\Delta \leftarrow \Delta \cup \{\gamma\}$, $Q \leftarrow Q \cup \{\gamma\}$. Go to (iii).

Algorithm ORBIT can easily be modified so as to construct all the orbits of G on Ω. Further, if $\Delta = \{\alpha_1,\ldots,\alpha_r\}$, ORBIT can be modified so as to save a set of elements $\{x_1,\ldots,x_r\}$ from G such that $\alpha_1 x_i = \alpha_i$. Provided that the algorithm is carefully implemented, it may be applied to groups having degrees well in excess of 100,000. The time complexity of the algorithm is $O(mn)$.

Although it is a straightforward task to generalize the algorithm so as to obtain orbits of G on Ω^k, $k \geq 2$, this is rarely practical because of the cardinality of Ω^k. If G is transitive and generators for G_α, $\alpha \in \Omega$, are known, the orbits of G on $\Omega\times\Omega$ may be obtained from the orbits of G_α on Ω by means of the following correspondence: If Γ is an orbit of G on $\Omega\times\Omega$, the correspondence $\Gamma \leftrightarrow \Gamma(\alpha) = \{\beta \mid (\alpha,\beta) \in \Gamma\}$ is a bijection between the orbits of G on $\Omega\times\Omega$ and the orbits of G_α on Ω.

Consider the problem of determining whether G acts regularly on Ω. A well-known theorem says that G is semiregular if and only if the centralizer of G in S_Ω is transitive (see Wielandt [37], pg.9). The following lemma is a consequence:

Lemma 2.1. Let $G = \langle X \rangle$ be a transitive group acting on the set Ω and suppose $\alpha \in \Omega$. If, for each x in X there is an element z in S_Ω such that

a) z centralizes X, and

b) $\alpha x = \alpha z$,

then G is regular.

The following algorithm of Sims is based on this lemma:

Algorithm REGULAR: Given a group G acting faithfully on the set $\Omega = \{\alpha_1,\ldots,\alpha_n\}$, and a generating set $X = \{g_1,\ldots,g_m\}$ for G, test if G acts regularly on Ω.

i) [Choose a point α] Let α be some point in Ω; $i \leftarrow 0$.

ii) [Take the next generator] $i \leftarrow i + 1$; If $i > m$, then G is regular, so stop. Otherwise, $\gamma \leftarrow \alpha g_i$; $j \leftarrow 0$.

iii) [Construct a mapping $z:\Omega\rightarrow\Omega$] $j \leftarrow j + 1$. If $j > n$ go to (iv). Otherwise, $\beta \leftarrow \alpha_j$; Let w be an element of G mapping α to β and define the image of β under z to be γw; Go to (iii).

iv) [Test whether z is a bijection] If z is a permutation of Ω which commutes with each element of X, go to (ii). Otherwise, G does not act regularly, so stop.

The time complexity of REGULAR is $O(m^2n)$. It is not difficult to devise a generalization of this algorithm which will test an intransitive group for being semiregular. Algorithm REGULAR is applicable to groups having degree up to 100,000 provided that the generating set X is small.

We now consider the problem of determining whether a transitive group G acts primitively on Ω. This corresponds to showing that there are no proper G-invariant partitions of Ω. Our approach is to determine the finest G-invariant partition such that the pair $\{\alpha,\beta\}$, $\alpha,\beta \in \Omega$, is contained in the same subset (block). The group is primitive if for a fixed α and for each choice of β from the set $\Omega - \{\alpha\}$, only the trivial partition is obtained.

Algorithm PRIMITIVE: Given a transitive group G acting faithfully on the set $\Omega = \{\alpha_1,\ldots,\alpha_n\}$, and a generating set $X = \{g_1,\ldots,g_m\}$ for G, test whether G is primitive.

i) [Choose α] $\alpha \leftarrow \alpha_1$; $i \leftarrow 1$.

ii) [Choose β] $i \leftarrow i + 1$; If $i > n$, then G is primitive, so stop. Otherwise, $\beta \leftarrow \alpha_i$, and create the partition $\alpha,\beta|\gamma_1|\ldots|\gamma_{n-2}$, where $\gamma_1,\ldots,\gamma_{n-2}$ are the distinct elements of $\Omega - \{\alpha,\beta\}$.

iii) [Initialize loop over the generators] $j \leftarrow 0$.

iv) [Apply a generator to the current partition] $j \leftarrow j + 1$; If $j > m$ go to (v). $g \leftarrow g_j$. Suppose the current partition is $\Delta_1|\Delta_2|\ldots|\Delta_t$; form the partition $\Delta_1 g|\Delta_2 g|\ldots|\Delta_t g$; If for each k we have $\Delta_k g \subseteq \Delta_\ell$ for some ℓ, go to (iv). Otherwise, form a new partition by merging Δ_p and Δ_q, $p \neq q$, where both Δ_p and Δ_q intersect some $\Delta_k g$ nontrivially; go to (iii).

v) [Proper partition?] If the partition consists of two or more sets, then it is a system of imprimitivity for G, so stop. Otherwise, go to (ii).

A particularly efficient version of this algorithm is described in Atkinson [1]. Various means of speeding up the algorithm are described by Atkinson. Suppose a subgroup $H \leq G_\alpha$ can be found. If $\Sigma = \{\beta_1,\ldots,\beta_p\}$ is a set of representatives for the orbits of H, then it suffices to let β run through the elements of $\Sigma-\{\alpha\}$ rather than the elements of $\Omega-\{\alpha\}$. A large subgroup of G_α can be computed cheaply using the random Schreier algorithm (section 3).

The time complexity of the primitivity test is $O(mn^2)$. Provided that all the speed-up tricks are used, algorithm PRIMITIVE may be applied to groups having degree up to 100,000.

It is important to be able to rapidly determine if G contains the alternating group A_Ω. The most useful result for this purpose is the following theorem of Jordan:

Theorem 2.2. Let G be a primitive group of degree n. If G contains a p-cycle, where $p \leq n-3$ is a prime, then G is the alternating or symmetric group of degree n.

This result can be used as the basis of a rather efficient test for A_n or S_n. Firstly, Algorithm PRIMITIVE is used to test G for primitivity. Assuming that it is, a number of random elements are examined in the hope of finding one that satisfies the conditions of the above theorem. Rather than looking directly for an element that is a p-cycle, it is much more effective to look for an element x containing a p-cycle

whose length is coprime with the lengths of all the other cycles of x. If, after examining a predetermined number of random elements, no p-cycle of the appropriate length has been found, some version of the Schreier algorithm (section 3) is invoked to determine the order of G. It is possible to design an algorithm along these lines which can recognize that a group is the alternating or symmetric group for degrees up to 100,000 (see Cannon [16]).

3 BASES AND STRONG GENERATING SETS

In this section we introduce some ideas which allow us to represent the set of elements of a finite permutation group G in a particularly compact manner. This representation enables us to test membership in G and to run through the elements of G without repetition. Further, the order of G can be obtained immediately from this representation.

A sequence $B = \{\beta_1,\ldots,\beta_k\}$ of distinct points from Ω is called a base for G if the only element of G that fixes B pointwise is the identity. If $B = \{\beta_1,\ldots,\beta_k\}$ is a base for G, it is convenient to define the following symbols:

$G^{(i)} := G_{\beta_1,\ldots,\beta_{i-1}}$, so that $G^{(1)} = G$ and $G^{(k+1)} = 1$;

$\Delta_i :=$ orbit of $G^{(i)}$ containing β_i (the i^{th} basic orbit);

$U_i :=$ a right transversal for $G^{(i+1)}$ in $G^{(i)}$.

A subset S of G is said to be a strong generating set for G relative to B if $G^{(i)} = \langle S \cap G^{(i)} \rangle$ $(i = 1,\ldots,k)$. Thus, S contains generators for each subgroup in the chain of stabilizers.

$$G = G^{(1)} > G^{(2)} > \ldots G^{(k)} > G^{(k+1)} = 1.$$

Given a base B and strong generating set S for G, the pair of sets (Δ_i,U_i) $(i = 1,\ldots,k)$ can be constructed simultaneously using a variant of Algorithm ORBIT. The notions of base and strong generating set were introduced by Sims in 1970 (see [30] and [31]).

The cardinality of the set B is called the length of the base. A regular group must have a base of length one, while the symmetric group of degree n must have a base of length $n-1$. However, the length

of a base is not a group invariant. One reason for the importance of the notion of a permutation group base is the observation that important classes of groups will more often than not have bases whose lengths are small compared to the degree. In particular, if the alternating groups are excluded, this is true for simple groups. For example, the length of a typical base for Held's group (order 4,030,387,200) in its degree 2058 permutation representation lies between 8 and 12. Babai has shown that a primitive but not 2-transitive group has a base of length at most $4n^{\frac{1}{2}}\log n$. It is an easy exercise to show that the number of strong generators is bounded by n^2.

Every element g of G has a unique expression as a product $u_k u_{k-1} \cdots u_1$ with $u_i \in U_i$ $(i = 1,\ldots,k)$. The following algorithm may be used to determine whether the permutation g is an element of G.

Algorithm STRIP: Given a permutation group G acting on a set Ω, and an element g of S_Ω, determine whether g is an element of G. It is assumed that the sets S, B, Δ_i $(i = 1,\ldots,k)$ and U_i $(i = 1,\ldots,k)$ are defined for G.

i) [Initialize] $i \leftarrow 0$; $h \leftarrow g$.

ii) [Find the next factor u_i in the product $u_k \cdots u_1$] $i \leftarrow i + 1$; If $\beta_i h \notin \Delta_i$, then g is not in G, so stop. Otherwise, $h \leftarrow h u_i^{-1}$, where u_i is the unique element of U_i such that $\beta_i h = \beta_i u_i$.

iii) [Reached end of base] If $i < k$, go to (ii). Otherwise, if $h = 1$ then g is in G, while if $h \neq 1$, then g is not in G.

This process is often referred to as stripping the element g. If on termination $h \neq 1$, then g is not an element of G, and h is called the residue of G relative to B and S.

In 1967, Sims [29] described a method for constructing a base and strong generating set for G which is based on the following lemma of Schreier (Hall [24], p. 96):

Lemma 3.1. Let G be a group with generating set X, and let H be a subgroup of G. If U is a right transversal for H in G such that U contains the identity element then H is generated by the set

$$S = \{ux\phi(ux)^{-1} \mid u \in U,\ x \in X\}$$

where $\phi(ux)$ is the unique element of U such that $Hux = H\phi(ux)$.

In theoretical terms, the method may be described as follows: For β_1 choose any point not fixed by every element of X. Use Algorithm ORBITS, to construct Δ_1 and U_1. Now apply Schreier's lemma to construct a set of generators for $G_{\beta_1} = G^{(2)}$. By induction we successively construct the base points $\beta_2,\ldots,\beta_k$ and generators for the stabilizers $G^{(3)},\ldots,G^{(k+1)}$. The algorithm terminates when we reach the trivial subgroup.

This method is impractical because of the large number of Schreier generators that will be constructed $(1 + [G:G_{\beta_1}](|X|-1)$ for $G_{\beta_1})$. In fact, however, a tiny fraction of these generators will suffice. Sims' idea was the following: As soon as a non-trivial generator is obtained for $G^{(2)}$, it is used to generate part of the orbit Δ_2 (and hence part of U_2). Sufficient information is now available to deduce a generator of $G^{(3)}$. If this generator is non-trivial, it is immediately used in the same way to deduce a generator for $G^{(4)}$ etc. Whenever a new Schreier generator t is formed, Algorithm STRIP is used to determine whether or not it lies in the subgroup of G generated by the currently known generators of $G^{(2)},\ldots,G^{(k+1)}$. If t has a non-trivial residue after the application of STRIP, then that residue is added to S. By this means, the number of generators actually saved for a subgroup $G^{(i)}$ $(i = 2,\ldots,k)$ is typically five or six.

This algorithm, labelled the Schreier algorithm by Sims, is sufficiently powerful for it to be applicable to groups having degrees up to 500. It has been implemented by a number of people and it is perhaps the most important existing algorithm for permutation groups. A number of variants of this algorithm will be briefly described. First, however we make the following observation: If g is an element of G and $B = \{\beta_1,\ldots,\beta_k\}$ is a base for G, then g is uniquely determined by the sequence, $\{\beta_1 g,\ldots,\beta_k g\}$. This sequence will be referred to as the base image representation of G.

Variant 1: Known base Schreier algorithm. Suppose a base B is known for the group G and it is necessary to compute strong generators for G relative to B. Most formation of products of permutations x and y which arise in the Sims algorithm may be replaced by computation of the base images corresponding to these products: $\{(\beta_1 x)y,\ldots,(\beta_k x)y\}$.

If k is significantly smaller than n, this can result in a considerably faster algorithm. This variant is particularly important when it is necessary to construct bases and strong generating sets for a number of distinct subgroups of a fixed group G, for a base for G is also a base for any subgroup of G. See Butler and Cannon [8] for details.

Variant 2: Schreier Todd-Coxeter algorithm. The standard form of the Sims algorithm is rather inefficient when one considers the fact that only a tiny fraction of the Schreier generators constructed are actually needed: the vast majority are redundant and are simply discarded. The problem is recognizing when a sequence of points B and a set of elements S constitutes a base and strong generating set for the group G. In 1974, Sims [32] described an approach to this problem which is based on the use of the Todd-Coxeter algorithm (see Cannon et al [17]) for enumerating cosets in a finitely presented group. This form of the Schreier algorithm was subsequently refined by Leon [26,27]. Given a sequence of points B and a set of elements S belonging to the group G this algorithm may be regarded as a means of verifying whether B and S constitute a base and strong generating set, respectively, for G. Commencing with $G^{(k)}$, the algorithm inductively constructs a presentation for each stabilizer $G^{(i)}$ $(i = k,\ldots,i = 1)$. The Todd-Coxeter algorithm is used to define $|\Delta_i|+1$ cosets of $G^{(i+1)}$ in the group defined by those relations currently known to hold for $G^{(i)}$. Letting representatives for these cosets act on β_i and comparing the resulting sequence of points with the orbit Δ_i, we are led either to a new relation for $G^{(i)}$, or a new strong generator for $G^{(i)}$. The process terminates for $G^{(i)}$ when the Todd-Coxeter algorithm terminates with exactly $|\Delta_i|$ cosets defined.

Because of overheads, this algorithm is often slower than the standard Schreier algorithm for groups having degree less than 100. As the degree increases the Schreier Todd-Coxeter algorithm becomes more competitive. For degrees greater than 1,000 it is typically several times faster than the standard Schreier algorithm. In particular, it enables us to construct bases and strong generating sets for groups of degree 10,000.

Variant 3: The random Schreier algorithm. In many situations it is useful to be able to rapidly construct an "approximation" to a base and strong generating set for a group G. We say that the sets B and S approximate a base and strong generating set for G if either B and

S already constitute a base and strong generating set, or they do so upon the addition of a small number of elements. In his work on the Schreier Todd-Coxeter algorithm Leon devised the following algorithm:

Algorithm APPROX: Given a permutation group G, construct an approximation for a base and strong generating set for G. The algorithm is to terminate after m consecutive unsuccessful attempts to find a new strong generator.

i) [Initialize]. Create a base B and strong generating set for the trivial group acting on Ω, i ← 0.

ii) [Consider a new random element]. i ← i + 1. If i > m then stop. Generate a random element g of G and apply Algorithm STRIP so as to obtain the residue h. If h = 1, i ← i + 1 and go to (ii). Otherwise, add h to S and, if necessary, add a new point to B; i ← 0; Go to (ii).

Random elements may be obtained by generating random words in the given generators X of G. It is desirable that the lengths of these random words vary with the lower bound on |G| given by the current values of B and S. It has been found experimentally that if the stopping parameter m is set to 20, then almost always this algorithm will return a complete base and strong generating set for G.

There are a number of important applications of this algorithm:

a) It can be used to quickly obtain a large subgroup H of a one-point stabilizer G_α in a transitive group G. Knowledge of the orbits of H may greatly reduce the number of pairs $[\alpha,\beta]$ that must be considered by the primitivity algorithm (algorithm PRIMITIVE).

b) If the order of G is known in advance, the random Schreier algorithm gives a very fast method for computing a base and strong generating set for G.

c) The efficiency of the Schreier Todd-Coxeter algorithm may be enhanced by first using the random Schreier algorithm to obtain a close approximation to a base and strong generating set.

For many applications it is desirable to have a particular sequence of points from Ω appear as an initial segment of a base for G. Given a base $B = \{\beta_1,\ldots,\beta_k\}$ and a sequence of points $C = \{\gamma_1,\ldots,\gamma_\ell\}$, there exists an efficient algorithm due to Sims [31] which constructs a strong generating set for G relative to the base $\{\gamma_1,\ldots,\gamma_\ell,\delta_1,\ldots,\delta_m\}$, where $\delta_1,\ldots,\delta_m$ are points from Ω chosen to complete the sequence

$\{\gamma_1,\ldots,\gamma_\ell\}$ to a base. At the heart of the Sims algorithm is a procedure which modifies the current strong generating set so as to correspond to the base obtained by interchanging two adjacent points in the current base. The algorithm involves mainly the computation of orbits. In [6], Butler describes an improvement to the algorithm whereby he precedes the Sims algorithm with the conjugation of G by an element g in S_Ω which maps as large an initial segment of B as possible into the corresponding initial segment of G.

The existence of a fast algorithm for changing base means that it is possible to obtain the pointwise stabilizer $G_{\gamma_1,\ldots,\gamma_\ell}$ of any sequence of distinct points from Ω very cheaply.

4 CONSTRUCTING SUBGROUPS I: NORMAL CLOSURE

Let H be a subgroup of G. The normal closure $ncl_G(H)$ may be constructed using the following algorithm which forms the closure of H under conjugation by the generators of G:

Algorithm NCL: Given a subgroup $H = \langle y_1,\ldots,y_s\rangle$ of the permutation group $G = \langle x_1,\ldots,x_r\rangle$, determine the normal closure of H in G. At the outset we assume that a base and strong generating set are known for H.

i) [Initialize] $K \leftarrow H$; $i \leftarrow 0$; $t \leftarrow s$.

ii) [Choose the next generator of K] $i \leftarrow i + 1$; If $i > t$, then $ncl_G(H) \leftarrow K$, and stop. Otherwise, $j \leftarrow 0$.

iii) [Choose the next generator of G] $j \leftarrow j + 1$. If $j > r$, go to (ii). Otherwise, $g \leftarrow y_i^{x_j}$. If $g \in K$, go to (iii).

iv) [Extend K] $t \leftarrow t + 1$; $y_t \leftarrow g$; $K \leftarrow \langle K,g\rangle$. Use the Schreier algorithm to construct a base and strong generating set for K. Go to (iii).

Algorithm STRIP is used to perform the element membership test in step (iii). The most time consuming part of this algorithm is the construction of new strong generating set each time K is extended. Usually a base will be known for K (since a base for G is also a base for K), so that for groups of moderate degree Variant 1 of the Schreier algorithm is appropriate. For groups of large degree it may be necessary to employ the Schreier Todd-Coxeter algorithm. In cases where the order of $ncl_G(H)$ is known in advance the random Schreier algorithm is the method of choice.

If $H = \langle y_1,\ldots,y_s\rangle$ and $K = \langle z_1,\ldots,z_t\rangle$ are normal subgroups